Signal und Bildverarbeitung

WS 2022/23

Übung 2

Gruppe: Drucker Fridolin & Klausgraber Claudia

1) Klassifizieren Sie Sprachen (textuelle Repräsentation) mittels Kompression. Wählen Sie dabei zumindest n=8 Klassen (d.h. Sprachen), für die Sie repräsentative Vergleichsdatensätze aufbereiten. Welche Vorverarbeitungen sind beim Aufbau der Vergleichsdatensätze bzw. Testdatensätze sinnvoll bzw. notwendig? Klassifizieren Sie dann Beispieltexte und werten Sie die Ergebnisse statistisch aus (exakte Treffer bzw. Rang, ev. Verwendung einer Konfusions-Matrix). Können die erzielten Ergebnisse bei vorliegender Testanzahl als statistisch signifikant betrachtet werden? Wie hängt die erzielbare Klassifikationsgenauigkeit mit der Anzahl der zu diskriminierenden Klassen zusammen?

Textklassifikation, also ein Text einem bestimmten Thema z.B. zuzuordnen, ist eine Methode des maschinellen Lernens, um große Textmengen vordefinierten Kategorien zuzuweisen. Es hilft dabei, große, meist unstrukturierte, Daten bzw. Texte zu ordnen und zu strukturieren

Klassifikation mittels Datenkompression bedeutet, Daten durch einen bestimmten Kompressionsalgorithmus einer Klasse zuzuordnen. Diese Methode beinhaltet nur Algorithmen, die ohne Verlust arbeiten, denn wenn dies nicht der Fall wäre, würde zwar die Datei kleiner werden, aber Informationen verloren gehen. Verlustbehaftete Kompressionsmethoden eignen sich nur für Medien, bei denen der Verlust für den Menschen nicht auffällt, beziehungsweise keine Rolle spielt, etwa bei Bildern, wenn gewisse Pixel weggelassen werden, die der Mensch nicht wahrnehmen kann (Menschen können zB nur eine maximale Anzahl an Graustufen-Pixel unterscheiden) oder bei MP3, wenn für den Menschen unhörbare Frequenzen weggelassen werden.

Die Klassifikation durch Kompression wird folgendermaßen durchgeführt: Zuerst erstellt man n Textdateien, die die Klassen repräsentieren. Danach wird eine Datei erstellt, die zugeordnet werden soll. Zuerst erstellt man von jeder Klassendatei eine komprimierte Datei, diese werden jeweils mit der Datei in eine Datei zusammengefügt, die zugeordnet werden soll, und ebenfalls komprimiert. Diejenige Datei, die das kleinste Delta innerhalb der Klasse mit der Datei hat, die zugeordnet werden soll, ist jene Klasse, zu der es gehört. Es gilt also: Zip(Textdatei + Zuordnungstext)- Zip(Textdatei) bzw. Zip(Trainingstext + Testdatei) – Zip(Traingstext).

Für diese Übung wurden n=8 Klassen vorgegeben, wobei jede Klasse eine Sprache darstellen soll. Hier wurden die Sprachen Chinesisch, Deutsch, Englisch, Estnisch, Französisch, Griechisch, Italienisch und Spanisch benutzt. Als Trainingstext wurde geschaut, dass es immer ca gleich lange Textdateien sind, immer etwa 4000 Zeichen. Es wurde der Anfang von Harry Potter – Stein der Weisen benutzt und ein Auszug aus Wikipedia über Fußball. Dieser wurde in die anderen sieben Sprachen mittels dem Pons Textübersetzter übersetzt. Wenn bei einer Datei zu wenig Zeichen waren, wurde der Rest mit dem Wikipedia Artikel zu Fußball aufgefüllt. Diese wurden jeweils einzeln komprimiert. Danach wurde die Testdatei erstellt – die ersten ca 4000 Zeichen von Hänsel und Gretel. Diese Datei wurde wieder in die anderen sieben Dateien übersetzt und abgespeichert. Nun wurde jeweils zu jeder Textdatei die Testdatei in die Textdatei hinzugefügt. Wichtig ist hier anzumerken, dass die beiden Texte jeweils wirklich in der gleichen Datei zusammengefügt werden. Danach wurden alle Dateien wieder komprimiert. Danach wurde eine Matrix erstellt, die jeweils die gleichen Sprachen als Referenz haben. Von gemeinsam komprimierten Textdateien wurde dann jeweils die komprimierte Textdatei abgezogen, wie in Abbildung 1 beispielhaft an der Sprache Chinesisch zu sehen, und der Unterschied in den entsprechenden Feldern eingetragen, wie in Abbildung 2 zu sehen.

