Metrici pentru compresia fişierelor

Prof. dr. Ion IVAN, ec. Daniel VERNIŞ, ec. Petrişor OPREA
Catedra de Informatică Economică, A.S.E., Bucureşti

Pentru a se putea compara algoritmii de compresie, trebuie definit un sistem de indici pentru măsurarea performanţelor acestora. În această lucrare sunt prezentaţi principalii indicatori folosiţi în compresia de date.
Cuvinte cheie : *metricile fişierelor, compresie de date, entropia.*

1. Introducere

Compresia de date este de multe ori privită ca un exerciţiu de reducere a redundanţei în fişier. De fapt, reprezintă cu mult mai mult. Compresia de date rezolvă una din problemele majore ale teoriei informaţiei : cum se poate codifica în mod eficient un mesaj, din punct de vedere al numărului de biţi pe simbol. Cînd un mesaj a fost exprimat în cît mai puţini biţi pe simbol putem spune că a avut loc o compresie optimală. Nu sunt biţi folosiţi ineficient. Acesta este scopul compresiei de date.

În literatura de specialitate, sunt folosiţi deseori termenii de compresie şi compactare. Între aceşti doi termeni există o diferenţă.

Prin definiţie, **compactarea datelor** este o tehnică de reducere a lungimii reprezentărilor fizice a datelor, cu păstrarea numai a unui subset din setul de date iniţial. Acest subset poartă denumirea de **informaţie relevantă**.

Prin **compresie de date** se înţelege un procedeu de compactare care este complet reversibil. Adică din fişierul compresat, se poate obţine prin aplicarea decompresiei fişierul iniţial. În continuare, **F** este fişierul iniţial iar **F'** este fişierul compresat.

Deci, dacă la compresie, fişierul se poate recupera integral după compresie, la compactare acest lucru nu mai este valabil. De exemplu, prin compactarea unui text se poate înţelege eliminarea spaţiilor nesemnificative. Pentru măsurarea lungimii fişierelor, în compresia de date, se folosesc două tipuri de lungimi:

– lungimea fişierului ca număr de simboluri, **L(F)** ;

– lungimea fişierului ca număr de biţi, $l(F)=\Sigma l(x_i)$, unde x_i reprezintă simbolul din fişierul **F** de pe poziţia **i**.

2. Indicatori de performanţă a algoritmilor

Performanţa unui algoritm de compresie este privită sub trei aspecte semnificative :

– modul în care se face compresia fiecarui fişier în parte ;

– comportamentul statistic global al produsului care implementează algoritmul ;

– resursele necesare pentru realizarea compresiei ;

Gradul de compresie

O expresie cantitativă a puterii algoritmului de a reduce lungimea **l(F')** faţă de **l(F)**, dată prin :

$$g = (l(F) - l(F')) / l(F)$$

Rata de compresie

Reprezintă ponderea pe care o are fişierul compresat în fişierul iniţial.

$$r = l(F') / l(F)$$

Lungimea minimală

Reprezintă lungimea sub care fişierul compresat devine mai lung decît fişierul iniţial. Se identifică deci

situaţia limită cînd algoritmul de compresie îşi pierde eficienţa.
Pentru coduri de compresie cu lungime fixă, se consideră :
– **n** numărul de simboluri distincte din fişier;
– **k** lungimea codului ;
– **m** lungimea fişierului ca număr de simboluri ;
Presupunem ca această lungime se memorează pe patru baiţi .
h(a) - lungimea codului iniţial, avînd valoarea 8 ;
Din inegalitatea:

$$n \cdot h(a) + n \cdot k + 4 + l(F') < l(F)$$

rezultă:

$$n \cdot (8 + k) + 4 + m \cdot k < m \cdot 8$$

şi se obţine lungimea minimală:

$$l_{\min} = \frac{n \cdot (8 + k) + 4}{8 - k} < m .$$

Dacă se consideră coduri de lungime variabilă asociate simbolurilor, se calculează o lungime medie a codurilor $\bar{k}$ şi se determină intervalul $[\bar{k} - 3\sigma, \bar{k} + 3\sigma]$. Vor rezulta valori limită precum şi un nivel mediu de încadrare a lungimii peste care compresia devine eficientă.
Lungimea minimală se poate obţine în acest caz prin tehnici de simulare.

