technische universität
dortmund

Fakultät für Informatik
Lehrstuhl für Algorithm Engineering (LS11)
http://ls11-www.cs.tu-dortmund.de

Textkompression mithilfe einer Variante von LZ78

Bachelorarbeit

Florian Kleine
Matrikelnummer: 157020

24. August 2016

Betreuer:
Prof. Dr. Johannes Fischer
Dominik Köppl

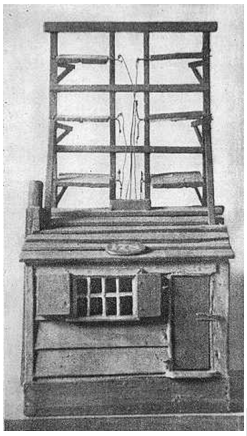
Inhaltsverzeichnis

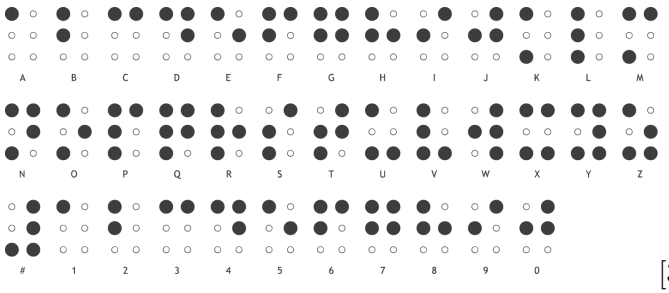
1 Einleitung

Die Datenkompression spielt in der Informatik eine große Rolle. Zwar sind die Datenträger im Vergleich zu früher um ein Vielfaches größer und vor allem günstiger geworden, stoßen bei den heute anfallenden riesigen Datenmengen aber immer noch an ihre Grenzen. Deshalb ist es sinnvoll die Daten mit geschickten Verfahren so zu komprimieren, dass sie später verlustfrei in den Ursprungszustand zurückübersetzt werden können. In dieser Arbeit soll es darum gehen, Texte mithilfe einer Variante des Lempel-Ziv78-Verfahrens in Faktoren zu zerlegen, zu kodieren und so verlustfrei zu komprimieren. Dieses Verfahren stützt sich auf der Eliminierung von Redundanzen, indem Teile des Textes durch Verweise auf vorher auftretende gleiche Teile ersetzt werden. Solche Verweise benötigen weniger Speicher als der Text, was so zu einer Kompression des kompletten Textes führt. Das aus dieser Arbeit entstehende Verfahren wird anschließend mit bereits vorhandenen Kompressionsverfahren (z.B. gzip und 7zip) verglichen.

1.1 Anekdote

Um die Bedeutung von Datenkompression zu verdeutlichen und zu zeigen, wie lange man sich über dieses Thema schon Gedanken macht, hier eine kleinen Anekdote:
Ende des 18. Jahrhunderts benötigte die britische Marine eine schnelle Kommunikation zwischen London und dem Marinestützpunkt an der Küste. Dies wurde durch eine Kette von Hütten auf Hügeln in etwa 5 Meilen Entfernung umgesetzt. Diese Hütten(sog. *Klappentelegraphen*1) hatten 6 Klappen auf dem Dach, die entweder geschlossen oder geöffnet waren. Durch diese 64 möglichen Kombinationen von offenen und geschlossenen Klappen konnte man Buchstanden des Braille-Alphabets1 darstellen. Da das Alphabet weniger als 64 Buchstaben hat, gab es freie Kombinationen von Klappenstellungen. Diese freien Kombinationen wurden dazu benutzt die Kommunikation zu beschleunigen. Oft benutzen Wörtern wie 'for' oder 'the' wurden freie Kombinationen zugewiesen. Anderen Worten wie 'father' oder 'mother' wurde die freie Kombination 'dot5' und die Kombination für 'f' bzw. 'm' zugewiesen. So erreichte man eine Kompression von 20% bei englischen Texten.[1, S.1-4]
Dieses Problem hat sich bis heute nicht geändert. Wie bekommen wir die Daten möglichst klein und somit die Übertragung möglichst schnell, ohne Informationen zu verlieren?


