INFORMACJA

Materiały KODA, A.Przelaskowski

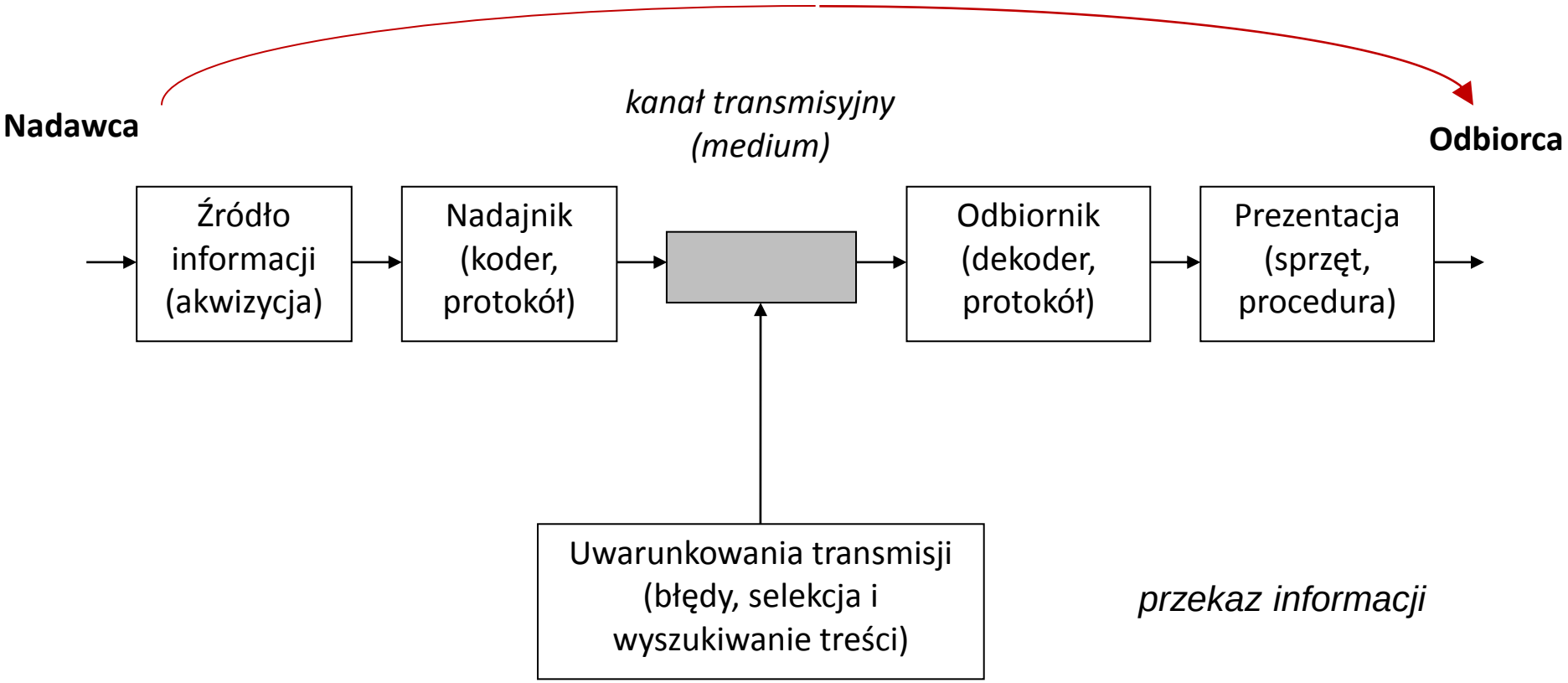
- Co to jest informacja?
- Statystyczna i semantyczna teoria informacji
- Koncepcje przesyłania informacji
- Modele źródeł
- Entropia
- Twierdzenia o kodowaniu
- Przykład BWT

Informacja

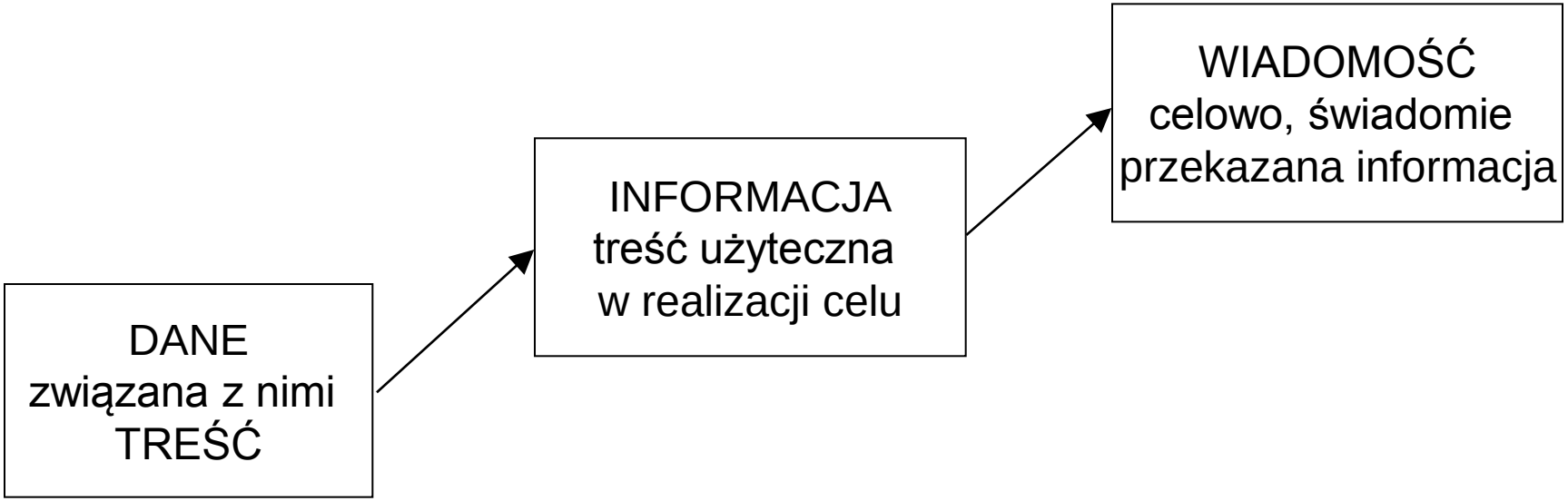
Co to jest informacja?

- Informacja to wszystko, co
  - służy bardziej skutecznej realizacji zamierzonego **celu**
  - jest użyteczne dla **odbiorcy** (użytkownika)
- Cechy: względność (subiektywność) - balans subiektywno-obiektywny, hierarchiczność
- **Koszt** pozyskania informacji (zwykle << korzyści)
- Informacja zawsze występuje w kontekście **przekazu** (podstawowy schemat nadawca-odbiorca)

Kontekst przekazu informacji



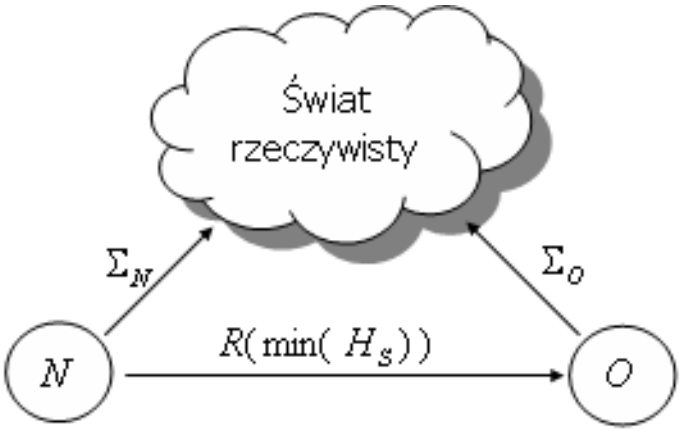
Rozumienie informacji



- dane nie zawsze wyrażają informację
- ważne jest odwołanie do treści (semantyki)
- czasami istotna jest tylko forma (syntaktyka, brak semantyki)
- przeciwieństwa: pragmatyka komputerowa i ludzka (przedmiot vs istota żywa)

