Friedrich-Koenig-Gymnasium Würzburg Oberstufenjahrgang 2024/2026

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

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 Polybius 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[1]. 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[2]. 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[3]. 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[4]. 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

[1] [17] Book X pp.42-43: 44. The Improvement introduced by Aeneas Tacticus
[2] [8] p.32: Cipher (American English)
[3] [14] p.36: 2.2.1 Introduction to Information Theory
[4] [11] 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[5]. 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 gesammten 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[6]. 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.

$$H = -\sum p_i \log(p_i) \tag{1}$$

Diese Gleichung gilt, wenn die Zeichen voneinander unabhängig sind[7].

[5] [12]: p.1: Introduction
[6] [2] p.40-41: 7 Lossy Compression Techniques
[7] [16] 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[8].

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[9].

[8] [14] p.40-44: 2.2.4 Higher-order Sources
[9] [14] 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 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[10].

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

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

Shannon stellt außerdem fest, dass mit zunehmender Länge N unserer Eingabe die durchschnittliche Länge des Codeworts H' sich der Entropie H annähert.

$$H' = \frac{1}{N} \sum (m_s p_s) \tag{4}$$

Hier ist p_s der Wert für die kumulative Wahrscheinlichkeitsfunktion P_i. m_s ist eine ganze Zahl, wo:

$$\log_2(\frac{1}{p_s}) \leq m_s < \log_2(\frac{1}{p_s}) + 1 \tag{5}$$

$$H \leq H' < H + \frac{1}{N} \tag{6}$$

Wenn diese Annahme richtig ist, bestätigt dies die Intuition, dass ein Komprimierungssystem bei längeren Eingaben umso effektiver ist, auch wenn das Komprimierungssystem nicht optimal ist. Es ist immer noch zu beachten, dass der Wert für die durchschnittliche Codewortlänge jedoch niemals unter den Wert für die Entropie H fallen kann, ohne dass Informationen verloren gehen.

2.1.1 Beispiel: Statisches Shannon Kodierung

Eingabe:

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

[10] [14] p.52-53: 2.4.2 Shannon's Method; [16] p.401-403: 9. The Fundamental Theorem for a Noiseless Channel

s_i	p_i	P_i	l_i	r-Darstellung von P_i	Code
a	0.22	0	3	$0.000\ldots$	000
b	0.20	0.22	3	$0.001\ldots$	001
c	0.18	0.42	3	$0.011\ldots$	011
d	0.15	0.60	3	$0.100\ldots$	100
e	0.10	0.75	4	0.11	1100
f	0.08	0.85	4	$0.1101\ldots$	1101
g	0.05	0.93	5	$0.11101\ldots$	11101
h	0.02	0.98	6	$0.111110\ldots$	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 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[11]. Dieses System ähnelt dem adaptiven Fano-Kodierungssystem mit der Brute-Force-Methode, das später behandelt wird.

[11] [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[12].

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	2	00
b	0.20	2	01
c	0.18	3	100
d	0.15	3	101
e	0.10	3	110
f	0.08	4	1110
g	0.05	5	11110
h	0.02	5	11111

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

[12] [4] p.5-6: Selection from N Equally Likely Choices; [14] p.55: Algorithm 3 Static Fano Encoding

