Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering

Kurs Information und Codierung

Datenkompression: Lauflängencodierung (run length encoding) und Huffman-Codierung

Studiengang IT
21.08.2017

https://olat.zhaw.ch/olat/...

Autoren: Prof. Dr. Marcel Rupf, Kurt Hauser

Dozenten: Dr. Jürg Stettbacher, Kurt Hauser

Lernziele

- **Die Studierenden können verlustlose und verlustbehaftete Kompressionsverfahren unterscheiden**
- **Sie kennen die Bedeutung des Quellencodierungstheorems**
- **Sie kennen das Verfahren der Lauflängencodierung (RLE; run length encoding) und können selbstständig eine Codierung vornehmen**
- **Die Studierenden kennen das Huffman-Verfahren und können es in Beispielen anwenden.**

Datenkompression

Kompressionsverfahren

- **Verlustlose Verfahren**

  -Allgemeine Verfahren
  -Jede Art von Daten ist komprimierbar; Kompressionsrate aber abhängig von der Art der Daten (Text, Bild, Ton, etc)
  -Ursprüngliche Daten können exakt rekonstruiert werden
  -Beispiele: pkzip,winzip,compress

- **Verlustbehaftete Verfahren**

  -Anwendungsspezifisch
  -Kompressionsrate oftmals wählbar
  -Ursprüngliche Daten können nur approximativ rekonstruiert werden
  -Beispiele: JPEG, MP3

Quellencodierungstheorem

Voraussetzung

X[0], X[1], X[2], ... mit **H** bit / Symbol Information

Quelle ohne Gedächtnis (DMS): **H** = H(X)
Quelle mit Gedächtnis: Entropierate **H** = $\lim_{n->\infty} H(X[0],...,X[n-1]) / n$

Quellencodierungstheorem (Shannon, 1948)

Die Quelle kann verlustlos codiert werden, solange die Coderate $R \geq H$. Umgekehrt, wenn $R < H$, kann die Quelle auf keinen Fall verlustlos codiert werden.

Das Theorem gibt Bedingungen für die Existenz von Quellencodes an. Es liefert aber keine Algorithmen.

Run-Length-Encoding (RLE): Text

Grundidee der Lauflängencodierung

oft Zeichenketten mit Folgen gleicher Zeichen

d … d ersetzen mit (n,d) n: Repetitionscounter, d: Datensymbol

Unterscheidung Zahl und Repetitionscounter?

escape-Charakter vor Repetitionscounter stellen, z.B. (@, n, d)

(escape, counter, data) normalerweise 3 Bytes lang

nur runs >3 durch (@, n, d) ersetzen

Unterscheidung @ und escape-Charakter?

z.B. @-Zeichen durch (@, 1, @) ersetzen

Beispiel

AAAAAABBBAAAACAABBBBBBBBCCCCC...

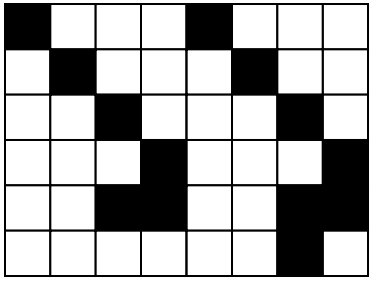
@6ABBB@4ACAA@8B@5C…

Run-Length-Encoding (RLE): Bild

RLE ist ein natürlicher Kandidat für Bildkompression

- benachbarte Pixels haben oft gleiche Farbe bzw. Intensität
- Pixelgrösse: binär (s/w), Byte (grau), 3 Bytes (RGB)

Beispiel: schwarz-weiss-Bild



Auflösung, gestartet wird mit weiss!

6,8,0,1,3,1,4,1,3,1,4,1,3,1,4,1,3,1,2,2,2,2,6,1,1.

oder: zeilenweise, z.B. Zeile 1: 0,1,3,1,3,eol

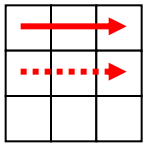
Beispiel: Graustufen-Bildzeile

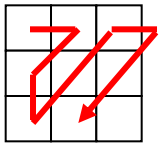
12,12,12,12,12,12,12,12,12,35,76,112,67,87,87,87,87,5,5,5,5,5,5,1,…

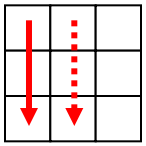
escape-Charakter ist z.B. 255 (1 Graustufe weniger):

