



Algorithmentheorie

16 - Kompressionsverfahren für Texte

Prof. Dr. S. Albers

Kompressionsverfahren für Texte

Verlustfreie Kompression

Original kann perfekt rekonstruiert werden

Beispiele: Huffman Code, Lauflängencodierung, arithmetische Codierung, Lempel-Ziv

Verlustbehaftete Kompression:

Unterschied zwischen Original und decodiertem Objekt, nutzt physiologische und psychologische Eigenschaften des Auges und Ohres aus, erlaubt höhere Kompressionsraten

Beispiele: JPEG, MPEG

Verlustfreie Kompressionsverfahren

Ziel:

Finde umkehrbare Codierung, so dass der Text wieder (verlustfrei) rekonstruiert werden kann.

Beispiele:

- Huffman-Code
- Lauflängen-Codierung
- Arithmetische Codierung
- Lempel-Ziv Codierung

Arithmetische Codierung

Ziel:

Bits der codierten Nachricht umgekehrt proportional zur Wahrscheinlichkeit der Nachricht.

Wahrscheinlichkeit groß - # Bits klein

Wahrscheinlichkeit klein - # Bits groß

Arithmetische Codierung

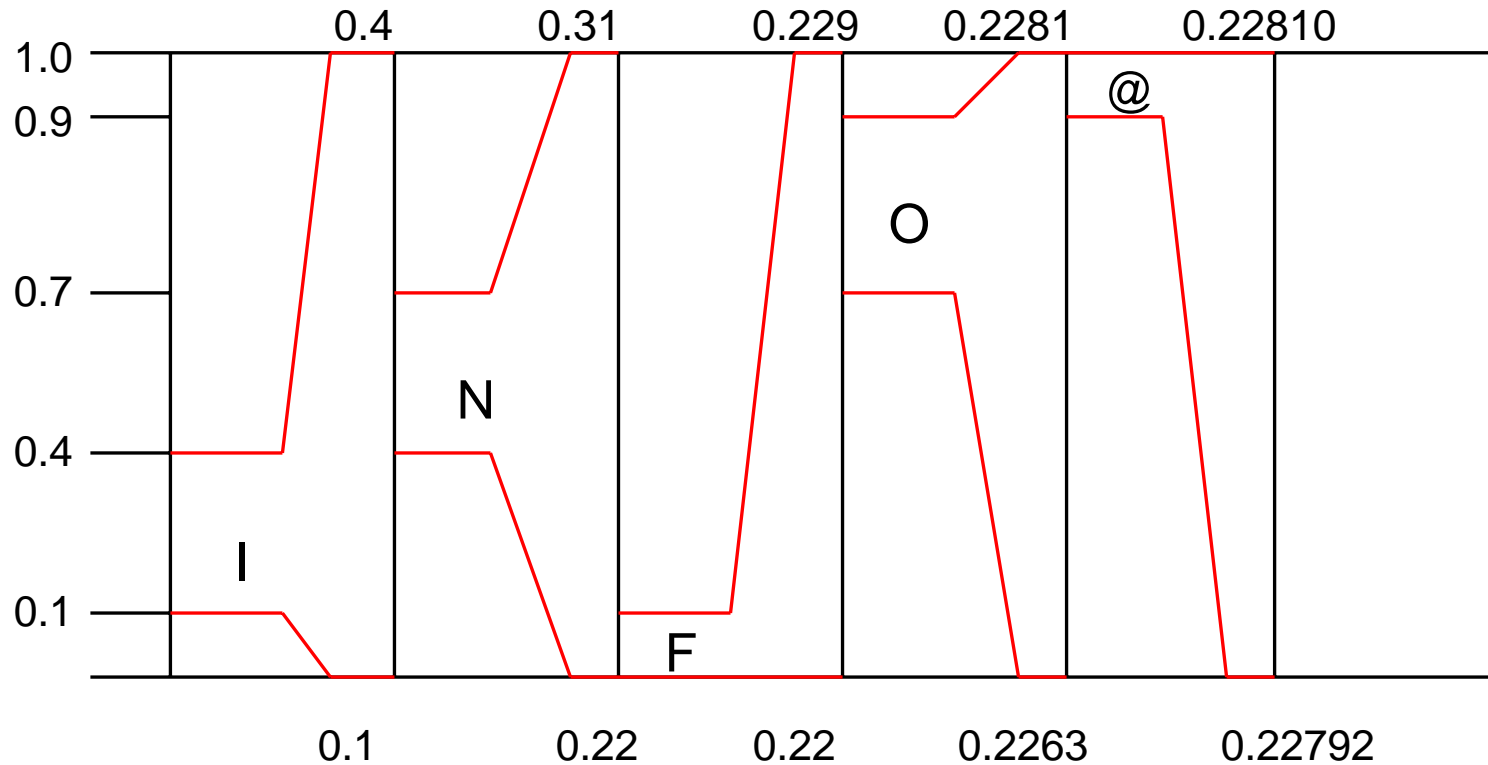
Idee:

Stelle Symbole und Symbolfolgen als Intervall $[l, r)$ bzw. als Element daraus dar.