I. Pragmatyka maszyny: informacja zobiektywizowana

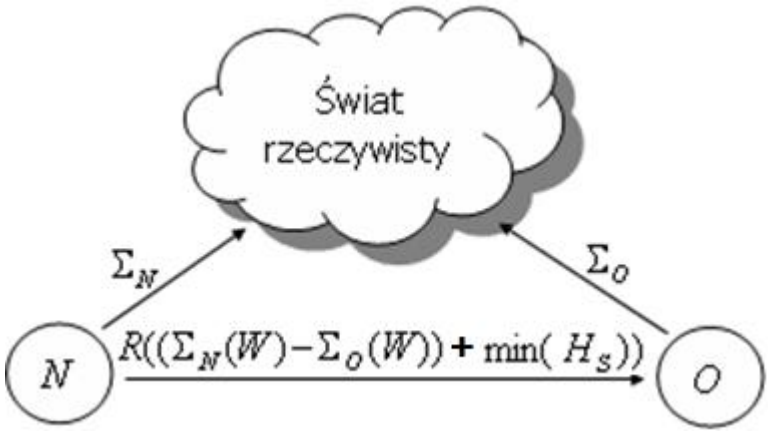
- Abstrahujemy od semantyki (dominuje syntaktyka)
- Probabilistyczny model źródła informacji
- **Informacja to poziom niepewności** odbiorcy w kontekście przekazu (wszystko co nieprzewidywalne jest informacją)



Model niepewności

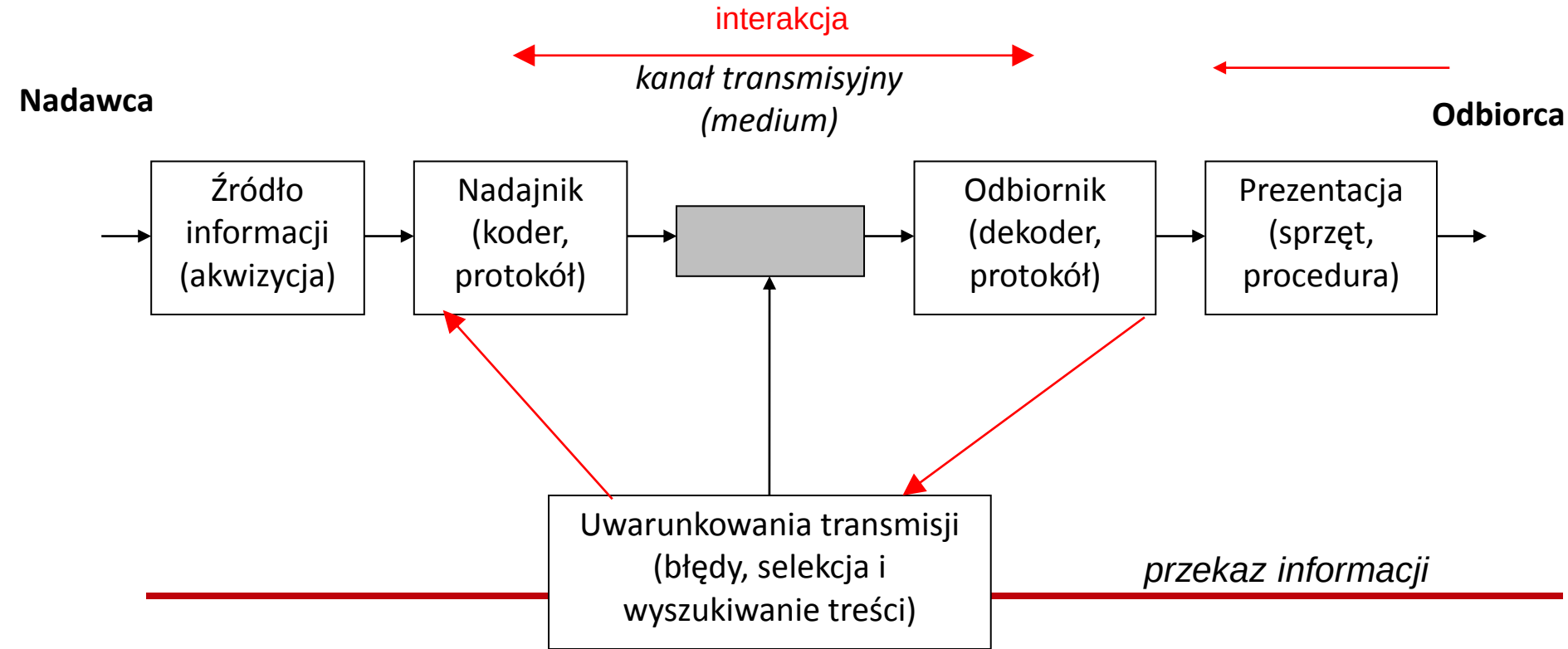
reprezentacja danych *R* źródła S o minimalizowanej entropii przesyłanej od nadawcy *N* do odbiorcy *O* przy z założenia zgodnych wartościach semantycznych ($\Sigma_N = \Sigma_O$), gdzie Σ jest funkcją semantyczną

II Pragmatyka ludzka: porządek znaczeń



Model semantyka+niepewność

reprezentacja informacji o minimalizowanej entropii źródła opisującego dodatkowo różnicę znaczeń u nadawcy i odbiorcy w odniesieniu do przesyłanej wiadomości *W*.



Pragmatyka I, czyli zacznijmy od niepewności: stochastyczne modele informacji

- Zmienna losowa *X*– proces (łańcuch) losowy (X_t,*t*)– pole losowe i szereg czasowy

Zmienna losowa to dowolna funkcja o wartościach rzeczywistych określona na zbiorze zdarzeń elementarnych

Proces losowy, inaczej funkcja losowa, to funkcja, której wartości leżą w przestrzeni definiowanej przez szereg zmiennych losowych. Inaczej to rodzina zmiennych losowych (X_t,t), gdzie t przebiega pewien przedział dziedziny. Jeśli jest to przedział czasowy – proces nazywamy szeregiem czasowym, jeśli zaś obszar przestrzeni – proces nazywamy polem losowym.

Łańcuch losowy to proces losowy zdefiniowany na dyskretnej przestrzeni stanów (ziarnisty zbiór zmiennych losowych rodziny).

