Belegarbeit zur Quellencodierungsmethode **"Run-Length Encoding"** im Modul "statistische Nachrichtentheorie"

Von: Monique Golnik (563075)

Dozent: Prof. Dr. Christoph Lange
Master Informations- Kommunikationstechnik WS 2021/22

18. Januar 2022

Inhaltsverzeichnis

Kapitel 1

Einleitung

In der heutigen Zeit ist es wichtiger den je sich mit der Komprimierung von Daten auseinander zu setzen. Dateien wie Bilder, Kurzvideos oder Musik gewinnen zunehmend an Beliebtheit und werden datentechnisch immer größer. Ein schnelles Senden zu jemand anderem gehört dabei mittlerweile zum Alltag. Auch wenn die gegenwärtige Speicherkapazität einem das Gefühl der Grenzenlosigkeit vermittelt, entspricht dies nicht der Realität - Kompressionsverfahren sind nicht mehr wegzudenken und gewinnen stetig an Bedeutung.

Gegenstand dieser Belegarbeit ist die nähere Betrachtung des Codierverfahrens **Run-Length Encoding**. Ziel ist es, dass ein Verständnis der Funktionsweise dieser Methode und dessen Grenzen vermittelt werden.

Zunächst wird in Kapitel 2 in die Thematik eingeführt und die grundlegenden Begrifflichkeiten der QuellenCodierung und Datenkompression definiert. In Kapitel 3 wird das Verfahren **Run-Length Encoding** erläutert, Vor- und Nachteile abgeschätzt sowie Anwendungsbeispiele dargestellt, um den Einsatz zu verdeutlichen.

Kapitel 2

Quellencodierung und Datenkompression

Bei der Quellencodierung ist es das Ziel, dass die Symbolentropie erhöht und so die Redundanz reduziert wird. Sie ist eine eineindeutige Darstellung der Quelleninformation in einer realisierbaren und möglichst redundanzfreien beziehungsweise -armen Form.

Die Aufgabe der Quellencodierung ist die Codierung eines Datensatzes, der von einer Informationsquelle abgegeben wurde, in einen Binärcode mit möglichst kleiner Stellenzahl. [Lan21, Seite 47 ff.]

„Bei der Datenkompression werden Dateien in eine alternative Darstellung überführt, die effizienter ist als die ursprüngliche. Ziel dieser Codierung ist es, sowohl den benötigten Speicherplatz als auch die Übertragungszeit zu verringern.“ [SE21]

Durch zwei unterschiedliche Ansätze, lässt sich so ein Codiergewinn erreichen:

- **Redundanz-Kompression**: Auf der Grundlage einer Redundanzreduktion lassen sich Daten nach der Kompression verlustfrei wieder dekomprimieren - eine solche Kompression ist nur möglich, wenn ein Datensatz sich wiederholende Zeichen beinhaltet
- **Irrelevanz-Kompression**: Bei diesem Ansatz werden irrelevante Informationen entfernt, um einen Datensatz zu komprimieren. Jedoch lässt sich dieser dadurch nicht Bit-genau wiederherstellen und ist somit verlustbehaftet

Im Folgenden Kapitel wird der Fokus auf die verlustfreie Kompressionsmethode **Run-Length Encoding** gerichtet und näher betrachtet.

Kapitel 3

Run-Length Encoding

3.1 Redundanz-Kompression durch RLE

Die Lauflängencodierung [1] beschreibt eine verlustfreie Kompression, die sinnvoll angewandt werden kann, wenn der gesendete Datensatz Sequenzen mit sich mehrfach wiederholenden Zeichen enthält.

Die RLE ist den sogenannten **Phrasencodern** zugeordnet, wie in Abbildung 3.1 dargestellt. Bei diesen wird nicht wie bei den Entropiecodern auf den mittleren Informationsgehalt geachtet, sondern auf eine syntaktische Einheit, die abgeschlossen ist.

