Quellencodierung

Ziele der Quellencodierung: Datenkompression, Bandbreite reduzieren, Übertragungszeit reduzieren

Mittlere Codewortlänge

x_n: Symbole, l_n: Codelänge, $P(x_n)$: Wahrscheinlichkeit von x_n

$$L = \sum_{n=0}^{N-1} P(x_n) \cdot l_n \quad \text{(Bit/Symbol)}$$

z.B.

x_0	10	2 Bit	$P(x_0) = 0.45$	$l(x_0) = 1.15$
x_1	110	3 Bit	$P(x_1) = 0.47$	$l(x_1) = 1.09$
x_2	1110	4 Bit	$P(x_2) = 0.08$	$l(x_2) = 3.64$

$L = P(x_0) \cdot l_0 + P(x_1) \cdot l_1 + P(x_2) \cdot l_2$
$= 0.45 \cdot 2 + 0.47 \cdot 3 + 0.08 \cdot 4 = 2.63$ Bit/Symbol

Redundanz

Redundanz = Mittlerecodewortlänge − Entropie
$R = L - H(x)$ (Bit/Symbol)
$R > 0$: Code kann noch verlustfrei komprimiert werden
$R = 0$: Code kann nicht mehr verlustfrei komprimiert werden
$R < 0$: Code wird verlustbehaftet komprimiert
z.B. Fortsetzung linkes Beispiel
Mittlerecodewortlänge: $L = 2.63$ Bit/Symbol
Entropie: $H(x) = P(x_0) \cdot l(x_0) + P(x_1) \cdot l(x_1) + P(x_2) \cdot l(x_2)$
$= 0.45 \cdot 1.12 + 0.47 \cdot 1.09 + 0.08 \cdot 3.64 = 1.32$ Bit/Symbol
Redundanz: $R = L - H(x) = 2.63 - 1.32 = 1.31$ Bit/Symbol

Präfixfreiheit

Voraussetzung für binäre Codes unterschiedlicher Länge.
Kein Code bildet den Anfang eines anderen Codes.

z.B. x_0 10, x_1 110, x_2 1110 } 110 10 1110 10

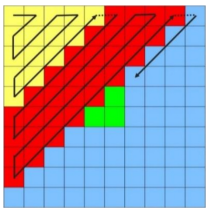
x_0 11, x_1 110, x_2 1110 } nicht unterscheidbar

Kompressionsrate

$$R = \frac{\text{Codierte Bits}}{\text{Originale Bits}} < 1 \quad ✓$$

Lauflängencodierung (RLE)

Runs werden mit Tokens codiert: (Marker, Anzahl, Code)
Als Marker wird ein selten verwendeter Code eingesetzt.
Bit/Token: Marker Bit + Zähler Bit + Zeichen Bit z.B. (A 6 R)
z.B. original: ...TERRRRRRMAUGGWXXXXL...
Komprimiert: TEA6RMA1AUA2GWA4XL
↳ Marker müssen immer codiert werden
Bit total: Header + (Anzahl Token · Token/Bit) + Einzelsymbole
ⓘ Es gibt keine Runs mit Länge 0, daher kann z.B. ein 4-Bit-Zähler eine Länge von 1 bis 16 abbilden



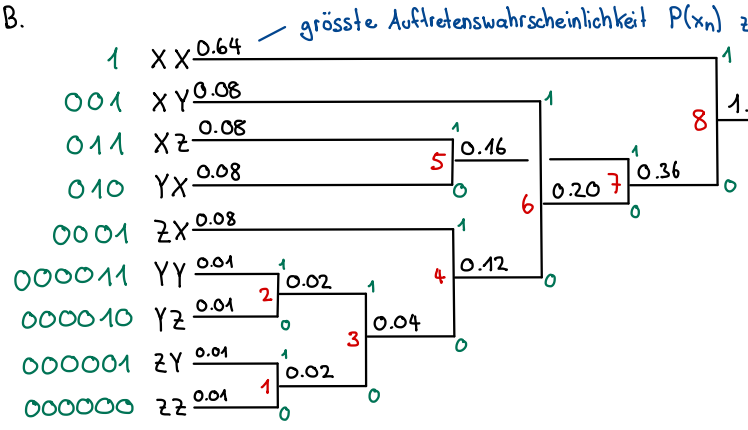
RLE mit 4-Bit-Zähler (1-16)
G15Y, G16R, G14R, G4B, G2G, G8B, G1G, G16B, G16B, G8B

Doppelsymbole

Wahrscheinlichkeit : $P(AA) = P(A) \cdot P(A)$
Entropie : $H(XX) = 2 \cdot H(X)$
Mittlere Codewortlänge : $L(X) = L(XX) : 2$
Redundanz : $R(X) = R(XX) : 2$
} wenn Vergleich zu ursprünglichem Symbol
z.B. $A = 0.08$, $P(AA) = 0.08 \cdot 0.08 = 0.64$

Huffman Codes

Häufige Symbole erhalten kurze Codes, Seltene Symbole erhalten lange Codes
Erzeugte Codes sind automatisch präfixfrei und optimal (es gibt keinen besseren präfixfreien Code)
Es werden immer die zwei kleinsten Codes zusammengefasst (Schluss muss 1 sein)
z.B.



