

# SEMINARARBEIT

im W-Seminar

WM Kryptologie

**im Fach**

Mathematik

**Thema:**

Quellenkodierung I - Huffman-/Shannon-Fano-Kodierung

**Verfasser / -in:** Kai Rasmussen

**Kursleiter / -in:** Oliver Manger

**spät. Abgabetermin:** Di., 11.11.2025

**Tag der Abgabe im  
OS-Büro:** \_\_\_\_\_

**Bewertung:**  Punkte für die W-Seminararbeit

(einfache Wertung, Ergebnis liegt zwischen 0 und 15 Punkten)

Punkte für die Präsentation

**Gesamtnote:**  Punkte

(Arbeit dreifach gewertet, Präsentation einfach gewertet, Summe geteilt  
durch 2, Ergebnis gerundet; Gesamtergebnis liegt zwischen  
0 und 30 Punkten)

Besprochen am: \_\_\_\_\_

\_\_\_\_\_  
Unterschrift der Kursleiterin / des Kursleiters

# Erklärung

Hiermit erkläre ich, dass ich die vorliegende Seminararbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die im Literaturverzeichnis angeführten Quellen und Hilfsmittel verwendet habe. Insbesondere versichere ich, dass alle Stellen der Seminararbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen sind, durch Angabe der Herkunft kenntlich gemacht wurden. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie Quellen aus dem Internet.

---

Ort, Datum

---

Unterschrift der Schülerin / des Schülers

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>4</b>
1.1	Zielsetzung . . . . .	5
1.2	Grundlegende Prinzipien . . . . .	5
1.2.1	Unterschied zwischen verlustfreie und verlustbehaftete Kompressi- onssysteme . . . . .	5
1.2.2	Informationsentropie . . . . .	5
1.2.3	Unterschied zwischen Higher-order und Zeroth-order Systeme . . .	6
1.2.4	Unterschied zwischen statische und adaptive Kodierungssysteme . .	6
<b>2</b>	<b>Grundlegende Kompressionssysteme</b>	<b>7</b>
2.1	Shannon Codierung . . . . .	7
2.1.1	Beispiel: Statisches Shannon Kodierung . . . . .	7
2.1.2	Adaptive Shannon Kodierung . . . . .	7
2.2	Fano Codierung . . . . .	9
2.2.1	Beispiel: Statisches Fano Kodierung . . . . .	9
2.2.2	Adaptive Fano Codierung . . . . .	10
2.3	Huffman Codierung . . . . .	11
2.3.1	Beispiel: Statisches Huffman Codierung . . . . .	11
2.3.2	Adaptive Huffman Codierung . . . . .	12
<b>3</b>	<b>Higher-order Systeme</b>	<b>13</b>
3.1	Dictionary Based Kodierung . . . . .	13
3.2	Grammatikbasierte Codierung . . . . .	13
<b>4</b>	<b>Vergleich der ursprünglichen Kompressionsalgorithmen</b>	<b>14</b>
<b>5</b>	<b>Schluss</b>	<b>15</b>

# 1 Einleitung

**Abstrakt - Der Vergleich zwischen den ursprünglichen Datenkompressionsalgorithmen, den Huffman- und Shannon-Fano-Kodierungssystemen, sowie eine Betrachtung der verschiedenen Verbesserungen und Anwendungen verlustfreier Kompressionssalgorithmen.**

