ROZDZIAŁ 2. TEORETYCZNE PODSTAWY KOMPRESJI BEZSTRATNEJ

Konstrukcja różnych metod kodowania danych polega przede wszystkim na określeniu sposobu realizacji dwu etapów bezstratnego procesu kompresji: modelowania i binarnego kodowania. Modelowanie polega na zbudowaniu takiego modelu kompresowanych danych, który najlepiej opisze rzeczywistą, zawartą w zbiorze danych informację. Model ten umożliwi eliminację różnego typu nadmiarowości uzyskując prostszą reprezentację danych oryginalnych. Metoda binarnego kodowania opierając się na analizie statystycznej przetwarzanych danych przyporządkowuje minimalny kod ciągowi symboli otrzymanych z modelowania. Wykorzystuje się tutaj przede wszystkim narzędzia analityczne dostarczane przez teorię informacji, której podstawy przedstawiono poniżej w punkcie 2.1.

Można by właściwie konstruować algorytmy kodowania bez definiowania pojęcia informacji, określając jedynie obiekty elementarne w kompresowanym zbiorze danych, szacując częstości ich występowania i konstruując kod jednoznacznie dekodowalny według pewnej reguły, mierząc jego efektywność. Zgodnie jednak z historyczną rolą prac Shannona oraz przyjętą filozofią kompresji, zarysowaną we wprowadzeniu, zasadniczym zabiegiem wstępnym przy konstrukcji algorytmów kompresji jest określenie czym jest informacja podstawowa, aby na tej podstawie zbudować dla niej reprezentację fundamentalną. Pojęcie informacji użytecznej oraz nieprzydatnej jest także kluczowe w kompresji danych medycznych, szczególnie obrazowych. Jakkolwiek w przypadku kompresji odwracalnej pojęcie informacji ma czysto statystyczny charakter i nie sposób mówić o informacji użytecznej, to jednak w opinii Autora konstruowanie technik bezstratnych na bazie teorii informacji daje jednolity opis zagadnienie kompresji, szczególnie w kontekście nowych technik stratnych-do-bezstratnych.

Ponadto w punkcie 2.2 przedstawiono ogólną charakterystykę technik kompresji bezstratnej, zwracając uwagę na elementy wspólne oraz różnice rozwiązań stosowanych w najbardziej znanych metodach kodowania danych.

2.1. Elementy teorii informacji

Pojęcie informacji

Definiowanie informacji

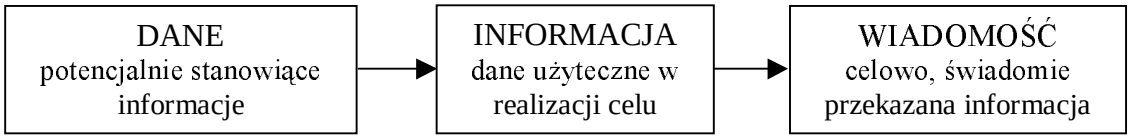
Informacja jest pojęciem pierwotnym, a więc ścisłe jej zdefiniowanie za pomocą prostszych pojęć nie jest możliwe. Pozostaje więc wyjaśnienie sensu tego pojęcia, które będzie odpowiadać jego intuicyjnemu rozumieniu.

Informacją nazywamy to wszystko, co można zużytkować do realizacji założonego celu poprzez właściwy (bardziej sprawny, efektywniejszy) wybór działań zmierzających do jego osiągnięcia. Można także powiedzieć, że informacją jest to wszystko, co jest użyteczne dla konkretnego odbiorcy, z jego punktu widzenia. Definicja taka jest jednak w dużym stopniu niejednoznaczna, kto bowiem ma decydować o użyteczności tego ‘wszystkiego’ i celu do realizacji, czy sam podmiot użytkujący, czy jakieś grono ekspertów, matka dobierająca właściwe lektury i programy dla dzieci? Ma więc informacja silny aspekt subiektywny, jakkolwiek można też formułować informacje w sensie bardziej zobiektywizowanym. Taka próba podejmowana jest zapewne w codziennych telewizyjnych programach informacyjnych,

takich jak Wiadomości, Fakty, Informacje czy Wydarzenia, kiedy to świadomie dokonuje się selekcji materiału z kryterium możliwie szerokiego kręgu odbiorców, dla którego staną się one rzeczywiście informacją (zróbmy takie upraszczające założenie, nie znam bowiem rzeczywistych intencji Autorów tych audycji).

Uzyskanie informacji wiąże się z pewnym kosztem. Koszt ten, wynikający z charakteru zjawiska, którego dotyczy informacja i postaci, jaką ona przyjmuje, jest jednak zazwyczaj znacznie mniejszy od korzyści wynikających z ich stosowania. Jako przykłady różnorodnych postaci informacji można podać zbiór liczb, słowa i zdania, parametry charakteryzujące stan fizyczny ciał lub ich układów, przebiegi wielkości fizycznych w funkcji czasu, dźwięki, obrazy, opisy zdarzeń, itp.

Aby lepiej zrozumieć sens pojęcia informacji, można dokonać rozróżnienia podstawowych pojęć jak na rys. 2.1:



Rys. 2.1. Wyjaśnienie pojęcia informacji. Zależność pomiędzy informacją oraz zbiorem danych potencjalnie stanowiącym informację i wiadomością jako celowo przekazaną informacją.

Względność informacji

To co stanowi informację dla realizacji pewnego celowego działania, może być bezużyteczne bądź nawet utrudniające realizację innego celowego działania. Przykładowo informacja o chorobie przekazana pacjentowi o silnej woli stanowi często pomoc w procesie leczenia, podczas gdy dla człowieka słabego psychicznie, załamanego, może stanowić zagrożenie dla dalszej terapii. Przekazanie wiadomości o zwycięstwie polskiej drużyny piłkarskiej dla wielu osób staje się rzeczywiście informacją z punktu widzenia jej użyteczności, dając satysfakcję, radość z sukcesu rodaków - naszego sukcesu, a nawet wzrost dumy narodowej i korzyści finansowe uruchamiając mechanizmy zakupu nowych piłkarzy przez kluby, itp. Dla innych osób zaś dane o wynikach piłkarskich nie są informacją, bo z racji braku jakiegokolwiek zainteresowania są one zupełnie nieprzydatne, nie istnieje po prostu żaden cel, któremu one mogą służyć.

Klasyfikacja informacji

Ze względu na postać, w jakiej wyrażona jest informacja, można wyróżnić informację ze zbiorem ziarnistym (zbiór o skończonej liczbie elementów) oraz informację ze zbiorem ciągłym (obok jednej informacji dowolnie blisko można znaleźć inne informacje). Z racji praktycznych zastosowań do kompresji danych w postaci cyfrowej, interesować nas będzie przede wszystkim pojęcie informacji ze zbiorem ziarnistym. Można dla niej określić tzw. ciągi informacji, które są ciągiem symboli ze zbioru informacji elementarnych (alfabetu) pojawiających się w pewnej kolejności, stanowiącej istotę informacji. Zbiór informacji elementarnych zawiera wszystkie podstawowe symbole wyrażające informację. Przykładowo:

- zbiór informacji elementarnych: {a,b,c},
- ciąg informacji: (a,a,c,a,b,b,b,c,c,a,c,b, ...)

Hierarchiczny charakter informacji

Informacja ma zazwyczaj charakter hierarchiczny i może być wyrażana, opisywana, przybliżana na różnych poziomach hierarchii. Przykładem takiej hierarchii może być sposób tworzenia całych programów, sesji pracy komputera, na które składa się określona informacja, budowana na kolejnych poziomach hierarchii danych według następującego schematu:

sygnał elektryczny $\rightarrow$ ciąg bitów $\rightarrow$ 8 - bitowe słowo (np. kod ASCII) $\rightarrow$ ciągi słów (złożone z kilku znaków) $\rightarrow$ zdania (instrukcje) $\rightarrow$ programy $\rightarrow$ sesja pracy komputera.

Modele źródeł informacji

Znajdowanie odpowiednich modeli źródeł danych, opisujących z dobrym przybliżeniem analizowane rzeczywiste zbiory (strumienie) danych, umożliwia konstruowanie efektywnych technik kompresji. Modele te pozwalają na mniej lub bardziej dokładne określenie ilości informacji zawartej w zbiorze danych, a co za tym idzie mniej lub bardziej adekwatne do rzeczywistości wyznaczenie minimalnej bitowej reprezentacji tej informacji, która stanowi naturalne ograniczenie efektywności koderów bezstratnych opartych na tym modelu.

Strumień zawierający n-bitowe wartości danych bez względu na ich fizyczne źródło i sposób ich rejestracji może być traktowany jako obiekt wygenerowany przez źródło S o alfabecie A_S składającym się z 2^n symboli reprezentujących wszystkie możliwe wartości danych w zbiorze. Dodatkowa charakterystyka kodowanych zbiorów danych jest związana ze sposobem generacji kolejnych danych w strumieniu wejściowym przez źródło.

Zasadniczo, aby zbudować model źródła informacji, potrzeba określić:

- strukturę zbioru postaci, jakie może przyjmować informacja,
- strukturę informacji pojedynczej,
- niedeterministyczny mechanizm wybierania pojawiającej się w danej chwili informacji pojedynczej ze zbioru postaci, jakie może ta informacja przyjmować.

Najczęściej w praktyce konstruując źródło informacji trzeba przyjmować rozwiązania kompromisowe pomiędzy wiernością obiektowi opisywanemu przez dane źródło, a prostotą modelu dającą łatwość i wygodę operacji wykonywanych na źródle, takich jak analiza i przetwarzanie generowanej przez nie informacji.

W przeszłości podejmowano próby budowy deterministycznych modeli źródeł informacji, wykorzystujące np. przebiegi sinusoidalne, falę prostokątną, deltę Diraca. Trzeba je jednak w ogólności uznać za nieprzydatne do celów kompresji (jak również w przeważającej liczbie innych zastosowań). Przy konstruowaniu źródeł informacji należy wykorzystać wszelką informację dostępną a priori, aby uprościć model statystyczny 'niewiadomej', czyli rzeczywistej informacji.

