Datenkompression nach Lempel–Ziv

Christian Näger

14. Januar 1997

1 Der LZ77–Algorithmus

Search buffer Fenster mit bereits codierten Zeichen. In diesem wird dann nach Übereinstimmungen mit den noch zu codierenden Zeichen gesucht.

Look-ahead buffer Fenster, daß sich unmittelbar an den search buffer anschließt und die noch zu codierenden Zeichen enthält.

Bei der *Codierung* wird zunächst der search buffer von rechts nach Übereinstimmungen mit den ersten Zeichen des look-ahead buffers durchsucht. Eine Übereinstimmung ist eine Folge von Zeichen, die im search buffer (und rechts darüber hinaus) gleich ist mit den ersten Zeichen des look-ahead buffers. Die Startposition (von rechts im search buffer) der längsten Übereinstimmung wird als Offset bezeichnet. Die Codierung erfolgt nun durch das Tripel (o,l,c). Dabei ist o der Offset, l die Länge der Übereinstimmung und c das Codewort für das nächste Symbol, das im look-ahead buffer nach der Übereinstimmung folgt. Anschließend wird der search und der look-ahead buffer um l+1 nach rechts bewegt, so daß das letzte kodierte Zeichen das letzte (rechteste) Zeichen des search buffers ist und der look-ahead buffer mit dem ersten nicht-codierten Zeichen beginnt.

Beispiel

Der erste Rahmen ist der search buffer (7), der zweite der look-ahead buffer (6).
... $\boxed{\text{cabraca}}\boxed{\text{dabrar}}$ rarrad ...
(0,0,C(d)) keine Übereinstimmung (d kommt im search buffer nicht vor)
...c $\boxed{\text{abracad}}\boxed{\text{abrarr}}$ arrad ...
(7,4,C(r))
...cabrac $\boxed{\text{adabrar}}\boxed{\text{rarrad}}$ ...
(3,5,C(d)) Vorsicht: hier ist die Länge (5) größer als der Offset (3), die Übereinstimmung geht also über den search buffer hinaus.

Die *Decodierung* erfolgt analog, indem man in der bereits decodierten Sequenz um den Offset zurückgeht und dann zeichenweise Symbole kopiert und hinten an die Sequenz anhängt, solange wie es die Länge im Tripel angibt. Zum Schluß muß noch das letzte Zeichen im Tripel decodiert werden.

Anwendungen (LZ77-Algorithmus mit Variationen): PKZip, Zip, Lharc, ARJ
Problem des Algorithmus: Wiederholungen, die weiter auseinander liegen, als der search buffer lang ist.

2 Die LZ78–Algorithmen

2.1 Der ursprüngliche LZ78–Algorithmus

Bei den LZ78–Algorithmen wird während der *Codierung* bzw. *Decodierung* ein separates aber identisches Wörterbuch aufgebaut.

Es werden Zeichen von der Quelle eingelesen und zu einer Sequenz konkateniert. Wenn ein neues Zeichen eingelesen und konkateniert wird, wird zunächst überprüft, ob die neue Sequenz bereits im Wörterbuch vorhanden ist. Wenn dem so ist, fährt man fort. Wenn nicht, so wird der Index der alten Sequenz und der letzte Buchstabe codiert, und die neue Sequenz wird als neuer Eintrag ins Wörterbuch aufgenommen. Man startet dann wieder mit einer leeren Sequenz.