Verlustfreie Datenkompressionssysteme sind für unser modernes Leben von enormer Bedeutung. Ohne diese Systeme hätten wir dieses technologische Niveau nicht erreichen können. Wie der Name schon sagt, handelt es sich dabei um Systeme, die den Speicherplatzbedarf von Daten reduzieren, ohne dabei Informationen zu verlieren. Die Grundlage verlustfreier Datenkomprimierungssysteme bilden die Codierungssysteme von Huffman, Shannon und Fano. Seit der Einführung dieser Systeme gab es jedoch zahlreiche Verbesserungen und unterschiedliche Ansätze für das Problem der verlustfreien Datenkomprimierung. Dieses Prinzip ist eng mit der Geschichte der Kryptologie verbunden. Bereits im 4. Jahrhundert v. Chr. können wir in Griechenland beobachten, wie Polybios von einem verbesserten Kommunikationssystem berichtet, das von Aeneas Tacticus eingeführt wurde. Dabei verwendeten zwei Personen, die sich in Sichtweite befanden, eine Fackel und zwei identische, beschriftete Gefäße, um in Kriegszeiten schneller zu kommunizieren<sup>1</sup>. Hier verwendeten sie Wasser und Gravuren in den Gefäßen, wobei eine bestimmte Wassermenge eine bestimmte Nachricht darstellte. Dies ist ein frühes Beispiel für eine kryptographische Chiffre, bei der eine bestimmte Eingabe, hier die Wassermenge, zu einer bestimmten Klartextnachricht entschlüsselt werden kann<sup>2</sup>. In ähnlicher Weise wurden viele Versionen des optischen Semaphors verwendet, wobei vordefinierte Interpretationen bestimmter Positionen verwendet wurden, um eine vorab festgelegte Nachricht zu übermitteln. Einer der Nachteile beider Systeme besteht darin, dass die übertragbaren Informationen begrenzt sind, da nicht jede mögliche Nachricht in eine Chiffre passt. Die Übermittlung einzelner Buchstaben würde sehr viel Zeit in Anspruch nehmen. Um die zu sendende Datenmenge zu reduzieren und gleichzeitig alle für die Erstellung einer Nachricht erforderlichen Informationen zu erhalten, entstand die Idee der verlustfreien Datenkompression. Der erste wichtige Kompressionsalgorithmus war die nach ihm benannte Kodierungssystem von Claude Shannon, das auf seinem Theorem zur rauschfreien Kodierung basiert (Englisch: noiseless coding theorem). Diese Theorem und Kompressionsalgorithmus bilden die Grundlage dessen, was später zum Gebiet der Informationstheorie werden sollte<sup>3</sup>. Etwa zur gleichen Zeit entwickelte Robert Fano das später als Fano-Kodierung bekannte Verfahren. Obwohl sich die Codierungssysteme von Shannon und Fano grundlegend unterscheiden, auch wenn sie in etwa auf die gleiche Weise funktionieren, werden sie oft zur Shannon-Fano-Codierung kombiniert, wobei meist das Codierungssystem von Fano gemeint ist. Beide Systeme sind jedoch suboptimal, wenn es um die resultierende Kompressionsgröße geht<sup>4</sup>. Das änderte sich, als David A. Huffman einen Elektrotechnik-Aufbaustudiengang zum Thema Informationstheorie bei Robert Fano besuchte. In diesem Kurs hatten die Studenten die Wahl zwischen einer Semesterarbeit und einer Abschlussprüfung. Das Thema der Semesterarbeit bestand darin, das optimale verlustfreie Komprimierungssystem für ein binäres System zu finden – genau das, was Fano und Shannon selbst zu erreichen

---

<sup>1</sup>[19] Book X pp.42-43: 44. The Improvement introduced by Aeneas Tacticus

<sup>2</sup>[9] p.32: Cipher (American English)

<sup>3</sup>[16] p.36: 2.2.1 Introduction to Information Theory

<sup>4</sup>[12] p.1: Introduction

versuchten. Huffman entschied sich, die Herausforderung anzunehmen und die Semesterarbeit zu schreiben. Das Ergebnis war eine der einflussreichsten Arbeiten auf diesem Gebiet: „A method for the construction of minimum-redundancy codes“. Er bewies, dass dies das optimale System zur Konstruktion eines solchen verlustfreien Komprimierungssystems war<sup>5</sup>. Seitdem gab es viele Verbesserungen auf diesem Gebiet, von denen wir einige im Abschnitt über higher-order Systeme und im Zusammenhang mit den adaptiven Versionen dieser ursprünglichen Systeme diskutieren werden. Diese Verbesserungen sind jedoch sehr unterschiedlich, sodass es unmöglich wäre, sie alle vorzustellen und die Wichtigkeit dieser Verbesserungen hängen stark vom Anwendungsbereich ab.

## 1.1 Zielsetzung

In der folgenden Seminararbeit werden zunächst die Funktionsweise der grundlegenden Kompressionssysteme Huffman, Shannon und Fano erklärt, und wie sie sich voneinander unterscheiden. Anschließend wird auf den Unterschied zwischen statische und adaptive Systeme eingegangen und wie die gesamten Systeme verbessert wurden und wie sie heutzutage eingesetzt werden. Danach werden die grundlegenden Kompressionssysteme anhand von Beispieldaten mit einem Java-Vergleichssystem verglichen, um herauszufinden, wie stark sich diese Systeme unterscheiden und ob es bestimmte Auffälligkeiten bei den Datenbeispielen gibt.

## 1.2 Grundlegende Prinzipien

### 1.2.1 Unterschied zwischen verlustfreie und verlustbehaftete Kompressionssysteme

Verlustfreie Kompressionssysteme sind eine Kompressionsmethode, bei der die ursprüngliche unkomprimierte Form in ihrer ursprünglichen Qualität rekonstruiert werden kann. Im Vergleich dazu sind die verlustbehaftete Kompressionssysteme, bei dem weniger relevante Informationen verworfen werden, um die Größe zu komprimieren<sup>6</sup>. Hier befassen wir uns ausschließlich mit verlustfreien Kompressionssysteme.

