

Verlustfreie Datenkompression durch Blocksortieren nach Burrows und Wheeler

Zur Geschichte

Das bekannte Verfahren von Ziv, Lempel und Welch (LZW-Kodierung), das seit den 1970er Jahren für die verlustfreie Kompression von Zeichenketten entwickelt worden ist, arbeitet in mehrererlei Hinsicht nicht optimal. Einerseits muß das Wörterbuch Ummengen von unwichtigen, nie wiederkehrenden Teil-Zeichenfolgen speichern; andererseits kann eine häufig vorkommende n -stellige Zeichenfolge erst nach ihrem n -ten Auftreten wirksam komprimiert werden ...

Das hier vorgestellte Verfahren ist seit 1994 bekannt; es wurde von Michael Burrows und David Wheeler in Cambridge entwickelt. Mithilfe dieser Methode lassen sich bspw. für Textdateien Rekord-Komprimierungsraten erzielen.

Das sehr elegante und auch effiziente Verfahren ist nicht patentiert; es hat Eingang in das Komprimierungstool *bzip2* gefunden.

Grundidee

Ein beliebig langer Zeichenstrom, der komprimiert werden soll, wird in Blöcke konstanter Länge zerlegt, die jeder für sich kodiert werden.

Die Burrows-Wheeler-Transformation (BWT) selbst – die das Kernstück des Algorithmus darstellt – komprimiert die Zeichenfolge noch nicht, sondern permutiert die Zeichen derart, daß ähnliche lokale Kontexte gleicher Zeichen/Zeichenfolgen gruppiert werden.

So können dann andere, herkömmliche (sehr einfache) Komprimierungstechniken, auf dem Zwischenergebnis aufbauend, gute Komprimierungsraten erzielen. Es sind dies vor allem Lauflängen- und *move-to-front*-Kodierung, bevor schließlich eine arithmetische oder Huffman-Kodierung den Rest besorgt. Gute Ergebnisse werden für möglichst große n erzielt; die Blöcke sollten in praxi schon einige Kilobytes groß sein.

Im folgenden wird die Hin- und Rücktransformation vorgestellt.

Die Burrows-Wheeler-Transformation $w \mapsto v$

Gegeben sei also ein Block der Länge n , d. h. ein n -Tupel w von Zeichen x_i eines Alphabets ($i = 1, \dots, n$):

$$w = x_1x_2\dots x_n$$

Gesucht ist eine Permutation $i \mapsto j(i)$ der Indizes $i \in \{1, \dots, n\}$ derart, daß nachfolgende Komprimierungsverfahren die Zeichenfolge $v = x_{j(i)}$ möglichst effizient weiterverarbeiten können.

Zunächst werden alle zyklischen Verschiebungen von w ermittelt,¹ man denke sich dies anhand einer $n \times n$ -Matrix M :

$$M = \begin{bmatrix} x_1 & x_2 & \cdots & x_{n-1} & x_n \\ x_2 & x_3 & \cdots & x_n & x_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n-1} & x_n & \cdots & x_{n-3} & x_{n-2} \\ x_n & x_1 & \cdots & x_{n-2} & x_{n-1} \end{bmatrix}$$

In *jeder* Zeile kommt offenbar jedes der Zeichen x_i vor; ebenso in jeder Spalte. Dies gilt auch für den Fall, daß die Zeilen (oder Spalten) von M permutiert werden sollten.

Die Grundidee ist nun, die Zeilen von M zu sortieren,² wodurch eine neue Matrix M' definiert ist. Die Sortier-Permutation der Zeilen ist dabei in nichttrivialen Fällen eindeutig bestimmt, denn dann sind alle Zeilen verschieden.³

Es ist klar, daß durch eine solche Sortierung gleiche bzw. ähnliche Zeilenanfänge (die ja das Kriterium für die Sortierung liefern) nah beieinander gruppiert werden. Dies wird aber – weil die meisten häufig vorkommenden Teilfolgen (Phrasen) eine Länge > 2 haben – auch für die Zeilenenden gelten.⁴

Daher wird nun als Ergebnis der Transformation für jede Zeile der sortierten Matrix M' das jeweils *letzte* Zeichen der Zeile ausgegeben, es entsteht eine n -stellige Folge $(y_i)_{i=1}^n$ der x_i . Viele aufeinanderfolgende y_i sind dabei identisch.