Teoria Shannona – podstawowe cele

- modelowanie (modele źródeł informacji)
  - stworzenie wiarygodnej, strukturalnej charakterystyki źródła informacji tłumaczącego dane źródłowe w sposób zwarty
  - *de facto* jest to model informacji w celu jej zakodowania
- obliczenia ilości informacji dostarczanej przez źródła
  - miary informacji
  - wyznaczanie teoretycznych wartości granicznych wydajności koderów tych źródeł
- opracowania przesłanek do konstrukcji kodów dla zdefiniowanych źródeł informacji
  - sugestie metod kodowania binarnego

Stara, dobra teoria (probabilistyczna teoria informacji)

Prace C. Shannona określiły matematyczne podstawy statystycznej teorii informacji formalizując pojęcia:

- bezstratnej kompresji źródeł informacji modelowanych **dyskretnym procesem losowym** (informacja ziarnista) - w tym modele źródeł, entropia, twierdzenia o kodowaniu źródeł

- stratnej kompresji źródeł informacji modelowanych **ciągłym procesem losowym** - w tym funkcja zniekształceń źródeł informacji R(D), średnia informacja wzajemna

Alternatywa matematyczna: aproksymacyjna teoria informacji Kołmogorowa

- proces stochastyczny zastąpiono klasą funkcji (sygnałów) *f* określonych w dziedzinie *T*, tj. przez

$$\Theta = \{ f(t) : t \in T \}$$

- Dowolna *f* jest aproksymowana i dyskretyzowana w koderze *K* przez

$$\tilde{f} = \mathcal{K}(f)$$

dobraną z sieci aproksymacji

$$\Theta_{\mathcal{K}} = \{ \tilde{f} : \exists_{f \in \Theta} \; \tilde{f} = \mathcal{K}(f) \}$$

przy ograniczeniu

$$\sup_{f \in \Theta} \| f - \mathcal{K}(f) \| \leq \varepsilon$$

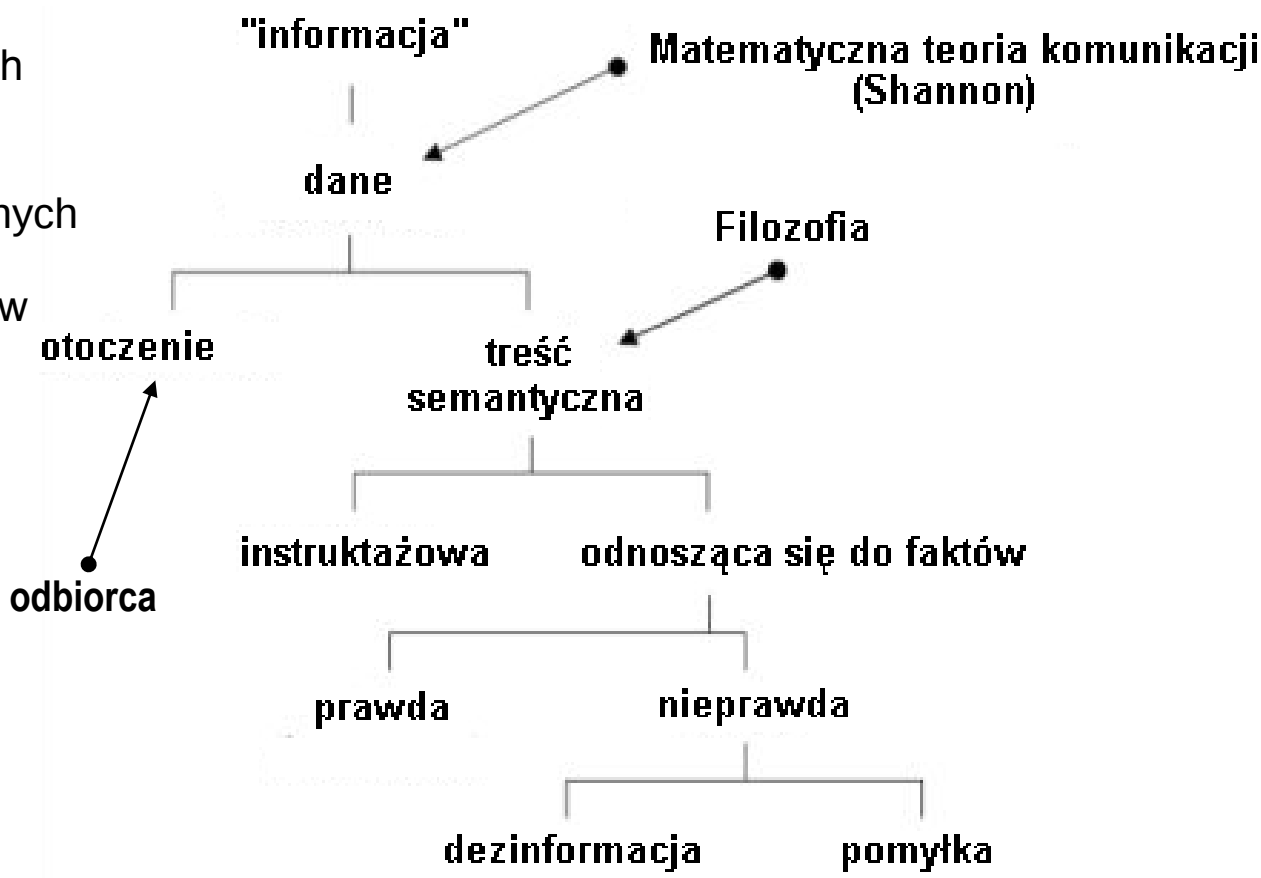
i minimalnym rozmiarze sieci.

- ε-entropia Kołmogorowa jako miara informacji:

$$H_{\varepsilon}(\Theta, \|\cdot\|) = \log_2 \left[\min_{\mathcal{K} \in Z_{\varepsilon, \|\cdot\|}} N(\Theta_{\mathcal{K}}) \right]$$

Alternatywny semantyczne

- Treść semantyczna:
  - początki to prace Bar-Hillela i Carnapa z lat 50
  - dodatkowo określane jest znaczenie poszczególnych symboli alfabetu źródła informacji
  - wykorzystanie ontologicznych i aksjologicznych (wartościowanie) aspektów rozumienia informacji
  - istotna jako uzupełnienie teorii matematycznych



Stanford Encyklopedia of Philosophy

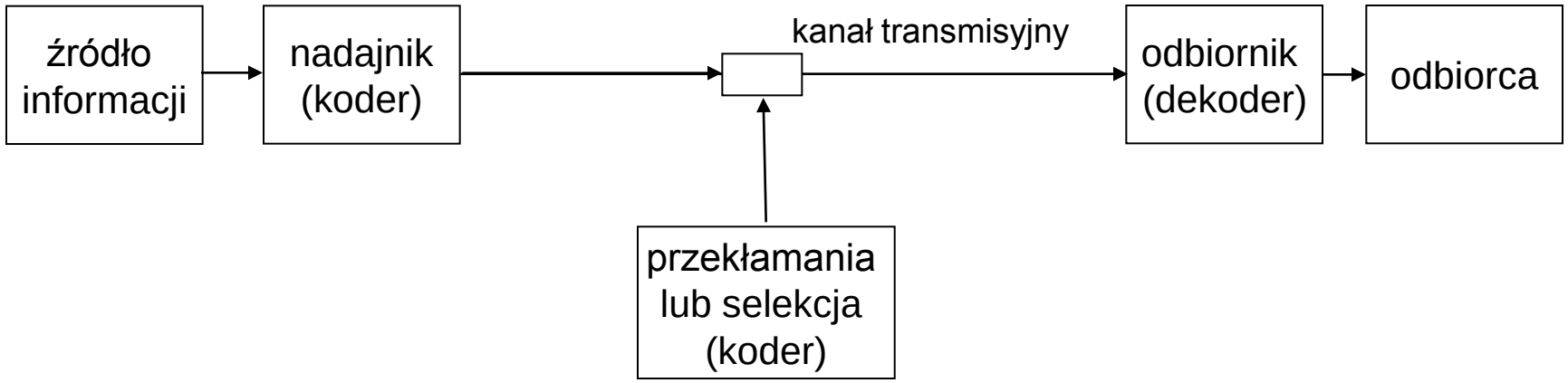
Uzupełnienia w teorii informacji

- Psychowizualny model odbiorcy
  - fizjologia, medycyna, psychologia
  - percepcja, przyswajanie informacji
- Użytkowy model odbiorcy
  - pragmatyka
  - odniesienie do zastosowań
- Modele lingwistyczne
  - gramatyki
  - słowniki
- Modele obliczeniowe, teoria liczb
  - operacje na przedziałach
  - predykcje
  - konteksty złożone
- Modele geometryczne
- Modele obiektowe
- Modele specyficzne
- ….

