Datenkomprimierung

Ronald Hasenberger, TU-Wien SON-004

Teil 1: Allgemeines

Bereits seit einiger Zeit sind zahlreiche Programme in Verwendung, welche die Komprimierung von Daten ermöglichen. Ich möchte in dieser Artikelgruppe auf die Grundlagen der Datenkomprimierung eingehen und im Anschluß daran grundsätzliche Verfahren beschreiben.

Arten von Datenkomprimierung

Grundsätzlich können 2 unterschiedliche Arten von Datenkomprimierung unterschieden werden: solche, bei denen die gesamte Information erhalten bleibt und solche, bei denen zwischen relevanter und irrelevanter Information unterschieden und der Verlust von letzterer akzeptiert wird. Im Bereich der Computertechnik im engeren Sinne (Datenbanken, Textverarbeitung,...) gibt es üblicherweise keine irrelevante Information, sodaß nur verlustfreie Verfahren eingesetzt werden können.

Im Bereich der Audio- und Videoverarbeitung (und damit auch im multimedialen Bereich der Computertechnik) kann in der Regel aber sehr wohl auf Teile der Information verzichtet werden, sodaß dort nicht verlustfreie Verfahren eingesetzt werden (z.B. bei der DCC).

Aufbau der Artikelgruppe

Der erste Artikel (Grundlagen der Datenkomprimierung, diese Ausgabe der PC·NEWS) behandelt zunächst allgemein die mathematischen Grundlagen zur (verlustfreien) Datenkomprimierung, wobei ich auf Begriffe wie den Informationsgehalt von Symbolen und Datenquellen eingehe. Das Ergebnis dieses Artikels ist im wesentlichen, daß bei einer Codierung der Daten mit konstanter Bitanzahl (z.B. 8 Bit für ein extended ASCII-Zeichen) mehr Speicherplatz[3] als notwendig belegt wird. Wem diese Aussage reicht und die Mathematik zu trocken ist, kann gleich beim dem zweiten Artikel (Quellcodierung und Anwendungen, PC·NEWS-38) weiterlesen.

In diesem Artikel werden grundlegende Verfahren der Quellcodierung (Shannon Fano und Huffman) beschrieben und mit der Kodierung mit konstanter Bitanzahl anhand eines Beispieles verglichen. Diese Verfahren sind die Grundlage von zahlreichen Datenkomprimierungsprogrammen, werden dort aber in etwas abgewandelter Form verwendet. Deshalb gehe ich in diesem Artikel außerdem auf die Einschränkungen dieser Verfahren ein und werde auch das Verfahren, welches die Firma Microsoft für DBLSPACE eingesetzt hat, kurz vorstellen.

Im letzten Artikel (Verlustbehaftete Verfahren der Datenkomprimierung, PC·NEWS-39) behandle ich verlustbehaftete Datenkomprimierungsverfahren, für Audio- (insbesondere Sprach-) und Videosignale.

Anmerkung: Diese Artikelserie kann auch geschlossen über den Literaturdienst als SON-004 bestellt werden.

[3] Ich spreche hier vereinfachend von Speicherplatz, letztendlich wird die Anzahl der Symbole verringert, die ich zur Darstellung benötige, und damit jeglicher Aufwand, der mit Speicherung und Datenübertragung verbunden ist.

Teil 2: Grundlagen der Datenkomprimierung

Mathematische Grundlagen

Diese sind vor allem für die verlustfreien Verfahren, wie sie im nächsten Artikel behandelt werden, von Interesse, da für diese Verfahren der Informationsgehalt (siehe in Kürze) nicht verändert werden darf und daher die effizienteste Kodierung für einen bestimmten Informationsgehalt gesucht werden muß (→ Quellcodierung).

Bei den verlustbehafteten Verfahren (siehe übernächster Artikel) wird der Informationsgehalt bewußt verändert; hier sind vor allem die subjektiven Auswirkungen der Veränderungen wesentlich, und die sind mathematisch nur schwer erfaßbar.

Informationsgehalt von Symbolen

Den Informationsgehalt eines Symbols kann man sich als die Überraschung, die beim Auftreten eines bestimmten Symbols auftritt[4] veranschaulichen. Konkret ist der Informationsgehalt eines Symbols m_i definiert als:

$$l[m_i] = ld \frac{1}{P[m_i]} = -ld\, P[m_i] \qquad \text{Formel 1}$$

mit $P[m_i]$: Wahrscheinlichkeit des Auftretens von m_i

Die Wahl der Basis des Logarithmus ist willkürlich, bei Wahl der Basis 2 ergibt sich die Einheit des Informationsgehalts zu 1 Bit[5].