Beispiel:

Symbol	$p(\text{Symbol})$	Intervall
F	0.1	$[0.0, 0.1)$
I	0.3	$[0.1, 0.4)$
N	0.3	$[0.4, 0.7)$
O	0.2	$[0.7, 0.9)$
@	0.1	$[0.9, 1.0)$

Arithmetische Codierung



$$[x, y) \xrightarrow{[u,o)} [x + (y - x)u, x + (y - x)o)$$

Arithmetische Codierung

Decodierung durch Umkehrung:

$$[v, w) \xrightarrow{[u, o)} \left[\frac{(v - u)}{(o - u)}, \frac{(w - u)}{(o - u)} \right)$$

Beispiel:

$$\begin{array}{l} [0.22792, 0.2281) \\ \xrightarrow{[0.1, 0.4):I} [0.4264, 0.427) \\ \xrightarrow{[0.4, 0.7):N} [0.088, 0.09) \\ \xrightarrow{[0.0, 0.1):F} [0.88, 0.9) \\ \xrightarrow{[0.7, 0.9):O} [0.9, 1.0) \\ \xrightarrow{[0.9, 1.0):@} [0.0, 1.0) \end{array}$$

Arithmetische Codierung

Es reicht natürlich ein Wert aus dem Intervall, z.B. der untere.

Die Gesamtnachricht wird zur Codierung in Teilzeichenfolgen zerlegt, die durch Stoppzeichen beendet werden.

Nachteil: Statisches Verfahren, passt sich nicht wechselnden Wahrscheinlichkeiten an.

Nachteil: Aufwendige Berechnungen.

Einfaches verlustfreies Kompressionsverfahren



Ersetze häufig auftretendes Muster durch kurzes
Codewort, verwende Wörterbuch für die Codeworte

abracadabracadabra
1 2 2 1

abra 1
cad 2

Wörterbuchbasiertes Kompressionsverfahren:

statisch: Wörterbuch wird vor Codierung festgelegt
und bleibt unverändert

dynamisch: Wörterbuch passt sich dem zu
komprimierenden Text dynamisch an

Lempel-Ziv: zip, TIFF (Image File Format)

Lempel-Ziv-Welch: Compress in Unix

Lempel-Ziv Idee:

Baue das Wörterbuch simultan mit der Kodierung des Textes auf;
anfangs seien für jeden Buchstaben des Alphabets Codenummern
im Wörterbuch

Einfaches verlustfreies Kompressionsverfahren



Kodierung von: a b a b c b a b a b

Wörterbuch

<u>Eintrag</u>	<u>#</u>	<u>Ausgabe</u>
a	1	
b	2	
c	3	

Einfaches verlustfreies Kompressionsverfahren



Decodierung von: 1 2 4 3 5 8

Wörterbuch

<u>Eintrag</u>	<u>#</u>	<u>Ausgabe</u>
a	1	
b	2	
c	3	

LZW Kodierung

Algorithmus Lempel-Ziv-Welch

Input: Ein Text $T=t_1\dots t_n$ über dem Alphabet Σ

Output: Die LZW Kodierung von T

Initialisiere Wörterbuch D mit Zeichen aus Σ ;

Initialisiere string $s = t_1$;

while *noch nicht alle Zeichen von T gelesen* **do**

 lies nächstes Zeichen c ;

if $s+c$ ist Anfangsstück eines Eintrags in D

then $s = s+c$ /* *bestimme aktuellen Match* */

else { gib $Codewort(s)$ aus;

 trage $s+c$ in D ein;

$s = c$ }

end while;

gib $Codewort(s)$ aus

LZW Decodierung

Input: Sequenz von Codewörtern

Output: Zeichenfolge über dem Alphabet Σ

Initialisiere Wörterbuch D mit Zeichen aus Σ ;

Lies lcw und gib Buchstaben $string(lcw)$ aus, den lcw codiert;

while *noch nicht alle Codewörter gelesen* **do**

 lies nächstes Codewort acw ;

if acw ist in D **then**

 { trage $string(lcw) + firstchar(string(acw))$ in D ein;
 gibt $string(acw)$ aus }

else /*Spezialfall */

 { trage $string(lcw) + firstchar(string(lcw))$ in D ein;
 gebe $string(lcw) + firstchar(string(lcw))$ aus }

$lcw = acw$

end while

Lempel-Ziv Kodierung: Beispiel

$T = \text{COCOA AND BANANAS}$

Wörterbuch D (anfangs)

m	$\#(m)$	m	$\#(m)$
A	000 001	O	001 111
B	000 010	:	:
C	000 011	S	010 011
D	000 100	:	:
:	:	Z	011 010
N	001 110		011 011

Lempel-Ziv Kodierung: Beispiel

$T = \text{COCOA AND BANANAS}$

m	#(m)	add to D	T
C	000 011	CO 011 100	OCOA AND BANANAS
O	001 111	OC 011 101	COA AND BANANAS
CO	011 100	COA 011 110	A AND BANANAS
A	000 001	A 011 111	AND BANANAS
	011 011	A 100 000	AND BANANAS
A	000 001	AN 100 001	ND BANANAS
N	001 110	ND 100 010	D BANANAS
D	000 100	D 100 011	BANANAS
	011 011	B 100 100	BANANAS

Lempel-Ziv Kodierung: Beispiel

m	$\#(m)$	add to D		T
B	000 010	BA	100 101	ANANAS
AN	100 001	ANA	100 110	ANAS
ANA	100 110	ANAS	100 111	S
S	010 011			

(T) = 000 011 | 001 111 | 011 100 | 000 001 | 011 011 | ...

Lempel-Ziv Eigenschaften

- Wörterbuch passt sich dynamisch an die zu komprimierende Zeichenkette an, d.h. es enthält schließlich die am häufigsten vorkommende Zeichenkette
- Wörterbuch (Code Tabelle) muss nicht übertragen werden. Bekannt sein muss nur die Anfangstabelle, alles weitere wird beim Decodieren dynamisch erzeugt.
- Codieren und Decodieren ist in linearer Zeit möglich
- LZ führt i.a. zu höheren Kompressionsraten als Huffman