Koncepcja probabilistyczna

- modelowanie źródeł informacji

Teoria informacji (probabilistyczna, komunikacji)



Stochastyczne modele źródeł informacji

- Źródło dostarcza (emituje) symbole z określonym prawdopodobieństwem p_i
- Informacja jest realizacją zmiennej losowej, procesu, łańcucha
- Modele dyskretne
  - stacjonarne: prawdopodobieństwo jest niezmienne względem przesunięć w czasie
  - ergodyczne: każda generowana sekwencja ma te same właściwości statystyczne

Model bez pamięci (DMS)

- alfabet źródła $A_S = \{a_1, a_2, \ldots, a_n\}$
- zbiór prawdopodobieństw $P_S = \{p_1, p_2, \ldots, p_n\}$

Rozszerzenie stopnia N źródła S

$$A_S^N = \underbrace{A_S \times A_S \times \cdots \times A_S}_{N}$$

łączenie symboli w bloki

Model z pamięcią (CSM)

Wykorzystywane są prawdopodobieństwa warunkowe $P(\cdot|\mathbf{c}^t)$

- alfabet źródła $A_S = \{a_1, a_2, \ldots, a_n\}$
- zasada określania kontekstu C
- zbiór kontekstów C dla źródła S postaci
  $A_C{}^S = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k\}$
- zbiór prawdopodobieństw warunkowych
  $P(a_i|\mathbf{b}_j) = N(a_i, \mathbf{b}_j)/N(\mathbf{b}_j)$

Model Markowa (rzędu m)

$P(a_i|\mathbf{c}_j) = P(a_i|a_{j1}, a_{j2}, \ldots, a_{jm})$

czyli że np. $P(s_{t+1}|\mathbf{c}^t) = P(s_{t+1}|s_t, s_{t-1}, \ldots, s_{t-m})$

Łańcuch (model) Markowa

- skończone łańcuchy losowe działające w czasie dyskretnym
- niepusty zbiór stanów odnosi się do naturalnego alfabetu zdarzeń i zależy od wielkości kontekstu modelu

$$A_C = \{\mathbf{b}_j\}_{j=1,\ldots,k}$$

- wyjście stanów jest zdeterminowane (alfabet obserwacji)

$$A_O = A_S = \{a_i\}_{i=1,\ldots,n}$$

- łańcuch Markowa rzędu *m* (wymiar wektora kontekstu **b**)

$$p_{j,i} = P(s_t = a_i \mid \mathbf{b}_j)$$

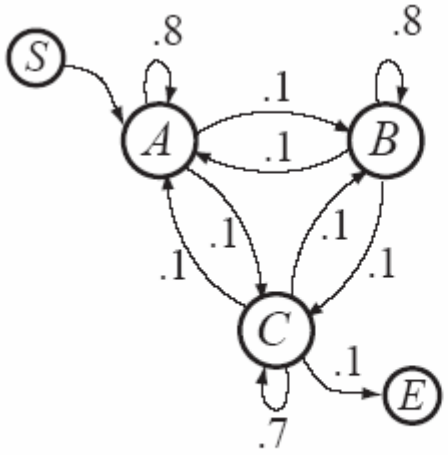
$$\text{gdzie } \mathbf{b}_j = s_{t-1,t-2,t-m} = (a_{j_1},\ldots,a_{j_m})$$

definiowany przez macierz przejść:

$$M = (p_{j,i})_{\mathbf{b}_j \in A_C, a_i \in A_S}, \qquad \sum_{a_i \in A_S} p_{j,i} = 1 \quad \forall j$$

Model Markowa

- następny stan zależy od obecnego, model ma zdeterminowane wyjście, nie zależy od *t*
- przykład modelu rzędu 1 z alfabetem {A,B,C}

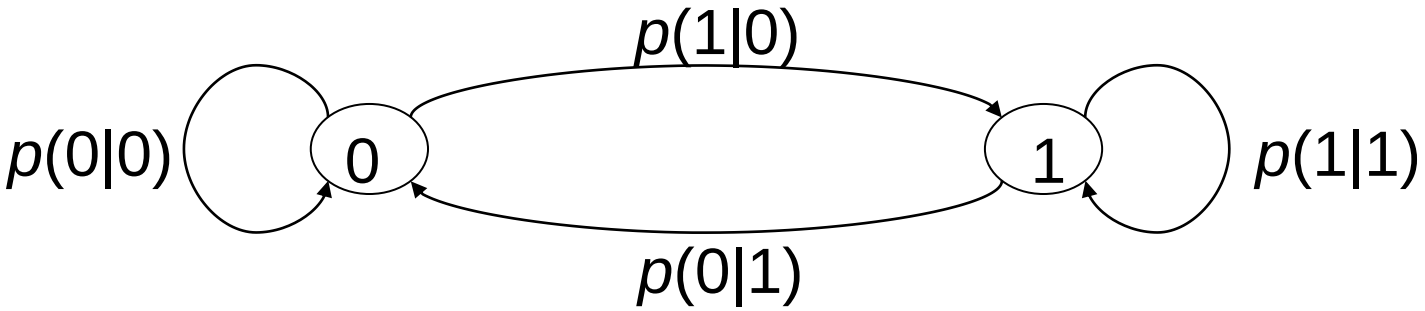


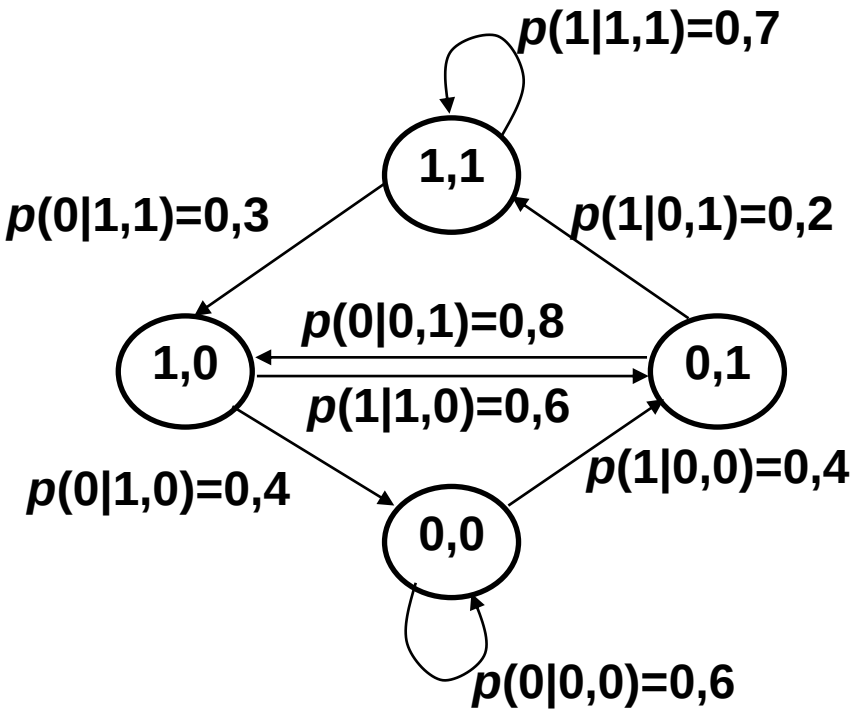
$b_j=a_{j1}$ \ a_i	S	A	B	C	E
S	0	1	0	0	0
A	0	.8	.1	.1	0
B	0	.1	.8	.1	0
C	0	.1	.1	.7	.1
E	0	0	0	0	1

