COMPRESIA datelor

Claudiu Soroiu

În cadrul acestui episod al serialului dedicat compresiilor de date vă vom prezenta compresia aritmetică, cea mai eficientă metodă entropică de compresie.

În articolul din numărul anterior am prezentat o serie de algoritmi pentru compresia datelor care foloseau *arbori Huffman*. Dezavantajul acestor algoritmi este acela că fiecare simbol generat de o sursă de informație *S* este codificat folosind un număr întreg de biți, fapt care duce la apariția unei diferențe mari între lungimea codificării unui șir de simboluri și entropia acestuia, dacă nu se consideră lungimea, în biți, a dicționarului.

Nu există algoritmi care să elimine complet redundața unei surse de informație deoarece, în primul rând, entropia unui șir de simboluri generat de o sursă este un număr real și bitul este o unitate atomică, și, în al doilea rând, trebuie transmis dicționarul pentru ca informația să poată fi reconstituită.

Cea mai eficientă metodă entropică de compresie a datelor care elimină aproape complet redundanța unei surse de informație este ***compresia aritmetică***.

Algoritmul de compresie

Compresia aritmetică a fost descoperită de către cercetătorii *Peter Elias*, *Jorma J. Rissanen* și *Richard C. Pasco*. Ideea care stă la baza acestei metode de compresie este aceea de codifica un șir de simboluri folosind un singur cod și nu câte un cod pentru fiecare simbol în parte. De fapt, orice șir de simboluri generat de o sursă de informație este codificat folosind un număr real cuprins în intervalul [0, 1).

Datorită faptului că această metodă de compresie este entropică avem nevoie de probabilitățile p_i ($0 \le i < m$, unde m este numărul de simboluri pe care le poate genera o sursă de informație *S*) de apariție ale simbolurilor.

Similar algoritmului *Huffman* se pot construi variantele *statică*, *semi-statică* și *dinamică* ale algoritmului de compresie aritmetică.

În continuare vom prezenta modul în care se realizează codificarea unui șir de simboluri generat de o sursă de informație *S* pentru varianta *statică*. În cazul variantei *semi-statice* probabilitățile se calculează pe baza simbolurilor generate de sursa de informație *S*, fiind nevoie de două parcurgeri ale șirului de simboluri la fel ca în cazul algoritmului *Huffman*.

Considerăm un interval $[a, b) \subseteq [0, 1)$, $b > a$. Fiecărui simbol care poate fi generat de o sursă de informație *S* i se atașează un subinterval al lui $[a, b)$ cu proprietatea că lungimea subintervalului corespunzător unui simbol este direct proporțională cu probabilitatea de apariție a simbolului respectiv și oricare două subintervale corespunzătoare la două simboluri distincte nu au puncte de intersecție.

Fie P_i probabilitatea cumulată a simbolului A_i, $P_0 = 0$ și $P_i = p_0 + ... + p_{i-1}$, $0 \le i < m$, și fie $l = b - a$ lungimea intervalului $[a, b)$. Din faptul că suma tuturor probabilităților de apariție ale simbolurilor A_i, $0 \le i < m$ este 1, rezultă că $P_i \in [0, 1)$.

Fiecărui simbol A_i i se va atașa intervalul $[a + l \cdot P_i , a + l \cdot (P_i + p_i))$. Lungimea subintervalului corespuzător unui simbol A_i este egală cu $l \cdot p_i$.

În cazul în care o sursă de informație nu generează toate simbolurile pe care le poate genera, atunci probabilitatea de apariție a unor simboluri poate fi 0 deci, lungimea subintervalelor atașate simbolurilor care nu sunt generate este 0, caz în care avem intervale degenerate, și, dacă p_i este 0 și p_{i+1} este diferit de 0, atunci intersecția dintre subintervalele corespunzătoare celor două simboluri este diferită de mulțimea vidă și nu mai sunt respectate condițiile enunțate mai sus. Dacă se elimină subintervalele de lungime 0, atunci condițiile sunt respectate.