Beispiel für Berechnungen:
Chinesisch x Chinesisch : 3,41-2,16 = 1,25
Chinesch x Deutsch: 3,63-2,06 = 1,57
Chinesisch x Englisch: 3,54-1,96=1,58
Chinesisch x Estnisch: 3,63-2,05= 1,58
Chinesisch x Französisch:3,63-2,06=1,57
Chinesisch x Griechisch: 4,05-2,57=1,48
Chinesisch x Italienisch:3,60-2,02=1,58
Chinesisch x Spanisch: 3,58-2,00=1,58

Abb1: Rechenschritte für die Testdatei Chinesisch.

	Chinesisch	Deutsch	Englisch	Estnisch	Französisch	Griechisch	Italienisch	Spanisch	Match?
Chinesisch	1,25	1,57	1,58	1,58	1,57	1,48	1,58	1,58	ja
Deutsch	1,56	1,22	1,36	1,35	1,36	1,47	1,36	1,36	ja
Englisch	1,4	1,2	1,11	1,23	1,2	1,33	1,21	1,22	ja
Estnisch	1,41	1,22	1,24	1,12	1,24	1,35	1,24	1,24	ja
Französisch	1,53	1,33	1,34	1,35	1,22	1,46	1,31	1,32	ja
Griechisch	1,83	1,87	1,88	1,87	1,87	1,53	1,88	1,88	ja
Italienisch	1,48	1,29	1,29	1,31	1,25	1,41	1,17	1,28	ja
Spanisch	1,48	1,29	1,29	1,28	1,26	1,41	1,27	1,19	ja

Abbildung 2: Die Matrix mit den ausgefüllten Zellen.

Man kann erkennen, dass jedes Dokument seiner Sprache zuordenbar ist. Somit ist eine Trefferquote von 100% gegeben. Damit liegt die Trefferquote bei weit über dem Durchschnitt und ist somit statistisch signifikant. Die Präzision, also die richtig positiven/alle Positiven, ist 1.

Dadurch, dass die gewählten Sprachen nicht sehr ähnlich waren, war die Unterscheidung nicht schwierig. Je ähnlich die Sprachen, bzw. je ähnlicher die Buchstaben der Brache, desto schwieriger wird die Unterscheidung. Die Idee ist nämlich, dass wenn ein Dokument einer Trainingsdatei ähnelt und sie Muster teilen, wird dies vom Komprimierungsalgorithmus ausgenutzt. Die Größe der Komprimierung der Trainingstexte zusammen mit der Trainingsdatei ist ein Indikator dafür, wie viele Muster übereinstimmen. Das heißt, je geringer der Anstieg, desto eher sollte das Label des Trainingstextes vergeben werden. Genau deshalb ist es bei ähnlichen Sprachen so schwierig, weil hier die geteilten Muster ähnlich bzw. gleich sind und irreführenderweise auch nur gering ansteigen. Vermutlich wäre dies zum Beispiel bei Niederländisch und Deutsch der Fall, da beide germanische Sprachen sind und Niederländisch generell als dem Deutsch am ähnlichsten gilt.

Ebenso spielt die Anzahl der Texte eine Rolle, denn je mehr Sprachen benutzt werden, desto eher kann es zu falschen Zuordnungen kommen. Je größer die Anzahl der Klassen, zu denen ein Text zugewiesen werden kann, desto wichtiger ist es, dass die Trainingsdateien dem entsprechen, was die Klasse darstellt. Für Sprachen eignen sich dafür etwa Pangramme, da diese im Idealfall das ganze Alphabet in einem bzw. wenigen Sätzen darstellen und es so ins Wörterbuch des Komprimierungsalgorithmus aufgenommen werden kann.