Beispiel

Quelle: wabba␣wabba␣wabba␣wabba␣woo␣woo␣woo␣

	Wörterbuch			Wörterbuch	
Output	Index	Eintrag	Output	Index	Eintrag
(0,C(w))	1	w	(4,5)	10	ba␣
(0,C(a))	2	a	(9,3)	11	wabb
(0,C(b))	3	b	(8,1)	12	a␣w
(3,2)	4	ba	(0,C(a))	13	o
(0,C(␣))	5	␣	(13,5)	14	o␣
(1,2)	6	wa	(1,13)	15	wo
(3,3)	7	bb	(14,1)	16	o␣
(2,5)	8	a␣	(13,13)	17	oo
(6,3)	9	wab			

2.2 LZW, die bekannteste Modifikation des LZ78

Im Gegensatz zu LZ78 wird hier das Wörterbuch mit den Elementen aus dem Alphabet der Quelle initialisiert. Der weitere Aufbau des Wörterbuchs geschieht ähnlich wie bei LZ78. Die Codierung erfolgt jedoch nicht mehr als Paar sondern nur noch durch den Index im Wörterbuch.

Beispiel

Alphabet: ␣, a, b, o, w
Quelle: wabba␣wabba␣wabba␣wabba␣woo␣woo␣woo␣
Output: 5 2 33 2 1 6 8 10 12 9 11 7 16 5 4 4 11 21 23 4

Index	Eintrag	Index	Eintrag	Index	Eintrag	Index	Eintrag
1	␣	8	bb	15	wabb	21	oo
2	a	9	ba	16	ba␣	22	o␣
3	b	10	a␣	17	␣wa	23	␣wo
4	o	11	␣w	18	abb	24	oo␣
5	w	12	wab	19	ba␣w	25	␣woo
6	wa	13	bba	20	wo		
7	ab	14	a␣w	21	oo		

Da nur der Output übertragen wird (und nicht das ganze Wörterbuch), muß bei der *Decodierung* darauf geachtet werden, daß wieder das exakt gleiche Wörterbuch wie bei der *Codierung* aufgebaut wird. Dies geschieht analog zum Aufbau des Wörterbuchs bei der *Codierung*. Lediglich in dem Fall, in dem auf einen Wörterbucheintrag zugegriffen wird, der noch nicht vollständig im Wörterbuch vorhanden ist, muß eine Ausnahmebehandlung erfolgen. Der fehlende Eintrag wird dabei durch den bereits vorhandenen Teil des Eintrags vervollständigt. (Beispiel: Codierung und anschließende Decodierung der Sequenz abababa ...)

Anwendungen

Dateikompression mit Unix `Compress` Man beginnt mit einer Wörterbuchgröße von 512 Einträgen, d. h. die Codewörter für die Indizes sind 9 bit lang, und verdoppelt die Größe des Wörterbuchs, sobald es voll ist.

Bildkompression im GIF (Graphic Interchange Format) Ursprünglich von Compuserve Information Service entwickelt, fand GIF schnell weite Verbreitung v. a. auch durch das WWW. Bei digitalisierten Bilder ist GIF jedoch relativ ineffizient.

Kompression bei Modemübertragung durch V.42 bis Internationaler Standard der Modemkommunikation umfaßt LZW–Kompression und Fehlerkorrektur. Die Kompression erfolgt durch flexible (verdoppelbare) Wörterbuchgrößen. Sie wird jedoch bei sich nicht wiederholenden Quellen (z. B. weil die Quelle schon komprimiert war) abgeschaltet, um Datenexpansion zu vermeiden.

3 Zusammenfassung

Die Lempel–Ziv–Algorithmen profitieren generell von sich wiederholenden Mustern in der Originaldatei. Solche Wiederholungen kommen vor allem in Textdateien, Computerprogrammen, bei Computer–Computer Kommunikation und in computergenerierten Bildern vor. Diese Anwendungsbereiche machen die LZ–Algorithmen zu einem der effektivsten und weitverbreitetsten Methoden der Datenkompression. Es gibt jedoch (meist natürliche) Quellen, bei denen kaum Wiederholungen vorkommen (gescannte Bilder, digitalisierte Sprache etc.). In diesen Fällen muß auf andere Kompressionsalgorithmen ausgewichen werden, die an diese Anforderungen besser angepaßt sind (JPEG, MPEG Audio etc.).

4 Quelle

K. Sayood Data Compression. Morgan Kaufmann, 1996