

## Quellenkodierung

### a) Redundanz

Ein wichtiges Mass in der Informationstheorie ist der Anteil einer Nachricht der keine Information erhält, die Redundanz  $R$  (Überfluss). Diese erhält man, indem man die relative Entropie von 1 abzieht. Sie gibt den Teil des Aufbaus der Nachricht an der nicht durch die Wahlfreiheit der Quelle, sondern eher von angenommenen statistischen Regeln bestimmt wird. Diesen Teil der Nachricht könnte man weglassen und die Nachricht wäre im Wesentlichen immer noch vollständig oder könnte vervollständigt werden. Die Redundanz ist somit die Grundlage für die Datenkompression.

In einem Binärcode gibt die Redundanz direkt an um wieviele bit die verschlüsselte Nachricht die minimale Anzahl benötigter Zeichen überschreitet.

*Beispiel:* Entropie der deutschen Sprache

Der mittlere Informationsgehalt der Summe aller Buchstaben inkl. Zwischenraum der deutschen Sprache ist 4,9bit, wenn die Buchstaben und der Zwischenraum gleichverteilt sind. Berücksichtigt man jedoch die Wahrscheinlichkeitsverteilung der Buchstabenfolgen in Tabelle 1 und die Häufigkeiten der Di- und Trigramme in Tabelle<sup>1</sup> 2 und 3, so erhält man  $H$  von nur 1,6bit. Die Redundanz der deutschen Schriftsprache ist somit  $4,9\text{bit} - 1,6\text{bit} = 3,3\text{bit}$ . Dies bedeutet, dass ein Text auch dann noch lesbar wäre wenn jedes zweite Zeichen fehlen würde.

er	409	ge	147
en	400	es	140
ch	242	ne	122
de	227	un	119
ei	193	st	116
nd	187	re	112
te	185	he	102
in	168	an	102
ie	163	be	101

Tabelle 2: Die häufigsten Digramme der deutschen Sprache (Häufigkeiten in ‰)

ein	122	sch	66	ind	46	sse	39	nic	31
ich	111	cht	61	enw	45	aus	36	sen	31
nde	89	den	57	ens	44	ers	36	ene	30
die	87	ine	53	ies	44	ebe	35	nda	30
und	87	nge	52	ste	44	erd	33	ter	30
der	86	nun	48	ten	44	neu	33	ass	29
che	75	ung	48	ere	43	nen	32	ena	29
end	75	das	47	lic	42	rau	32	ver	29
gen	71	hen	47	ach	42	ist	31	wir	29

Tabelle 3: Die häufigsten Trigramme der deutschen Sprache (Häufigkeiten in ‰)

Die Redundanz an einem deutschen Text soll im folgenden Beispiel verdeutlicht werden, durch schrittweise Reduzierung der Redundanz bleibt der Text immer noch lesbar:

Wird die Redundanz reduziert ist das Lesen viel mühsamer  
 WIRD DIE REDUNDANZ REDUZIERT IST DAS LESEN VIEL MÜHSAMER  
 WIRDDIEREDUNDANZREDUZIERTISTDASLESENVIELMÜHSAMER  
 WI DD ER DU DA ZR DU IE TI TD SL SE VI LM HS ME

### b) Redundanzmindernde Quellenkodierungsverfahren

Die Quellenkodierung bezweckt eine Datenreduktion, auch Datenkompression genannt, mit dem Ziel, Speicher- oder Übertragungskapazitäten ökonomisch zu nutzen. Unter Bildkodierung wird die Quellenkodierung digitaler Bilder verstanden. In der Bildkodierung werden redundanzreduzierende und irrelevanzreduzierende Verfahren angewendet. Ein Verfahren heisst redundant, wenn der Dekodierer (Empfänger) aus den empfangenen Daten und der Kenntnis des Kodierverfahrens das Bild fehlerfrei rekonstruieren kann. Ein Verfahren zur Irrelevanzreduktion ist irreversibel, d.h. das dekodierte Bild entspricht nicht exakt dem Original sondern ist mit einem Informationsverlust behaftet, was sich in einem Qualitätsverlust bemerkbar machen kann. Die Redundanzreduktion ist unabhängig vom Empfänger, sie hängt allein von den statistischen Eigenschaften der Quelle ab. Die Irrelevanzreduktion orientiert sich an den physiologischen Eigenschaften des Empfängers und an seinen Bedürfnissen. Die heute realisierten Verfahren zur Bilddatenkompression enthalten untrennbar verknüpft Redundanz und Irrelevanz. Verfahren zur Redundanzreduktion sind z. B. Huffman-Kodierung und die Lauflängen-Kodierung.