Algoritmul de compresie aritmetică se bazează pe această împărțire a unui interval.

Faptul că avem intervale degenerate nu va influența funcționalitatea algoritmului, deoarece o sursă de informație nu va genera niciodată simboluri care au probabilitatea de apariție 0.

Algoritmul de codificare constă în alegerea unui subinterval al intervalului [0, 1) corespunzător primului simbol generat de o sursă *S* și apoi, ca nou interval se consideră

subintervalul ales și se alege subintervalul corespunzător celui de-al doilea simbol.

Algoritmul de compresie aritmetică folosit pentru a codifica un șir de simboluri generat de o sursă de informație *S* este următorul:

- considerăm intervalul care are extremitatea din stânga *a* și lungimea *l*;
- fie p_i probabilitățile de apariție ale simbolurilor care pot fi generate de o sursă de informație *S*;
- fie P_i probabilitățile cumulate ale simbolurilor;
- pentru fiecare simbol A_i generat de sursa de informație *S* execută:
  - $a \leftarrow a + l \cdot P_i$;
  - $l \leftarrow l \cdot p_i$;
- *rezultat* $\leftarrow a + l / 2$.

În ciclul repetitiv prezentat anterior se schimbă intervalul inițial cu subintervalul corespunzător simbolului generat de sursa de informație *S*.

Datorită faptului că intervalele atașate simbolurilor care au probabilitatea de apariție diferită de 0 sunt disjuncte, fiecare simbol este unic determinat de orice număr care aparține subintervalului corespunzător.

În teorie se folosește ca interval inițial intervalul [0, 1), deci cu capătul din stânga 0 și lungimea 1.

După parcurgerea șirului de simboluri generat de o sursă de informație se transmite un număr real din intervalul [0, 1) care reprezintă codificarea șirului de simboluri. Numărul real trebuie transmis cu o precizie foarte mare. Acest număr este dat, de obicei, de mijlocul ultimului interval calculat cu ajutorul algoritmului prezentat anterior: $a + l/2$.

Algoritmul de codificare mai poate fi construit folosind limitele inferioară și superioară ale unui interval în locul limitei inferioare și lungimii intervalului. În acest caz, după linia în care se calculează lungimea intervalului se adaugă linia $b \leftarrow a + l$.

Dacă intervalul inițial este [0, 1), atunci precizia cu care trebuie calculată limita din stânga pentru un simbol cu probabilitatea p_i, $0 \le i < m$, este de $\lceil -\log_2 p_i \rceil$ biți.

Dacă analizăm modul de construire al numărului real plecând de la intervalul [0; 1), atunci precizia cu care trebuie transmis numărul care reprezintă mijlocul ultimului interval calculat este de $\left\lceil n \cdot \left(-\sum_{i=0}^{m-1} p_i \cdot \log_2 p_i \right) \right\rceil + 1$ biți, unde *n* reprezintă numărul de simboluri generate de sursa *S*.

Se poate observa foarte ușor că diferența dintre entropia șirului generat de sursa de informație S și numărul mediu de biți necesari transmiterii unui simbol este foarte mică (mai mică de 1 bit).

De exemplu, fie o sursă de informație *S* care poate genera simbolurile `'a'`, `'b'` și `'c'` cu probabilitățile 1/2, 1/4 și 1/4.