3. Entropia fişierelor

Pentru fişiere, entropia reprezintă numărul de biţi pe simbol ce s-ar obţine printr-o compresie optimală a mesajului.
Este foarte uşor a se determina, din acest moment, şi gradul de redundanţă în fişier. Dacă un fişier de date se reprezintă în cod ASCII, adică 8 biţi pe simbol, se poate obţine gradul de redundanţă în fişier astfel:

$$R_F = 8 - H$$

măsurată în biţi pe simbol, unde H se determină prin relaţia lui Shannon [1] astfel:

$$H = -\Sigma f_i \cdot \log(f_i)$$

Entropia poate fi considerată o metrică pentru fişierele de date, datorită proprietăţilor :

$$H(X) \geq 0 \; ; \; H(X) \leq -\Sigma(1/n)\log_2(1/n) .$$

unde **1/n** reprezintă probabilitatea de apariţie a simbolului x_i în condiţii de echiprobabilitate.
Considerăm două alfabete $A = \{x_1 \; x_2 \; \ldots \; x_n\}$ şi $B = \{y_1 \; y_2 \ldots \; y_p\}$ şi $A \cap B = \phi$.
Se construiesc pe cele două alfabete mulţimile de fişiere F_A, F_B.
Fie $F_1 \in F_A$ şi $F_2 \in F_B$ cărora li se asociază cîmpurile de probabilitate X_1 respectiv X_2. În aceste condiţii

$$H(X_1,X_2)=H(X_1) + H(X_2).$$

Dacă alfabetele **A** şi **B** sunt oarecare (nu mai au proprietatea de independenţă)

$$H(X_1,X_2)=H(X_1) + H(X_2/X_1).$$

unde $H(X_2/X_1)$ este entropia alfabetului sursă **B** condiţionat de apariţia simbolurilor din alfabetul **A**.
Deci $H(X_1,X_2) \leq H(X_1) + H(X_2)$

$$H(X_1/X_2) = H(X_2/X_1) + H(X_1) - H(X_2)$$

$$0 \leq H(X_1) - H(X_1/X_2) \leq H(X_1)$$

Aceste proprietăţi permit efectuarea de cuantificări pe fişiere şi o ierarhizare a acestora în raport cu gradul de nedeterminare.
De exemplu, se consideră fişierele :
F_1={ a a a a b b b b c c c c d d d d }
F_2={ a a b b b b c c c c c c c c d d }
F_3={ a b b b b c c c c d d d d d d d }
În care $lg(F_1) = lg(F_2) = lg(F_3)$, unde **lg()** este funcţia de lungime a unui fişier care indică numărul de simboluri conţinute.
Fişierele F_1, F_2, F_3 sunt definite pe alfabetul A = { a, b, c, d }. Frecvenţele de apariţie a simbolurilor în cele 3 fişiere sunt date în tabelul 1. :

Tabelul 1.

Simbol	F1	F2	F3
a	4	2	1
b	4	4	4
c	4	8	4
d	4	2	7

Probabilităţile, ca frecvenţe relative de apariţie sunt date în tabelul 2.

Tabelul 2.

Simbolul	F1	F2	F3
a	1/4	1/8	1/16
b	1/4	1/4	1/4
c	1/4	1/2	1/4
d	1/4	1/8	7/16
Entropia	1.999	1.749	1.771

Prin concatenarea fişierelor $\mathbf{F_1}$, $\mathbf{F_2}$, $\mathbf{F_3}$ rezultă fişierul $\mathbf{F_C}$ caracterizat prin valorile date în tabelul 3.

Tabelul 3.

Simbolul	Frecventa	f_i
a	7	7/48
b	12	12/48
c	16	16/48
d	13	13/48
Entropia	-	1.943

Omogenitatea fişierelor în raport cu distribuţia de probabilitate a simbolurilor poate fi considerată un criteriu de alegere a unui algoritm de compresie de date.