Model probabilistyczny

Przyjmując probabilistyczny model źródła informacji zakładamy, że informacja jest realizacją pewnej zmiennej losowej (procesu losowego) o określonych własnościach statystycznych. Dla takich modeli obowiązują twierdzenia graniczne, prawa wielkich liczb, z których wynika, że przy dostatecznie dużej liczbie niezależnych obserwacji częstości występowania określonych postaci informacji będą zbliżone do prawdopodobieństw ich występowania. Częstościowe określanie prawdopodobieństw jest tym dokładniejsze, im liczniejsze zbiory są podstawą do określenia prawdopodobieństw.

Trzeba również zastrzec, iż interesują nas jedynie modele z pełną informacją statystyczną. Istnieją też inne probabilistyczne modele informacji, ale są one zasadniczo mało skuteczne.

Proces generacji informacji modelowany za pomocą źródła informacji *S* polega na emisji przez źródło sekwencji symboli wybranych ze skończonego alfabetu według pewnych reguł. Określone są one przez parametry procesu stochastycznego skojarzonego z tym modelem źródła (rozkład i jego momenty, charakterystyka stacjonarności, itd.).

Model bez pamięci - DMS

Najprostszą postacią źródła informacji *S* jest dyskretne źródło bez pamięci DMS (ang. discrete memoryless source), w którym symbole sukcesywnie emitowane przez źródło są statystycznie niezależne. Źródło DMS jest całkowicie zdefiniowane przez zbiór wszystkich możliwych symboli zmiennej losowej *s*, czyli alfabet $A_S = \{a_1, a_2, ..., a_n\}$ oraz zbiór wartości prawdopodobieństw występowania poszczególnych symboli alfabetu $P_S = \{p_1, p_2, ..., p_n\}$, gdzie $\Pr(s = a_i) = P(a_i) = p_i$, $p_i \geq 0$ i $\sum_{i=1}^{n} p_i = 1$.

Źródłem bez pamięci jest więc na przykład źródło *S* opisane przez: $A_S = \{0,1\}$ oraz $P_S = \{\frac{1}{5}, \frac{4}{5}\}$.

Można sobie wyobrazić, że źródło o alfabecie A_S generuje zamiast pojedynczych symboli bloki *N* symboli z alfabetu źródła, czyli struktura pojedynczej informacji jest ciągiem *N* dowolnych symboli źródła. W takim przypadku można zdefiniować nowe źródło S^N o n^N elementowym alfabecie, zawierającym wszystkie możliwe *N* - elementowe ciągi symboli. Rozszerzony alfabet tego źródła jest następujący: $A_S^N = \sum_{i=1}^{n^N} \overline{a}_i = \sum_{i_{1..N}=1}^{N} (a_{i_1}, a_{i_2}, ..., a_{i_N})$, a prawdopodobieństwo *i*-tego elementu alfabetu wynosi $P(\overline{a}_i) = P(a_{i_1})P(a_{i_2}) \cdots P(a_{i_N})$. Źródło S^N jest nazywane *N*-tym rozszerzeniem źródła *S*.

Przykładem alfabetu źródła będącego potrójnym rozszerzeniem binarnego źródła S_2 jest zbiór $A_{S_2}^3 = \{0,1\}^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$.

Model z pamięcią - CSM

Jakkolwiek model DMS spełnia założenia prostej, wygodnej w analizie struktury, to jednak w wielu zastosowaniach jest nieprzydatny ze względu na małą zgodność z charakterem opisywanej informacji. Założenie o statystycznej niezależności kolejnych zdarzeń emisji symbolu jest bardzo rzadko spełnione w praktyce. Kolejne wartości w zbiorze danych wynikają z pewnych zależności treściowych przekazywanej informacji. Aby lepiej wyrazić rzeczywistą informację zawartą w zbiorze danych, konstruuje się więc tzw. źródło z pamięcią - CSM (ang. conditional source model).

Źródła z pamięcią, najczęściej ograniczoną, pozwalają z większą dokładnością przewidzieć pojawienie się poszczególnych symboli alfabetu źródła (strumień danych staje się lepiej określony przez model źródła). Koncepcja pamięci źródła może być zrealizowana poprzez określenie kontekstu (czasowego, przestrzennego), który ma wpływ na prawdopodobieństwo wystąpienia konkretnych symboli w danej chwili w strumieniu wejściowym. Taki kontekst może być stały lub też definiowany jako funkcja symboli

wcześniej wyemitowanych przez źródło. Wpływ kontekstu na bieżący symbol może być opisany przez rozkłady prawdopodobieństw warunkowych. W przypadku statystycznego kodowania zbiorów tekstowych, 'dobrym' kontekstem okazują się najczęściej najbliższe, kolejne symbole w strumieniu danych wejściowych. Przy kompresji danych obrazowych najlepszy kontekst danego piksela stanowią zazwyczaj piksele sąsiednie w kierunku horyzontalnym, wertykalnym oraz diagonalnym. Realizacja takiego kontekstu wiąże się z pewną złożonością algorytmu kodowania, który musi uwzględniać zależności pomiędzy wartościami pikseli w porządku nie wynikającym bezpośrednio z kolejności przeglądania pikseli obrazu w algorytmie kodowania.

Źródło *S* z pamięcią jest określone poprzez:

- alfabet symboli źródła $A_S = \{a_1, a_2, ..., a_n\}$,
- zbiór wartości kontekstu X czyli alfabet symboli kontekstu postaci $A_X = \{b_1, b_2, ..., b_k\}$,
- prawdopodobieństwa warunkowe $P(a_i \mid b_j)$ dla i =1,2,...,*n*, j =1,2,...,*k*, przy czym

$$P(a_i \mid b_j) = \frac{N(a_i, b_j)}{N(b_j)}, \tag{2.1}$$

  $N(a_i, b_j)$ - liczba łącznych wystąpień symbolu a_i i kontekstu b_j, $N(b_j)$ - liczba wystąpień kontekstu b_j,
- zasadę określania kontekstu *X* w każdej 'chwili czasowej' przez funkcję $f(\cdot)$ symboli już wyemitowanych przez źródło.

Kontekst *X* określany przez $f(\cdot)$ może być konstruowany na różny sposób. Istotnym parametrem jest tutaj rząd kontekstu oznaczony przez *m*, który określa liczbę symboli tworzących kontekst. Załóżmy, że źródło emituje sekwencję danych wejściowych $s_1, s_2, ..., s_i, ...$. Kontekst stanowią często symbole bezpośrednio poprzedzające wartość kodowaną s_i. Wtedy szacujemy model $\Pr(s_i = a_l \mid x_i)$, gdzie $x_i = (s_{i-1}, s_{i-2}, ..., s_{i-m})$, a alfabet kontekstu X^m jest zbiorem wszystkich możliwych *m*-elementowych sekwencji symboli źródła informacji: $x_i = b_j = (a_{j1}, a_{j2}, ..., a_{jm})$, $j_p (p = 1, ..., m) = 1, 2, ..., n$. Istnieje bardzo wiele możliwości konstruowania źródła z pamięcią. Sposób określenia kontekstu może zmieniać się dynamicznie w trakcie procesu kodowania, np. zależnie od lokalnej statystyki. Popularną techniką jest też kwantyzacja kontekstu, kiedy to liniowa kombinacja pewnej liczby sąsiednich symboli warunkuje wystąpienie symbolu dając de facto model warunkowy pierwszego rzędu, jakkolwiek zależny od kilku czy kilkunasto-elementowego sąsiedztwa.

Prostym przykładem źródła z pamięcią może być źródło opisane w sposób następujący:
$A_S = \{0,1,...,255\}$, $A_X = \{31,95,159,223\}$, czyli $A_X \subset A_S$, oraz stała funkcja $f(\cdot)$ określająca kontekst dla kolejnej danej wejściowej s_i jako skalarną wartość $x_i = b_j \in A_X$ zależną od jednoelementowego kontekstu według zależności:

$$x_i = f(s_1, s_2, ..., s_{i-1}) = f(s_{i-1} = a_t) = \begin{cases} 31 & \text{dla} \quad 0 \le a_t < 64 \\ 95 & \text{dla} \quad 64 \le a_t < 128 \\ 159 & \text{dla} \quad 128 \le a_t < 192 \\ 223 & \text{dla} \quad 192 \le a_t \le 255 \end{cases} \tag{2.2}$$

dla i=2, ... , L (L - długość zbioru danych) oraz $x_1 = 31$.

Wartości warunkujące prawdopodobieństwo wystąpienia poszczególnych symboli z alfabetu A_S zależne są od wartości poprzedniej danej w strumieniu wejściowym, przy czym jedynie czteroelementowy alfabet wartości kontekstu ma zaletę niewielkiej liczby stanów modelu prawdopodobieństwa warunkowego.

W powyższych przykładów wynika bardzo powszechnie stosowana realizacja źródła CSM zwana źródłem Markowa.

Model źródła Markowa

Źródło Markowa rzędu *m* jest źródłem, w którym prawdopodobieństwo wystąpienia symbolu a_i źródła zależy od skończonej liczby *m* poprzednich symboli w strumieniu wejściowym, czyli od X^m - kontekstu rzędu *m*. Prawdopodobieństwo to wyznaczone jest przez zbiór prawdopodobieństw warunkowych (oznaczenia jak przy definicji źródła CSM):

$$P(a_i \mid b_j) = P(a_i \mid a_{j_1}, a_{j_2}, ..., a_{j_m}), \qquad \text{gdzie} \quad i, j_t (t = 1, ..., m) = 1, ..., n \qquad (2.3)$$

Można rozważać źródło jako znajdujące się w pewnym stanie, zależnym od skończonej liczby występujących poprzednio symboli. Dla źródła Markowa pierwszego rzędu znajduje się *n* takich stanów, jeden dla każdego z symboli alfabetu źródła, dla drugiego stopnia n^2 takich stanów, jeden dla każdej pary symboli z alfabetu źródła i analogicznie dla źródła *m*-tego rzędu- n^m stanów.