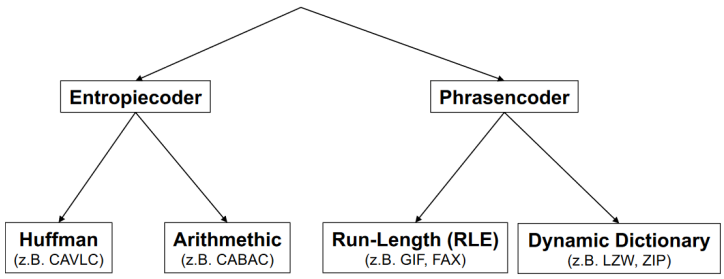

Abbildung 3.1: Klassifikation von verlustlosen Kompressionsmethoden [Bro21, Seite 8]

Bei dieser Art der Codierung wird eine Folge von gleichen Zeichen, Buchstaben oder Zahlen durch ein Symbol und die Angabe über die Häufigkeit der gleichen Symbole substituiert. Die Abfolge von identischen Zeichen und der Länge der jeweiligen Sequenz, die in einem sogenannten Run Counter gespeichert wird, wird als **Run** und die Anzahl der Wiederholungen werden als Lauflänge beziehungsweise **Run Length** bezeichnet.

[1] (Run-Length Encoding (RLE), Run-Length Coding (RLC), Lauflängencodierung = Synonyme

Aufeinanderfolgende Vorkommen eines Datensatzes $d = Run$ (3.1)

Anzahl der Wiederholungen n von $d = RunLength$ (3.2)

Diese Form der Kompression eignet sich unter anderem sehr gut um Redundanzen in Grafiken und Bilddateien mit wenigen Farben zu beseitigen.

Ist die zu komprimierende Grafik sehr detailreich, besitzt also sehr viele Farben oder Kontraste, ist die Lauflängencodierung nicht mehr geeignet, da so nur sehr selten gleiche Zeichen in einer Sequenz aufeinander folgen.[Gmb12] Auch für Texte eignet sich diese Methode eher nicht, da hier zwar Farben eine untergeordnete Rolle spielen, jedoch die aufeinander folgenden Buchstaben sehr selten gleich sind und so einzelne Zeichen nicht effizient zusammengefasst werden können.[Lan21, Seite 61]

Bei der **Kompression** werden identische Zeichen so lange eingelesen, bis sich eines ändert - Zeichen und Anzahl die gleich sind, werden festgehalten. Während der **Dekompression** wird der festgehaltene Wert ausgelesen und die dementsprechende Anzahl an Bits ausgegeben.[Gmb12] Der von der Informationsquelle gesandte Datensatz ist verlustfrei wieder hergestellt.

Sendet die Quelle neben beispielsweise Buchstaben auch Ziffern, muss das Verfahren um Sonderzeichen erweitert werden, so dass die resultierende Angabe weiterhin eindeutig als Lauflänge gekennzeichnet ist. [Lan21, Seite 62] Ein Beispiel zur Verdeutlichung findet sich in Abbildung 3.2.

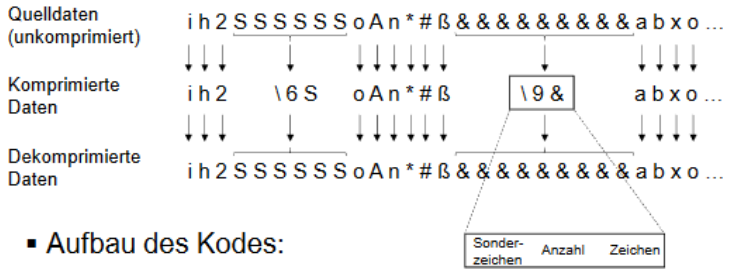

Abbildung 3.2: Beispiel Verfahrenserweiterung
[Lan21, Seite 62]

In der Abbildung 3.2 enthält der Datensatz Buchstaben, Ziffern und Zeichen. Damit dennoch sinnvolle Sequenzen zusammengefasst werden und der Speicherbedarf verringert werden kann, wird in diesem Beispiel mit dem Sonderzeichen *backslash* codiert.

Der Kompressionsgrad von Run-Length Encoding hängt stark von der Charakteristik der Quelldaten ab. [Lan21, Seite 62] Wie sich dieser äußert

wird im Abschnitt **Einsatz & Anwendungsbeispiele** verdeutlicht. Folgend werden jedoch erst die Vor- und Nachteile dieser Methode aufgeführt.