Se defineşte **indicatorul normat al entropiei** fişierului **F**, fişier definit pe un alfabet cu **n** simboluri.

$$I_H = \frac{H(F)}{\lg_2 n}$$

În continuare se prezintă programul **entropia.cpp**, care calculează entropia fişierelor şi gradul de redundanţă a acestora.

Calculul entropiei fişierelor - entropia.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define LOG(x) 3.32*log10(x)
#define ENTROPIA(x) -(x*LOG(x))
FILE *input;
unsigned int TabelaContor[256];
void CitireFisier(void);
double Analiza(void);
void Utilizare (void);
main(int argc,char *argv[])
{double Rezultat;
if (argc==1) Utilizare();
input=fopen(argv[1],"rb");
if (!input) printf("\nFisier de intrare inexistent.");
if (!input) exit(-1);
printf("\n  Citire fisier ...");
CitireFisier();
printf("\n  Analiza date ....");
Rezultat=Analiza();
printf("\nFisierul \"%s\" are o entropie de %3.3f
biti pe bait.",argv[1],Rezultat);
printf("\nEste posibil sa se obtina o rata medie
de compresie");
printf("\nfolosind algoritmul Huffman standard
de : %2.2f%%\n\n\n",100 - (Rezultat*100)/8);
fclose(input);
return(1);
}
void CitireFisier()
{int ch;
while((ch=getc(input))!=EOF)
TabelaContor[ch]++;
}
double Analiza()
{double accum=0.0;
double freq;
long fsize=0L;
unsigned int z;
fsize=ftell(input);
for(z=0;z<256;z++)
        if (TabelaContor[z])
                {
                freq=(double)
TabelaContor[z]/fsize;
                accum+=(double)
ENTROPIA(freq);
                }
return accum;
}
void Utilizare()
{printf("\n\n\n");
printf("  \'Entropia fisierelor\' v1.0 by Daniel
Vernis\n");
printf("  Sintaxa : entropia [fisier]\n\n");
printf("     Acest program calculeaza entropia
fisierelor\n");
printf("  folosind ecuatia entropiei introdusa de
Shannon.\n"); exit(1);
}
```

4. Erori ale compresiei de date

Transmisia datelor folosind canalele discrete fără zgomot este din nefericire, un model nu prea realistic în sistemele de comunicaţie. Actualele sisteme de transmisii de date, sunt înclinate spre două tipuri de erori :

– **erori de fază**, în care codul unui simbol este pierdut sau adăugat ;

– **erori de amplitudine**, în care codul unui simbol este corupt .

Gradul în care erorile de canal degradează transmisia este un parametru important în alegerea metodei de compresie a datelor.

Susceptibilitatea la erori a algoritmilor de compresie, depinde în mare măsură de tipul algoritmului : static sau dinamic.

Algoritmi statici

Este în general cunoscut că algoritmii de tip Huffman, au proprietatea de autocorecţie, prin care erorile de transmisie tind să nu se propage mai departe. Într-un algoritm static, prin sincronizare se înţelege identificarea în acelaşi mod, la emisie şi la recepţie, a cuvîntului transmis.

În continuare este prezentat un exemplu pentru ilustrarea abilităţii algoritmului Huffman standard de recuperare a erorii de fază. Se foloseşte mesajul **bcdaeb**, codificat după algoritmul Huffman astfel :

a	**1**
b	**011**
c	**010**
d	**001**
e	**000**

Şirul va fi codificat **0110100011000011**. Figura 1 demonstrează impactul pierderii primului bit, celui de-al doilea sau celui de-al patrulea. Punctele delimitează codurile simbolurilor interpretate.

011.010.001.1.000.011. codificarea **bcdaeb**
1.1.1.0.001.1.000.011. bitul 1 pierdut, interpretare **aacdaeb**
010.1.000.1.1.000.011. bitul 2 pierdut, interpretare **caeaaeb**
011.1.000.1.1.000.011. bitul 4 pierdut, interpretare **baeaaeb**

Fig. 1. Recuperarea erorilor de fază