Verfahren zur Redundanz – und Irrelevanzreduktion sind die Prädikationsverfahren (Differenz-Plus-Code Modulation (DPCM) und Transformationsverfahren (lineare Transformations-Kodierung mit diskreten Kosinusfunktionen (DCT)).

<sup>1</sup> [3] Friedrich L. Bauer: Entzifferte Geheimnisse. 2. erweiterte Auflage, Springer Berlin-Heidelberg, 1997

## Huffman-Kodierung

Das Huffman-Verfahren kodiert häufig vorkommende Zeichen mit wenigen und seltene Zeichen mit mehr Bits. Zur Festlegung der jeweiligen Bit-Kodierungen sind die Zeichenhäufigkeiten zu ermitteln. Dafür gibt es drei Wege:

### 1. Statisches Verfahren

Hier wird eine einmal berechnete Häufigkeitsverteilung im Kompressor und Expander benutzt und zwar unabhängig von den zu übertragenden Daten. Der Vorteil besteht darin, dass sich der Code sowohl im Kompressor als auch im Expander „fix programmieren“ lässt, was eine schnelle Verarbeitung ermöglicht. Andererseits ist eine hinreichend präzise Voraussage von Zeichenhäufigkeiten bei universellen Programmen kaum möglich. Daher finden diese Verfahren in der Praxis ihre Anwendung höchstens für Hilfsfunktionen. LHarc etwa verwendet statische Huffman-Kodierung nicht zur Übertragung der eigentlichen Daten, sondern nur zur Nachbearbeitung von (bei anderen Verfahren entstehenden) Offset und Längenangaben.

### 2. Dynamisches Verfahren

Bei diesem Verfahren wird die Häufigkeitsverteilung für jede Datei neu berechnet. In einem ersten Durchlauf ermittelt der Kompressor die Zeichenhäufigkeiten und konstruiert daraus eine entsprechende Code-Tabelle, in einem zweiten Lauf komprimiert er die Datei dann gemäss dieser Code-Tabelle. Ein solches Verfahren kommt beispielsweise innerhalb von PKZIP zur Nachkodierung literal übertragener Bytes zum Einsatz. Da das Kodierungsschema sich erst im Laufe der Kompression ergibt, muss man neben den eigentlichen Daten auch die dazugehörige Code-Tabelle übertragen.

### 3. Adaptive Verfahren

Adaptive Verfahren versuchen die Vorteile der beiden obigen Verfahren miteinander zu verbinden. Sie verändern das Codierungsschema im Verlauf der Kompression anhand der gelesenen Daten. So verwenden sie zu Beginn der Kompression einen festen Code und bestimmen dann jeweils nach dem Verarbeiten einer vorgegebenen Anzahl von Bytes anhand der inzwischen ermittelten Häufigkeiten einen neuen. Der Expander kennt den Anfangscode und kann damit die ersten Zeichen dekodieren. Gleichzeitig berechnet er, wie der Kompressor, die Häufigkeiten der übertragenen Zeichen und daraus den nächstfolgenden Code.