OPTIONAL – nur für Interessierte/Expert*innen: [20] Klassifizieren Sie Beispielbilder (8bit Grauwert) mittels Kompression. Wählen Sie zumindest n=4 Klassen, für die Sie repräsentative Vergleichsdatensätze aufbereiten. Welche Vorverarbeitungen sind beim Aufbau der Vergleichsdatensätze bzw. Testdatensätze sinnvoll bzw. notwendig? Klassifizieren Sie dann Beispieltexte und werten Sie die Ergebnisse statistisch aus (exakte Treffer bzw. Rang). Können die erzielten Ergebnisse bei vorliegender Testanzahl als statistisch signifikant betrachtet werden? Wie hängt die Klassifikationsgenauigkeit mit der Anzahl der zu diskriminierenden Klassen zusammen?

Bei den Bildern wird ähnlich wie bei den Texten vorgegangen. Auch hier wird der ASCII-Code der Bilder in Textdateien gespeichert und komprimiert. Hier war die Vorgabe n=4, wobei hier für jede Klasse zwei Bilder ausgesucht wurden, die einfach und ähnlich sind. Sie unterscheiden sich nur durch das Verändern des Ansichtswinkels.

Grundsätzlich ist es wichtig, dass die verwendeten Bilder alle gleich groß sind (hier 128 x 128 Pixel), den gleichen Farbraum haben (hier 8 Bit) und Auflösung(16K). Es wurde auf die Datenbank „Columbia Object Image Library (COIL-100) Dataset“ in Graubildern zurückgegriffen, damit die Bilder all diese Eigenschaften erfüllen, wie etwa die Ente, wie in Abb. 3 zu sehen.

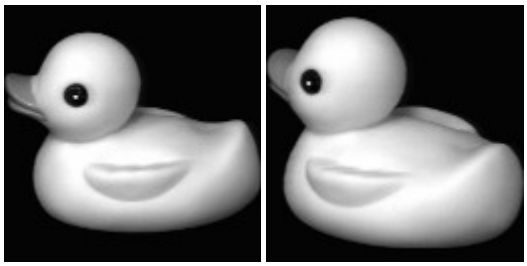


Abb. 3: Das Trainingsbild „Ente1“ und Testbild „Ente2“ aus der Coil-Datenbank.

Die Klassen, die gewählt wurden, lauten Auto, Katze, Riegel und Ente. Wieder wurde eine Matrix angelegt und nach der gleichen Vorgehensweise gerechnet: Zip(Trainingsbild + Testbild) – Zip(Trainingsbild).

Hier ist anzumerken, dass die Bilder als ASCII-Code gespeichert werden, also nicht direkt mit den Bildern gearbeitet wird. Und auch hier ist es wieder wichtig, dass die beiden Textdateien in eine einzige Textdatei zusammengefügt werden. Es wurde eine Matrix erstellt und die Deltas der Konvertierungsraten eingetragen, wie in Abbildung 4 zu sehen.

	Auto 1	Katze 1	Riegel 1	Ente 1	Match?
Auto 2	4,445	4,767	4,715	4,707	Ja
Katze 2	7,7	7,868	7,826	7,813	nein
Riegel 2	6,993	7,305	7,281	7,2	nein
Ente 2	9,625	9,819	9,783	9,773	nein

Abb.4: Matrix der Bilder mit den unterschiedlichen Deltas der Komprimierungsraten.

Wie in Abb. 4 zu sehen, wurde leider nur ein Paar erkannt (Auto 1 & 2), wobei dies nicht statistisch signifikant ist. Alle anderen Klassen wurden leider nicht erkannt. Somit ergeben sich vier Fehler der 1. Art (die Klasse wurde nicht erkannt). Die Matrix wird in True False, Negativ False, True Positiv und True Negativ eingeteilt, wie in Abb. 5 zu sehen. Die Genauigkeit des Modells wird mit der Formel TP + TN/ TP+FP+TN+FN berechnet und ist mit 0,625 nicht sehr hoch.