Modele przełączane

Wykorzystanie jednego dobrze dobranego modelu źródła danych czasami okazuje się mało skuteczne. Nawet dynamiczna modyfikacja zbioru prawdopodobieństw określających źródło w trakcie kodowania, uwzględniająca lokalne własności zbioru danych, może być niewystarczająca. W pewnych zastosowaniach, gdy zbiór danych jest realizacją bardziej złożonych procesów niestacjonarnych wykorzystuje się kilka modeli źródeł przełączanych przy kodowaniu poszczególnych partii strumienia danych. Wykorzystując wiedzę a priori lub też lokalną charakterystykę może zmieniać model źródła informacji na najbardziej skuteczny i tym samym zwiększać efektywność kompresji. Kodery, o których tutaj mowa, przełączają modele bez pamięci, modele z pamięcią i różnymi kontekstami itd. Oczywiście schemat przełączania modeli musi być znany w algorytmie dekodującym i dokładnie powtórzony, a więc musi być zapisany w strumieniu wyjściowym kodera.

Metody kompresji z modelami przełączanymi są zazwyczaj bardzo złożone i okazują się najczęściej efektywne przy kompresji bardzo dużych zbiorów danych ze względu na konieczność rzetelnego określenia poszczególnych modeli (chyba że mamy dobrze określoną wiedzę a priori).

Miara ilości informacji

Miara ilości informacji związanej ze źródłem probabilistycznym winna mieć dwie intuicyjne cechy:

- więcej informacji wynika z pojawienia się mniej prawdopodobnego symbolu,
- informacja związana z wystąpieniem kilku niezależnych zdarzeń jest równa sumie informacji zawartej w każdym ze zdarzeń.

Informacja $I(a_i)$ związana z wystąpieniem pojedynczego symbolu a_i źródła *S* określona jest w zależności od prawdopodobieństwa wystąpienia tego symbolu

$p_i = P(s = a_i)$ jako $I(a_i) = \log\frac{1}{p_i}$, $p_i \neq 0$. Jest to tzw. informacja własna (ang. self-similarity) zdefiniowana przez Shannona.

W przypadku strumienia danych do określenia ilości informacji wykorzystuje się pojęcie entropii. Zasadniczo, dla kolejnych wartości s_i pojawiających się w strumieniu wejściowym przybliżonych przez źródło informacji *S* o alfabecie symboli $A_S = \{a_1, a_2, ..., a_n\}$ entropia jest określona przez $H(S) = \lim_{m\to\infty} \frac{1}{m} I_m$, gdzie

$$I_m = -\sum_{j_{1\ldots m}=1}^{n} P(a_{j_1}, a_{j_2}, ..., a_{j_m}) \log P(a_{j_1}, a_{j_2}, ..., a_{j_m}) = \tag{2.4}$$

$$= -\sum_{i_1=a_1}^{a_n}\sum_{i_2=a_1}^{a_n}\ldots\sum_{i_m=a_1}^{a_n} \Pr(s_1 = i_1, s_2 = i_2, ..., s_m = i_m) \log \Pr(s_1 = i_1, s_2 = i_2, ..., s_m = i_m).$$

Tak określona entropia nosi nazwę entropii łącznej, gdyż jest wyznaczana za pomocą łącznego prawdopodobieństwa wystąpienia kolejnych symboli z alfabetu źródła. Zależność (2.4) jest jednak niepraktyczna, gdyż nie sposób określić prawdopodobieństwo łącznego wystąpienia każdej możliwej (określonej przez alfabet) kombinacji symboli źródła w rzeczywistym zbiorze danych. Wymaga to albo dużej wiedzy a priori na temat charakteru zbioru danych, które mamy kompresować, albo nieskończenie dużej liczby danych do analizy (czytaj nieskończenie długiej analizy). Należałoby więc zbudować model źródła informacji, który potrafi określić prawdopodobieństwo łącznego wystąpienia dowolnie długiej i każdej możliwej sekwencji symboli tegoż źródła. Uproszczone, bardziej praktyczne postacie zależności na entropię, aproksymujące wartość entropii łącznej dla danego źródła informacji, wynikają z uproszczonych modeli źródeł.

Entropia źródła może być też rozumiana jako średnia ilość informacji przypadająca na symbol źródła, jaką należy koniecznie zapewnić, aby usunąć wszelką nieokreśloność ze zbioru danych. Podstawa logarytmu używanego w definicjach miar określa jednostki używane do wyrażenia ilości informacji. Jeśli jako podstawę użyjemy 2, wówczas informację zawartą w danym symbolu można wyrażać w bitach (czyli na ilubitach można tę informacje zapisać), a entropia według równania (2.4) z logarytmem o podstawie 2 wyraża w bitach na symbol średnią ilość informacji zawartą w zbiorze danych.

Dla poszczególnych modeli źródeł informacji można określić ilość informacji generowanej przez te źródła. Należy jednak podkreślić, iż obliczona dla konkretnego źródła ilość informacji tym lepiej będzie przybliżać rzeczywistą informację zawartą w pewnym zbiorze danych (określaną asymptotycznie miarą entropii łącznej), im wierniejszy model źródła informacji został skonstruowany.

Entropia źródła DMS

Jeżeli przybliżymy zbiór danych źródłem DMS, wówczas uśredniając informację własną po wszystkich symbolach alfabetu źródła wyznaczymy średnią ilość informacji przypadającą na symbol źródła, czyli entropię tego źródła:

$$H(S_{\mathrm{DMS}}) = -\sum_{i=1}^{n} P(a_i) \log_2 P(a_i), \tag{2.5}$$

gdzie *n* oznacza ilość symboli a_i w alfabecie. Zachowuję się przy tym następującą konwencję: dla $P(a_i) = 0$ wartość $-0 \cdot \log 0 \equiv 0$, gdyż $\lim_{\phi\to 0^+} \phi \log 1/\phi = 0$. Entropię tą można nazwać entropią bezwarunkową (od użytej formy prawdopodobieństwa). W

przypadku, gdy źródło bez pamięci nie najlepiej opisuje kodowany zbiór danych obliczona entropia źródła DMS jest wyraźnie większa od entropii łącznej, czyli nie jest w tym przypadku najlepszą miarą informacji. Rzeczywista informacja zawarta w zbiorze danych jest pomniejszona o informacje zawarte w kontekście wystąpienia kolejnych symboli.

Zależności pomiędzy danymi w tym strumieniu lepiej opisuje model z pamięcią, a entropia tego źródła jest bliższa rzeczywistej ilości informacji zawartej w kompresowanym zbiorze danych. Zależność pomiędzy entropią łączną $H(S) = H(X,S)$ źródła o zdefiniowanym kontekście X, warunkową $H(S \mid X)$ oraz tzw. entropią brzegową $H(X)$ jest następująca:

$$H(X,S) = H(S \mid X) + H(X). \tag{2.6}$$

Przykładem miary informacji źródeł z pamięcią będzie entropia wyznaczona dla źródeł Markowa, znajdujących bardzo częste zastosowanie w praktyce kompresji.

Entropia źródła Markowa

Aby obliczyć za pomocą modelu źródła Markowa ilość informacji (średnio na symbol źródła) zawartą w kodowanym zbiorze danych, zamiast szacowania entropii łącznej wykorzystuje się zbiór prawdopodobieństw warunkowych zakładając rząd m modelu źródła. Określa się wówczas tzw. entropię warunkową źródła znajdującego się w pewnym stanie $(a_{j1}, a_{j2}, ..., a_{jm})$ jako:

$$H(S \setminus s_{j1}, s_{j2}, ..., s_{jm}) = -\sum_{i=1}^{n} P(s_i \mid s_{j1}, s_{j2}, ..., s_{jm}) \log_2 P(s_i \mid s_{j1}, s_{j2}, ..., s_{jm}). \tag{2.7}$$

Następnie oblicza się średnią entropię warunkową źródła S jako sumę ważoną entropii warunkowych po kolejnych stanach źródła wynikających ze wszystkich możliwych konfiguracji (stanów) kontekstu X^m:

$$A_X^m = \bigcup_{j_{1...m}=1}^{n} (a_{j_1}, a_{j_2}, ..., a_{j_m}) = \bigcup_{i_1=a_1}^{a_n} \bigcup_{i_2=a_1}^{a_n} .... \bigcup_{i_m=a_1}^{a_n} (s_1 = i_1, s_2 = i_2, ..., s_m = i_m), \tag{2.8}$$

przy czym wagami są prawdopodobieństwa przebywania źródła w danym stanie:

$$H(S \mid X^m) = \sum_{A_x^m} P(a_{j_1}, a_{j_2}, ..., a_{j_m}) H(S \mid a_{j_1}, a_{j_2}, ..., a_{j_m}), \tag{2.9}$$

czyli

$$H(S \mid X^m) = -\sum_{A_x^m} \sum_{i=1}^{n} P(a_i, a_{j_1}, a_{j_2}, ..., a_{j_m}) \log_2 P(a_i \mid a_{j_1}, a_{j_2}, ..., a_{j_m}). \tag{2.10}$$

Tak określona średnia entropia warunkowa źródła Markowa rzędu m jest mniejsza lub równa entropii bezwarunkowej. Jest ona pomniejszona o średnią ilość informacji zawartą w kontekście wystąpienia każdego symbolu strumienia danych.