S A A A A A A A A B B B B B B B B B C C C C B B B B B B C E

- skończona liczba stanów zależy od rzędu modelu i rozmiaru alfabetu A_S

Przykłady modelu Markowa





Miara ilości informacji (entropia)

- informacja własna symbolu a_i: $I(a_i) = \lg(1/p_i)$
- średnia informacja symboli źródła:

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

- entropia źródła Markowa

$$H(S|a_{j_1}, a_{j_2}, \ldots, a_{j_m}) = -\sum_{i=1}^{n} P(a_i|a_{j_1}, \ldots, a_{j_m}) \log_2 P(a_i|a_{j_1}, \ldots, a_{j_m})$$

(w pewnym stanie)

$$H(S|C^{(m)}) = \sum_{A_{C^{(m)}}^{S}} P(a_{j_1}, \ldots, a_{j_m}) H(S|a_{j_1}, \ldots, a_{j_m})$$

(uśredniamy po wszystkich stanach)

Entropia (łączna)

$$H(S) = \lim_{m \to \infty} \frac{1}{m} I_m$$

gdzie

$$I_m = -\sum_{j_1=1}^{n} \sum_{j_2=1}^{n} \cdots \sum_{j_m=1}^{n} P(a_{j_1}, a_{j_2}, \ldots, a_{j_m}) \lg P(a_{j_1}, a_{j_2}, \ldots, a_{j_m}) =$$

$$= -\sum_{j_1,\ldots,j_m=1}^{n} \Pr(s_1 = a_{j_1}, \ldots, s_m = a_{j_m}) \lg \Pr(s_1 = a_{j_1}, \ldots, s_m = a_{j_m})$$

Relacje entropii hipotetycznej i modelowanej

$$H(S) \leq H(S|C^{(m)}) \leq H(S_{\mathrm{DMS}})$$

Zasada: modele Markowa **możliwie** wysokiego rzędu są szczególnie użyteczne

Twierdzenie o bezstratnym kodowaniu źródła

Zał: ***S* - ergodyczne źródło** z alfabetem o t wygenerowanych elementach i entropii $H(S)$; **kod *K*** - bloki po N ($N \leq t$) symboli alfabetu S są jednocześnie kodowane za pomocą binarnych słów kodowych dających kod jednoznacznie dekodowalny.

Teza: Możliwa jest taka konstrukcja kodu K, że p**oprzez dobór odpowiednio dużej wartości *N*** średnia liczba bitów reprezentacji kodowej przypadająca na symbol tego źródła $L_K(S)$ leży dowolnie blisko $H(S)$, czyli dla dowolnie małego $\delta > 0$ zachodzi

$$H(S) \leq L_K(S) < H(S) + \delta$$

Ponadto, $H(S) \leq L_K(S)$ jest spełniona dla dowolnego jednoznacznie dekodowalnego kodu K przypisującego słowa kodowe N-elementowym blokom symboli źródła

- Zasada 1: zalecane są kody strumieniowe (blokowe) jako różnowartościowe przekształcenie ciągu wejściowego w słowo(a) kodowe
- Zasada 2: należy stosować wiarygodne modele źródeł informacji

Cel zasadniczy – opisać (by usunąć) nadmiarowość

Nadmiarowość (precyzyjnie)

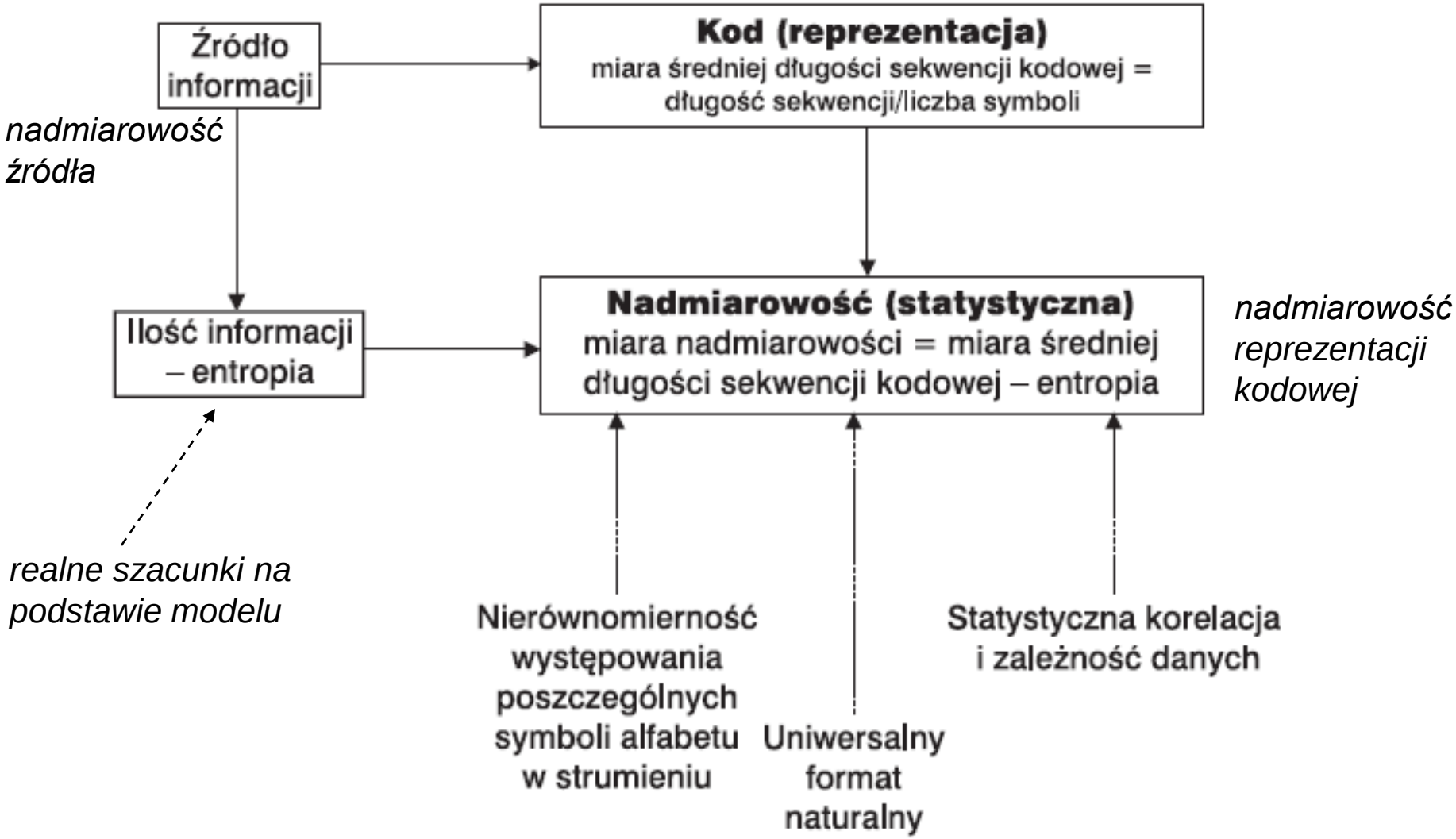
Nadmiarowość (reprezentacji) źródła

Nadmiarowość statystyczna sekwencji danych źródła informacji określana jest jako różnica pomiędzy średnią bitową reprezentacji danych źródłowych (znane) i entropią tego źródła (???)