	Auto 1	Katze 1	Riegel 1	Ente 1	Match?
Auto 2	TP	TN	TN	TN	Ja
Katze 2	FP	FN	TN	TN	nein
Riegel 2	FP	TN	FN	TN	nein
Ente 2	FP	TN	TN	FN	nein

Abb.5: Konfusionsmatrix der Bilderpaare, eingeteilt in TF, NF, FP und TP.

Wahrscheinlich waren die Bilder nicht einfach genug für diese Aufgabe oder die Klassen an sich doch zu unterschiedlich. Wenn das Bild zu komplex ist, werden zu viele Zeichen im Wörterbuch gespeichert und sobald dieses voll ist, wird ein neues angefangen. Eventuell ist das hierbei passiert. Wenn noch einfachere Motive gewählt worden wären, hätte es vielleicht geklappt.

Davon abgesehen wäre es vielleicht leichter gewesen, wäre eine Klasse durch mehrere Bilder repräsentiert worden, also z.B. 3 Bilder, die immer zu einer Klasse gehören. So hätte die Anzahl der TP-Treffer evtl. erhöht werden können, da in manchen Papern angegeben wird, dass eine Zuweisung als korrekt gewertet wird, sobald ein TP-Match mit einem Trainingsbild der Klasse stattfindet. Die Anzahl der zu differenzierenden Klassen hat wieder den Einfluss, dass je mehr Klassen es gibt, desto wahrscheinlich ist ein Outliner und desto schwieriger wird das Klassifizieren, da sich die Bilder nicht zu sehr ähneln dürfen, also die Klassen klar voneinander abgegrenzt sein müssen. Somit spielen die Motive ebenso, bzw. implizit, eine wichtige Rolle, durch die die Klassen repräsentiert werden sollen. Wie bei uns ersichtlich, ist es naheliegend, dass je komplexer das Motiv ist, desto komplexer ist die Zuweisung.

Wenn der verwendenden Datensatz nicht aus einer vorgefertigten Datenbank stammt, müssen die Bilder noch vorbereitet werden, also auf die gleiche Größe, Farbraum und Auflösung gebracht werden. Laut verschiedenen Papern hat auch die Brightness einen Einfluss, die man am besten durch eine Gammakorrektur anpassen kann. Da jedes Bild auch eine andere Textur hat, bzw. jede Klasse, kann man auch nach der Textur klassifizieren.

2.3 a) [5] Komprimieren Sie die Sequenz dabbababababbbaaaababababccdd (n=4 Symbole: {a,b,c,d}) händisch mittels Lempel-Ziv Kompression. Berechnen Sie die erzielbare Kompressionsrate.

Die Lempel-Ziv-Welch-Methode, manchmal auch als LZW-Algorithmus oder einfach LZW bezeichnet, wird häufig in Grafikformaten verwendet, um Daten zu komprimieren bzw. die Datenmenge zu verringern. Abraham Lempel und Jacob Ziv entwickelten und veröffentlichten einen großen Teil der Funktionalität dieses Algorithmus im Jahr 1978.

Eine verlustfreie Komprimierungsmethode ist LZW. Es wird z. B. in TIFF und dem GIF-Bildformat verwendet. Da das Wörterbuch erst zur Laufzeit erstellt wird und somit formatunabhängig ist, eignet es sich für jede Art von Daten.

Die eingetragenen Zahlen entsprechen der ASCII-Code-Tabelle, welche beim Eintrag 255 endet. Neue Zeichenketten werden ab 256 weitergezählt.

Kompression und Code-Transformation (Lempel-Ziv)