În figura 1 se poate observa modul de codificare a șirului de simboluri `'abac'`. Precizia cu care trebuie transmis rezultatul este de 7 biți (rezultatul este 0,3046875 = $0{,}0100111_2$ ceea ce înseamnă că se transmit biții 0100111,

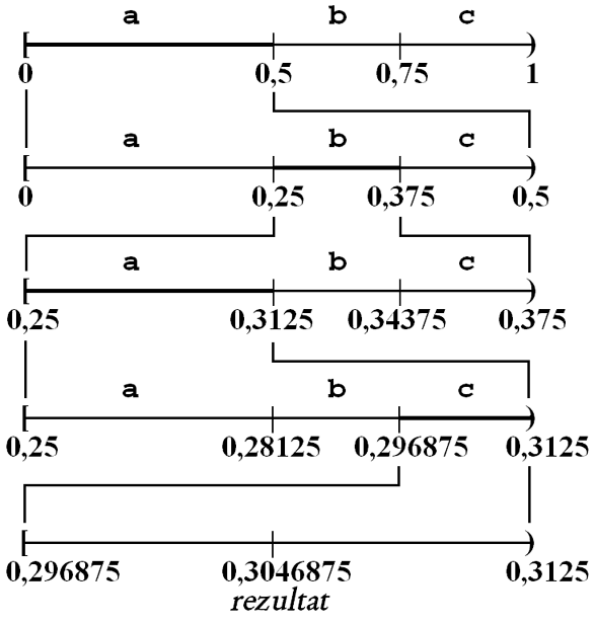

Figura 1

deoarece numărul este cuprins între 0 și 1 și ne interesează doar porțiunea care se află după virgulă).

Pe lângă rezultatul obținut în urma codificării mai trebuie transmis *dicționarul* care este format din probabilitățile de apariție ale simbolurilor care pot fi generate de sursa de informație *S*.

Motivul pentru care rezultatul îl reprezintă mijlocul ultimului interval calculat, și deci o creștere a preciziei cu un bit, este acela că în practică există situații în care reprezentarea în baza 2 a celor două limite ale intervalului, folosind aceeași precizie, sunt egale pentru că microprocesoarele existente realizează operații cu numere reale cu o precizie finită de până la 80 de biți.

Din punct de vedere teoretic este suficient ca rezultatul să fie constituit de limita inferioară a ultimului interval găsit, calculată cu precizia de $\left\lceil n \cdot \left(-\sum_{i=0}^{m-1} p_i \cdot \log_2 p_i \right) \right\rceil$ biți, deoarece, bazându-ne pe cele enunțate anterior, un simbol căruia i s-a atașat intervalul [a, b) este unic determinat de orice număr real care aparține intervalului.

Se poate pune întrebarea "*De ce nu se poate lua ca rezultat limita superioară a intervalului corespunzător ultimului simbol codificat*?".

Răspunsul este acela că limita superioară a intervalului corespunzător ultimului simbol codificat nu aparține intervalului, aceasta aparține intervalului corespunzător simbolului care urmează în ordine lexicografică.

Algoritmul de decompresie

Pentru a decodifica un număr real, cuprins între 0 și 1, a cărui lungime în biți se cunoaște, trebuie să avem probabilitățile de apariție ale simbolurilor care au fost folosite în procesul de codificare și numărul total *n* al simbolurilor care au fost codificate.

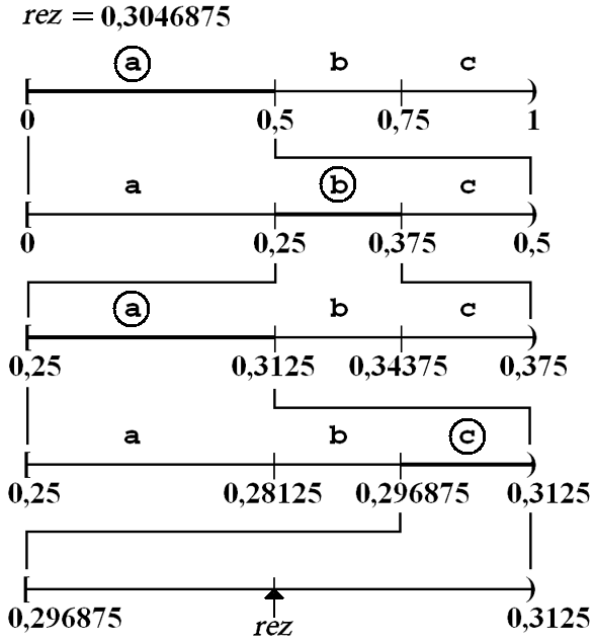

Figura 2

La începutul procesului de decodificare considerăm intervalul $[a, a + l)$, unde $a = 0$ și $l = 1$. Fiecărui simbol îi corespunde un subinterval al acestui interval.