Nadmiarowość reprezentacji kodowej (względem modelu źródła)

Nadmiarowość zakodowanej reprezentacji sekwencji danych, uzyskanej za pomocą kodu wykorzystującego określony model źródła informacji, jest to różnica pomiędzy średnią bitową reprezentacji kodowej i entropią modelu tego źródła

Nadmiarowość kodowania (orientacyjnie)



Przykład: nadmiarowość reprezentacji obrazów

- przestrzenna (jednoobrazowa i międzyobrazowa)
- czasowa, pojawiająca się wskutek korelacji obrazów sekwencji rejestrowanej w kolejnych chwilach czasowych
- spektralna, wynikająca z korelacji komponentów w obrazach wielokomponentowych (kolorowych, pseudokolorowych, innych);
- percepcyjna, powodowana niedoskonałością narządu wzroku odbierającego informację;
- semantyczna, powstająca na poziomie interpretacji informacji

Zasada: scharakteryzować wstępnie nadmiarowość, którą chcemy usunąć

Metody (praktyka!!!) – warunek konieczny, czyli jednoznaczna dekodowalność

Jednoznaczna dekodowalność kodu – zasady konstrukcji/weryfikacji

1) Kody symboli: liczba słów odpowiada liczbie symboli, różne słowa kodowe

$K_1(\{a_1,a_2,a_3,a_4\})=A_{K2}=\{00,01,10,11\}$ $K'_1(a_1,\ldots,a_4) = 11$

2) Analiza długości słów: nierówność **Krafta-MacMillana**

- warunek konieczny jednoznacznej dekodowalności kodu:

$$\sum_{i=1}^{n} 2^{-L_i} \leq 1$$

$A_{K3} = \{0,10,11,?\}$

$A_{K'3} = \{0,10,11,111\}$

- istnieje kod przedrostkowy dla dowolnych całkowitych L_i >0 spełniających tę nierówność

3) Kombinacja słów nie jest kombinacją innych słów

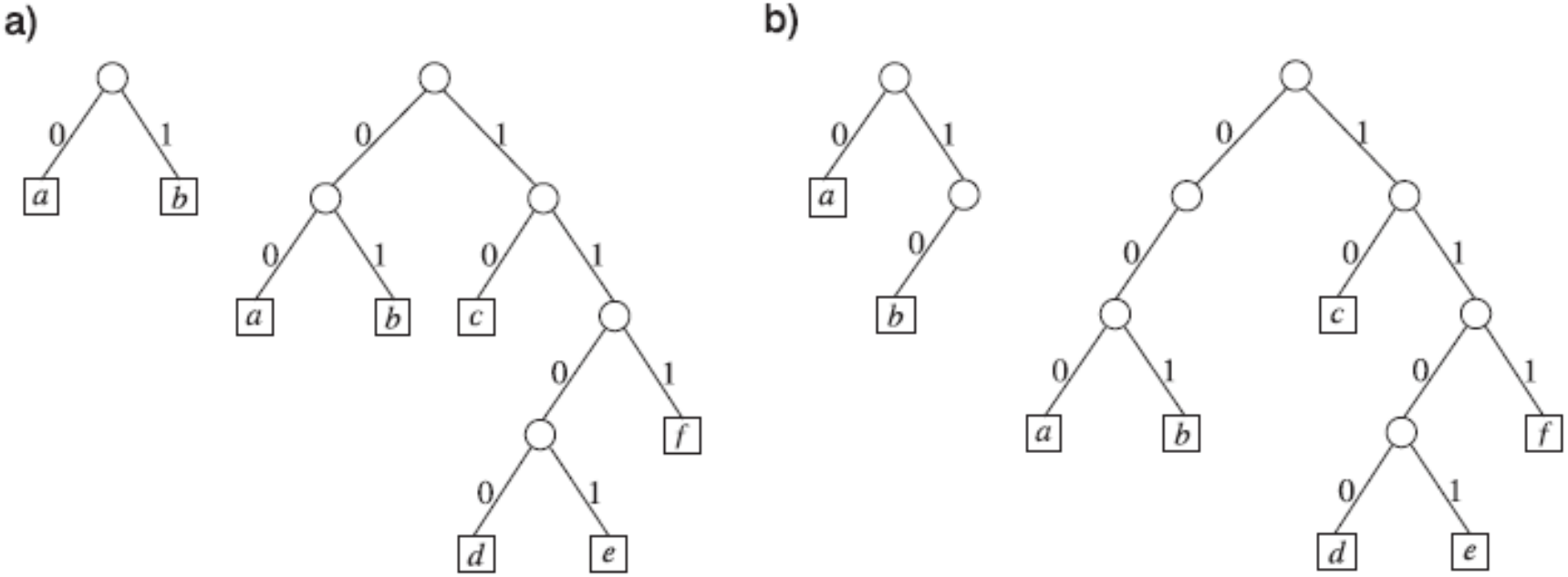
$A_{K2}= \{1,10,00\}$, $A_{K'2}= \{1,10,01\}$

Najlepiej: kody przedrostkowe

- 4) kody przedrostkowe (bezprzedrostkowe, *prefix codes, prefix condition codes*, *prefix-free codes, comma-free code, instantaneous codes*)

$$A_{K4} = \{1,01,001,0001,00001,\ldots\ldots\}$$

- 5) kody drzew binarnych



Rys. 2.3. Przykłady drzew: a) lokalnie pełnych; b) nie będących lokalnie pełnymi

Metody– wstępne przekształcenie danych w kompresji bezstratnej

Transformacja Burrowsa-Wheelera

- BWT metodą wstępnej dekompozycji danych, uzupełnionej zwykle przekształceniem MTF oraz kodowaniem entropijnym
- Inaczej *block-sorting method* stosowana w WinImp, UHBC, ABC, 7-Zip, PAC, SBC, UHBC, ZZIP i innych
- Cel: utworzenie reprezentacji pośredniej bardziej podatnej na kodowanie poprzez porządkowanie danych w blokach:
  - pojawiają się serie symboli identycznych lub zbliżonych
  - zachowany jest alfabet źródła, rozszerzony o dodatkowy wskaźnik pozycji bloku startowego

Algorytm BWT (prosty)

- Podział strumienia danych na bloki *n* elementowe (zależny od właściwości danych i wymagań implementacji)
- Dla każdego bloku: tworzenie ko-bloków dodatkowych (*n*-1 z przesunięciem o jedną pozycję w lewo i zawijaniem)
- Sortowanie (niezależne) każdego zestawu *n* ko-bloków
- Formowanie reprezentacji pośredniej jako:
  - ciąg symboli z ostatniej pozycji *n* ko-bloków
  - wskaźnik bloku startowego

Przykład: ….„adam”….

- ciąg źródłowy dzielimy na bloki *n*=4 symboli ($s_i \in A_S$)
- z początkowej postaci b_0=„adam” tworzone są trzy ko-bloki: b_1=„dama”, b_2=„amad”, b_3=„mada”
- po sortowaniu otrzymujemy:

Posor-towane bloki b_j	Elementy bloków	Symbol z pierwszej pozycji v_P[i]	Symbol z ostatniej pozycji v_O[i]	Indeks pozycji bloku $index(b_j)$
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**

- reprezentacja pośrednia: ($\mathbf{s}_{BWT}$, s_{idx}), gdzie $\mathbf{s}_{BWT}$ =„mdaa”, s_{idx} = 2 (indeks pozycji bloku b_1)

Inne przykłady

```
t: hat acts like this:<13><10><1
t: hat buffer to the constructor
t: hat corrupted the heap, or wo
W: hat goes up must come down<13
t: hat happens, it isn't  likely
w: hat if you want to dynamicall
t: hat indicates an error.<13><1
t: hat it removes arguments from
t: hat looks like this:<13><10><
t: hat looks something like this
t: hat looks something like this
t: hat once I detect the mangled
```

final char (L)	sorted rotations
a	n to decompress. It achieves compression
o	n to perform only comparisons to a depth
o	n transformation} This section describes
o	n transformation} We use the example and
o	n treats the right-hand side as the most
a	n tree for each 16 kbyte input block, enc
a	n tree in the output stream, then encodes
i	n turn, set $L[i]$ to be the
i	n turn, set $R[i]$ to the
o	n unusual data. Like the algorithm of Man
a	n use a single set of probabilities table
e	n using the positions of the suffixes in
i	n value at a given point in the vector $R
e	n we present modifications that improve t
e	n when the block size is quite large. Ho
i	n which codes that have not been seen in
i	n with ch appear in the {\em same order
i	n with ch. In our exam
o	n with Huffman or arithmetic coding. Bri
o	n with figures given by Bell~\cite{bell}.