### 1.2.2 Informationsentropie

Die Informationsentropie beschreibt das Verhältnis zwischen den Wahrscheinlichkeiten der Eingabesequenz und der Länge des Codeworts. Bei einem optimalen verlustfreien Kompressionssystem entspricht die durchschnittliche Codewortlänge dem Wert der Informationsentropie. Dieses Konzept war eines der wichtigsten in Shannons Grundlagen der Informationstheorie. In dem Moment, in dem die durchschnittliche Codewortlänge unter den Wert der Entropie  $H$  fällt, geht Information verloren, sodass das Kompressionssystem nicht mehr verlustfrei ist.

$$H = - \sum p_i \log(p_i) \quad (1)$$

Diese Gleichung gilt, wenn die Zeichen voneinander unabhängig sind<sup>7</sup>.

---

<sup>5</sup>[14]: p.1: Introduction

<sup>6</sup>[2] p.40-41: 7 Lossy Compression Techniques

<sup>7</sup>[18] p.396-399: 7 The Entropy of an Information Source

### **1.2.3 Unterschied zwischen Higher-order und Zeroth-order Systeme**

In einem zeroth-order System wird jedes Zeichen unabhängig voneinander codiert, wobei Korrelationen zwischen den Zeichen in der Eingabesequenz nicht berücksichtigt werden. Ein higher-order System ist in der Lage, Muster über mehrere Zeichen hinweg zu erkennen und zu nutzen<sup>8</sup>.

### **1.2.4 Unterschied zwischen statische und adaptive Kodierungssysteme**

In einem statischen Kodierungssystem verfügen wir vor der Kodierung der Sequenz über Kenntnisse hinsichtlich der Wahrscheinlichkeiten der Quellsequenz. In einem adaptiven System ist die einzige Eingabe, über die der Kodierer verfügt, die Quellsequenz selbst. Das System passt dann das Kodierungsschema an die Änderungen in der Häufigkeit der Eingabezeichen an<sup>9</sup>.

---

<sup>8</sup>[16] p.40-44: 2.2.4 Higher-order Sources

<sup>9</sup>[16] p.60-61: 2.5 Adaptive Coding

## 2 Grundlegende Kompressionssysteme

### 2.1 Shannon Codierung

Auch wenn die von Claude E. Shannon vorgeschlagene Shannon-Methode aufgrund ihrer Ineffizienz hinsichtlich der resultierenden Kompressionsgröße nicht häufig verwendet wird, hat sie dennoch eine gewisse Relevanz in diesem Bereich. Dieses Kompressionssystem basiert auf einem Top-Down-Konstruktionssystem, wodurch es schneller zu berechnen ist. Dieses System liefert eine eindeutige Zahlenfolge, indem es eine kumulative Wahrscheinlichkeitsfunktion  $P_i$  verwendet und dann die ersten  $l_i$  Ziffern der  $r$ -ten Dezimaldarstellung der kumulativen Wahrscheinlichkeitsfunktion nimmt, um das entsprechende Codewort zu erstellen<sup>10</sup>.

$$P_i = \begin{cases} 0 & \text{for } i = 1 \\ P_{i-1} + p_{i-1} & \text{for } 2 \leq i \leq m \end{cases} \quad (2)$$

$$l_i = \lceil \log_r(p_i^{-1}) \rceil \quad (3)$$

#### 2.1.1 Beispiel: Statisches Shannon Kodierung

Eingabe:

- $\mathcal{S} = \{a, b, c, d, e, f, g, h\}$
- $\mathcal{P} = \{0.22, 0.20, 0.18, 0.15, 0.10, 0.08, 0.05, 0.02\}$
- $\mathcal{A} = \{0, 1\} \rightarrow r = 2$

$s_i$	$p_i$	$P_i$	$l_i$	$r$ -Darstellung von $P_i$	Code
a	0.22	0	3	0.000...	000
b	0.20	0.22	3	0.001...	001
c	0.18	0.42	3	0.011...	011
d	0.15	0.60	3	0.100...	100
e	0.10	0.75	4	0.11	1100
f	0.08	0.85	4	0.1101...	1101
g	0.05	0.93	5	0.11101...	11101
h	0.02	0.98	6	0.111110...	111110