3.2 Vor- und Nachteile gegenüber anderen Verfahren

Das RLE-Verfahren zeichnet sich durch seine Geschwindigkeit sowie Einfachheit bei der Implementierung aus und ermöglicht zudem eine signifikante Datenkompression, wie im nachfolgendem Kapitel **Einsatz & Anwendungsbeispiele** veranschaulicht wird. Zudem gehört sie zu den verlustfreien Kompressionsmethoden, die bei der Dekomprimierung eine vollständige Wiederherstellung des gesendeten Datensatzes ermöglicht.

Auch wenn RLE nennenswerte Vorteile bietet, ist es jedoch zur Kompression komplexer[2] (Farb-)Bilder sowie Texte ungeeignet- auch wenn das Verfahren weiterentwickelt wurde und dadurch die Zunahme des Datenvolumens bei Reihen ohne Wiederholungen kein Problem mehr darstellt, so sind die Kompressionsraten bei solchen Bildern doch denkbar schlecht. [Koe06] Der sinnvolle Einsatz von RLE beschränkt sich daher auf Datenströme, die über einen nennenswerten Anteil an **Runs** verfügt. Bei sehr kleinen Zeichenfolgen kann es passieren, dass sich die Daten sogar vergrößern. Hierbei müsste vorher eine Mindestanzahl von Werten die zusammengefügt werden sollen, gesetzt werden, um zu ermitteln, ab wann bei einem vorgesehen Einsatz die Lauflängencodierung sinnvoll ist oder sogar eine Verschlechterung zu erwarten wäre.

3.3 Einsatz & Anwendungsbeispiele

Die Lauflängencodierung findet Anwendung bei verschiedenen Grafikdateiformaten, wie beispielsweise dem TIFF[3], dem TGA[4]- und dem Bitmap[5]-Dateiformat. In diesem Kapitel werden Beispiele aus unterschiedlichen Bereichen dargestellt, um zu veranschaulichen, wo sich die Lauflängencodierung eignet und wie sie angewandt wird.

Im Bereich der **Computergrafik** wird das RLE- Verfahren bei speicherintensiven Rastergrafiken angewendet und ist besonders effizient, wenn es sich um Grafiken mit wenigen aber großflächigen Farben handelt. Hierbei wird die Methode auf die RGB-Farbwerte übertragen. Der Vorgang kann dabei sogar noch verbessert werden, in dem eine Wegoptimierung durch die Wahl des effektivsten Laufweges anhand eines Algorithmus vorgenommen

[2] komplex hier im Sinne von detailreich oder verrauscht
[3] TIFF = Tagged Image File Format, Standardformat für Druckproduktion
[4] Targa Image File, z.B. Vorschaubild
[5] Grafik, die von einzelnen Pixeln in einem Raster dargestellt werden

wird. Dieser Laufweg könnte beispielsweise zeilen-sequenziell, Zick-Zack oder mäanderförmig[6] (Abbildung 3.3) sein.

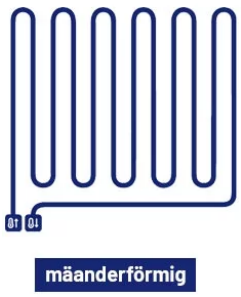

Abbildung 3.3: Wegoptimierung: Mäanderförmig
[hei]

In der Abbildung 3.4 werden zwei 48 Byte Grafiken dargestellt und die Effizienz der Datenkomprimierung unter Anwendung von RLE verglichen. Linksseitig befinden sich die Grafiken untereinander und rechts daneben der zugehörige RLE-Code. Die RGB-Farbwerte stellen die zu sehenden Farben dar - sind viele gleichfarbige Pixel (hier als Kästchen dargestellt) aufeinander folgend, können diese gut zusammengefasst werden, in dem die Anzahl vor dem Farbwert übertragen wird.

In dieser Grafik wird schnell deutlich, dass ein detailreicheres Bild ungeeigneter für die RLE-Codierung ist, da kaum gleichfarbige Pixel aufeinander folgen. Die Datenreduktion beträgt im oberen Grafikbeispiel gerade einmal 11%. In der unteren, in der sich sehr viele nebeneinander liegende blaue Pixel befinden, beträgt die Datenreduktion effiziente 55%.