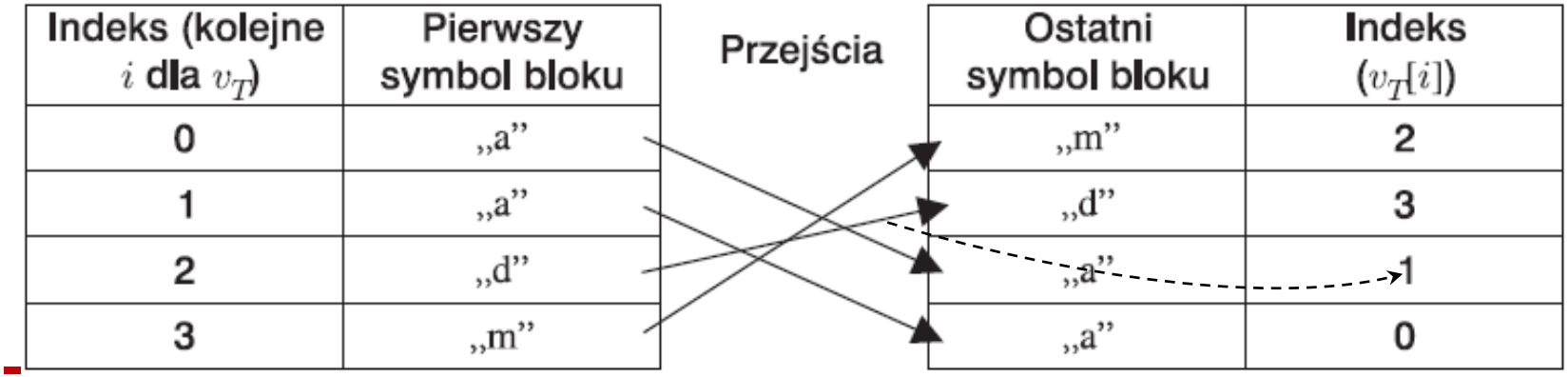
Rekonstrukcja „adam”

- reprezentacja wejściowa: („mdaa”,2)
- pozycja bloku startowego równa 2 oznacza, że pierwszym symbolem bloku źródłowego jest „a”
- posortowanie „mdaa” daje wektor początkowych symboli bloków „aadm”; znaczy to, że pozycja b_2 to 1 (ostatni jego symbol „d” jest drugim zdekodowanym symbolem b_0)
- pierwszy symbol b_2 to „a”; stąd pozycja b_3 to 3, a więc trzeci element b_0 to „a” (z ostatniej pozycji b_3)
- ponieważ pozycja b_0 to 0, możemy odczytać ostatni symbol „m”; mamy więc pełną rekonstrukcję: b_0=„adam”

Algorytm BWT (odwrotny)

- Odczytanie pierwszego symbolu
- Ustalenie wektora porządkowych symboli ko-bloków
- 1) Odtworzenie danych źródłowych według zasady: pierwszy symbol ko-bloku *i* jest ostatnim symbolem ko-bloku (*i*+1)%4 oraz symbolem b_0 na pozycji *i*
- 2) Pomocna w implementacji struktura wektora transformacji:

  v_T [*index*(b_i)] = *index*(b_{i+1}) i odczytanie ostatnich symboli kolejnych ko-bloków



Przykładowa implementacja

```
int n = 4, 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;
      }
}
```

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

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

Korzyści

- wydobywanie nadmiarowości (różnicowanie prawdopodobieństw, długie serie)
- uporządkowanie kontekstowe danych źródłowych
- wykorzystanie kontekstu długiego (*n*-1) i ‘bezprzyczynowego - wprzód’
- stosunkowo mała złożoność (problem sortowania)
- dobór rozmiaru bloków na podstawie ch-ki danych (możliwość zrównoleglenia obliczeń)
- duża efektywność dla danych tekstowych
- podatność na kodowanie entropijne

Move To Front (MTF)

- Dynamiczna zmiana alfabetu (ostatnio kodowany na początek)
- Reprezentacja: ciąg indeksów aktualnego alfabetu przypisywanych kolejnym symbolom wejściowym
- Przykłady: „cccaaaag” (pozycja w alfabecie początkowym a-0,c-2,g-6) → „2,0,0,1,0,0,0,6” (indeksujemy od 0)

Encoding		
Symbol	Code	List
a	0	a b c d
b	1	b a c d
a	1	a b c d
a	0	a b c d
c	2	c a b d
a	1	a c b d
b	2	b a c d
a	1	a b c d
d	3	d a b c

Decoding		
Code	Symbol	List
0	a	a b c d
1	b	b a c d
1	a	a b c d
0	a	a b c d
2	c	c a b d
1	a	a c b d
2	b	b a c d
1	a	a b c d
3	d	d a b c

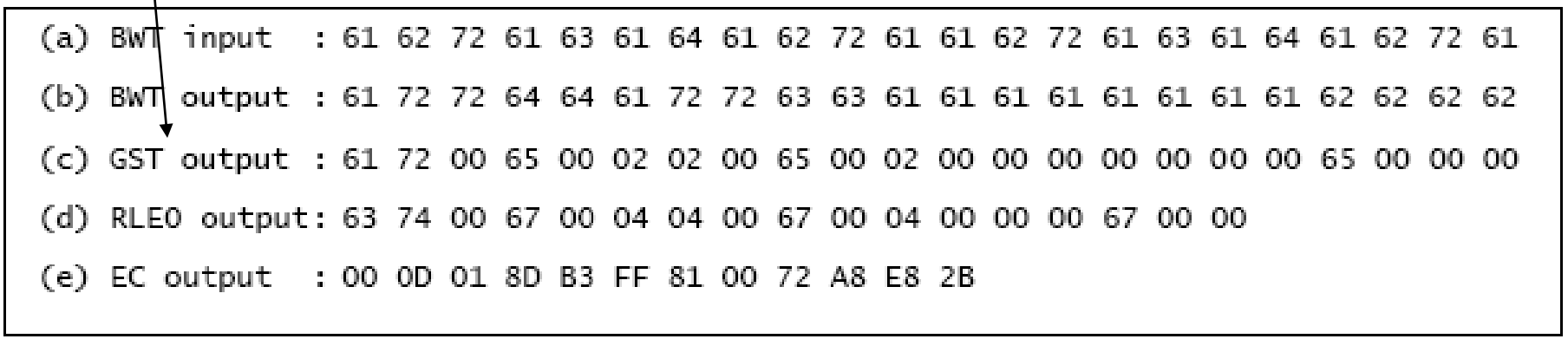
Inne modyfikacje

- RLE0, czyli kodowanie serii zer występujących po MTF
- DC (distance coder), czyli kodowanie odległości