Tabelle 1: Angepasst aus Beispiel von [7] p.1101: Table III

#### 2.1.2 Adaptive Shannon Kodierung

Travis Gagie schlägt eine Methode zur Erstellung eines adaptiven Shannon-Kodierungssystems vor. Bei dieser Methode werden die Aufgaben in Vordergrund- und Hintergrundaufgaben unterteilt. Die Vordergrundaufgabe berücksichtigt das Gewicht der vorherigen Zeichenanzahl und aktualisiert die Gewichte entsprechend dem neu eingegebenen Zeichen. Befindet sich das Zeichen noch nicht im Kodierungsschema, wird der Knoten zur Zeichenanzahl

<sup>10</sup>[16] p.52-53: 2.4.2 Shannon's Method; [18] p.58-61: 9. The Fundamental Theorem for a Noiseless Channel

hinzugefügt. Im Hintergrund wird für jedes im Vordergrund verarbeitete Zeichen ein Zeichen im Hintergrund verarbeitet. Die Gewichtung eines bestimmten Zeichens  $a$  wird dann aktualisiert, wenn das Zeichen zuvor aufgetreten ist, oder es wird mit der Gewichtung, die der Wahrscheinlichkeit des Zeichens entspricht, zum Komprimierungsbaum hinzugefügt<sup>11</sup>. Dieses System ähnelt dem adaptiven Fano-Kodierungssystem mit der Brute-Force-Methode, das später behandelt wird.

---

<sup>11</sup>[5] p.3-4: III Dynamic Shannon Coding



## 2.2 Fano Codierung

Die Fano-Kodierungsmethode basiert wie die Shannon-Kodierungsmethode auf einem Top-Down-Konstruktionssystem. Die Fano-Methode beginnt mit einem Sequenzcode  $\mathcal{S}$ , der nach absteigender Wahrscheinlichkeit in  $\mathcal{P}$  sortiert ist. Im binären System teilen wir diese Liste dann in zwei Untergruppen mit ungefähr gleicher Wahrscheinlichkeit  $\mathcal{S}_0$  und  $\mathcal{S}_1$  und damit auch  $\mathcal{P}_0$  und  $\mathcal{P}_1$ . Ihre Codes beginnen dann entweder mit 0 oder 1. Dieser Vorgang wird dann so lange wiederholt, bis die Unterteilungen den einzelnen Zeichen entsprechen. Hier ist die Codewortlänge gleich  $\log_2 m$ , wenn  $m$  eine Potenz von zwei ist. Wenn  $m$  keine Zweierpotenz ist, ist die Länge des Codeworts eine der beiden ganzen Zahlen, die  $\log_2 m$  am nächsten kommen<sup>12</sup>.

### 2.2.1 Beispiel: Statisches Fano Kodierung

- $\mathcal{S} = \{a, b, c, d, e, f, g, h\}$
- $\mathcal{P} = \{0.22, 0.20, 0.18, 0.15, 0.10, 0.08, 0.05, 0.02\}$
- $\mathcal{A} = \{0, 1\} \rightarrow r = 2$

$s_i$	$p_i$	$L_i$	Code
a	0.22		
b	0.20		
c	0.18		
d	0.15		
e	0.10		
f	0.08		
g	0.05		
h	0.02		

Tabelle 2: Angepasst aus Beispiel von [7] p.1101: Table III

---

<sup>12</sup>[4] p.5-6: Selection from N Equally Likely Choices; [16] p.55: Algorithm 3 Static Fano Encoding