În continuare se caută subintervalul căruia îi aparține numărul care trebuie decodificat. După ce s-a găsit acest subinterval se transmite simbolul corespunzător acestuia și noul interval devine cel găsit. Acest pas se execută până în momentul în care am decodificat n simboluri.

Dacă nu se transmite numărul n de simboluri care au fost codificate, în momentul compresiei alfabetul sursei S de informație se poate extinde cu un simbol suplimentar care are semnificația de *sfârșitul codificării* care se va codifica după ce sursa S nu mai generează simboluri. În concluzie, în momentul în care numărul se va afla în subintervalul corespunzător simbolului de *sfârșit de codificare*, decodificarea se va încheia.

În figura 2 se poate observa modul de decodificare a numărului 0,3046875 pentru alfabetul format din simbolurile `'a'`, `'b'` și `'c'`, care au probabilitățile 1/2, 1/4 și 1/4, și numărul de simboluri care au fost codificate $n = 4$.

La primul pas se observă că numărul ***rez*** aparține subintervalului [0, 0,5), subinterval corespunzător simbolului `'a'`, deci acest simbol se va transmite și noul interval va deveni [0, 0,5).

La al doilea pas numărul ***rez*** aparține subintervalului [0,25, 0,375) care corespunde simbolului `'b'`, acest simbol se va transmite și noul interval este [0,25, 0,375).

La al treilea pas numărul ***rez*** aparține subintervalului [0,25, 0,3125) care corespunde simbolului `'a'`, acest simbol se va transmite și noul interval este [0,25, 0,3125).

La al patrulea pas numărul ***rez*** aparține subintervalului [0,296875, 0,3125) care corespunde simbolului `'c'`, acest simbol se va transmite și noul interval este [0,296875, 0,3125). În acest moment se încheie procesul de decodificare deoarece au fost decodificate $n = 4$ simboluri. În timpul decodificării a fost transmis șirul de simboluri `'abac'`.

Detalii de implementare

Datorită faptului că pentru date de dimensiuni mari precizia cu care trebuie transmis rezultatul devine foarte mare, calculatoarele nu dețin resurse suficiente pentru a efectua operații cu numere reale cu precizie mare. De exemplu, există situații când trebuie folosită o precizie de $8 \cdot 10^5$ biți sau chiar mai mare, în timp ce calculatoarele moderne au o precizie de cel mult 128 de biți. În acest caz, algoritmii prezentați anterior nu se pot utiliza în practică.

Toate implementările performante ale acestui algoritm nu utilizează operații cu numere reale, ci folosesc operații cu numere întregi reprezentate pe 16 sau 32 de biți pentru a evita erorile de calcul care apar când se efectuează operații cu numere reale reprezentate în virgulă mobilă. În unele implementări, cele două operații de înmulțire, care apar în algoritmii prezentați, au fost înlocuite cu adunări și deplasări pe biți. Cu toate că s-a obținut o scădere a timpului de execuție pentru date de dimensiuni mari, acești algoritmi nu mai codifică informația la fel de eficient.

Pentru a implementa acest algoritm fără a folosi numere reale vom utiliza limitele superioare și inferioare ale intervalelor în locul limitelor inferioare și lungimilor intervalelor.

Datorită faptului că pentru date de dimensiuni mari nu putem păstra în memorie tot rezultatul codificării, ajungem la concluzia că acesta trebuie transmis pe parcursul codificării simbolurilor generate de sursa de informație.