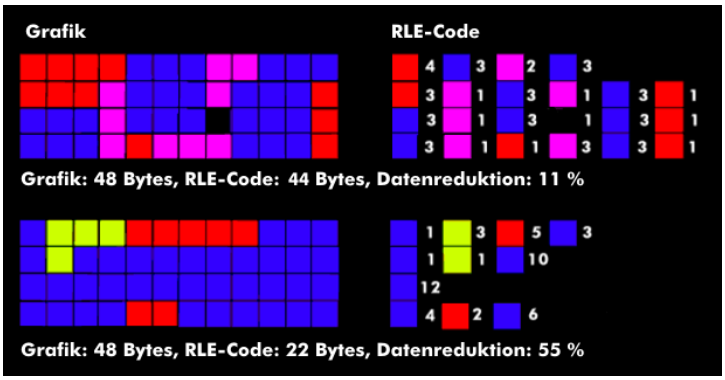

Abbildung 3.4: Beispiel mit geringer und höherer Effizienz
[Gmb12]

[6]Mäander = Bezeichnung einer Flussschlinge in einer Abfolge weiterer Flussschlingen

Diese Art der Kompression lässt sich nicht nur auf Grafiken anwenden, sondern auch auf Zeichenketten, wie beispielsweise die **Abfolge bestimmter Nukleinbasen**. Hierbei werden anstatt der RGB-Farbwerte ASCII-Zeichen genutzt. Im folgenden Beispiel wird die Sequenz der Nukleinbasen (U,G,C,A) zur Verdeutlichung codiert und ausgewertet.[Juc21]

Als Beispieldatensatz dient die Basenabfolge **A G G U A** C G .., wobei auf Grund der Übersichtlichkeit nur die ersten 5 Zeichen codiert und ausgewertet werden. Der dazugehörige *ASCII-Code* lautet dementsprechend **01000001 01000111 01000111 01010101 01000001** und ist bereits bei nur 5 Buchstaben sehr lang. Es wäre nun möglich eine binäre Codierung vorzunehmen, da in diesem Beispiel nur vier unterschiedliche Zeichen (Buchstaben) auftauchen, würden 2 Bit für eine eindeutige Zuordnung ausreichen:

Zahl	Binär	Zeichen
0	00	A
1	01	C
2	10	G
3	11	U

Tabelle 3.1: Binärcodierungstabelle für Beispiel-Zeichen

Dadurch würde sich die betrachtete Sequenz auf **00 10 10 11 00** kürzen. Die Kompressionsrate ist bereits hier merklich gestiegen, wurden zuvor noch 8 Bit benötigt, sind es jetzt nur noch 2, was bedeutet, dass der erforderlicher Speicherplatz auf 25% komprimiert werden konnte. Jedoch darf hierbei nicht vergessen werden, dass die Zuordnungstabelle bisher nur dem Sender bekannt ist, diese muss für eine Dekompression allerdings auch dem Empfänger bekannt sein - Daher muss die Tabelle beispielsweise im Dateikopf mit übertragen werden. Das entspricht daher einen tatsächlichen Bedarf von $4 * (2 + 8)$ Bit $= 40$ Bit allein für die Tabelle - bis hier würde sich bei dieser Übertragung wider erwartend eher ein Nachteil ergeben (Bei der Kürze des hier genutzten Datensatzes).

Da die binäre Codierung ineffizient zu sein scheint, wird nun zum Vergleich mit der **RLE-Methode** codiert:

Code	Zeichen	Code	Zahl
00	A	00	1
01	C	01	2
10	G	10	3
11	U	11	4

Tabelle 3.2: RLE-Codiertabelle für Beispiel-Zeichen

Daraus ergibt sich **1**A **2**G **1**U **1**A. Es wird deutlich, dass sich der zu übertragende Datensatz merklich verkürzt hat, in dem die jeweilige Anzahl der aufeinanderfolgenden Buchstaben angegeben wird. Bei der Übertragung wird zuerst die Anzahl und dann das dazugehörige Zeichen gesendet.Der Speicherbedarf reduziert sich hierbei auf 32 Bit + Dateikopf. Auf den ersten Blick wüsste man jedoch nicht, ob es sich gerade um eine Zahl oder einen Buchstaben handelt. *Merke:* Ob beispielsweise eine 01 als C oder als 2 interpretiert wird, wird durch die Position der Zeichenkette festgelegt.