Der Logarithmus erweist sich zumindestens insofern als sinnvoll, als sich dadurch der Informationsgehalt eines zusammengesetzten Symbols als Summe der Informationsgehalte der Einzelsymbole ergibt[6]. Durch

[4] D.h. desto wahrscheinlicher das Auftreten eines Symbols ist, desto geringer ist der Informationsgehalt dieses Symbols. Man kann sich diese Aussage plausibel machen, indem man den Informationsgehalt einer Alarmleitung zur Feuerwehr betrachtet: Der Zustand "kein Alarm" ist der normale und enthält nur wenig Information, da ohnehin niemand (ernsthaft) mit Feuer rechnet. Der Zustand "Feueralarm" hingegen hat, mit der entgegengesetzten Argumentation, einen hohen Informationsgehalt; auch wenn von den Handlungsnotwendigkeiten abgesehen wird.

[5] Die Identität mit dem Bit, welches in der Digitaltechnik verwendet wird, ist auch nicht zufällig: ein binäres (zweiwertiges) Signal mit gleichen Wahrscheinlichkeiten für "1" und "0" hat den Informationsgehalt

$$l["1"] = -ld\, P["1"] = -ld\, 0{,}5 = 1$$

$$l["0"] = -ld\, P["0"] = -ld\, 0{,}5 = 1$$

[6] Bei statistischer Unabhängigkeit der Einzelsymbole ergibt sich die Wahrscheinlichkeit eines zusammengesetzten Symbols zu

$$P[m_0, m_1, \ldots, m_n] = \prod_{i=0}^{n} P[m_i]$$

und damit der Informationsgehalt des zusammengesetzten Symbols zu

$$l[m_0, m_1, \ldots, m_n] = -ld\, P[m_0, m_1, \ldots, m_n] =$$

$$-ld \prod_{i=0}^{n} P[m_i] = \sum_{i=0}^{n} -ld\, P[m_i] = \sum_{i=0}^{n} l[m_i]$$

Damit ergibt sich für ein aus 2 binären Signalen mit gleichen Wahrscheinlichkeiten für "0" und "1" ein Informationsgehalt von

$$l[0,0] = l[0] + l[0] = 2,\ l[0,1] = l[0] + l[1] = 2,$$

$$l[1,0] = l[1] + l[0] = 2 \text{ und } l[1,1] = l[1] + l[1] = 2.$$

diese Definition hat ein aus 2 binären Signalen mit gleichen Symbolwahrscheinlichkeiten für "0" und "1", wie man es auch erwarten würde, einen Informationsgehalt von 2 Bit.

Informationsgehalt einer Datenquelle[7]

Der mittlere Informationsgehalt H_S einer Datenquelle ist definiert als Mittelwert der Informationsgehalte der von der Quelle gelieferten Symbole[8]

$$H_s = \sum_i P[m_i] \cdot l[m_i] = \sum_i -P[m_i] \cdot \mathrm{ld}\, P[m_i]. \quad \text{Formel 2}$$

Die Extremfälle sind hier eine Quelle, die immer das selbe Symbol m[9] liefert; diese Datenquelle hat einen Informationsgehalt von

$$H_s = P[m] \cdot l[m] = -1 \cdot \mathrm{ld}\, 1 = 0, \quad \text{Formel 3}$$

was mit Blickrichtung auf die anfängliche Plausibilisierung (Überraschung beim Auftreten eines Symbols) auch logisch erscheint, da man ohnehin schon weiß, daß dieses Symbol kommen wird, d.h. keine Überraschung mit dem Auftreten dieses Symbols verbunden ist.

Andererseits ergibt sich für eine Quelle mit gegebener Anzahl von Symbolen das Maximum an Informationsgehalt jedenfalls bei gleicher Wahrscheinlichkeit für alle Symbole. Zum Beispiel kommt eine binäre Quelle, die Symbole "0" und "1" mit einer Wahrscheinlichkeit $P[m_i]=0{,}5$ für m_0="0" und m_1="1", auf einen Informationsgehalt von

$$H_{s,max} = -0{,}5 \cdot \mathrm{ld}\, 0{,}5 - 0{,}5 \cdot \mathrm{ld}\, 0{,}5 = 1. \quad \text{Formel 4}$$

Quellenverzeichnis

Vorlesung und Skriptum "Grundlagen Nachrichtentechnischer Signale", VO382.756, UE382.767: H. Weinrichter, F. Hlawatsch; Sommersemester 1990

Fortsetzung in den **PC·NEWS***-38.*

Zu diesem Thema passend folgt ein Beitrag aus der Praxis, gelesen im FIDO-Net:

COMPRESSION DETECTION

David Nghiem

Does anyone know of a way to detect whether a file is compressed, particularly by these programs: pkzip, arj, lharc, lzh? I am writing a program that I would like to support compressed files so a routine would be handy.