[2]



[3]

Abbildung 1: Links sieht man einen Klappentelegraphen. Rechts das englische Braille-Aphabet.

2 Theoretische Grundlagen

Im Folgenden werden Datenstrukturen und Operationen eingeführt, die wir in dieser Arbeit benötigen werden.

2.1 Operationen auf Strings

Sei das Alphabet Σ definiert als eine Menge von Zeichen, dann bezeichnet Σ^* die Menge aller Worte, die aus dem Alphabet gebildet werden können. Jedes dieser Worte bezeichnet man als String. Sei s ein String mit der Länge n.

2.1 Definition (Länge) Sei $|s|$ die Länge des String s, das heißt die Anzahl von Zeichen in s.

2.2 Definition (Leerstring) Sei $\varepsilon \in \Sigma^*$ der leere String. Es gilt $|\varepsilon| = 0$.

2.3 Definition (Symbolzugriff) Für $x \in \mathbb{N}$ und $x \leq n$ sei $s[x]$ das x-te Zeichen aus s.

2.4 Definition (Teilstring) Für $x, y \in \mathbb{N}$ und $1 \leq x < y \leq n$ sei $s[x, y]$ die Zeichenfolge vom x-ten bis zum y-ten Zeichen aus s. s[x] und s[y] einschließlich.

2.5 Definition (Suffix) Für $x \in \mathbb{N}$ und $x \leq n$ sei $s[x..]$ das x-te Suffix von s. Also gilt $s[x..] = s[x, n]$.

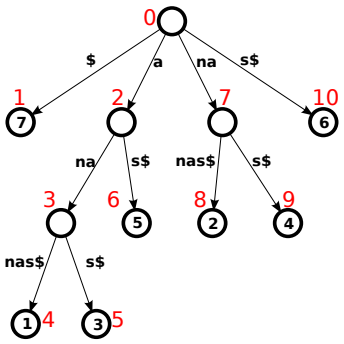
2.2 Suffix-Tree

2.6 Definition (Suffix-Tree) Ein Suffix-Tree eines Strings s ist ein Baum mit n Blättern. Alle inneren Knoten erfüllen folgende Bedingungen:

- Jeder Knoten hat mindestens 2 Kinder.
- Jede Kante ist mit einem nicht-leeren Teilstring von s markiert.

- Die Markierung ausgehender Kanten eines Knotens beginnen nicht mit dem gleichen Zeichen.
- Die Konkatenation von allen Zeichen auf dem Pfad von der Wurzel zum Blatt i ist das i-te Suffix $s[i..]$ von s.

2.7 Beispiel (Suffix-Tree) $s =$ `ananas$`



[4]

Abbildung 2: Diese Abbildung zeigt den Suffix-Tree zu $s =$ `ananas$`. Die Blätter sind hierbei nummeriert. Der String auf dem Pfad von der Wurzel zum Blatt i repräsentiert das Suffix $s[i..]$. Die roten Zahlen an den Knoten sind durch eine Pre-Order-Nummerierung entstanden und sind die IDs der Knoten.

3 Idee der LZ78 Variante

Um die Grundidee des Verfahrens, das wir in dieser Arbeit entwickeln, zu erläutern, betrachten wir im Folgenden zunächst LZ78.

3.1 LZ78

LZ78 wurde 1978 von Jacob Ziv und Abraham Lempel erfunden und ist ein Verfahren zur Textkompression. Es benutzt dabei ein *adaptives Wörterbuch*. Das heißt es wird je nach Eingabetext ein Wörterbuch erzeugt. Dieses Wörterbuch besteht aus Faktoren, die auf vorherige, längste Vorkommen des selben Substrings verweisen und an diesen ein Zeichen anhängen. Diese Verweise benötigen weniger Speicher als der Originaltext, wodurch es zu einer Kompression kommt.[5, S.53ff]

Der (naive) Algorithmus durchläuft den String bzw. den Eingabetext T und ersetzt Redundanzen durch Verweise auf das längste vorherige Vorkommen des selben Teils und hängt das nächste Zeichen an diesen Verweis an. Dadurch wird ein Baum erzeugt, der sog. LZ78-Trie. Dieses Verfahren wird nun am Beispiel von $T =$ `ananas$` verdeutlicht.