dabbababababbbaaaababababccdd **25 Zeichen**

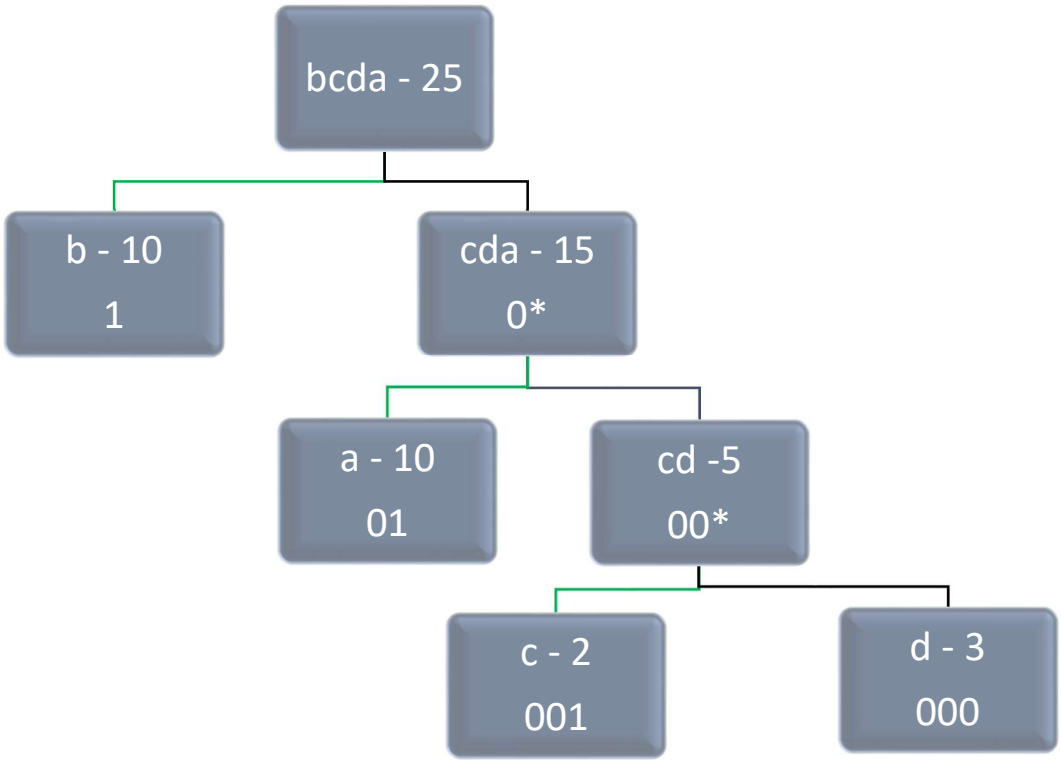
aktuelles Zeichen	nächstes Zeichen	im Wörterbuch	Ausgabe	neue codierung
d	a	N	d	<256> da
a	b	N	a	<257> ab
b	b	N	b	<258> bb
b	a	N	b	<259> ba
a	b	Y 32;	257 (ab)	<260> aba
a	ba	Y35;	260 (aba)	<261> abab
a	b	Y 32;	258 (bb)	<262> bbb
b	b	Y 33;	259 (ba)	<263> baa
a	a	N	a	<264> aa
a	a	Y	264 (aa)	<265> aab
a	b	Y 36;	259 (ba)	<266> bab
a	b	y 32	266 (bab)	<267> babc
c	c	n	c	<268> cc
c	d	n	c	<269> cd

d	d	n	d	<270> dd
d	0	n	d	0
			20 Zeichen	
Originallänge		25		
Komp. Länge		20		
			Platzersparnis	
Komp Reate		**1,25**	20%	

2.3 b) [8] Transformieren Sie die Sequenz dabbababababbbaaaababababccdd (n=4 Symbole: {a,b,c,d}) händisch mittels Huffmann-Coding in eine komprimierte Darstellung und geben Sie dabei auch den Huffmann-Baum an. Berechnen Sie die erzielbare Kompressionsrate bzw. die mittlere Codewort Länge. Bei welcher eigenständig gewählten 10-stelligen Sequenz ist die Kompressionsrate maximal bzw. bei welcher 10-stelligen Sequenz ist die Kompressionsrate minimal? Von welcher Beschaffenheit der Daten hängt die erzielbare Kompressionsrate ab?

David Huffman entwickelte 1952 die Huffman-Kodierungsmethode der Entropiekodierung. Sie gibt Quellensymbolen Codewörter variabler Länge. Es handelt sich um einen Präfixcode, der häufig zur verlustfreien Kompression verwendet wird. Ähnlich wie bei früheren Entropiekodierungen werden Zeichen, die häufiger vorkommen, mit weniger Bits dargestellt als Zeichen, die selten vorkommen.

dabbababababbbaaaababababccdd



Variable	Anzahl	Wahrscheinlichkeit
A - 01	10	0,4

B - 1	10	0,4
C - 001	2	0,08
D - 000	3	0,12

25*2^3 = ursprünglich = 200

Bei der ursprünglichen Codierung erhalten wir 50 Bits, die wir an Speicherplatz benötigen (00,01,10,11). Dann vergleiche es mit den Bits, welche ich in Huffman erzielen kann.