În primul rând, având în vedere că se folosesc numere întregi fără semn, trebuie să alegem o reprezentare pentru intervalul [0, 1). În cazul în care folosim întregi pe 16 biți, este de preferat ca acest interval să fie reprezentat prin intervalul $[0, 10000_{16}]$ din care vom considera numai numerele naturale, adică toate numerele întregi fără semn din acest interval (0, 1, ..., 65535 = $FFFF_{16}$).

Vom considera că un număr întreg reprezentat pe 16 biți reprezintă valoarea unui număr real cuprins între 0 și 1 reprezentat cu o precizie de 16 biți. În aceste condiții cel mai apropiat număr de 1 reprezentat cu o precizie de 16 biți este $0{,}1111111111111111_2 = 0{,}99998$.

Având reprezentarea intervalului [0, 1) și cunoscând faptul că suma tuturor probabilităților de apariție a simbolurilor generate de o sursă de informație S este 1 (adică lungimea intervalului) aceste probabilități trebuie recalculate astfel încât suma lor să fie 65535, deci vor fi înmulțite cu acest număr. Din punct de vedere teoretic suma probabilităților ar trebui să fie 65536, dar în cazul practic nu se întâmplă acest lucru deoarece este vorba de o mulțime și nu de un interval și între 65535 și 65536 nu mai avem nici un număr. În cazul intervalului [0, 1) există un număr real cu suficient de multe cifre zecimale care să-l aproximeze pe 1.

După efectuarea calculelor exemplificate anterior trebuie avut în vedere faptul că avem la dispoziție un număr limitat de numere din interval și o precizie de maxim 16 biți. Cu aceste valori calculate, dacă se folosește algoritmul de codificare prezentat, în câțiva pași se va ajunge la depășire inferioară (*underflow*) datorită împărțirilor.

Considerând rezultatul codificării unui șir de simboluri $a_1, a_2, ..., a_n$ care au probabilitățile de apariție $p_1, p_2, ..., p_n$ și rezultatul codificării aceluiași șir de simboluri la care s-a mai adaugat un simbol a_j, se poate observa că diferența de lungime dintre cele două rezultate este aproximativ egală cu $\lceil -\log_2 p_j \rceil$, unde p_j este probabilitatea de apariție a simbolului a_j, și toți biții primului rezultat cu excepția ultimilor cel mult 2 biți sunt identici cu biții cei mai semnificativi ai celui de-al doilea rezultat.

Ținând cont de observația anterioară, pe parcursul codificării, în momentul în care cei mai semnificativi biți ai limitelor intervalului de la iterația curentă sunt identici, atunci aceștia se vor transmite și limitele intervalului se vor deplasa la stânga cu un număr de poziții egal cu numărul de biți care au fost identici. Cei mai nesemnificativi biți ai limitei din dreapta a intervalului curent se vor completa, după deplasare, cu 1 pentru ca această limită să aproximeze mai bine limita reală a subintervalului curent.

Datorită faptului că probabilitățile de apariție au fost recalculate și sunt numere întregi, atunci numărul de biți identici ai celor două numere care reprezintă capetele unui interval nu depășesc 16 biți.