## Huffman-Baum

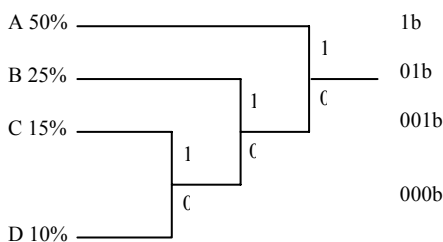
Stehen die Häufigkeiten der vorkommenden Zeichen fest, können die Bitkombinationen in passenden Längen verteilt werden, wobei seltene Zeichen besonders viele Bits erhalten. Am Beginn stehen die beiden seltensten. Sie bekommen eine 0b beziehungsweise 1b. Nun fasst man beide zusammen und ersetzt sie in der Aufstellung der Häufigkeiten durch ihre Summe. Dann werden wieder die beiden seltensten gesucht, eins um 0b das andere um 1b erweitert und die beiden zusammengefasst. Dieses Verfahren endet, wenn nur noch ein Zeichen übrig ist. Bekommt eine Zusammenfassung von Zeichen eine 0b oder 1b an ihren Code gehängt, so bedeutet das, dass die Codes aller zugehörigen Zeichen um dieses Bit verlängert werden.

Die Ermittlung der Codes aus den Häufigkeiten der Zeichen führt zu einem Huffman-Baum Dieser Baum dient zum Kodieren und Dekodieren nach Huffman.

Beim Kodieren geht man vom Zeichen aus zur Wurzel und erhält so den Binärcode, der für das Zeichen stehen soll. Die Länge des Codes ist die Länge des zurückgelegten Weges.

Das Dekodieren beginnt an der Wurzel mit dem Einlesen eines Bits. Je nach Inhalt verzweigt man im Baum nach der einen (1b) oder anderen (0b) Seite, liest das nächste Bit und folgt wieder entsprechend der nächsten Verzweigung. Am Ende, gewissermassen auf einem Blatt des Baumes, steht das gesuchte Zeichen.

*Beispiel:*



Bei der Ermittlung der Huffman-Codes für die vier Zeichen von A bis D erhält man folgenden Baum, wenn die Häufigkeiten, mit denen sie vorkommen, 50%, 25%, 15% und 10% sind.

Schematischer Aufbau eines Huffman-Baums:

C und D sind die seltensten Zeichen. Also beginnen sie mit 1b beziehungsweise mit 0b, werden zusammengefasst und haben zusammen 25% Häufigkeit. Jetzt sind B und (C + D) mit je 25% die seltensten. Wiederum wird 1b beziehungsweise 0b angehängt und danach B und (C + D) zusammengefasst. Dieser Vorgang wird solange wiederholt, bis auch das letzte Element mit der grössten Häufigkeit in den Baum eingebunden wurde.

## Laufmängenkodierung (Run-Length-Encoding)

Die Laufmängenkodierung ist ein sehr einfaches Verfahren - nicht besonders effizient, aber dafür sehr schnell. Zeichen, die mehrmals hintereinander vorkommen, werden nur einmal zusammen mit einem Zähler kodiert. Da auch der Zähler Platz kostet, bleiben zwei- oder dreimal vorkommende Zeichen uncodiert. Bei Grafikdateien ist diese Komprimierung oft recht wirksam. Strukturlose Flächen werden damit durch wenige Bytes ersetzt.

Es gibt zwei Wege den Zähler zu kodieren:

### 1. Die 7-Bit-Laufmängenkodierung

Wenn die Zeichenbytes aus dem Bereich von 00h-7Fh sind, ist das für die Kodierung des Zählers nutzbar, indem man mehrfach vorkommende Bytes durch den Zähler mit gesetztem achtem Bit gefolgt vom eigentlichen Zähler kodiert. Dieses Verfahren ist besonders für Textdateien geeignet, in denen selten Bytes mit gesetztem achtem Bit vorkommen. Solche Sonderzeichen sind mit vorangestelltem Zähler 81h zu kodieren.

### 2. Die Standard-Laufmängenkodierung

Die Standard-Laufmängenkodierung kodiert mehrfach vorkommende Bytes in drei Bytes: das ursprüngliche Byte, 90h, Zähler. Da der maximale Zählerstand 255 sein kann, müssen grössere Vorkommen unterteilt werden. Weiterhin ist für 90h eine besondere Kodierung nötig, da dieses Byte normalerweise bedeutet: „Zähler folgt“. 90h wird daher als 90 00h kodiert. Einige Kompressionsprogramme wenden diese Laufmängenkodierung auf die Originaldaten an, bevor die eigentliche Kompression zum Zuge kommt.