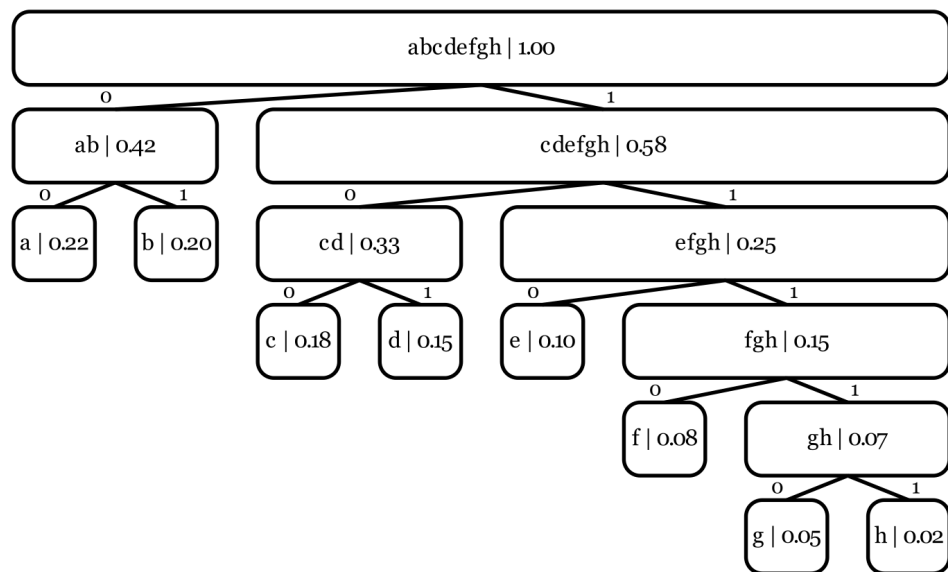


Abbildung 1: Fano Baum von Tabelle 2; Mit Manim Community v0.19.0 gemacht

### 2.2.2 Adaptive Fano Codierung

Luis Rueda schlägt zwei verschiedene Möglichkeiten vor, um ein adaptives Fano-Kodierungssystem zu erstellen: die sogenannte Brute-Force-Methode und die sogenannte Greedy-Methode. Bei der Brute-Force-Methode berechnet das System bei jeder Änderung des Zeichenzählers den entsprechenden Kompressionsbaum neu<sup>13</sup>. Dies ist natürlich ziemlich ineffizient, aber funktioniert immer um ein statisches in ein adaptives Kompressionssystem zu wandeln. Bei der Greedy-Methode beginnen wir mit einer Liste des Eingabealphabets und initialisieren die Wahrscheinlichkeiten jedes Zeichens als gleich. Immer wenn ein Zeichen codiert wird, wird ein bestimmtes Partitionierungsverfahren aufgerufen, das das Codewort ausgibt. Je nach verwendetem Codealphabet können verschiedene Partitionierungsverfahren zum Einsatz kommen. Der Einfachheit halber sagen wir einfach, dass dieses System die Wahrscheinlichkeiten auf die eine oder andere Weise optimal partitioniert, aber weitere Informationen zu verschiedenen Partitionierungsverfahren finden Sie in [15]. Das Ergebnis ist jedoch, dass wir bei der Codierung eines Zeichens in unserer Eingabesequenz nicht mehr unseren Komprimierungsbaum für jedes neue Eingabezeichen aktualisieren müssen. Es werden nur die Änderungen im Gewicht eines bestimmten Zeichens berücksichtigt und alle erforderlichen Änderungen im aktuellen Kodierungsschema<sup>14</sup>.

Beispiel

<sup>13</sup>[15] p.1659-1660: 2.1 A brute-force method for adaptive Fano coding

<sup>14</sup>[15] p.1660: 2.2 The greedy encoding algorithm

## 2.3 Huffman Codierung

In Huffmans System beginnen wir mit einem Sequenzcode  $\mathcal{S}$ , der nach absteigender Wahrscheinlichkeit sortiert ist, und weisen jedem dieser Zeichen einen Knoten in  $\mathcal{Q}$  zu.

$$p_i \geq p_{i+1} \geq \dots \geq p_m \quad (4)$$

$$q_i \leftarrow s_i, (\tau_i = p_i) \quad (5)$$

Wir nehmen dann maximal  $r$ , mindestens jedoch 2 Zeichen mit der geringsten Wahrscheinlichkeit, verbinden sie mit einem neuen Knoten und fügen diesen neuen Knoten wieder in die richtige Position entsprechend seinem Gewicht in das Array  $\mathcal{Q}$  ein.

$$q_{new} \leftarrow q_{newj}, \tau_{new} = \sum_{i=1}^j \tau_i \quad (6)$$

Man kann diesen Vorgang wiederholen, bis  $\tau_{new} = 1$  ist. Am Ende kann man den Kompressionsbaum visualisieren, indem man die Kinder jedes Knotens nachverfolgen, bis man die Quellzeichen erreicht<sup>15</sup>.

### 2.3.1 Beispiel: Statisches Huffman Codierung

Eingabe:

- $\mathcal{S} = \{a, b, c, d, e, f, g, h\}$
- $\mathcal{P} = \{0.22, 0.20, 0.18, 0.15, 0.10, 0.08, 0.05, 0.02\}$
- $\mathcal{A} = \{0, 1, 2, 3\}$

$s_i$	$p_i$	$L_i$	Code
a	0.22	1	1
b	0.20	1	2
c	0.18	1	3
d	0.15	2	00
e	0.10	2	01
f	0.08	2	02
g	0.05	3	030
h	0.02	3	031

Tabelle 3: Beispiel von [7] p.1101: Table III

In Abbildung 2 stellen die grünen Knoten die einzelnen Zeichen mit ihren entsprechenden Wahrscheinlichkeiten  $p$  dar, während die violetten Knoten die Verbindungsknoten  $q$  mit ihrem entsprechenden Gewicht  $\tau$  darstellen. Bei der Kodierung und Dekodierung verwenden wir diesen Baum als Übersetzungshilfe, wobei der am weitesten links stehende Knoten 0 ist, dann 1 usw.