Algoritmul de codicare a unui șir de simboluri generate de o sursă de informație *S* folosind aritmetica numerelor întregi fără semn pe 16 biți este următorul:

```
a ← 0
b ← FFFF16
l ← b - a
transmite numărul n de simboluri pe care le va genera
                                                  sursa S
fie pi probabilitățile de apariție ale simbolurilor care pot fi
                                  generate de o sursă de informație S
fie Pi probabilitățile cumulate ale simbolurilor
transmite probabilitățile pi
cât timp sursa S generează simboluri execută
   fie i numărul de ordine al simbolului generat de S
   a ← a + l · Pi
   l ← l · pi
   b ← a + l
   fie c numărul de biți semnificativi identici ai lui a și b
   dacă c > 0 atunci
      transmite cei mai semnificativi c biți ai lui a
      deplasează a și b cu c poziții la stânga
      setează pe 1 cei mai nesemnificativi c biți ai lui b
   sfârșit dacă
sfârșit cât timp
fie c numărul de biți semnificativi identici ai lui a și b
dacă c > 0 atunci
   transmite cei mai semnificativi c biți ai lui a
   deplasează a și b cu c poziții la stânga
   setează pe 1 cei mai nesemnificativi c biți ai lui b
sfârșit dacă
r ← (b - a)/2
transmite r
```

Algoritmul de decodificare a unui număr real, bazat pe observațiile anterioare este următorul:

```
a ← 0
b ← FFFF16
l ← b - a
r ← 0
k ← 15
fie n numărul de simboluri care au fost codificate
fie pi probabilitățile de apariție ale simbolurilor care pot fi
generate de o sursă de informație S
fie Pi probabilitățile cumulate ale simbolurilor
cât timp n > 0 execută
   citeste 1 bit
   r ← r + (1 deplasat cu k poziții la stânga)
   k ← k - 1
   dacă există un unic subinterval al intervalului [a, b)
   care îl include pe r atunci
      fie i numărul de ordine al simbolului corespunzător
                                        subintervalului determinat
      transmite simbolul cu numărul de ordine i
      a ← a + l · Pi
      l ← l · pi
      b ← a + l
      fie c numărul de biți semnificativi identici ai lui a și b
      dacă c > 0 atunci
         deplasează a, b și r cu c poziții la stânga
         setează pe 1 cei mai nesemnificativi c biți ai lui b
         k ← k + c
      sfârșit dacă
      n ← n - 1
   sfârșit dacă
sfârșit cât timp
```

În practică se folosesc implementări ale algoritmilor de codificare și decodificare mult mai performante bazate pe folosirea noțiunilor de aritmetică legate de modul de calcul folosind baze de numerație puteri ale lui 2 (de obicei se lucrează în bazele 2^8 sau 2^{16}) în funcție de tipul de calculator și sistem de operare pe care se va folosi programul de compresie / decompresie.

Compresia aritmetică se folosește în combinație cu alte metode și se obține o reducere considerabilă a conținutului unui fișier sau a datelor transmise printr-o rețea de calculatoare (internet, rețea locală etc.).

Această metodă de compresie a datelor nu este foarte răspândită deoarece compania *IBM* deține patente de inventator asupra câtorva algoritmi performanți de compresie aritmetică.

Limba română				Limba engleză			
a	8,18%	î	0,61%	ș	0,64%	a	8,20%
ă	2,94%	j	0,17%	t	5,63%	b	1,06%
â	0,96%	k	0,14%	ț	0,81%	c	3,44%
b	1,23%	l	4,58%	u	7,39%	d	3,63%
c	3,94%	m	3,62%	v	1,28%	e	12,41%
d	3,45%	n	6,69%	w	0,08%	f	2,35%
e	12,05%	o	4,87%	x	0,88%	g	1,81%
f	2,33%	p	2,92%	y	0,09%	h	3,50
g	1,25%	q	0,03%	z	0,80%	i	7,68%
h	0,10%	r	10,16%				
i	9,74%	s	2,44%				

j	0,19%	s	7,06%		
k	0,39%	t	9,69%		
l	4,48%	u	2,87%		
m	2,81%	v	1,24%		
n	7,64%	w	1,35%		
o	7,14%	x	0,21%		
p	2,03%	y	1,89%		
q	0,09%	z	0,05%		
r	6,68%				

Tabelul 1

Compresie aritmetică adaptivă

Șirul de simboluri, care urmează a fi codificat, generate de o sursă de informație *S* trebuie parcurs de două ori: la prima parcurgere se calculează frecvențele (probabilitățile) de apariție ale simbolurilor, iar la a doua parcurgere se codifică șirul. Din această cauză, timpul de execuție al programelor de compresie este mare datorită faptului că accesul la discurile fizice, rețea etc. este mai lent decât citirea și prelucrarea datelor care există în memoria calculatorului.