Nach Huffman: 000 01 1 1 01 1 01 1 01 1 1 1 01 01 01 01 1 01 1 01 1 001 001 000 000 konnte ich **42** Bits zählen.

Komp. Rate: 4,7619

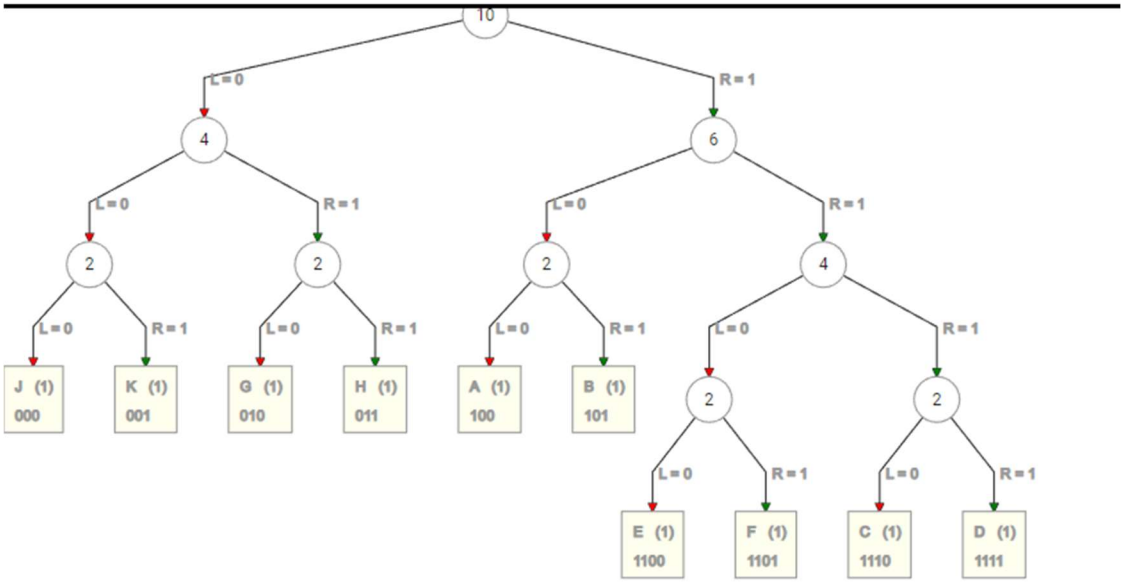
Platzersparnis: 79%

Mittlere Codewortlänge: 0,4*2 + 0,4 + 0,08*3 + 0,12*3 = 1,8

Eine **maximale** Kompressionsrate (10 Stellen) kann ich erzielen, wenn ein einzelner Buchstabe maximal oft auftritt. Also 10a = 10Bits. Daraus ergibt sich eine Komp. Rate von 1 und eine Mittlere Codewortlänge von 1.

Variable	Anzahl	Wahrscheinlichkeit
A - 1	10	1
B - 01	0	0
C - 001	0	0
D - 000	0	0

Das Gegenteil tritt auf bei einer **minimalen** Kompressionsrate (10 Stellen) und wenn alle gleichmäßig unterschiedlich sind. Also 10 unterschiedliche Buchstaben.



Mittlere Codewortlänge = 0,1*3*6 + 0,1*4*4= 3,4

Komp. Rate = 160 / (6*3+4*4) = 4,709

Platzersparnis = 78.75%
Bits komp. = 34

Unkomprimiert: 10*2^4= 160 bits

2.3c) [6] Komprimieren Sie händisch die Sequenz 11110101011111000001111000 1010 (30 Stellen, n=2 Symbole: {0,1}) mittels Runlength Coding. Berechnen Sie die erzielbare Kompressionsrate. Wie kann das Runlength Coding auf eine Symbolmenge n>2 erweitert werden und was ist dabei zu beachten? Demonstrieren Sie Ihre Erweiterung an einem selbst gewählten BSP und berechnen Sie die erzielbare Kompressionsrate. Von welcher Beschaffenheit der Daten hängt die erzielbare Kompressionsrate ab?