Jednocześnie entropia warunkowa źródła Markowa rzędu m jest niemniejsza niż entropia łączna wyznaczona dla konkretnego zbioru danych, opisanego prawdopodobieństwami łącznego wystąpienia dowolnej konfiguracji symboli źródła informacji - równanie (2.4). Jeśli przyjęty kontekst rzędu m nie obejmuje wszystkich zależności pomiędzy symbolami strumienia wejściowego, wówczas model Markowa jest jedynie przybliżeniem źródła informacji, a wyznaczona entropia warunkowa jest większa od entropii łącznej. W praktyce oznacza to, że zbudowanie bezstratnego kodera w oparciu o model Markowa wyższego rzędu niż m może (hipotetycznie) poprawić jego efektywność. Tą

zależność pomiędzy różnymi rodzajami entropii, związanymi z modelami źródeł informacji opisującymi z większym lub mniejszym przybliżeniem informację zawartą w konkretnym zbiorze danych, jest następująca:

$$H(S) \le H(S \mid X^m) \le H(S_{\mathrm{DMS}}). \tag{2.11}$$

Zastosowanie modeli CSM wyższych rzędów zazwyczaj lepiej określa rzeczywistą informację zawartą w zbiorze danych, co pozwala zwiększyć potencjalną efektywność algorytmów kompresji. W zależności od charakteru kompresowanych danych właściwy dobór kontekstu może wtedy zmniejszyć graniczną długość kodu. Stosowanie rozbudowanych modeli CSM napotyka jednak na duże trudności w konkretnych implementacjach, wynikające przede wszystkim z faktu, iż ze wzrostem rzędu liczba współczynników opisujących model rośnie wykładniczo. Wiarygodne statystycznie ich określenie zaczyna być jedynie pobożnym życzeniem, a niewiarygodny, trudny do adaptacji rozbudowany model może dać gorszą efektywność kompresji od modeli prostszych.

Ograniczenie efektywności metod bezstratnego kodowania

Algorytmy kodowania wykorzystując opisane wyżej modele źródeł informacji tworzą wyjściowy strumień kodowy, który jest sekwencją bitową o skończonej długości, utworzoną z bitowych słów kodowych charakterystycznych dla danej metody kodowania. Algorytm kodowania (koder) jest realizacją kodu. **Kodem** nazwiemy regułę (funkcję) przyporządkowującą ciągowi symboli wejściowych (opisanych modelem źródła informacji o zdefiniowanym alfabecie) bitową sekwencję wyjściową (kodową). W tzw. kodach symboli o zmiennej długości (ang. variable-length symbol codes) sekwencja wyjściowa powstaje poprzez przypisanie kolejnym symbolom pojawiającym się na wejściu odpowiednich słów kodowych o różnej długości bitowej w schemacie jeden symbol - jedno słowo. W kodach strumieniowych pojedyncze słowo kodowe jest przypisywane całemu ciągowi symboli wejściowych. Może to być ciąg symboli o stałej lub zmiennej długości (np. w koderach słownikowych), a w skrajnym przypadku wszystkim symbolom strumienia wejściowego odpowiada jedno słowo kodowe, będące jednoznacznie dekodowalną reprezentacją oryginału (tak jest w koderach arytmetycznych).

W metodach kompresji, w przeciwieństwie do np. technik szyfrowania, istnieje warunek konieczny, aby kod bitowy (tj. bitowa sekwencja wyjściowa powstała w wyniku realizacji reguły kodu na strumieniu wejściowym) był jednoznacznie dekodowalny. Oznacza to, iż na podstawie wyjściowej sekwencji bitowej kodera *K* realizującego ustalony kod można jednoznacznie odtworzyć zbiór oryginalny symboli wejściowych, czyli jeśli dla ciągów symboli wejściowych s_i' i s_i'' o wartościach z alfabetu $A_S = \{a_1,...,a_n\}$ i długości k przyporządkowano bitową sekwencję kodową $c \in C$ (C - zbiór wszystkich sekwencji bitowych generowanych przez koder *K*) to:

$$K(s_1', s_2', ..., s_k') = K(s_1'', s_2'', ..., s_k'') = c \;\Rightarrow\; s_i' = s_i'' \quad \text{dla} \quad i = 1,2,...,k. \tag{2.12}$$

PRZYKŁAD 2.1. Kody jednoznacznie dekodowalne.
Koder K_1 przyporządkuje czterem symbolom alfabetu źródła informacji *S* następujące binarne słowa kodowe: $A_{b1} = \{0,01,001,0011\}$. Rozpatrzmy krótką sekwencję kodową (zwaną czasem kodem wyjściowym) utworzoną ze słów kodowych alfabetu A_{b1}: $c_1 = 001$, który może być dekodowany jako 0 i 01 (pierwszy i drugi symbol) lub też 001 (trzeci symbol). Kod realizowany przez koder K_1 nie jest więc kodem jednoznacznie dekodowalnym, a można to

stwierdzić na podstawie analizy alfabetu postaci słów kodowych danego kodu. Dla koderów wykorzystujących inne zbiory binarnych słów kodowych można w analogiczny sposób stwierdzić, że:

$A_{b2} = \{1,01,001,0001\}$ - kod jest jednoznacznie dekodowalny,

$A_{b3} = \{0,10,110,111\}$ - kod jest jednoznacznie dekodowalny,

$A_{b4} = \{0,10,11,111\}$ - kod nie jest jednoznacznie dekodowalny (wyjściowa sekwencja bitowa może nie być jednoznacznie zdekodowana),

$A_{b5} = \{00,01,10,11\}$ - kod jest jednoznacznie dekodowalny,

$A_{b6} = \{1,10,100,101\}$ - kod nie jest jednoznacznie dekodowalny.

Przy konstruowaniu nowych technik odwracalnej kompresji bardzo interesującym jawi się pytanie o granicę możliwej do uzyskania efektywności kompresji. Intuicyjnie wiadomo, że nie można stworzyć nowej reprezentacji danych o dowolnie małej długości. Trzeba przecież jakoś opisać zawartą w zbiorze informację. Okazuje się, że entropia łączna $H(S)$ wyznaczona według równania (2.4) dla kodowanego zbioru danych stanowi graniczną wartość średniej długości kodu. Mówią o tym twierdzenia Shannona o kodowaniu źródeł, w tym szczególnie istotne twierdzenie o kodowaniu źródła bezszumnego [1]. Dotyczy ono granicznej efektywności kodowania osiąganej poprzez kodowanie dużych bloków symboli alfabetu źródła (N-tego rozszerzenia źródła) za pomocą binarnych słów kodowych. Zamiast przypisywania słów kodowych pojedynczym symbolom alfabetu jak w koderach symboli koncepcja kodera przechodzi w kierunku realizacji kodu strumieniowego, wykorzystując przy tym w możliwie najlepszy sposób zależności pomiędzy poszczególnymi symbolami.

Twierdzenie o kodowaniu źródła bezszumnego

Niech S będzie ergodycznym źródłem z alfabetem o n elementach i entropii $H(S)$. Rozważmy kodowanie bloków po N ($N \le n$) symboli alfabetu źródła jednocześnie za pomocą binarnych słów kodowych dających kod jednoznacznie dekodowalny. Mamy więc do czynienia ze zmianą modelu źródła, tzw. rozszerzeniem źródła. Zmienia się struktura informacji pojedynczej, czyli inaczej elementarny obiekt przenoszący (reprezentujący) informację. Wówczas dla dowolnego $\delta>0$ możliwe jest, poprzez dobór odpowiednio dużej wartości N, skonstruowanie kodu w taki sposób, że średnia liczba bitów przypadająca na symbol tego źródła $\overline{L}_S$ spełnia następujące równanie:

$$H(S) \le \overline{L}_S < H(S) + \delta \,. \tag{2.13}$$

Okazuje się, że zawarta w tym twierdzeniu sugestia o zwiększaniu efektywności kodowania poprzez konstruowanie coraz większych rozszerzeń źródła (rosnące N) jest cenną wskazówką pokazującą kierunek optymalizacji odwracalnych koderów, jednak w dosłownej realizacji jest niepraktyczna, głównie ze względu na problem skutecznego określenia prawdopodobieństw łącznego wystąpienia N symboli źródła na podstawie strumienia danych do kompresji.

Z twierdzenia o kodowaniu źródła bezszumnego jasno wynika, że każde źródło danych może być bezstratnie kodowane przy użyciu kodu jednoznacznie dekodowalnego, którego średnia liczba bitów na symbol źródła jest dowolnie bliska, ale nie mniejsza niż entropia źródła (określona na podstawie prawdopodobieństw występowania poszczególnych symboli i grup symboli źródła) wyrażona w bitach. Jest to naturalne ograniczenie wszystkich metod bezstratnej kompresji. Oczywiście aplikowany model źródła danych winien jak najlepiej charakteryzować (przybliżać) zbiór danych rzeczywistych. Należy więc podkreślić względność tego twierdzenia wobec przyjętego modelu, podobnie jak całości prezentowanych

rozważań, tj. szacowania ilości informacji i związku z budowanym na tej podstawie koderem. Innymi słowy kluczowym staje się wybór struktury obiektu elementarnego, a następnie sposób skutecznej estymacji opartego na nim modelu statystycznego.

Ponieważ najczęściej nie sposób wiarygodnie określić wartości entropii łącznej dla zbiorów rzeczywistych, uproszczone modele źródeł informacji, które opisują kodowane zbiory i są wykorzystywane w konstrukcji algorytmów kompresji, pozwalają aproksymować średnią ilość informacji w bitach na symbol (entropia warunkowa lub bezwarunkowa), która stanowi nieprzekraczalną granicę średniej długości kodu dla kodera budowanego w oparciu o koncepcję danego źródła.

Próbując osiągnąć coraz większą skuteczność kompresji metod bezstratnych można działać w dwu zasadniczych kierunkach, które ogólnie można nazwać doskonaleniem modelu źródła informacji przybliżającego coraz wierniej kompresowany zbiór danych, nawet kosztem rosnącej złożoności modelu. Najpierw potrzeba coraz dokładniej modelować wpływ kontekstu, zarówno za pomocą rozbudowanych, dynamicznych (adaptacyjnych) modeli predykcyjnych, jak też coraz szybciej i pełniej określanych probabilistycznych modeli Markowa nawet wyższych rzędów. Działania te muszą być skojarzone z nowymi rozwiązaniami kodów binarnych tworzonych na podstawie tych modeli, pozwalających osiągnąć minimalną długość bitowej sekwencji nowej reprezentacji danych.