Pozycja w sekwencji wejściowej				**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Wejście				a	a	b	a	c	b	b	b	a	b	a	a	c
Odległość jako wskaźnik pozycji	1	3	5	1	2	3	5	8	1	1	2	2	0	1	0	0
Ucięte końcowe 0 oraz odległość jako liczba pustych pozycji	1	2	3	1	1	1	3	6	1	1	1	1	0	1		
Zastosowanie RLE	1	2	3	2	-	1	3	6	3	-	-	1	0	1		

Efekty

Global structure transformation (np. MTF)



```
(a) BWT input  : 61 62 72 61 63 61 64 61 62 72 61 61 62 72 61 63 61 64 61 62 72 61
(b) BWT output : 61 72 72 64 64 61 72 72 63 63 61 61 61 61 61 61 61 62 62 62 62
(c) GST output : 61 72 00 65 00 02 02 00 65 00 02 00 00 00 00 00 00 00 65 00 00 00
(d) RLE0 output: 63 74 00 67 00 04 04 00 67 00 04 00 00 00 67 00 00
(e) EC output  : 00 0D 01 8D B3 FF 81 00 72 A8 E8 2B
```

Fig. 2. Transformed data of the input string "abracadabraabracadabra" by the different stages

J. Abel, Improvements to the Burrows-Wheeler Compression Algorithm: After BWT Stages, http://www.data-compression.info/JuergenAbel/Preprints/Preprint_After_BWT_Stages.pdf

Rezultaty

Block size (bytes)	bits/character	
	book1	Hector corpus
1k	4.34	4.35
4k	3.86	3.83
16k	3.43	3.39
64k	3.00	2.98
256k	2.68	2.65
750k	2.49	-
1M	-	2.43
4M	-	2.26
16M	-	2.13
64M	-	2.04
103M	-	2.01

M. Burrows and D.J. Wheeler, **A Block-sorting Lossless Data Compression Algorithm,**, **SRC Research Report, 1994**

Programme	Total CPU time/s		Total compressed size (bytes)	mean bits/char
	compress	decompress		
compress	9.6	5.2	1246286	3.63
gzip	42.6	4.9	1024887	2.71
Alg-C/D	51.1	9.4	856233	2.43
comp-2	603.2	614.1	848885	2.41

Przykładowe wyniki

(Darek Pryczek i Mariusz Gwiazda w ramach projektu KODA)

Metoda	rozmiar po kompresji	średnia bitowa	czas kompresji (s)	czas dekompresji (s)
allcalgarycorpus (3 141 622)				
MTF+RLE0+AA	879 002	2.24	4.19	2.02
MTF+RLE0+SA	914 438	2.33	**3.5**	**1.39**
MTF+AA	949 328	2.42	4.84	2.81
MTF+SA	1 032 707	2.63	3.69	1.77
DC+AA	861 148	**2.19**	4.73	1.45

allcalgarycorpus (3 141 622)		BR	t_k (s)	t_d (s)
DC+AA	861 148	2.19	4.73	1.45
WinRAR	764 588	1.95	3.5	2.1
Bzip2	862 695	2.20	1.8	**0.5**
WinRK	650 362	**1.66**	71	51
WinZip	1 041 027	2.65	**1.5**	1

allsilesia (211 938 580)				
DC+AA	52 469 347	1.98	382	119
WinRAR	50 808 227	1.92	225	62.2
Bzip2	54 573 629	2.06	142	37.1
WinRK	43 611 153	**1.65**	4 472	4 160
WinZip	68 979 726	2.60	**34**	**16**

Kodowanie plików banalnych i trudnych

¤	rozmiar·po·kompresji¤	średnia·bitowa¤	czas·kompresji·(s)¤	czas·¶ dekompresji·(s)¤
a.txt·(10·485·760)¤				
MTF+RLE0+AA¤	51¤	0.000039¤	8.05¤	0.55¤
WinRAR¤	5·205¤	0.003971¤	1.7¤	0.21¤
Bzip2¤	49¤	**0.000037¤**	**0.8¤**	**0.17¤**
WinRK¤	970¤	0.000740¤	50¤	47¤
ab.txt·(10·485·760)¤				
DC+AA¤	59¤	**0.000046¤**	9.66¤	**0.39¤**
WinRAR¤	5·264¤	0.004016¤	1.7¤	0.8¤
Bzip2¤	373¤	0.000285¤	20.3¤	0.6¤
WinRK¤	969¤	0.000739¤	51¤	50¤
WinZip¤	10·303¤	0.007861¤	**1¤**	1¤
ala.txt·(10·485·760)¤				
DC+AA¤	98¤	**0.000083¤**	24.6¤	**0.406¤**
WinRAR¤	5·344¤	0.004077¤	1.9¤	0.76¤
Bzip2¤	798¤	0.000609¤	38¤	0.7¤
WinRK¤	1·165¤	0.000889¤	58¤	57¤
WinZip¤	20·476¤	0.015622¤	**1¤**	1¤
random.txt·(10·485·760)¤				
MTF+SA¤	10·131·731¤	**7.730¤**	19.4¤	11.9¤
WinRAR¤	10·485·834¤	8.00¤	20.3¤	0.13¤
Bzip2¤	10·530·234¤	8.03¤	12.5¤	3.6¤
WinRK¤	10·680·775¤	8.15¤	1177¤	1098¤
WinZip¤	10·487·478¤	8.00¤	**3¤**	**1¤**

Dopasowanie rozmiaru bloku

Rozmiar okna BWT [Kb]	rozmiar po kompresji	średnia bitowa	czas kompresji (s)	czas dekompresji (s)
allcalgarycorpus (3 141 622)				
1	1 696 885	4.32	3.83	3.69
2	1 481 334	3.77	3.33	2.41
4	1 316 443	3.35	3.09	1.83
8	1 190 558	3.03	2.97	1.42
16	1 094 942	2.79	2.94	1.19
32	1 019 484	2.6	3.17	1.08
64	961 041	2.45	2.86	1.05
128	914 436	2.33	3.14	1.14
256	891 818	2.27	3.47	1.22
512	871 773	2.22	3.84	1.28
1024	862 773	2.2	4.2	1.36
2048	859 989	2.19	4.42	1.38
3141	861 148	2.19	4.59	1.44
4092	861 148	2.19	4.61	1.42

dickens (10 192 446)				
1	6 299 616	4.94	15.5	12.2
2	5 487 887	4.31	14.3	7.76
4	4 871 178	3.82	12.9	5.92
8	4 393 763	3.45	12.4	4.72
16	4 008 650	3.15	11.7	4.03
32	3 694 972	2.9	11.5	4.3
64	3 434 636	2.7	12.3	3.59
128	3 208 875	2.5	12.6	3.98
256	3 019 270	2.37	13.7	4.39
512	2 864 002	2.25	15.4	4.67
1024	2 728 439	2.14	16.6	4.83
2048	2 621 946	2.06	17.4	4.94
4092	2 556 958	2.01	18.1	5.13
6144	2 533 841	1.99	18.7	5.24
8192	2 517 244	1.98	19.2	5.28
10240	2 469 726	1.94	20.1	5.42

samba (21 606 400)				
1	10 649 876	3.94	27.6	23.9
2	9 034 768	3.35	23.8	15
4	7 847 258	2.91	21.7	11.4
8	6 968 159	2.58	20.2	8.81
16	6 287 917	2.33	19.5	7.47
32	5 757 402	2.13	19.5	6.66
64	5 341 941	1.98	19.8	6.47
128	5 046 303	1.87	22.4	7.08
256	4 841 271	1.79	25.2	7.08
512	4 682 494	1.73	31	7.5
1024	4 618 408	1.71	32.5	7.81
2048	4 547 436	1.68	33.4	8.11
4092	4 427 763	1.64	34.7	8.38
8192	4 356 896	1.61	37.2	8.69