Ein einfaches Verfahren zur verlustfreien Komprimierung ist die Lauflängenkodierung (RLE). Sie ist hilfreich bei der Verdichtung längerer Symbolwiederholungen. Da sie auf der absoluten Häufigkeit und nicht auf der relativen Häufigkeit von Symbolen basiert, fällt sie nicht in die Kategorie der Entropie-Kodierer.

Sequenz→11110 1010 1111 100000 111 1000 1010 = 30 Bits

Zusammengefasst/nachher → 4 11111 5 5 4 3 1111

4101010515041301010 = 19

4+ - + - + - 5+ 5- 4+ 3- + - + - = 5 patterns

2^3 Bit = 19*2^3 = 152

Komp Rate = 0,1974

SSSSSSSSSSDFFFFFFFFFFFFFFDDDDDDDAAA 35 Zeichen

4 patterns 35 * 2^4 = 560 bits

10SD13F7D3A

10 patterns 11 *2^10 = 11264 bits

Komp. Rate = 0,0497

2.3 [6] Berechnen Sie die Entropie der 30-stelligen Sequenz 221111226611122333345645112111 (n=6 Symbole: {1,2,3,4,5,6}). Bei welcher 10-stelligen Sequenz ist die Entropie maximal bzw. bei welcher 10-stelligen Sequenz ist die Entropie minimal? Welche Auswirkungen hat die Entropie in Bezug auf die erzielbare Kompression? Ist für die erzielbare Kompressionsrate dabei immer lediglich die Auftrittswahrscheinlichkeit entscheidend?

Die Entropie ist eine physikalische Größe. Um sie zu veranschaulichen wird dieser Begriff häufig von einer informationstheoretischen Perspektive beschrieben.

Um die sechs möglichen Zustände eines Würfels zu beschreiben, wären 3 Bit erforderlich, wobei aber zwei der 23=8 möglichen Bitmuster gar nicht verwendet werden. Der Informationsgehalt sollte in diesem Fall also einen nicht-ganzzahligen Wert zwischen 2 und 3 annehmen.

Die Gesamtentropie setzt sich wegen des Logarithmus aus Einzelentropien zusammen.

Entropie Berechnung nach Shannon:

Zahl/wurf	Häufigkeit	Wahrscheinlichkeit	gekürzt		Prozent	Inidivduelle Entropie
1	12	12/30"	6/15"	0,40	40%	0,4
2	7	7/30"	7/30"	0,23	23%	0,233
3	4	4/30"	2/15"	0,13	13%	0,133
4	2	2/30"	1/15"	0,07	7%	0,067
5	2	2/30"	1/15"	0,07	7%	0,067
6	3	3/30"	1/10"	0,10	10%	0,1

Mittlere Entropie, Bits: 2,26

H(X) = -[(0.4log20.4)+(0.233log20.233)+(0.133log20.133)+(0.067log20.067)+
(0.067log20.067)+(0.1log20.1)]
H(X) = -[(-0.529)+(-0.49)+(-0.388)+(-0.26)+(-0.26)+(-0.332)]
H(X) = -[-2.25936]
H(X) = 2.25936

Daraus können wir ablesen, dass wir 90 Bits brauchen, um den String zu codieren.

Die Metrische Entropie in dem Beispiel ist 0,07531

Sequenz	Entropie
2211112266	1.52193
3456451121	2.44644
1111111111	0
1234567890	3.32193

Grundsätzlich kann man sagen, dass eine niedrige Entropie einer guten Kompressionsrate entgegenkommt. Aufgrund aller vorherigen Beispiele behaupten wir, dass die Auftrittswahrscheinlichkeit nicht das einzige Maß für eine Kompressionsrate ist. Auch wenn hier die Entropie so berechnet wurde. Ein weiterer wichtiger Faktor für die Kompressionsrate ist die Clusterung der Variablen. Noise verschlechtert die Kompressionsrate mehr, als wenn die unwahrscheinlichen Variablen nebeneinander liegen.