Konkluzje na temat nadmiarowości

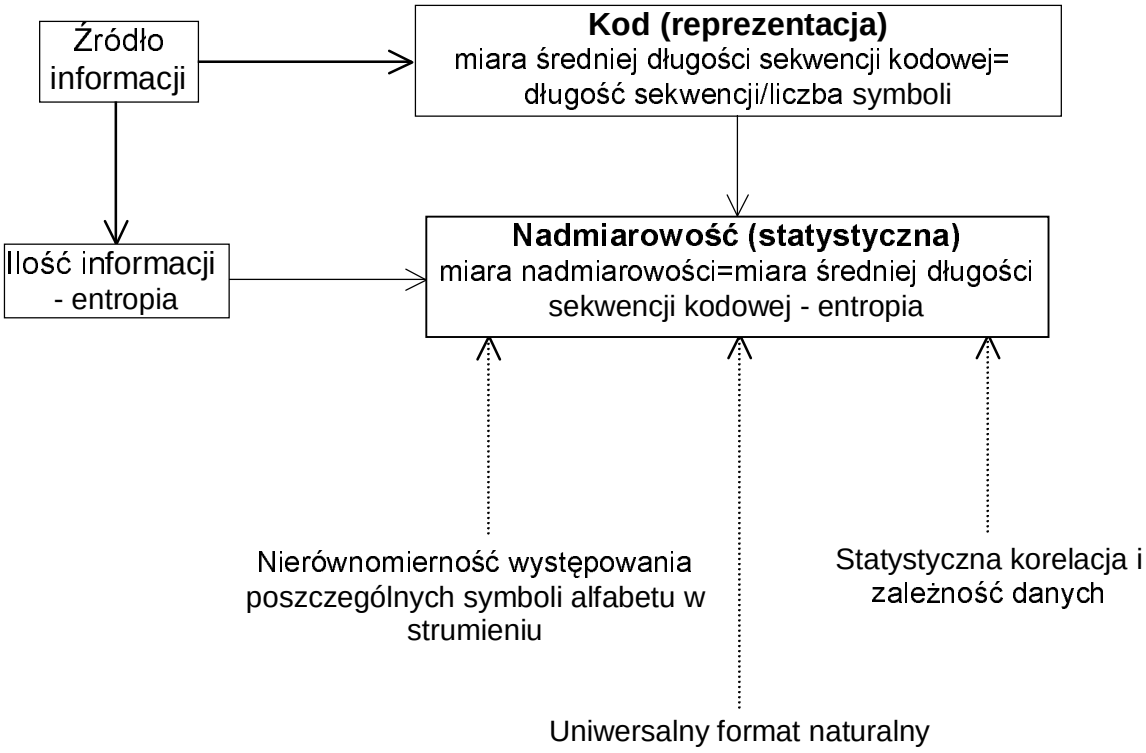
Uzyskanie kompresji konkretnego zbioru danych możliwe jest dzięki występującej w oryginalnej reprezentacji zbioru danych różnego rodzaju nadmiarowości. Proces kompresji sprowadza się więc zasadniczo do skutecznej redukcji czy też w najlepszym przypadku całkowitej eliminacji tej nadmiarowości.

W praktyce informacja jest podawana ze źródeł pierwotnych w postaci, która na ogół nie nadaje się ani do dalszego przetwarzania, ani do przesyłania. Dlatego z zasady dokonywane jest przekształcenie podanej przez źródło informacji w ciąg sygnałów (symboli) elementarnych o formie dogodnej do zapamiętania i przekształcania. Aby tego dokonać, potrzebna jest pewna reguła przyporządkowania poszczególnym postaciom informacji symboli elementarnych z pewnego alfabetu. Za pomocą t ej reguły tworzony jest ciąg symboli przyporządkowany danej informacji. Powstaje oryginalna reprezentacja zbioru danych.

Najprostsze przykłady takich reguł i reprezentacji to: przyporządkowanie naturalnym pojęciom opisującym świat ciągów liter, słów, układających się w sensowne zdania, a także opisanie informacji o charakterze ziarnistym za pomocą ciągów liczbowych, np. w systemie dwójkowym.

Najczęściej w systemach gromadzenia danych stosowany jest uniwersalny format danych, uwzględniający różnorodny charakter rejestrowanych zjawisk - możliwą dynamikę danych do zapisu, kolejność pojawiania się tych danych, możliwość wygodnego odczytu itd., co powoduje często znaczną redundancję reprezentacji w odniesieniu do konkretnego, pojedynczego zbioru danych. Ponadto, charakter informacji wyrażonej w pewnej reprezentacji w konkretnym zbiorze danych powoduje, że istnieją zazwyczaj pewne zależności pomiędzy danymi (najczęściej kolejnymi, ale nie tylko) w zbiorze. Wynikają one np. z reguł języka polskiego, w którym znacz nie częściej statystycznie rzecz biorąc po literce 't' występuje literka 'a' niż 'z' itp. Z kolei w obrazach prezentujących pewne struktury o rozmiarach większych niż piksel wartości sąsiednich pikseli są często ze sobą skorelowane. Zależność ta może być rozpatrywana zarówno w zakresie statystycznego rozkładu wartości danych w zbiorze (prosty histogram) - rozkład ten jest często nierównomierny w swojej

wersji globalnej, a już na pewno lokalnie, jak też poprzez określenie wpływu wartości występujących w pewnym sąsiedztwie (kontekst) kodowanej aktualnie wartości na tę wartość (histogram wielowymiarowy, warunkowy). Sposób określania liczbowej miary nadmiarowości w sensie statystycznym oraz zasadnicze, bezpośrednie jej przyczyny przedstawia rys.2.2.



Rys. 2.2. Ilościowe szacowanie statystycznej nadmiarowości oryginalnej reprezentacji kompresowanych danych oraz główne źródła tej nadmiarowości.

Dokładniejsza analiza nadmiarowości silnie zależy od rodzaju zbioru danych i charakteru zawartej tam informacji. Przykładowo, dla zbiorów danych obrazowych można wyróżnić następujące typy nadmiarowości:

- nadmiarowość przestrzenną (jednoobrazową i międzyobrazową), związaną z występowaniem zależności pomiędzy wartościami sąsiednich pikseli, zarówno w obrębie jednego obrazu, jak też serii obrazów kolejnych (zbiory danych trójwymiarowych);
- nadmiarowość czasową, pojawiającą się w związku z korelacją czasową pomiędzy kolejnymi obrazami sekwencji czasowej;
- nadmiarowość spektralną, związaną z występowaniem korelacji pomiędzy chromatycznymi składowymi w obrazach kolorowych,

a także występującą na wyższym, semantycznym poziomie

- nadmiarowość wynikającą z faktu, iż nie cała informacja zawarta w obrazie jest istotna z punktu widzenia celu ich wykorzystywania (nadmiarowość semantyczna).

Przykładem nadmiarowości semantycznej może być treść prezentowana w obrazach medycznych. Zdarza się, że znaczna ilość informacji zawarta w tych obrazach nie jest istotna diagnostycznie, a więc usunięcie jej z obrazu nie zmniejsza jego wartości diagnostycznej. W

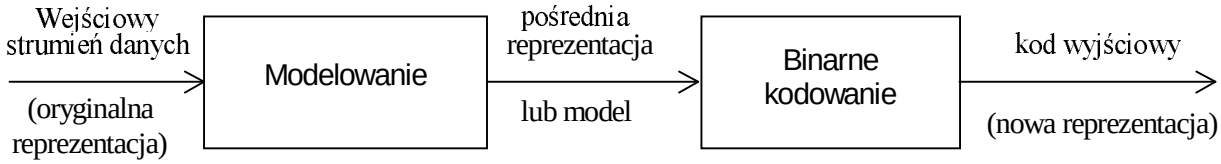
niektórych rodzajach badań (np. scyntygraficznych) w dużych obszarach obrazu występuje jedynie szum wynikający z metody pomiarowej, bądź nie związane z żadną informacją diagnostyczną artefakty, mogące nawet utrudniać dalszą analizę obrazów w systemach medycznych. Ich eliminacja może więc nawet poprawić jakość obrazu, choć przy analizie statystycznej poziom korelacji i wyznaczone prawdopodobieństwa mogą wskazywać na wcale niemałą informację zawarta w tych obszarach. Semantyczne ok reślanie informacji użytecznej (a co za tym idzie także nadmiarowości) ma znaczenie nadrzędne w interpretacji diagnostycznej.

2.2. Bezstratne metody kompresji

Metodyka konstrukcji efektywnych metod kompresji jest skierowana głównie w kierunku możliwie dokładnego przybliżenia ogólnego i lokalnego charakteru danych za pomocą różnych modeli lub miar, czy - mówiąc inaczej - wydobycia rzeczywistej (w myśl definicji) informacji związanej z pojawieniem się każdej kolejnej danej w strumieniu wejściowym oraz efektywnego jej opisania. Koniecznym jest wyrugowanie wszelkich nadmiarowości. Często techniki bezstratnej kompresji danych nazywane są kodowaniem.

Pierwsza faza procesu bezstratnej kompresji danych obrazowych polega na usuwaniu nadmiarowości poprzez: uproszczenie formatu zapisu danych, wprowadzenie dobrych modeli statystycznych i predykcyjnych, zbudowanie słownika z najczęściej występującymi frazami (ciągami symboli), wykorzystanie wiedzy dostępnej a priori na temat kompresowanego zbioru danych itd. Na podstawie uzyskanego modelu tworzy się następnie binarną sekwencję kodową poprzez przypisanie słów kodowych pojedynczym symbolom alfabetu źródła, całemu strumieniowi danych lub też jego poszczególnym częściom.