<sup>15</sup>[7] p.1011: Generalization of the Method; [16] p.50: Algorithm 1 Static Huffman Encoding

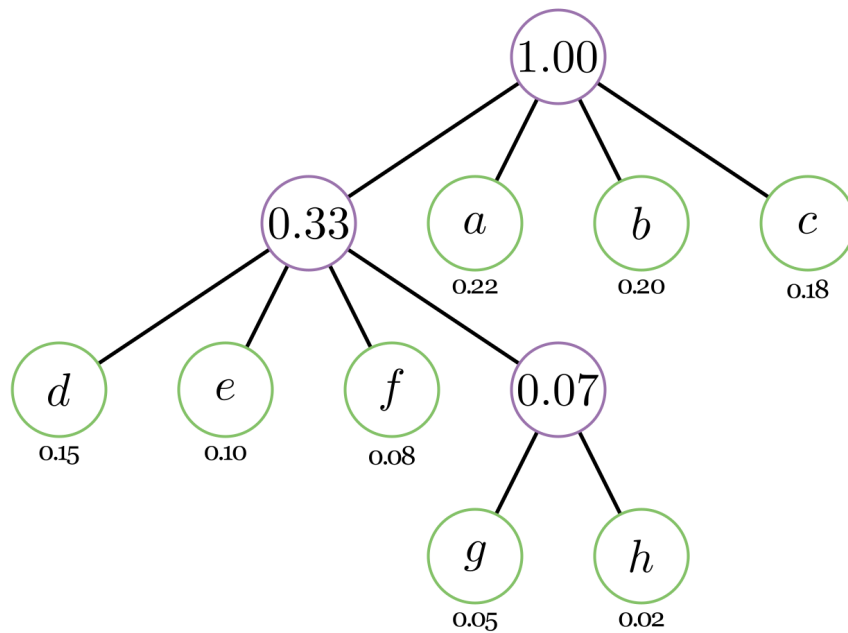


Abbildung 2: Huffman Baum von Tabelle 3; Mit Manim Community v0.19.0 gemacht

### 2.3.2 Adaptive Huffman Codierung

Die adaptive Huffman-Kodierung oder dynamische Huffman-Kodierung ist im Vergleich zur statischen Version ein System, das die Konstruktion des Huffman-Baums an Änderungen in der Anzahl der Zeichen anpasst, anstatt eine vorsortierte Liste mit vorgegebenen Wahrscheinlichkeiten zu verwenden. Diese von Robert G. Gallager vorgeschlagene Theorie basiert auf der Geschwister-Eigenschaft (Englisch: sibling property), wonach der Code für die aktuelle Zeichenanzahl optimal ist, wenn die Knoten in absteigender Reihenfolge aufgelistet werden können und jeder Knoten an seinen Geschwisterknoten angrenzt<sup>16</sup>. In einem binären System enthält jeder Knoten oder jedes Geschwisterpaar 5 verschiedene Komponenten. Zwei sind die aktuellen Zähler der untergeordneten Knoten, zwei sind Verknüpfungen zu den untergeordneten Knoten und die letzte ist eine Verknüpfung zum übergeordneten Knoten. Jede der Verknüpfungen verfügt außerdem über eine zusätzliche Eigenschaft, die angibt, ob es sich um den 0. oder den 1. untergeordneten Knoten handelt. Als Eingabe für unser adaptives System haben wir den Huffman-Baum eines um eins reduzierten gelesenen Zeichens und des nächsten Zeichens. Die Ausgabe ist dann der Huffman-Baum mit der aktualisierten Zählung. Wenn mit der Erhöhung der Zeichenanzahl festgestellt wird, dass die Anzahl die Anzahl des nächsthöheren Geschwisterpaares überschreitet, werden diese beiden Knoten vertauscht, wodurch sich der Code und der resultierende Huffman-Baum ändern<sup>17</sup>. Dieser Algorithmus wurde dann von Donald E. Knuth in seiner Arbeit über dynamische Huffman-Kodierung weiterentwickelt. Hier füllt er einige Lücken, die Gallagers adaptives System hinterlassen hat, wie z. B. die Verringerung der Zeichenanzahl<sup>18</sup>.