Jason Tackaberry, 1:222/140

PCN-407/ARC.EXE

```
Module:     WHICHARC.C
Subject:    tries to determine the archiver
            used to compress files
Author:     Heinz Ozwirk
            modified for SNIPPETS by Bob Stout
Status:     public domain
Started:    28.09.1991   13:35:57
Modified:   13.10.1991   14:15:57
Prototype:  int WhichArc(char *pName)
   pName    address of full path name of file to examine
   Result   -1:      file not found
            UNKNOWN: unknown packer
            ARC:     ARC or PKARC
            ARJ:     ARJ
            LHA:     LHARC or LHA
            ZIP:     PKZIP
            ZOO:     Zoo
```

LHARC/LHA

No archive header. WhichArc examines the checksum of the first file header. If the checksum is valid and if the string -lh?- is found, LHA or LHARC is assumed.

ARJ

If a file starts with 0x60, 0xEA, ARJ is assumed.

ZIP

If the file begins with "PK", PKZIP is assumed.

ZOO

Zoo'ed archives always start with "ZOO x.xx Archive". WhichArc only looks for "ZOO".

ARC

No header. Files starting with 0x1A are assumed to be ARCed.

```
#include <stdio.h>
#include <string.h>

typedef unsigned char BYTE;

#define strNcmp(s1,s2,n) !strncmp((const char *)(s1), (s2),
(n))

enum {UNKNOWN, ARC, ZOO, ARJ, LHARC, LHA, ZIP}

WhichArc(char *pName)
{
      FILE  *fp;
      BYTE  header[128];
      int   c, i, n;

      memset(header, 0, sizeof(header));
      fp = fopen(pName, "rb");
      if (fp == NULL)
            return -1;
      n = fread(header, sizeof(BYTE),
                sizeof(header) - sizeof(BYTE), fp);
      fclose(fp);

      if (n <= 0)
            return -1;

      if (n >= 7 && n >= header[0] + 2)
      {
            for (c = 0, i = header[0]; i--;
                 c += (header+2)[i])
                  ;
            if ((BYTE)(c & 0x00FF) == header[1]       &&
                  strNcmp(&header[2], "-lh", 3) &&
                  '-' == header[6])
            {
                  return (header[5] > '1') ? LHA : LHARC;
            }
      }

      if (n >= 2)
      {
            if (strNcmp(header, "\x60\xEA", 2))
                  return ARJ;
            if (strNcmp(header, "PK", 2))
                  return ZIP;
      }

      if (n >= 3 && strNcmp(header, "ZOO", 3))
            return ZOO;

      if (n >= 25 && *header == 0x1A) return ARC;

      return UNKNOWN;
}
```

Auf der Diskette PCN-407 finden Sie auch eine erweiterte Version dieses Programms, das insgesamt folgende Formate erkennt: ARC, ARJ, LHA, ZIP, ZOO, PAK, ARC7, SFXARC, SFXARJ, SFXLHA, SFXZIP, SFXPAK. □

---

[7] Eine Datenquelle im hier verwendeten Zusammenhang ist alles, was Daten liefern kann. Darunter ist also eine Black Box zu verstehen, aus der Symbole herauspurzeln (jenes Ding, welches in nachrichtentechnischen Überlegungen oft verwendet wird). Einschränkungen für die Realisierungsmöglichkeiten dieser Black Box bestehen kaum, sie kann ebenso durch ein File realisiert werden, aus dem ein Programm (in diesem Zusammenhang die Datensenke) Daten ausliest, aber ebenso durch ein Programm welches Daten in ein File schreibt (in diesem Fall sind die Verhältnisse genau ausgetauscht) oder durch einen ADC der ein beliebiges Signal wandelt und damit als Datenquelle für die nachfolgenden Stufen agiert.

[8] Es werden Symbole m_i mit Wahrscheinlichkeiten $P[m_i]$ gesendet.

[9] mit einer Wahrscheinlichkeit P[m]=1