Kompressionsverfahren für Texte

Prof. Dr. S. Albers

Prof. Dr. Th. Ottmann

Zeichenkettenverarbeitung

- Suche in Texten, Textindizes
- Mustererkennung (Pattern-Matching)
- Verschlüsseln
- Komprimiern
- Analysieren (Parsing)
- Übersetzen

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:

- Huffmann-Code
- Lauflängen-Codierung
- Arithmetische Codierung
- Lempel-Ziv Codierung

Verlustbehaftete Kompressionsverfahren

Gegensatz: Verlustbehaftete Kompression für Bilder, Audio, Video

Beispiele:

- JPEG
- MPEG
- MP3

Arithmetische Codierung

Ziel:

Bits der codierten Nachricht umgekehrt proportional zur Wahrscheinlichkeit der Nachricht.

Wahrscheinlichkeit groß - # Bits klein

Wahrscheinlichkeit klein - # Bits groß

Arithmetische Codierung

Beispiel: Huffmann Code

Symbol	p(Symbol)	Code
A	0.01	0
B	0.99	1

AAAA → 1111

BBBB → 0000

Arithmetische Codierung

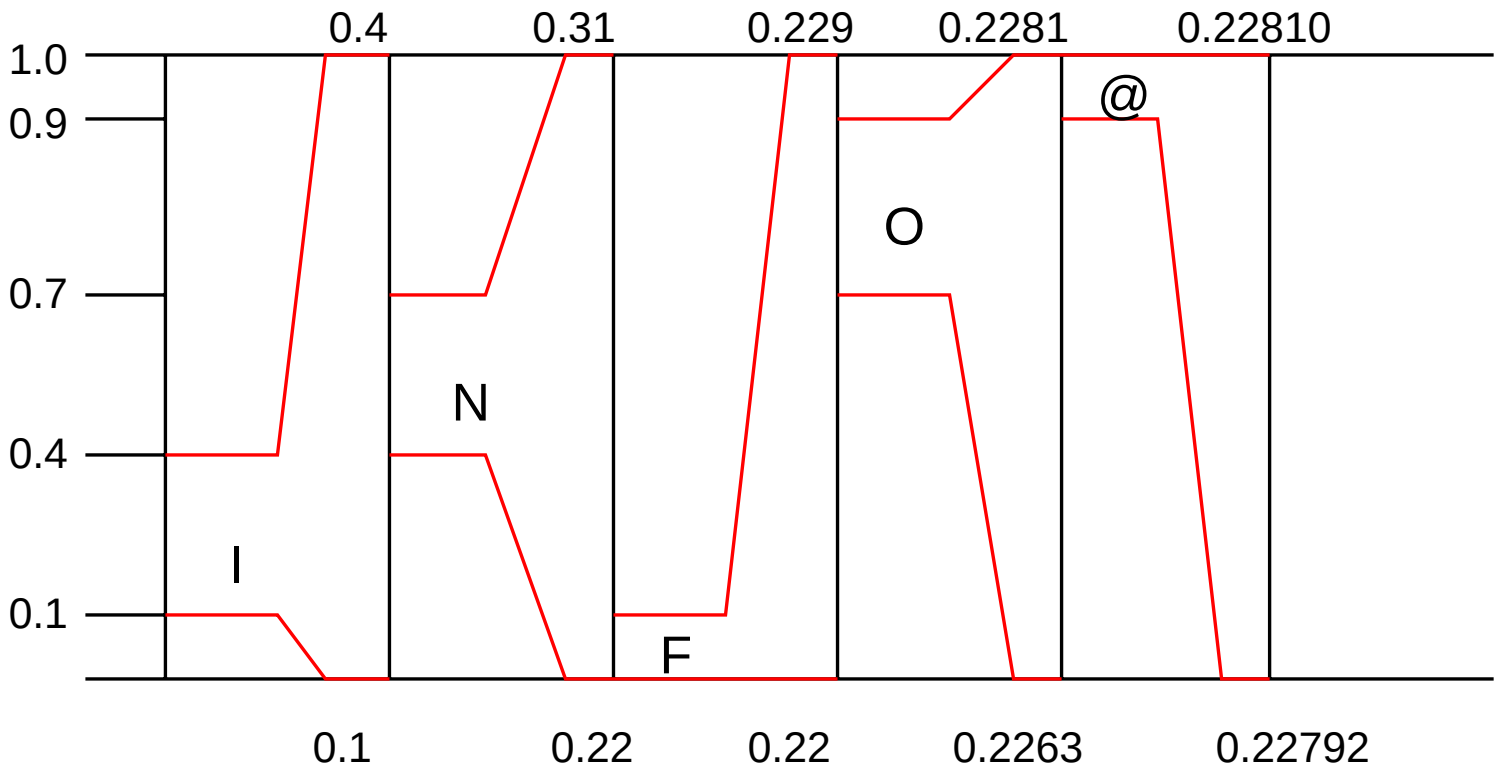
Idee:

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

Beispiel:

Symbol	p(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:

$$[v,w) \xrightarrow{[u,o)} [(v-u)/(o-u),(w-u)/(o-u))$$

Beispiel:

$$[0.22792,0.2281)$$

$$\xrightarrow{[0.1,0.4):I} [0.4264,0.427)$$

$$\xrightarrow{[0.4,0.7):N} [0.088,0.09)$$

$$\xrightarrow{[0.0,0.1):F} [0.88,0.9)$$

$$\xrightarrow{[0.7,0.9):O} [0.9,1.0)$$

$$\xrightarrow{[0.9,1.0):@} [0.0,1.0)$$

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

abra	1
cad	2

Einfaches verlustfreies Kompressionsverfahren

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

Einfaches verlustfreies Kompressionsverfahren

Lempel-Ziv Idee:

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

Einfaches verlustfreies Kompressionsverfahren

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

Wörterbuch

Eintrag	#
a	1
b	2
c	3

Ausgabe

Einfaches verlustfreies Kompressionsverfahren

Decodierung von: 1 2 4 3 5 8

Wörterbuch

Eintrag	#
a	1
b	2
c	3

Ausgabe

LZW Kodierung

Algorithmus Lempel-Ziv-Welch
Input: Ein Text $T=t_1 \ldots t_n$ über dem Alphabet Σ
Output: Die LZW Kodierung von *T*

Initialisiere Wörterbuch *D* mit Zeichen aus Σ;
Initialisiere string *s = t1:*
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 *act*;
    if *act ist nicht in D* **then** /* Spezialfall*/
        { trage *string(lcw)* + *firstchar(string(lcw))* in *D* ein;
          gebe *string(lcw)* + *firstchar(string(lcw))* aus }
    else { trage *string(lcw)* + *firstchar(string(act))* in *D* ein;
          gibt *string(act)* aus }
    lcw = acw
end while

Lempel-Ziv Kodierung: Beispiel

T = 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 = COCOA AND BANANAS

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

Lempel-Ziv Kodierung: Beispiel

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

(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 Anfangtabelle, alles weitere wird beim Decodieren dynamisch erzeugt.

- Codieren und Decodieren ist in linearer Zeit möglich

- LZ führt i.a. zu höheren Kompressionsraten als Huffmann