În cazul pierderii primului bit, are loc resincronizarea bupă cel de-al treilea bit. În şir s-a înlocuit simbolul **b** cu grupul **aa**. Cînd al doilea bit este pierdut, primii 8 biţi ai şirului sunt greşit interpretaţi. Resincronizarea se produce la bitul nouă. În cazul pierderii bitului 4, au loc aceleaşi efecte ca în cazul pierderii bitului al doilea. Însă, primul simbol al şirului, **b** este citit corect. Efectul erorilor de amplitudine asupra şirului **bcdaeb** este prezentat în figura 2. La erorile de amplitudine, biţii 1, 2 şi 4 sunt schimbaţi, nu pierduţi ca în cazul erorilor de fază. Astfel, bitul 1 este transformat din **0** în **1**, bitul 2 din **1** în **0**, iar bitul 4 din **0** în **1**.

011.010.001.1.000.011.
codificarea **bcdaeb**
1.1.1.010.001.1.000.011.
bitul 1 alterat, interpretare **aaacdaeb**
001.010.001.1.000.011.
bitul 2 alterat, interpretare **dcdaeb**
011.1.1.000.1.1.000.011.
bitul 4 alterat, interpretare **baaeaaeb**

Fig. 2. Recuperarea erorilor de amplitudine

Autosincronizarea poate fi discutată mai bine prin următoarele definiţii.

Dacă **s** este un sufix al unei codificări şi există secvenţele de cod Γ şi Δ, putem spune: $s\Gamma=\Delta$, unde Γ desemnează secvenţa de sincronizare a lui **s**. De exemplu, pentru şirul prezentat anterior, **1** este secvenţă de sincronizare pentru **01**, în vreme ce **000001** şi **011** sînt secvenţe de sincronizare pentru sufixul **10**. Dacă fiecare sufix are o secvenţă de sincronizare, atunci codul este **complet autosincronizat**. Dacă nici unul sau doar cîteva din sufixele indicate au secvenţe de sincronizare atunci codul este **nesincronizat** sau **parţial sincronizat**.

De asemenea, orice cod prefix care este complet autosincronizat se va sincroniza cu probabilitate 1, dacă şirul sursă constă într-o succesiune de mesaje independente.

Algoritmi dinamici

Algoritmii dinamici sunt de departe cei mai afectaţi de erorile de transmisie. De exemplu, în cazul algoritmului Huffman adaptiv, deşi receptorul se poate resincroniza cu emiţătorul, în scopul localizării corecte a unui început de codificare, informaţia pierdută reprezintă cîţiva biţi sau cîteva simboluri.

Din această cauză, la recepţie vor fi redefinite greşit codificările simbolu-

rilor din punctul întîlnirii erorii, fiind compromis întreg şirul.
Lempel şi Ziv recunosc că principalul dezavantaj al algoritmului LZ78 este susceptibilitatea la eroare.

5. Concluzii

În literatura de specialitate recentă, entropia este folosită drept criteriu de neomogenitate a fişierelor. Analiza entropică permite stabilirea apartenenţei la o anumită clasă de entropie şi în acest fel se va asocia direct componenta program corespunzătoare algoritmului care realizează compresie performantă. Se fundamentează construirea metricilor pentru compresie care ţine seama de neomogenitatea fişierelor. De asemenea, este necesar studiul corelaţiei dintre factorii care influenţează atît compresia cît şi decompresia datelor .

Bibliografie

1. C.Shannon, W.Weaver, Entropy, The Mathematical Theory of Communication, Urbana, Illinois, University of Illinois 1949
2. Abrams, S. Tren, A methodology for interactive computer service measurement, Communication of ACM, vol. 20, nr.12, Decembrie 1977.
3. I.Ivan, D.Verniş, Analiza comparată a algoritmilor de compresie date, PC World, nr.12, Decembrie 1995.
4. I.Ivan, M. Popescu, Metrici software, Byte, nr.5, Mai 1996.
5. I.Gh.Roşca, C.Apostol, B. Ghilic-Micu, Prelucrarea fişierelor în Pascal, Editura Tehnică, Bucureşti 1994.