Wie schon die obige Überlegung gezeigt hat, sind in jeder Zeile und in jeder Spalte alle Zeichen x_i anzutreffen. Dies gilt nun insbesondere für die erste Spalte F der Matrix M' ; dort liegen alle Zeichen sortiert vor. Es gilt aber auch für die letzte Spalte L von M' ; die Zeichen sind hier nicht mehr sortiert, aber aus der Permutation⁵ in L kann die ursprüngliche Zeichenfolge $(x_i)_{i=1}^n$ wieder rekonstruiert werden (näheres hierzu im Kapitel zur Rücktransformation).

Als **Ergebnis** der Burrows-Wheeler-Transformation kann die Folge $v = L = (y_i)_{i=1}^n$ angegeben werden. Zusätzlich muß noch diejenige Zeile gemerkt werden, die die ursprüngliche (nicht-rotierte) Zeichenfolge w enthält; dies kann durch Angabe ihres Zeilenindex k innerhalb M' geschehen.

Die Burrows-Wheeler-Rücktransformation $v \mapsto w$

Das Problem der inversen Transformation stellt sich wie folgt:

Gegeben ist nun eine Zeichenkette $v = (y_i)_{i=1}^n$ und ein Index k .

Gesucht ist die ursprüngliche Zeichenkette $w = x_1x_2\dots x_n$.

Mit v ist genau die letzte Spalte L der unbekanntenen Matrix M' (vgl. Kapitel zur Vorwärts-Transformation) gegeben. Durch Sortieren der y_i erhält man genau

¹In Implementierungen dieses Algorithmus werden diese "Rotationen" kaum explizit gespeichert werden, kann man doch einfach (mittels modulo-Operationen) den Zugriff auf eine rotierte Zeichenkette simulieren. Auf die rotierten Zeichenketten wird nämlich nur lesend zugegriffen, d. h. sie verändern sich nicht. Das Matrixelement α_{ij} enthält jedenfalls das Zeichen $\alpha_{ij} := x_{((i+j-2) \bmod n)+1}$.

²Offenbar muß auf dem Alphabet der Zeichen von w eine Ordnung definiert sein, üblicherweise die alphabetische.

³Im trivialen Fall sind alle Zeilen gleich, denn alle Zeichen x_i sind dann identisch und die Permutation ist beliebig.

⁴Genau dieser Tatbestand wird im weiteren als Basis für die Komprimierung durch andere Verfahren genutzt.

⁵Man weiß ja, daß F und L dieselben Zeichen (nur in jeweils anderer Permutation) enthalten, und daß F sortiert ist!

die erste Spalte F von M' . Unbekannt ist noch die Abfolge der gesuchten Zeichenkette w .

Nun läßt sich ein trickreiches Verfahren anwenden, dessen Hauptschleife wie folgt funktioniert:

- Man befindet sich in Zeile j der unvollständigen Matrix M' (am Anfang mit dem Startwert $j := k$ initialisiert).
- Das Zeichen $z = F(j)$ (also die j -te Komponente von F bzw. das erste Zeichen der j -ten Zeile in M') wird ausgegeben.⁶
- Wir befinden uns in der j -ten Zeile und haben dort das a -te Vorkommen von z innerhalb der Spalte F (von oben herab gezählt). Nun suchen wir in der Spalte L das a -te Vorkommen von z (von oben herab gezählt); die so gefundene Zeile sei das neue j .