<sup>16</sup>[6] p.6: 2. The Sibling Property Definition

<sup>17</sup>[6] p.17-21: Adaptive Huffman Codes

<sup>18</sup>[11] p.163-165: Introduction

## 3 Higher-order Systeme

### 3.1 Dictionary Based Kodierung

Storer und Szymanski legen in ihrer Arbeit „Data Compression via Textual Substitution“ einen Grundstein für die dictionary-based Kompressionsverfahren. Ihr System führt mehrere Komponenten hinzu: externe Makroschemata, interne Makroschemata, ein Wörterbuch (dictionary) und ein Skelett. Das externe Makroschema ermöglicht die Kodierung einer Quellzeichenfolge unter Verwendung eines Wörterbuchs, eines Speichers für Referenzzeichenfolgen und eines Skeletts, einer Kombination aus Zeichen des Eingabealphabets und Zeigern auf das Wörterbuch. Die internen Makroschemata ermöglichen Zeigern auf wiederholte Abschnitte derselben Zeichenfolge<sup>19</sup>. Mit diesem System können wir den Grad der Redundanz in unserer Eingabesequenz verringern.

LZ77/Deflate

### 3.2 Grammatikbasierte Codierung

Bei der grammatikbasierten Codierung (Englisch: Grammar Based Coding) verwenden wir das Konzept der formalen Sprachen und Grammatiken aus der Informatik, wobei eine Grammatik  $G$  aus einer Liste der Terminale  $T$  oder einem Alphabet  $A$  oder  $\Sigma$ , einer Liste von Nicht-Terminale  $N$ , einer Liste von Produktionsregeln  $P$  und einem Startknoten  $s$  besteht<sup>20</sup>. Mit diesem higher-order System codieren wir nicht die Zeichenkette  $x$  selbst, sondern die Grammatik  $G_x$ . Da die Grammatik nur die Zeichenkette  $x$  konstruiert, wird sie als kontextfreie Grammatik bezeichnet<sup>21</sup>.

$$L(G_x) = \{x\} \quad (7)$$

In diesem Zusammenhang besteht das Ziel darin, die kleinstmögliche kontextfreie Grammatik für eine bestimmte Zeichenfolge  $x$  zu erstellen. Die Größe einer Grammatik  $|G_x|$  entspricht der Anzahl der Symbole auf der rechten Seite der Produktionsregeln  $P$ . Dieses Problem wird als „Smallest Grammar Problem“ (SGP) bezeichnet<sup>22</sup>. Das in [3] angegebene Beispiel lautet:

$$P = \{ \langle S \rangle \rightarrow \langle B \rangle \langle B \rangle \langle A \rangle, \langle A \rangle \rightarrow \text{a rose}, \langle B \rangle \rightarrow \langle A \rangle \text{ is} \} \quad (8)$$

Hier ist  $|G_x| = 14$  die kleinste Grammatik, um die Zeichenfolge „a rose is a rose is a rose“ zu bilden.

---

<sup>19</sup>[20] p.929-932: 2 The Model and Basic Definitions

<sup>20</sup>[16] p.80-81: 2.6.6 Grammar-Based Coding

<sup>21</sup>[10] p.2-4: Introduction

<sup>22</sup>[3] p.1-2: Introduction

## 4 Vergleich der ursprünglichen Kompressionsalgorithmen

Beispiel	ASCII-1965 <sup>23</sup>	USASCII-8 <sup>24</sup>	UTF-32 <sup>25</sup>
Wordnik <sup>26</sup>	16.57	18.93	75.74
Shakespeare <sup>27</sup>	28	43.03	172.12

Tabelle 4: Liste mit Beispielen zum Vergleich der verschiedenen Kompressionssysteme mit ihrer entsprechenden Speichernutzung in Mb.

Kompressionssystem	Prozesse <sup>29</sup>	Größe <sup>30</sup>	$\bar{L}$ Avg. L <sup>31</sup>	Prozentsatz eingespart <sup>32</sup>
Huffman	—	—	—	—
Shannon	—	—	—	—
Fano	—	—	—	—

Tabelle 5: Vergleich der verschiedenen Kompressionssysteme mit Wordnik

Kompressionssystem	Prozesse	Größe	$\bar{L}$ Avg. L	Prozentsatz eingespart
Huffman	—	—	—	—
Shannon	—	—	—	—
Fano	—	—	—	—

Tabelle 6: Vergleich der verschiedenen Kompressionssysteme mit Shakespeare

<sup>23</sup>[13] p.423-425: 23.2 ASCII-1965; also ISO/IEC 646: [8] p.6-7: Table 1 (Berechnet: Anzahl der Zeichen multipliziert mit 7)

<sup>24</sup>[13] p.431-433: 23.11 USASCII-8 (Berechnet: Anzahl der Zeichen multipliziert mit 8)

<sup>25</sup>[21] p.77: 2.5.1 UTF-32 (Berechnet: Anzahl der Zeichen multipliziert mit 32)

<sup>26</sup>[1]: Wordnik Liste öffentlich zugänglicher englischer Wörter

<sup>27</sup>[17]: Die gesammelten Werke von William Shakespeare

<sup>28</sup>Die Datei aus Project Gutenberg enthält einige Formatierungszeichen, die nicht im Standard-ASCII-1965 enthalten sind.