Es empfiehlt sich die RLE-Methode zu optimieren, um so noch effizientere Ergebnisse zu erzielen. Hierbei werden die Einsen vor den Buchstaben entfernt, da diese unnötig Speicherkapazität brauchen und zudem keinen Beitrag zur Aussagekraft leisten. Des Weiteren bietet sich die Idee an, eine gemeinsame Codiertabelle mit 3 Bit zu erstellen, um Verwechslungen gänzlich zu vermeiden:

Binär	Zeichen
000	A
001	C
010	G
011	U
100	2
101	3
110	4

Tabelle 3.3: gemeinsame Codiertabelle

Daraus resultiert ein komprimierter Datenstrom mit A **2**G U A . Die Größe des Dateikopfes verringert sich ebenfalls, da die Tabelle, die zum Decodieren mit gesandt werden muss, nun kleiner ist: $6 * 3$ Bit $= 18$ Bit + Dateikopf.

Die Beispiele könnten in unterschiedlichsten Ausmaßen weiter geführt werden, jedoch liegt der Fokus hierbei darauf, ein Verständnis der Funktionsweise in unterschiedlichen Anwendungsgebieten zu vermitteln. Ebenfalls konnten durch die aufgeführten Beispielszenarien bereits erste Vor- und Nachteile angedeutet werden, welche im nächsten Abschnitt zusammengefasst werden.

Kapitel 4

Fazit

An den aufgeführten Anwendungsbeispielen wird deutlich, dass dieses Codierverfahren nicht perfekt ist, jedoch bereits einen ursprünglich hohen Speicherplatzbedarf deutlich komprimieren und der Datensatz ohne Verluste wieder dekomprimiert werden kann.

Derzeit ist leider noch kein perfektes Codierverfahren bekannt - aber vielleicht kommt dieses noch!

Kapitel 5

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Belegarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Berlin, den 18. Januar 2022

Monique Golnik

Abbildungsverzeichnis

Tabellenverzeichnis

Quellenverzeichnis

[Bro21] Benjamin Bross. *Bildkompression*. deutsch. Frauenhofer Heinrich-Hertz Institute. Mai 2021. URL: https://moodle.htw-berlin.de/pluginfile.php/1216458/mod_resource/content/2/MMK_SoSe2021_3_Bildkompression.pdf.

[Gmb12] DATACOM Buchverlag GmbH. „Lauflängencodierung“. In: (Okt. 2012). URL: https://www.itwissen.info/RLE-run-length-encoding-Lauflaengencodierung.html.

[hei] kocer heiztech. „Funktionsweise einer Fußbodenheizung“. In: (). URL: https://www.kocer-heiztech.at/blog/fussbodenheizungen/.

[Juc21] Frank Juchim. *Kompression und Lauflängencodierung - Codierung und Datenübertragung*. Apr. 2021. URL: https://www.youtube.com/watch?v=DRH1eAd1Psk.

[Koe06] Mirjam Koeck. *RLE - Runlength-Encoding*. Deutsch. Digital Media for Artists. Apr. 2006. URL: http://www.dma.ufg.ac.at/app/link/Grundlagen:2D-Grafik/module/12946?step=all.

[Lan21] Prof. Dr. Christoph Lange. *Einführung in die Informations- und-Kodierungstheorie I*. deutsch. HTW. Nov. 2021. URL: https://moodle.htw-berlin.de/pluginfile.php/1361027/mod_resource/content/1/SNT_HTW_Lange_WS_2021_22_04_InfKodTheorie_1.pdf.

[SE21] IONOS SE. „Datenreduktion durch Deduplikation und Kompression“. In: *Digital Guide IONOS by 11* (Jan. 2021). URL: https://www.ionos.de/digitalguide/server/knowhow/datenreduktion-durch-deduplikation-und-kompression/.