Durch die n Teilphrasen der Länge 2, die (zeilenweise in M') durch das Aufeinanderfolgen⁷ von $L(j)$ und $F(j)$ festgelegt sind, hat man sich durch die ursprüngliche Zeichenkette w gehandelt und diese damit rekonstruiert.

Die Schleife terminiert nach n Iterationsschritten. Die sukzessiv erzeugte Ausgabe stellt genau die ursprüngliche Zeichenkette w dar.

Ein Beispiel

$$w = ANANAS$$

Vorwärts-Transformation: Die quadratische Matrix der zyklischen verschobenen Worte w sieht hier wie folgt aus:

$$M = \begin{bmatrix} A & N & A & N & A & S \\ N & A & N & A & S & A \\ A & N & A & S & A & N \\ N & A & S & A & N & A \\ A & S & A & N & A & N \\ S & A & N & A & N & A \end{bmatrix} \begin{matrix} (1) \\ (4) \\ (2) \\ (5) \\ (3) \\ (6) \end{matrix}$$

Die zeilen-sortierte Matrix M' sieht folgendermaßen aus:⁸

$$M' = \begin{bmatrix} A & N & A & N & A & S \\ A & N & A & S & A & N \\ A & S & A & N & A & N \\ N & A & N & A & S & A \\ N & A & S & A & N & A \\ S & A & N & A & N & A \end{bmatrix} \begin{matrix} (1) \\ (2) \\ (3) \\ (4) \\ (5) \\ (6) \end{matrix}$$

$F \qquad \qquad \qquad L$

Wir müssen uns als Ausgabe zusätzlich zur letzten Spalte⁹

$$L = SNNA AA =: v$$

den Zeilenindex der Originalzeichenkette w merken, in diesem Fall $k = 1$.

(Leicht ist zu sehen, daß v optimal lauffängen- oder MTF-kodierbar ist.)

⁶Für die initiale Konstellation $j = k$ ist dies offenbar das erste Zeichen x_1 der ursprünglichen Zeichenkette w .

⁷durch die zyklische Verschiebung von w

⁸nur zufällig ist hier die Original-Zeichenkette w ausgerechnet in der ersten Zeile stehengeblieben

⁹nur in diesem Beispiel ist L hier zufällig absteigend sortiert

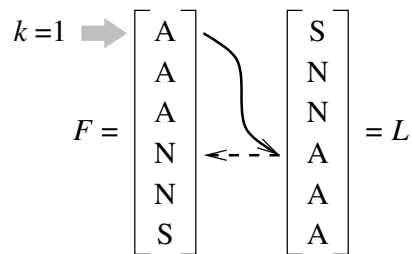
Rückwärts-Transformation: Gegeben ist $v = S N N A A A$ und der Startindex $k = 1$.

Durch Sortierung von $L := v$ ist auch die erste Spalte F von M' bekannt:

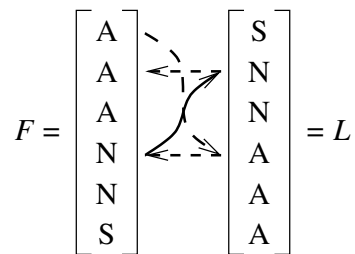
$$F = \begin{bmatrix} A \\ A \\ A \\ N \\ N \\ N \\ S \end{bmatrix} = L = \begin{bmatrix} S \\ N \\ N \\ A \\ A \\ A \\ A \end{bmatrix}$$

Man setzt $j := k(=1)$, gibt $z := F(j) = F(1) = A$ aus; man findet dieses erste A (der Spalte F) als erstes A innerhalb der Spalte L zuerst in Zeile $j = 4$. (a)

Man gibt $z := F(j) = F(4) = N$ aus; (b) man findet dieses erste N (der Spalte F) als erstes N innerhalb der Spalte L zuerst in Zeile $j = 2$.



a)



b)

Und so weiter ...

Schließlich hat man sukzessive die Zeichen A, N, A, N, A, S ausgegeben, was der Original-Zeichenkette $w = A N A N A S$ entspricht.