<sup>29</sup>Berechnungsprozesse sind relativ, verschiedene Arten von Prozessen dauern unterschiedlich lange, daher sollte diese Zahl eher vage interpretiert werden.

<sup>30</sup>Speichernutzung nach Komprimierung ohne Berücksichtigung des Speicherplatzes, der vom Komprimierungsbaum oder Kodierungsschema belegt wird. Der Speicherplatz, den diese im Verhältnis zu einer größeren Datenmenge beanspruchen, ist so gering, dass wir diesen geringen zusätzlichen Speicherbedarf im Grunde ignorieren können.

<sup>31</sup>Durchschnittscodewortlänge

<sup>32</sup>Prozentualer Anteil der komprimierten Speichergröße im Vergleich zum kleinsten arbeitenden Zeichensatz in Tabelle 2.

## 5 Schluss

Auch wenn die ursprünglichen Kompressionssysteme von Huffman, Shannon und Fano heute nicht mehr so häufig in ihrer ursprünglichen Form verwendet werden, ist ihr Einfluss in den modernen Kompressionssystemen von heute nach wie vor deutlich spürbar.

higher-  
order  
systems

# Literatur

- [1] URL: <https://raw.githubusercontent.com/wordnik/wordlist/refs/heads/main/wordlist-20210729.txt> (besucht am 23.04.2025).
- [2] Guy E Blelloch u. a. “Introduction to data compression”. In: *Computer Science Department, Carnegie Mellon University* 54 (2001).
- [3] Moses Charikar u. a. “The smallest grammar problem”. In: *IEEE Transactions on Information Theory* 51.7 (2005), S. 2554–2576.
- [4] Robert M Fano. *The transmission of information*. Bd. 65. Massachusetts Institute of Technology, Research Laboratory of Electronics . . . , 1949.
- [5] Travis Gagie. “Dynamic shannon coding”. In: *European Symposium on Algorithms*. Springer. 2004, S. 359–370.
- [6] Robert Gallager. “Variations on a theme by Huffman”. In: *IEEE Transactions on Information Theory* 24.6 (2003), S. 668–674.
- [7] David A Huffman. “A method for the construction of minimum-redundancy codes”. In: *Proceedings of the IRE* 40.9 (1952), S. 1098–1101.
- [8] *ISO/IEC 646:1991 Information technology - ISO 7-bit coded character set for Information Interchange*. The International Organization for Standardization und The International Electrotechnical Commission. 1991.
- [9] Robert E Joyce. *Committee on National Security Systems Glossary*. National Security Agency. Fort Meade, MD, United States of America, 2022.
- [10] John C Kieffer und En-Hui Yang. “Grammar-based codes: A new class of universal lossless source codes”. In: *IEEE Transactions on Information Theory* 46.3 (2000), S. 737–754.
- [11] Donald E Knuth. “Dynamic Huffman Coding”. In: *Journal of algorithms* 6.2 (1985), S. 163–180.
- [12] Stanislav Krajčí u. a. “Performance analysis of Fano coding”. In: *2015 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2015, S. 1746–1750.
- [13] Charles E Mackenzie. *Coded-Character Sets: History and Development*. Addison-Wesley Longman Publishing Co., Inc., 1980.
- [14] Inna Pivkina. “Discovery of Huffman codes”. In: *Mathematical Association of America* (2014).
- [15] Luis Rueda und B John Oommen. “A fast and efficient nearly-optimal adaptive Fano coding scheme”. In: *Information Sciences* 176.12 (2006), S. 1656–1683.
- [16] Luis Gabriel Rueda. “Advances in data compression and pattern recognition”. Diss. Carleton University, 2002.
- [17] William Shakespeare. *The Complete Works of William Shakespeare*. Project Gutenberg, 1994.
- [18] Claude Elwood Shannon. “A mathematical theory of communication”. In: *The Bell system technical journal* 27.3 (1948), S. 379–423.
- [19] Evelyn S Shuckburgh. *The Histories of Polybius*. Bd. 2. MacMillan und Co., 1889.



- [20] James A Storer und Thomas G Szymanski. “Data compression via textual substitution”. In: *Journal of the ACM (JACM)* 29.4 (1982), S. 928–951.
- [21] *The Unicode Standard, Version 17.0*. The Unicode Consortium. San Francisco, CA, United States of America, 2025.