Ogólny schemat bezstratnych metod kompresji przedstawiono na rys. 2.3. W niektórych rozwiązaniach schemat może być bardziej złożony. Po wstępnej dekompozycji może nastąpić budowanie modelu z pamięcią dla pośredniej reprezentacji danych lub też modelowanie może być przeplatane z kodowaniem. Zasadniczo w modelowaniu istotne są kontekstowe zależności danych sąsiednich w strumieniu wejściowym lub też w przestrzeni opisywanej przez zbiór danych (czy też skojarzonej z tym zbiorem), np. pewne metryczne zależności w przestrzeni koloru i w przestrzeni geometrycznej w obrazach.



Rys. 2.3. Blokowy, ogólny schemat bezstratnych metod kompresji.

Dla obrazów stosuje się szereg metod eliminacji nadmiarowości przestrzennej poprzez dekompozycję obrazu do postaci, która jest bardziej podatna na entropijne i słownikowe metody kodowania. Są wśród nich metody dzielące wartości pikseli obrazu na szereg map bitowych (ang. bit plane encoding), które szczególnie dla najstarszych bitów są bardzo silnie skorelowane, a więc podatne na silną redukcję rozmiaru ich reprezentacji. Inną grupę stanowią algorytmy predykcyjne: wartość, która ma się pojawić jako następna w strumieniu

danych wejściowych jest przewidywana na podstawie najbardziej z nią skorelowanych danych (najczęściej najbliższych w przestrzeni obrazu), które pojawiły się już wcześniej w kodowanym strumieniu (warunek przyczynowości, konieczny do rekonstrukcji obrazu oryginalnego podczas dekodowania). Schemat kodowania predykcyjnego prowadzący do praktycznych implementacji nazywany jest DPCM (ang. differential pulse code modulation). Pierwociny systemu DPCM zostały opracowane wiele lat temu w Bell Laboratories [2]. Stosowany w predykcji kontekst nie otacza danego piksela z każdej strony ze względu na konieczność zachowania warunku przyczynowości. Alternatywnym rozwiązaniem są znacznie nowsze techniki interpolacyjne HINT (ang. hierarchical interpolation) [3,4,5], w których kodowane są kolejne wersje obrazu o coraz większej rozdzielczości, przez co do predykcji mogą być wykorzystane wartości pikseli otaczające dany piksel z każdej strony, najczęściej jednak nie należące do najbliższego sąsiedztwa. Otrzymane w wyniku predykcji obrazy różnicowe są zazwyczaj kodowane przy wykorzystaniu algorytmów Huffmana lub kodowania arytmetycznego (w wersji statycznej).

Podsumowując, zbiór rozwiązań fazy modelowania można podzielić na cztery zasadnicze grupy:

- z prostym modelem statycznym (np. w metodach: RLE, Huffmana, Shannona-Fano, kodowaniu arytmetycznym 'bez pamięci'),
- z rozbudowanym modelem statystycznym wyższych rzędów (np. w koderach arytmetycznych 'z pamięcią', wykorzystujących modele Markowa rzędu *m*),
- ze słownikiem (metody słownikowe),
- ze wstępną dekompozycją danych - tworzona jest pośrednia reprezentacja (metody predykcyjne i interpolacyjne, kodowanie map bitowych, łączone kodowanie transformacyjne, przekształcenie Peano, stratne/bezstratne kodowanie resztkowe, odwracalne filtracje pasmowe).

Faza binarnego kodowania jest najczęściej realizowana w trzech wariantach:

- przypisanie słów kodowych poszczególnym pojedynczym symbolom alfabetu źródła (Huffmana, Shannona-Fano) - stałej liczbie symboli wejściowych odpowiada kod o zmiennej długości,
- przypisanie określonej reprezentacji (najczęściej o stałej długości) fragmentom strumienia wejściowego o zmiennej długości (RLE, kodowanie słownikowe) - zmiennej liczbie symboli wejściowych przypisane są słowa kodowe o stałej długości; modyfikacją tych metod są adaptacyjne kodery słownikowe o zmiennym rozmiarze słownika (a więc także indeksów), jak również adaptacyjne RLE realizowane w postaci kodów Golomba (ang. Golomb codes): ciągom symboli wejściowych o zmiennej długości przyporządkowane są sekwencje (słowa) o różnej liczbie bitów.
- kod wyjściowy jako jedno binarne słowo kodowe tworzone sukcesywnie dla całego strumienia wejściowego (kodowanie arytmetyczne).

Adaptacyjność algorytmów kodowania

Zasadniczo, niemal każdy ze składników schematu kompresji może być zaimplementowany w wersji statycznej lub adaptacyjnej. Algorytm kompresji jest adaptacyjny, jeśli struktura składników schematu bądź ich parametrów zmienia się lokalnie w ramach wejściowego strumienia danych adaptując do lokalnej charakterystyki danych. Adaptacyjność pozwala na zwiększenie efektywności kosztem wzrostu stopnia złożoności algorytmu kodującego.

Ze względu na poziom wykorzystania zasad adaptacyjności algorytmy kodowania realizowane są zazwyczaj w jednej z trzech podstawowych postaci (dotyczy dowolnego elementu schematu kompresji odwracalnej):

A. Statycznej:

- element jednakowy dla całej grupy zbiorów danych o zbliżonym charakterze, np. grupa tekstowych raportów pracy pewnego biura, czy też grupa obrazów z telekonferencji przedstawicieli firmy Y,
- brak konieczności dopisywania parametrów elementu do kodu wyjściowego;

B. Póładaptacyjnej:

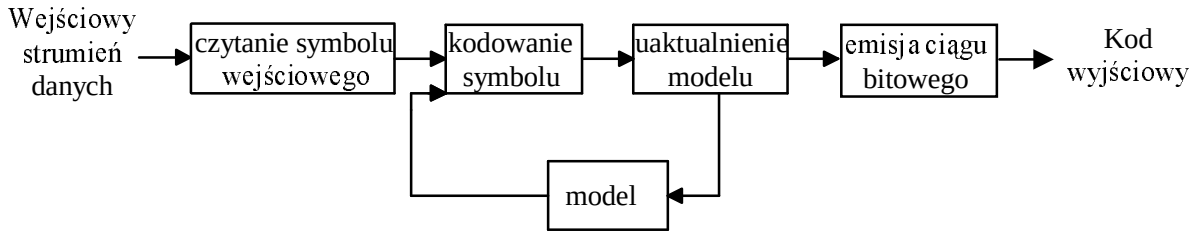
- element jednakowy w obrębie całego zbioru danych, wyznaczony na podstawie wiedzy *a priori*, wstępnej jego analizy, itp.;
- konieczność dopisania parametrów elementu do reprezentacji wyjściowej kodera;

C. Adaptacyjnej:

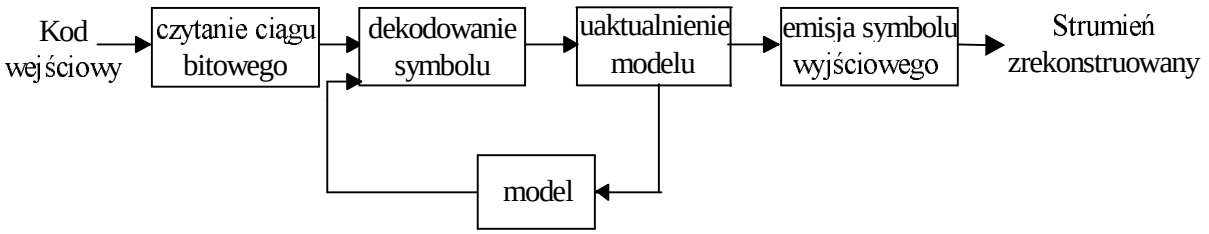
- element zmienny, dynamicznie dostosowany do lokalnej statystyki konkretnego zbioru danych na podstawie bieżącej analizy strumienia danych kodowanych,
- brak konieczności dopisywania parametrów elementu do kodu wyjściowego (w przypadku przyczynowej adaptacji wstecz) lub też konieczność dopisywania parametrów elementu (dla adaptacji wprzód).

Poprzez dokładniejsze dostosowanie do lokalnej charakterystyki danych uzyskuje się wierniejszy model, co z kolei pozwala lepiej opisać rzeczywistą informację zawartą w zbiorze danych. Skuteczna technika binarnego kodowania oparta na tym modelu pozwala uzyskać mniejszą długość kodu, a więc zwiększyć efektywność kompresji. Odbywa się to zazwyczaj kosztem znacznego rozbudowania algorytmu, zwiększenia ilości koniecznych obliczeń (czasochłonność, trudności realizacji w czasie rzeczywistym), a także zwiększenia wymagań na sprzęt realizujący algorytm (konieczne większe zasoby pamięci operacyjnej).

W modelowaniu adaptacyjnym można wykorzystać wstępne przeglądanie danych lub też informację dostępną *a priori* do ustalenia początkowego stanu modelu przybliżającego charakter danych. Model ten jest stale modyfikowany w miarę analizy kolejnych danych ze strumienia wejściowego. Ogólny schemat techniki kompresji wykorzystującej model adaptacyjny przedstawiony jest na rys. 2.4 i 2.5.



Rys.2.4. Ogólny schemat kompresji adaptacyjnej.



Rys.2.5. Ogólny schemat adaptacyjnej dekompresji.

Klasyczne metody kompresji odwracalnej zostały przedstawione w kolejnych rozdziałach. O niektórych innych, użytecznych koncepcjach kodowania wspomniano poniżej.

Przykłady innych koncepcji koderów odwracalnych

Bardzo prosta koncepcja kodowania długości sekwencji może być także realizowana w wersji adaptacyjnej, zorientowanej dodatkowo na reprezentację binarną strumienia danych. Przyjmuje ona postać **kodów Golomba** w wersji pierwotnej [6] oraz wielu rozszerzeniach, m.in. [7,8].