ⓘ Wenn Einzelsymbol durch Doppelsymbol ersetzt wird, kann die Redundanz verkleinert werden.
Doppelsymbol XY, $X = 0.4$ und $Y = 0.2 \rightarrow XY = 0.4 \cdot 0.2 = 0.8$
Doppelsymbol XY ergibt Bit/2 Symbol für Vergleich mit Bit/Symbol, Bit/2 Symbol durch 2 dividieren.
Einzelsymbol zu Doppelsymbol alle möglichen Kombinationen

$H = 0.64 \cdot 0.644 + 4 \cdot (0.08 \cdot 3.644) + 4 \cdot (0.01 \cdot 6.644) = 1.84$ Bit/2 Symbol
$L = 0.64 \cdot 1 + 0.08 \cdot 3 + 0.08 \cdot 3 + 0.08 \cdot 3 + 0.08 \cdot 4 + 0.01 \cdot 6 + 0.01 \cdot 6 + 0.01 \cdot 6 + 0.01 \cdot 6 = 1.92$ Bit/2 Symbol
$R = 1.92 - 1.84 = 0.08$ Bit/2 Symbol

LZ77

Alle Zeichen werden durch Tokens fixer Länge ersetzt. Token: (Offset, Länge, Zeichen)

1.) Längste Übereinstimmung mit dem Vorschau-Buffer im Such-Buffer suchen.

2.) Verschiebung um Übereinstimmung + nächstes Zeichen

z.B. Encoder: AMAMMMAAAMMMTAAT...

Such-Buffer	Vorschau-Buffer
	A M A M M
A	M A M M M
A M	A M M M A
A M A M M	M A A A M
A M A M M M A A	A M M M T

LZ77-Token

Offset	Länge	Zeichen	Wert
0	0	A	A
0	0	M	M
2	2	M	AMM
4	2	A	MAA
6	4	T	AMMMT

Decoder:

Offset	Länge	Zeichen
0	0	A
0	0	M
2	2	M
4	2	A
6	4	T

Buffer
A
A M
A M A M M
A M A M M M A A
A M A M M M A A A M M M T

LZ77-Token: 4Bit (Suchbuffer 10, $\log_2(10)$) + 3 Bit (Vorschaubuffer 5, $\log_2(5)$) + 8 Bit = 15 Bit

Kompressionsrate: $R = \frac{\text{Anzahl Token} \cdot \text{Bit pro Token}}{\text{Anzahl Zeichen} \cdot \text{Bit pro Zeichen}} = \frac{5 \cdot 15}{13 \cdot 8} = \frac{75}{128} = 0.586 < 1$ ✓

8 Bit ASCII Tabelle

Character	Decimal Number	Binary Number	Character	Decimal Number	Binary Number
blank space	32	0010 0000	^	94	0101 1110
!	33	0010 0001	-	95	0101 1111
"	34	0010 0010	`	96	0110 0000
#	35	0010 0011	a	97	0110 0001
$	36	0010 0100	b	98	0110 0010
A	65	0100 0001	c	99	0110 0011
B	66	0100 0010	d	100	0110 0100
C	67	0100 0011	e	101	0110 0101
D	68	0100 0100	f	102	0110 0110
E	69	0100 0101	g	103	0110 0111
F	70	0100 0110	h	104	0110 1000
G	71	0100 0111	i	105	0110 1001
H	72	0100 1000	j	106	0110 1010
I	73	0100 1001	k	107	0110 1011
J	74	0100 1010	l	108	0110 1100
K	75	0100 1011	m	109	0110 1101
L	76	0100 1100	n	110	0110 1110
M	77	0100 1101	o	111	0110 1111
N	78	0100 1110	p	112	0111 0000
O	79	0100 1111	q	113	0111 0001
P	80	0101 0000	r	114	0111 0010
Q	81	0101 0001	s	115	0111 0011
R	82	0101 0010	t	116	0111 0100
S	83	0101 0011	u	117	0111 0101
T	84	0101 0100	v	118	0111 0110
U	85	0101 0101	w	119	0111 0111
V	86	0101 0110	x	120	0111 1000
W	87	0101 0111	y	121	0111 1001
X	88	0101 1000	z	122	0111 1010
Y	89	0101 1001	{	123	0111 1011
Z	90	0101 1010	|	124	0111 1100
[	91	0101 1011	}	125	0111 1101
/	92	0101 1100	~	126	0111 1110
]	93	0101 1101			

LZW

1.) Zeichen-Kette im Wörterbuch suchen

2.) Neuer Eintrag im Wörterbuch, Index: Wörterbuch-Identifikator, Token: Verweis, String: Token

z.B. Encoder: AMAMMMAAAMMMTAAT...

Decoder: (65), (77), (256), (77), (257), (65), (258), (77), (84), (261)

Index	Eintrag
...	...
65	A
77	M
84	T
...	...

Vorinitialisierung

Fortsetzung →

Index	Eintrag	Token
256	AM	65
257	MA	77
258	AMM	256
259	MM	77
260	MAA	257
261	AA	65
262	AMMM	258
263	MT	77
264	TA	84
265	AAT	261
266	T..	

Kann nicht übermittelt werden und kann für alle Berechnungen ignoriert werden.

grösster Wörterbuch Index

LZW-Token: $\log_2(265) = 9$ Bit

Index	Eintrag
...	...
65	A
77	M
84	T
...	...

Vorinitialisierung

Fortsetzung →

Token	Index	Eintrag	Output
65	256	AM	A
77	257	MA	M
256	258	AMM	AM
77	259	MM	M
257	260	MAA	MA
65	261	AA	A
258	262	AMMM	AMM
77	263	MT	M
84	264	TA	T
261	265	AA_	AA

Kompressionsrate: $R = \frac{\text{Anzahl übertragene Token (ohne Vorinitialisierung)} \cdot \text{Bit pro Token}}{\text{übertragene Anzahl Zeichen} \cdot \text{Bit pro Zeichen}} = \frac{10 \cdot 9}{15 \cdot 8} = \frac{90}{120} = 0.75 < 1$ ✓

letztes Zeichen "T" wird nicht übermittelt