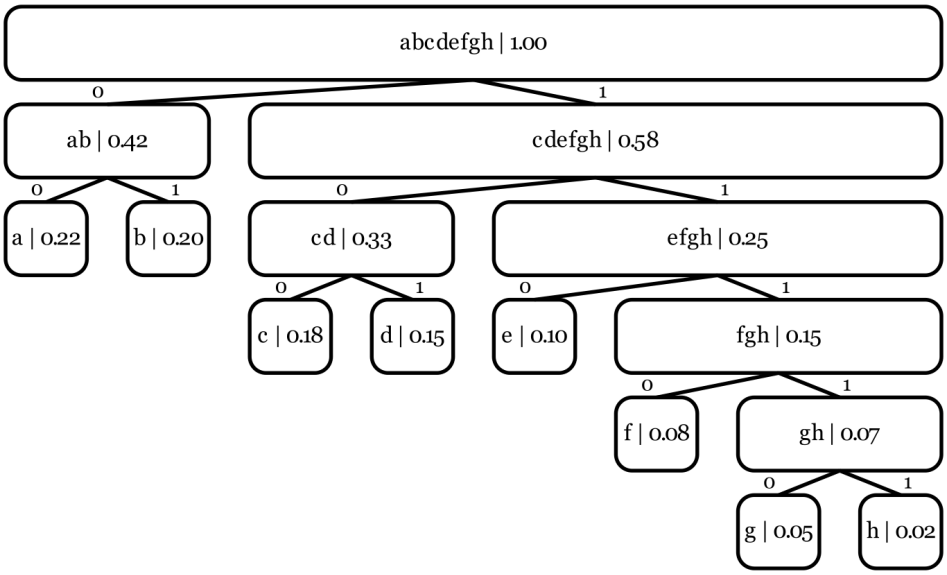

Abbildung 1: Fano Baum von Tabelle 2; Mit Manim Community v0.19.0 gemacht

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[13]. Dies ist natürlich ziemlich ineffizient, aber funktioniert immer um einen 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 [13]. 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[14].

[13] [13] p.1659-1660: 2.1 A brute-force method for adaptive Fano coding
[14] [13] 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.

$$p_i \geq p_{i+1} \geq \cdots \geq p_m \tag{7}$$

$$q_i \leftarrow s_i, (\tau_i = p_i) \tag{8}$$

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.

$$q_{new} \leftarrow q_{new\,j}, \tau_{new} = \sum_{i=1}^{j} \tau_i \tag{9}$$

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[15].

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 τ darstellen. Bei der Kodierung und Dekodierung verwenden wir diesen Baum als Übersetzungshilfe, wobei der am weitesten links stehende Knoten 0 ist, dann 1 usw.

[15] [7] p.1011: Generalization of the Method; [14] p.50: Algorithm 1 Static Huffman Encoding

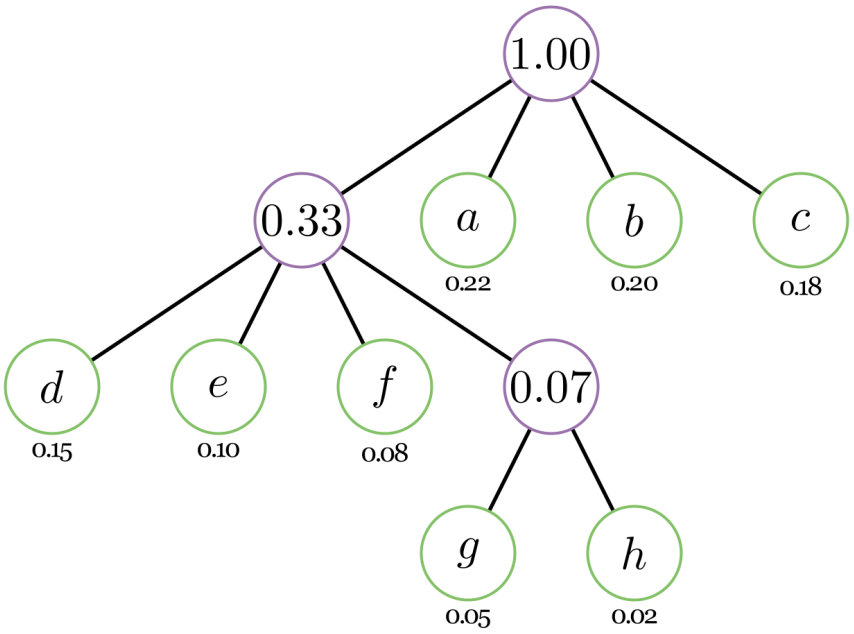

Abbildung 2: Huffman Baum von Tabelle 3; Mit Manim Community v0.19.0 gemacht

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[16]. 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[17]. 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[18].