Dla parametru m będącego dodatnią liczbą całkowitą kod Golomba rzędu m - G_m koduje liczbę całkowitą $y \geq 0$ w dwóch częściach: unarnej reprezentacji $u = \lfloor y/m \rfloor$ jako przedrostka (prefiksu) oraz zmodyfikowanej binarnej reprezentacji $r = y \bmod m$ jako sufiksu (zapisanej na $\lfloor \log_2 m \rfloor$ bitach, jeśli $r < 2^{\lceil \log_2 m \rceil} - m$ lub $\lceil \log_2 m \rceil$ bitach w pozostałych przypadkach). Zasadnicza koncepcja dotyczy więc kodowania nieujemnych liczb całkowitych przy założonym rzędzie kodu. Prześledźmy przykład wyznaczania słowa kodowego dla liczby y=13 przy rzędzie m=3. Unarny przedrostek będzie wyglądał następująco: ponieważ $\lfloor t/m \rfloor = \lfloor 13/3 \rfloor = 4$, to unarna reprezentacja (patrz tabela 2.1) jest postaci **11110.** Dalsze bity słowa to przedstawiony w konwencji binarnej wynik operacji: $y \bmod m = 13 \bmod 3 = 1$ na $\lceil \log_2 m \rceil = 2$ bitach (gdyż nie jest spełniony warunek $1 < 2^{\lceil \log_2 3 \rceil} - 3$), czyli sekwencja **10.** Daje to ostatecznie słowo kodowe **1111010** przypisane liczbie 13 według G_3. Więcej przykładów tworzenia kodów Golomba zamieszczono w tabeli 2.1.

Szczególny przypadek kodów Golomba, pozwalający na bardzo prostą i szybką realizację procedur kodowania/dekodowania, uzyskuje się przy $m = 2^k$. Oznaczmy go przez G_{2k}. Kodowanie nieujemnej liczby całkowitej y sprowadza się w tym przypadku do podziału binarnej reprezentacji liczby y na dwie części, mniej znaczących bitów o długości k oraz grupę pozostałych, bardziej znaczących bitów. Następnie, słowo kodowe powstaje poprzez sklejenie unarnej reprezentacji wartości powstałej z grupy bardziej znaczących bitów bezpośrednio z binarną reprezentacją k najmłodszych bitów. Przykładowo, dla y=6 kod G_{2k} dla parametru k=2 powstaje w sposób następujący: dzielimy $y = 6_{10} = 110_2$ w postaci binarnej na dwie części (z k=2 bitową grupą bitów młodszych) **1·10**, starszą część zapisujemy w postaci unarnej **10** i dołączamy do niej młodsze bity **10**, co daje ostateczną postać słowa kodowego **10·10** według kodu Golomba.

Tabela 2.1. Przykładowe kody Golomba różnych rzędów dla kolejnych dodatnich liczb całkowitych.

Nieujemna liczba całkowita y	Kody Golomba			
	m=1 (tylko kod unarny, nie ma części binarnej)	m=2	m=3	m=4
0	0	0·0	0·0	0·00
1	10	0·1	0·10	0·01
2	110	10·0	0·11	0·10
3	1110	10·1	10·0	0·11
4	11110	110·0	10·10	10·00
5	111110	110·1	10·11	10·01
6	1111110	1110·0	110·0	10·10
7	11111110	1110·1	110·10	10·11
8	111111110	11110·0	110·11	110·00
9	1111111110	11110·1	1110·0	110·01
...				

Innym przykładem zmodyfikowanego kodu Golomba jest elementarny kod Golomba. Niech dla parametru m (rząd kodu) będącego dodatnią liczbą całkowitą EG_m oznacza kod o ‘zmienno-zmiennej’ (ang. variable-to-variable) długości, zdefiniowany na rozszerzonym binarnym alfabecie postaci $\{1,01,...,0^l,...,0^{m-1}1,0^m\}$, gdzie 0^l oznacza ciąg zer o długości l. Opisuje on sekwencje bitowe pojawiające się na wejściu kodera. Kod EG_m przypisuje rozszerzonemu symbolowi 0^m wartość 0, podczas gdy symbole 0^l1, $0 \le l < m$ są kodowane za pomocą jedynki, po której występuje wartość l w zmodyfikowanej binarnej reprezentacji (opisanej wyżej, jak w G_m).

Ogólnie, kody Golomba są optymalne dla źródeł informacji opisanych rozkładami gęstości prawdopodobieństwa opadającymi wykładniczo (rozkłady geometryczne) dla nieujemnych wartości całkowitych n, tj. rozkładami postaci $Q(n) = (1-\rho)\rho^n$, gdzie $0 < \rho < 1$. Jednoparametryczne kody Golomba dają wtedy najkrótszą możliwą średnią długość kodu, jeżeli rząd m jest równy: $m = \left\lfloor \log(1+\rho) / \log(\rho^{-1}) \right\rfloor$. Rząd ten można dobierać adaptacyjnie [9].

Kody Golomba znalazły zastosowanie w metodzie odwracalnej kompresji obrazów według standardu JPEG-LS [7], głównie ze względu na możliwość uzyskania znaczącej skuteczności binarnego kodowania bez konieczności konstrukcji złożonych modeli statystycznych, często niedookreślonych, o bardzo dużych kosztach obliczeniowych. Przyjmując model reszt predykcyjnych jako dwustronny rozkład geometryczny (ang. two-sided geometric distribution - TSGD) stwierdzono dużą efektywność adaptacyjnie dobieranych (rozszerzono nieco alfabet przy kodowaniu płaskich obszarów obrazów) kodów Golomba [7], przy bardzo małych kosztach obliczeniowych.

Transformacja Burrows-Wheeler'a (ang. *Burrows-Wheeler Transform* - BWT) [10] jest klasyczną metodą dekompozycji danych do postaci pośredniej, która może być skuteczniej zakodowana za pomocą statystycznych koderów binarnych (arytmetycznego, Huffmana). BWT przekształca blok danych do formatu, który jest podatny na kodowanie, głownie ze względu na formowanie długich ciągów powtarzających się symboli postaci pośredniej. Symbole te są takie same jak w reprezentacji wejściowej, są jednak ustawione w innej kolejności. Uzyskuje się to poprzez trzy kolejne etapy dekompozycji. Po pierwsze następuje generacja z wejściowego bloku danych o n elementach n-1 dodatkowych bloków danych o tej samej liczności i poprzestawianych symbolach według zasady: drugi blok zawiera przesunięte o jedną pozycję w lewo elementy bloku pierwszego (wejściowego), przy czym początkowy symbol z pierwszego przechodzi na ostatnią pozycję bloku drugiego. Tak utworzony drugi blok nazwijmy blokiem startowym. Trzeci blok budowany jest na podstawie drugiego w analogiczny sposób, itd. W kolejnym etapie bloki są sortowane, np. według kodów ASCII kolejnych elementów bloku, gdy mamy do czynienia z tekstami, lub też według kolejności cyfr na poszczególnych pozycjach w przypadku bloków danych liczbowych, np. pomiarowych. Ciąg symboli znajdujących się na ostatniej pozycji w tak posortowanych blokach wraz ze wskaźnikiem (indeksem) pozycji bloku startowego stanowi wyjściową postać transformaty BW - reprezentację pośrednią algorytmu kompresji, bardziej podatną na kodowanie. Zgodnie z powyższym schematem dekompozycji dla prostego bloku danych pliku tekstowego postaci: b_0 =„a,d,a,m” tworzone są dodatkowo jeszcze trzy bloki: blok startowy b_1 =„d,a,m,a”, oraz b_2 =„a,m,a,d” i b_3 =„m,a,d,a”. Posortowane bloki zostały przedstawione w tabeli 2.2 dając wyjściową postać BWT: o_{BWT} =„m,d,a,a” oraz indeks $s\_idx = 2$.

Tabela 2.2. Posortowane w algorytmie BWT bloki danych, utworzone z wejściowego bloku danych postaci b_0 =„a,d,a,m”.

Początkowe oznaczenia bloków	Bloki	Symbole na pierwszej pozycji w bloku (v_P)	Symbole na ostatniej pozycji w bloku (v_O)	Indeksy posortowanych bloków (pogrubiony dla bloku startowego)
b_0	„a,d,a,m”	‘a’	‘m’	0
b_2	„a,m,a,d”	’a’	’d’	1
b_1	„d,a,m,a”	’d’	’a’	**2**
b_3	„m,a,d,a”.	’m’	’a’	3

Odwracalność transformacji BW oraz realne korzyści związane z jej stosowaniem choć początkowo trudno dostrzegalne, okazują się jednak możliwe (zobacz [11][12][14]). Na wejściu procedury odwrotnej BWT pojawia się ciąg symboli o_{BWT}. Ponieważ znany jest wektor v_O symboli na ostatnich pozycjach posortowanych bloków, a także wskaźnik bloku startowego, od razu można odtworzyć pierwszy symbol b_0 (tj. rekonstruowanego bloku danych) jako ‘a’ (wiadomo, że ostatni symbol bloku startowego jest pierwszym symbolem b_0 zgodnie z BWT), a także wektor v_P symboli z początkowej pozycji bloków z tabeli 2.2 (ponieważ są to posortowane symbole odczytanego wektora v_O). Wiedza ta jest wystarczająca do wyznaczenia tzw. wektora transformacji, który umożliwia pełną rekonstrukcję b_0. Wektor transformacji v_T składa się z elementów określających porządek ustawienia posortowanych bloków, gdzie $v_T[i]$ oznacza indeks bloku następnego utworzonego według zasady BWT (tj. przez przesunięcie o 1 w lewo elementów danego

bloku), czyli jeśli $i = index(b_j)$ to $v_T[i] = index(b_{j+1})$. Ponieważ znane są początkowe symbole bloków b_j, które przechodzą na ostatnią pozycję w b_{j+1} (symbole z ostatniej pozycji zostały odczytane), można bez przeszkód wyznaczyć v_T. Dla rozważanego przykładu wskaźnik bloku startowego równy 2 oznacza, że ‘a’ musiało przejść z pozycji początkowej bloku o indeksie 0 lub 1 (tam jest ‘a’). Ponieważ te same reguły sortowania obowiązują na wszystkich kolejnych pozycjach bloków, przy wystąpieniu takiego samego symbolu na ostatniej pozycji kilku bloków wcześniejsze wystąpienie tego symbolu na pozycji pierwszej przechodzi na pierwsze wystąpienie tegoż symbolu na pozycji ostatniej, to samo z drugim wystąpieniem itd. W przykładzie odczytane ‘a’ (z ostatniej pozycji) występuje w blokach o indeksie 3 i 4, oznacza to, że przeszły one z bloków o początku ‘a’ o indeksach odpowiednio 0 i 1. Tak więc $v_T[0] = 2$, $v_T[1] = 3$ i analogicznie $v_T[2] = 1$, a $v_T[3] = 0$. Wektor transformacji pokazano w tab. 2.3.