	1	2	3	4	5
Textteil	a	n	an	as	$
Faktorzerlegung	(-,a)	(-,n)	(1,n)	(1,s)	(-,$)

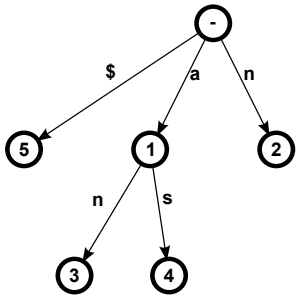


Abbildung 3: Links sieht man die Faktorisierung nach LZ78 und rechts den entsprechenden LZ78-Tree. Der i-te Faktor (x, s) wird zu einem neuen Knoten i mit x als Elternknoten und die Kante (x, i) wird mit s beschriftet.

3.1 Definition (Faktor (LZ78)) Ein *Faktor* ist ein Tupel (x, s) mit $x \in \{1..z\} \cup \{-\}$ und $s \in \Sigma$. x bezeichnet hierbei den Elternknoten im LZ78-Tree und s die Kantenbeschriftung der Kante von x zum neu entstehenden Knoten. z ist die Anzahl der Faktoren. Die Zahlen in den Knoten stehen für die Faktoren.

3.2 LZ78 Variante

Im Gegensatz zu LZ78 kann diese Variante (im Folgenden LZ78V genannt) auch mehr Zeichen an eine Ersetzung anhängen. Im Beispieltext $T =$ `ananas$` folgt nach einem `'n'` immer ein `'a'`. Dies spiegelt sich im Suffix-Tree2 durch die Kantenbeschriftung `'na'` wider. Mit LZ78V können wir nun beim ersten Lesen von `'n'` den Faktor (-,`na`) erstellen, denn nach `'n'` kann nichts anderes kommen. Es ergibt sich folgende Faktorzerlegung:

	1	2	3
Textteil	a	na	nas$
Faktorzerlegung	(-,a)	(-,na)	(2,s$)

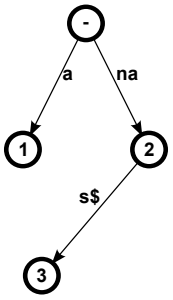


Abbildung 4: Links sieht man die Faktorisierung nach LZ78V und rechts den entsprechenden LZ78V-Tree. Der i-te Faktor (x, s) wird zu einem neuen Knoten i mit x als Elternknoten und die Kante (x, i) wird mit s beschriftet. Im Gegensatz zum LZ78-Tree kann s aber mehr als ein Zeichen beinhalten.

Den LZ78V-Tree können wir mit einem zweidimensionalen Array A repräsentieren. So wäre $A[i][1] = x$ und $A[i][2] = s$ der i-te Faktor (x, s) in der Array-Darstellung. Im Beispiel $T =$ `ananas$` ergibt sich:

	1	2	3
$A[1]$	-	-	2
$A[2]$	a	na	s$

Literatur

[1] Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text Compression*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[2] Bild von einem Klappentelegraphen. https://cms.sachsen.schule/typoecke2/typo-experimente/informationuebertragung-mit-dem-klappentelegraph/was-ist-ein-klappentelegraph/. abgerufen am: 24.08.2016.

[3] Braille-Aphabeth. http://www.pharmabraille.com/pharmaceutical-braille/the-braille-alphabet/. abgerufen am: 24.08.2016.

[4] Visualisierung von Suffix - Trees. http://visualgo.net/suffixtree.html. abgerufen am: 01.08.2016.

[5] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold, New York, NY, USA, 1994.