[16] [6] p.6: 2. The Sibling Property Definition
[17] [6] p.17-21: Adaptive Huffman Codes
[18] [10] 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 füht 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 Makro-Schemata ermöglichen Zeigern auf wiederholte Abschnitte derselben Zeichenfolge[19]. Mit diesem System können wir den Grad der Redundanz in unserer Eingabesequenz verringern. LZ77/Defla

3.2 Block-Sorting Kompressionsalgorithmus

Der Block-Sorting-Kompressionsalgorithmus, auch bekannt als Burrows-Wheeler-Transformation (BWT), ist eine Methode zum Sortieren einer Eingabesequenz in eine Form, die für die Komprimierung mit anderen Methoden besser geeignet ist. Wir beginnen mit einer Eingabesequenz $\mathcal{X} = \{x_1 \to x_M\}$ und erstellen eine $M \times M$-Matrix. Jede Zeile der Matrix ist die Eingabesequenz, verschoben um eine Spalte im Gegensatz zur Zeile davor. Anschließend sortieren wir die Matrix alphabetisch und bezeichnen sie als A. Die Ausgabe der Transformation ist die letzte Spalte von A, die wir als T bezeichnen, und die Zeile, in der $\mathcal{X}$ zu finden ist, die wir als k bezeichnen. Bei der Umkehrung der Transformation sortieren wir T alphabetisch in einer neuen Liste L und notieren die Position des Zeichens in L in T in einer neuen Liste F. Mit Hilfe von F können wir die ursprüngliche Sequenz $\mathcal{X}$ rekonstruieren[20].

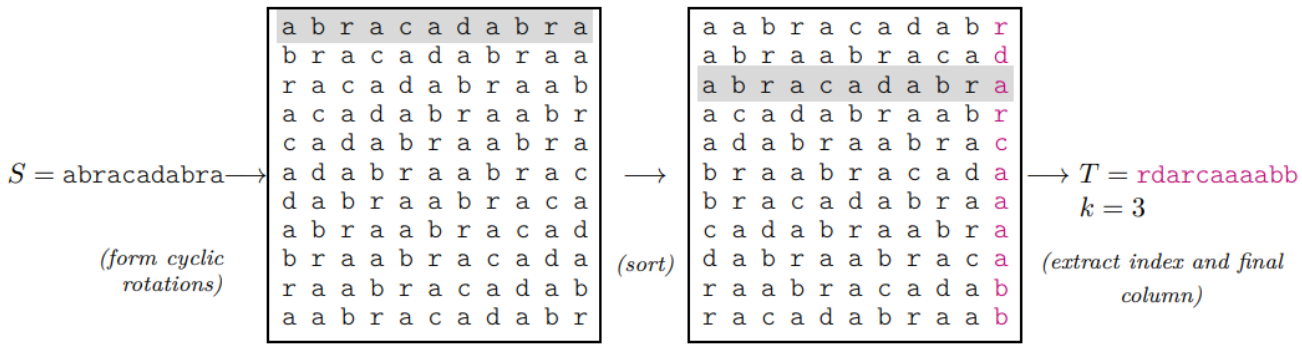

Abbildung 3: Beispiel aus [18]

3.3 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 Σ, einer Liste

[19] [19] p.929-932: 2 The Model and Basic Definitions
[20] [14] p.79-80: 2.6.5 Block Sorting Compression; [18] p.33-34: Block Compression

von Nicht-Terminale N, einer Liste von Produktionsregeln P und einem Startknoten s besteht[21]. 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[22].

$$L(G_x) = \{x\} \tag{10}$$

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[23]. Das in [3] angegebene Beispiel lautet:

$$P = \{< S > \rightarrow < B >< B >< A >, < A > \rightarrow \text{ a rose}, < B > \rightarrow < A > \text{ is } \} \tag{11}$$

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

3.4 Prediction with Partial Matching

Die Methode der "Prediction with Partial Matching"verwendet einen Suchbaum, der als "Trie"bezeichnet wird, mit einer maximalen Tiefe D, wobei jeder Knoten eine bestimmte Wahrscheinlichkeitsverteilung in Abhängigkeit von seinen Eltern sowie Vine-Zeiger auf die darüber liegende Ebene aufweist. Diese Methode funktioniert in mehreren Schritten[24]:

1. Erstellen Sie einen leeren Baum.
2. Wiederholen Sie dies, bis das Ende erreicht ist:

   Lesen Sie die nächste Eingabe.