Tabela. 2.3. Ustalenie wektora transformacji BWT dla b_0=„a,d,a,m”. Strzałką zaznaczono przejścia symboli z początku bloku b_j na koniec bloku b_{j+1}.

Indeks (kolejne i dla v_T)	Pierwszy symbol bloku	Przejścia	Ostatni symbol bloku	Indeks ($v_T[i]$)
0	‘a’		‘m’	2
1	‘a’		‘d’	3
2	‘d’		‘a’	1
3	‘m’		‘a’	0

Procedura wyznaczenia wektora transformacji wygląda następująco:

```
int n = 4;
int VT[] = {-1, -1, -1, -1};
char VO[] = "mdaa";

void wyznaczenie_wektora_transformacji()
{
      for (int i = 0; i < n; i++) {
            symbol = VO[i];
            pozycja = szukaj_symbol_w_VP(symbol);
            VT[pozycja] = i;
        }
}
```

Rekonstrukcja b_0 za pomocą v_T jest już prosta: rozpoczynając od elementu wektora v_O wskazywanego przez indeks bloku startowego $idx = s\_idx$, czyli $v_O[idx] = v_O[s\_idx] =$ ’a’, ustalamy następnie kolejny indeks według v_T, tak że $idx = v_T[idx]$ i odczytujemy ponownie $v_O[idx]$ (jako drugi symbol b_0), znów uaktualniamy idx według v_T, odczytujemy $v_O[idx]$ itd. Pozwala to jednoznacznie zdekodować słowo b_0=„a,d,a,m”. Przykładowa procedura rekonstrukcji b_0 za pomocą wektora transformacji wygląda więc następująco:

```
int n = 4;
int VT[] = {2, 3, 1, 0};
char VO[] = "mdaa";
```

```
int s_idx = 2;

void rekostrukcja_bloku()
{
   int idx = s_idx;
      for (int i = 0; i < n ; i++) {
         cout << VO[idx];
         idx = VT[idx];
      }
}
```

Sortowanie bloków wpływa na kumulację powtórzeń symboli alfabetu bloku w reprezentacji wyjściowej BWT (widać to chociażby w naszym przykładzie, gdzie dwa symbole ‘a’ znalazły się obok siebie). *n*-1 symboli danego bloku poprzedzających ostatni jego element stanowi kontekst tego elementu ustawiony obok najbardziej podobnych kontekstów bloków sąsiednich. Jest więc duża szansa, że i ten ostatni symbol sąsiednich bloków będzie podobny. Dłuższe ciągi powtarzających się symboli są łatwiejsze do zakodowania – rośnie więc efektywność kompresji wejściowego bloku danych. Często jeszcze przed kodowaniem arytmetycznym czy Huffmana stosuje się dodatkowo przekształcenie MTF (ang. move to front), pozwalające nieco zwiększyć ich skuteczność. Polega ono na monitorowaniu listy wszystkich możliwych symboli alfabetu. Dla każdego kolejnego symbolu reprezentacji pośredniej przepisuje na wyjście jego pozycję na liście, po czym przysuwa go na początek listy. Jeśli więc przykładowo aktualne miejsca na liście symboli 'g' i 'z' są odpowiednio 25 i 86, to ciągowi pośredniej postaci po BWT równej "g,g,g,z,z,g,g" odpowiada po przekształceniu MTF ciąg symboli "25,0,0,86,0,1,0" (indeks początku listy jest równy 0).

Obok wielu interesujących cech BWT szczególną uwagę należy zwrócić na fakt, iż wszystkie elementy danego bloku wpływają na kolejność symboli na wyjściu, czyli kontekst jest równy rozmiarowi bloku. Ponadto, w skład kontekstu wchodzą zarówno dane pojawiające się wcześniej, jak i później w strumieniu wyjściowym. Nie ma więc jednostronnego ograniczenia kontekstu ze względu na warunek przyczynowości.

Ogólnie daje się zauważyć tendencję, że większe rozmiary bloków dają zazwyczaj lepszą (efektywniejszą) kompresję, tj. generowane są wówczas dłuższe ciągi identycznych symboli (szczególnie dla jednorodnych zbiorów danych). Zwiększenie rozmiarów bloku w algorytmie BWT napotyka jednak na pewne problemy realizacyjne. Cały blok trzeba przechowywać w pamięci operacyjnej, a tworzenie kolejnych bloków z bloku wejściowego jest *de facto* ustawianiem wskaźników tychże bloków na strukturze bloku wejściowego. Rozmiar bloku musi być więc dostosowany do sprzętu, na jakim zrealizowany jest koder. Ponadto czas sortowania wydłuża się przy większych blokach (zazwyczaj bowiem liczba obliczeń przy sortowaniu jest proporcjonalna do $O(N \log N)$).

Metoda BWT nie jest objęta prawami patentowymi, może być więc powszechnie stosowana, co zwiększa jej użyteczność. Najlepiej przy tym nadaje się do kompresji plików tekstowych i może uzyskiwać efektywność kompresji porównywalną z metodami o prostych modelach statystycznych. Jest także przedmiotem wielu prac badawczych nad efektywnymi metodami kompresji, m.in. [12,13,14].

Często kodery odwracalne wykorzystują strukturę maszyny (automatu) o ograniczonej liczbie stanów **FSM** (ang. Finite State Machine) do modelowania kontekstu za pomocą liniowej czy nieliniowej predykcji [15], jak również do tworzenia binarnej sekwencji kodowej [16]. FSM w swojej koncepcji nawiązuje silnie do teorii źródła znajdującego się w pewnym

stanie, które następnie emituje symbol czy grupę symboli zgodnie z aktualnym modelem prawdopodobieństw przechodząc do kolejnego stanu. Alfabet często jest binarny, model statystyczny opiera się na strukturze możliwych stanów opisujących źródło i prawdopodobnych przejść pomiędzy nimi, natomiast przebywanie i oscylowanie źródła wokół stanów najbardziej prawdopodobnych powoduje wysyłanie na wyjście najkrótszej informacji bitowej stanowiącej zapis aktywności źródła emitującego określony zbiór danych. Za pomocą FSM można zrealizować np. pomysł kodowania Huffmana, jak również inne koncepcje opisujące pracę źródła informacji, w tym nawet kodowanie arytmetyczne [16].

Bibliografia

1. C. E. Shannon, *A Mathematical theory of communications*, Board of Trustees of the University of Illinois, 1949.
2. C. C. Cutler, *Differential Quantization for Television Signals*, U.S. Patent 2,605,361, 29 lipca 1952.
3. T. Endoh, and T. Yamazaki, *Progressive Coding Scheme for multilevel images*, Proc. Picture Coding Symposium, 21-22, 1986.
4. P. Roos, M. A. Viergever, M. C. A. van Dijke, and J. H. Peters, *Reversible Intraframe Compression of Medical Images*, IEEE Trans. Medical Imaging, 7(4):328-336, 1988.
5. P. Roos, M. A. Viergever, *Reversible Interframe Compression of Medical Images: A Comparison of Decorrelation Methods*, IEEE Trans. Medical Imaging, 10(4):538-547, 1991.
6. S.W. Golomb, *Run-Length Encodings*, IEEE Trans. Inf. Theory, IT-12:399-401, July 1966.
7. M. Weinberger, G. Seroussi, G. Sapiro, *The LOCO-I Lossless Image Compression Algorithm: Principles and Standarization into JPEG-LS*, IEEE Trans. Image Proces. 9(8), 1309-1324, 2000.
8. G. Seroussi, M.J. Weinberger, *On Adaptive Strategies for an Extended Family of Golomb-type Codes*, HP Labs Technical Reports, HPL-97-08, 1997.
9. P. Howard, J.S. Vitter, *Fast and efficient lossless image compression*, Proceedings of IEEE Data Compression Conference, 351-360, 19993.
10. M. Burrows, D. J. Wheeler, *A Block-sorting Lossless Data Compression Algorithm*, *SRC Research Report 124*, Digital Systems Research Center, Palo Alto, Ca., May 1994.
11. M. Nelson, *Data Compression with the Burrows-Wheeler Transform*, Dr. Dobbs's Journal, pp. 46-50, September 1996.
12. J. Seward, *On the performance of BWT sorting algorithms*, Proceedings of IEEE Data Compression Conference, 173-182, 2000.
13. Z. Arnavut, *Move-to-front and inversion coding*, Proceedings of IEEE Data Compression Conference, 193-202, 2000.
14. S. Deorowicz, *Improvements to Burrows-Wheeler compression algorithm*, Software-Practice and Experience, 30(13):1465-1483, 2000.
15. I. Tabus, J. Rissanen, J. Astola, *Adaptive L-predictors based on Finite State Machine context selection*, Proc. ICIP'97 International Conference on Image Processing, pages 401-404, 1997.
16. M.J. Gormish, J.D. Allen, *Finite State Machine Binary Entropy Coding*, Proceedings of IEEE Data Compression Conference, 449, 1993 (http://www.rii.ricoh.com/~gormish/pdf/dcc93_fsm.pdf).