255,9,12,35,76,112,67,255,4,87,255,6,5,1…

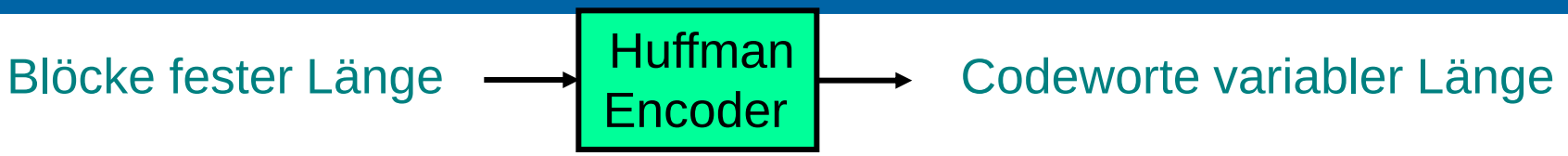
RLE-Scanning

Fax (RLE+Huffmann): 

JPEG: 

?: 

Huffman-Codierung



Grundidee

Häufig vorkommende Blöcke werden mit **kurzen** Codeworten codiert.

Problem mit Codeworten variabler Länge: Synchronisation!

Es werden **präfixfreie** Quellencodes bevorzugt.

Kein Codewort ist ein Präfix (Vorsilbe) eines anderen Codeworts.

Eindeutig und **sofort** (ohne Verzögerung) decodierbar.

Beispiel

Der Code { [0], [10], [110], [1110], [1111] } ist präfixfrei.

Detektion Codewortende, wenn „0“ kommt oder nach vier „1“

Huffman-Codes sind optimal

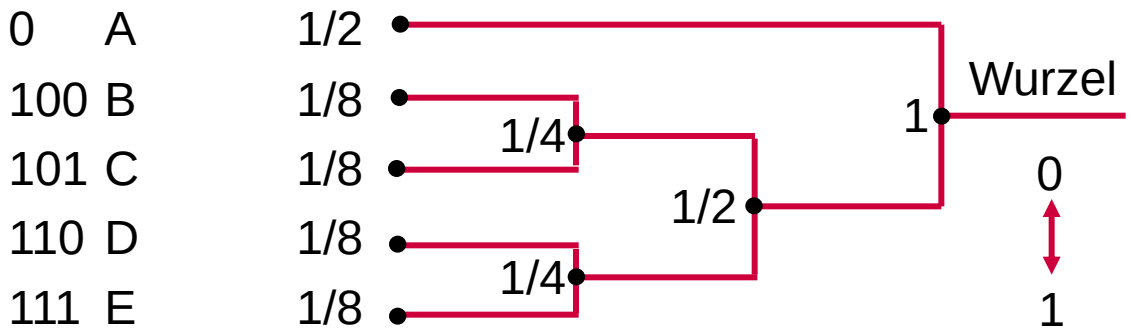
Haben minimale mittlere Codewortlänge unter allen präfixfreien Codes.

Huffman-Algorithmus

D. Huffman (1952)

1. Symbole nach Wahrscheinlichkeiten ordnen und Knoten eines Baums zuweisen
2. zwei Symbole mit kleinster Wahrscheinlichkeit in neuem Symbol zusammenfassen, neuer Knoten hat Summe der Wahrscheinlichkeiten
3. wenn nur noch 2 Symbole bzw. Knoten übrig bleiben, 4. sonst 2.
4. von der Wurzel aus bei jeder Verzweigung nach oben eine „0“ und nach unten eine „1“ eintragen (auch umgekehrt möglich)

Beispiel: DMS mit H(X) = 2 bit / Symbol, Alphabet *A* = {A,B,C,D,E}



R = E[L]
= (1/2)·1+4·(1/8)·3
= 2 bit / Symbol

Code ist optimal!

Huffman-Codierung (2)

Rate bzw. mittlere Codewortlänge

$$R = E[L] = \sum_{m=1}^{M} P_X(x_m) \cdot L(x_m)$$

L(x): Länge des dem Symbolwert x zugeordneten Codeworts

Codierung von n Symbolen einer DMS

$H(X) \leq R < H(X) + 1/n$

gilt auch für Quellen mit Gedächtnis, wenn H die Entropierate darstellt
=> R -> H(X) von oben, Gleichheit wenn $P_X(x_m)=2^{-\alpha}$

M-wertige Huffman-Codes => Algorithmus einfach erweiterbar

Nachteile

1. Huffman-Codes hängen stark von der Quellenstatistik ab
2. Quellenstatistik muss im voraus bekannt sein (ev. zuerst „messen“)
3. Komplexität wächst exponentiell mit der Blocklänge n