   Verwenden Sie die Wahrscheinlichkeitsverteilung des aktuellen Knotens, um die Eingabe zu codieren.

   Aktualisieren Sie die Wahrscheinlichkeitsverteilung und den entsprechenden Vine-Knoten.

   Suchen Sie den erforderlichen Knoten, der der Eingabe entspricht. Wenn die Tiefe maximal ist, wechseln Sie zum Knoten des Vine-Zeigers. Wenn der erforderliche Knoten nicht existiert, erstellen Sie einen und aktualisieren Sie den Baum, damit er funktioniert.

   Setzen Sie einen Zeiger vom aktuellen Knoten zum Knoten des Kindes und gehen Sie zu diesem Knoten.

Dies lässt sich grafisch darstellen, wobei die gepunkteten Linien die Vine-Zeiger und die normalen Zeiger den Pfad anzeigen, den der Algorithmus durchlaufen hat. Der schattierte Knoten ist die Position am Ende des Algorithmus:

[21] [14] p.80-81: 2.6.6 Grammar-Based Coding
[22] [9] p.2-4: Introduction
[23] [3] p.1-2: Introduction
[24] [18] p.111-113: 6.3.1 Basic Operation

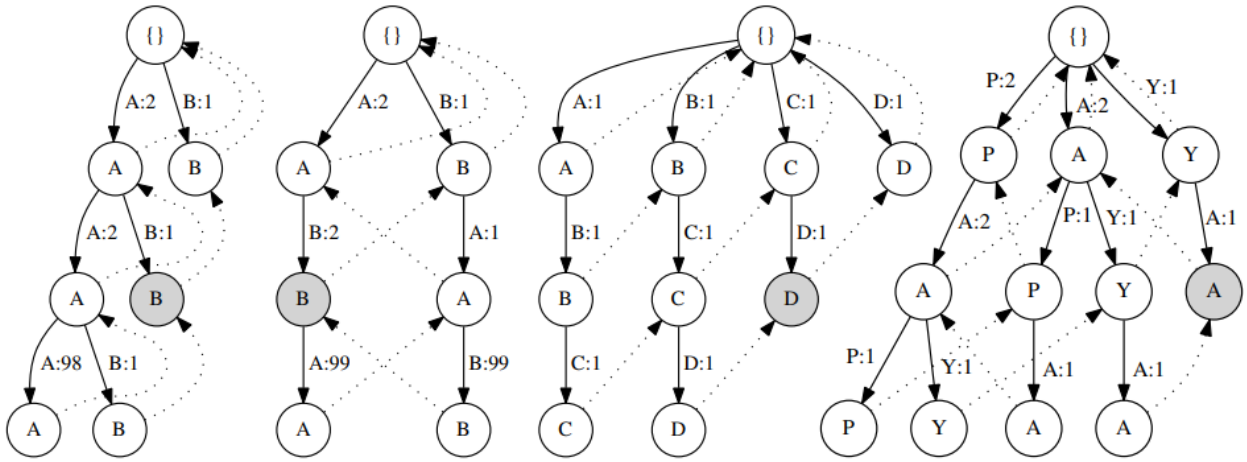

Abbildung 4: Beispiele aus [18]

4 Vergleich der ursprünglichen Kompressionsalgorithmen

Beispiel	**Speicherbedarf**[25]	**Entropie** H
Wordnik[26]	20.5	4.25
Shakespeare[27]	44.6	4.84

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

Kompressionssystem	**Prozesse**[28]	**Größe**[29]	$\overline{L}$ **Avg. L**[30]	**Prozentsatz**[31]
Huffman	2565462	11.0	4.29	53.6%
Shannon	2565500	12.0	4.70	58.7%
Fano	2565349	11.4	4.44	55.5%

Tabelle 5: Vergleich der verschiedenen Kompressionssysteme mit Wordnik

[25] Berechnet Anzahl der Eingabezeichen multipliziert mit 8

[26] [1]: Wordnik Liste öffentlich zugänglicher englischer Wörter

[27] [15]: Die gesammelten Werke von William Shakespeare

[28] Berechnungsprozesse sind relativ, verschiedene Arten von Prozessen dauern unterschiedlich lange, daher sollte diese Zahl eher vage interpretiert werden.

[29] 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.

[30] Durchschnittscodewortlänge

[31] Prozentualer Anteil der komprimierten Speichergröße im Vergleich zum Zeichensatz in Tabelle 4.

Kompressionssystem	**Prozesse**	**Größe**	$\overline{L}$ **Avg. L**	**Prozentsatz**
Huffman	5575693	27.2	4.88	61.0%
Shannon	5576446	29.6	5.31	66.4%
Fano	5575268	27.4	4.91	61.4%

Tabelle 6: Vergleich der verschiedenen Kompressionssysteme mit Shakespeare

5 Schluss

Auch wenn die ursprünglichen Komprimierungssysteme von Huffman, Shannon und Fano heute nicht mehr so häufig in ihrer ursprünglichen Form verwendet werden, bilden sie doch die Grundlage für moderne verlustfreie Komprimierungssysteme. Huffman zeigt uns, dass es in einem zeroth-order Komprimierungssystem einen optimalen Komprimierungsalgorithmus gibt, der mit einem Bottom-up-Konstruktionssystem funktioniert. Die Kodierungssysteme von Fano und Shannon sind zwar nicht so effektiv, wenn es darum geht, sich dem Wert der Entropie anzunähern, aber sie bieten einen intuitiveren Überblick und eine intuitivere Methode zur Implementierung eines verlustfreien Kompressionsalgorithmus. Anhand der verwendeten Beispiele können wir jedoch sehen, dass der Unterschied zwischen ihren Kompressionsgrößen zunächst recht gering ist, aber mit zunehmender Dateigröße zunimmt. In diesem Beispiel können wir auch sehen, wie Shannons Theorie funktioniert: Je größer die Datei wird, desto mehr Speicherplatz wird eingespart. Wenn wir uns higher-order Modelle ansehen, können die Dinge viel komplizierter und speichereffizienter werden, aber am Ende kommt es fast immer zu einem zeroth-order Komprimierungssystem, nur dass die Eingabe effektiver komprimiert werden kann als die ursprüngliche Eingabe.

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] Robert E Joyce. *Committee on National Security Systems Glossary*. National Security Agency. Fort Meade, MD, United States of America, 2022.

[9] 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.

[10] Donald E Knuth. "Dynamic Huffman Coding". In: *Journal of algorithms* 6.2 (1985), S. 163–180.

[11] Stanislav Krajči u. a. "Performance analysis of Fano coding". In: *2015 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2015, S. 1746–1750.

[12] Inna Pivkina. "Discovery of Huffman codes". In: *Mathematical Association of America* (2014).

[13] 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.

[14] Luis Gabriel Rueda. "Advances in data compression and pattern recognition". Diss. Carleton University, 2002.

[15] William Shakespeare. *The Complete Works of William Shakespeare*. Project Gutenberg, 1994.

[16] Claude Elwood Shannon. "A mathematical theory of communication". In: *The Bell system technical journal* 27.3 (1948), S. 379–423.

[17] Evelyn S Shuckburgh. *The Histories of Polybius*. Bd. 2. MacMillan und Co., 1889.

[18] Christian Steinruecken. "Lossless data compression". Diss. University of Cambridge, 2015.

[19] James A Storer und Thomas G Szymanski. "Data compression via textual substitution". In: *Journal of the ACM (JACM)* 29.4 (1982), S. 928–951.