Pentru a evita parcurgerea de două ori a șirului de simboluri generate de sursa *S* și, deci, scăderea performanței aplicației de compresie, probabilitățile se pot calcula adaptiv pe măsură ce simbolurile sunt codificate.

Există cel puțin două posibilități de a implementa varianta adaptivă a acestui algoritm.

O primă posibilitate este aceea de a inițializa la început toate probabilitățile de apariție cu constante astfel încât suma lor să fie 65535 (în cazul implementărilor care folosesc operații aritmetice pe 16 biți) iar pe parcursul codificării, după fiecare pas, acestea trebuie recalculate în funcție de simbolul care a apărut ultima dată.

O a doua posibilitate este aceea de a extinde alfabetul sursei de informație *S* cu un nou simbol, a cărui semnificație este ***NYT*** (***N**ot **Y**et **T**ransmitted* - nu a fost transmis încă). La începutul algoritmului, se inițializează toate probabilitățile de apariție ale simbolurilor care pot fi generate de sursa *S* cu 0, iar probabilitatea de apariție a simbolului ***NYT*** cu 65535. Pe parcursul codificării șirului de simboluri, la apariția unui simbol a cărui probabilitate de apariție este 0, se transmite codul parțial al simbolului ***NYT*** urmat de codul simbolului care a apărut și apoi se recalculează probabilitățile de apariție ale simbolurilor în funcție de simbolul care a apărut. În cazul în care simbolul nu are probabilitatea de apariție 0, acesta se codifică și apoi se recalculează probabilitățile.

Pentru recalcularea probabilităților de apariție a simbolurilor generate de o sursă de informație *S* se pot folosi diferite modele statistice, în funcție de informația pe care o transmite sursa.

În cazul în care se cunoaște faptul că sursa de informație transmite un text se pot folosi pentru modelul static probabilitățile de apariție ale simbolurilor din limba în care a fost scris textul respectiv.

De exemplu, dacă un text a fost scris în limba română și nu s-au folosit diacritice, atunci, probabilitatea de apariție a simbolului `'a'` va deveni egală cu suma probabilităților de apariție a simbolurilor `'a'`, `'â'` și `'ă'`.

În cazul în care se poate ști că dacă apare un simbol a_i este foarte probabil să apară un simbol a_j, atunci, pentru a micșora dimensiunea codificării este indicat să se modifice probabilitatea de apariție a simbolului a_j.

În tabelul 1 se pot observa probabilitățile de apariție ale caracterelor în limba română și în limba engleză. Nu am inclus în tabel probabilitățile de apariție ale simbolurilor speciale `'.'`, `'!'`, `'?'`, `':'`, `';'` etc. deoarece probabilitatea acestora de apariție este destul de mică și poate fi calculată adaptiv. Se poate observa faptul că dimensiunea codificării unui singur simbol generat de o sursă de informație *S* este cu atât mai mică, cu cât probabilitatea lui de apariție este mai mare.

Modelele statistice cu performanțe ridicate folosite în cadrul compresiei textelor se bazează, la fiecare pas, pe ultimele *k* simboluri apărute, *k* fiind cuprins, de obicei, între 0 și 6.

Unii dintre cei mai performanți algoritmi de compresie a textelor care se bazează pe modele statistice avansate sunt ***DMC*** (***D**ynamic **M**arkov **C**oder*) și ***PPM*** (***P**artial **P**attern **M**atching*).

Majoritatea programelor de compresie bazate pe codificarea aritmetică folosesc modelul ***cvasi-aritmetic***. În cadrul acestui model, probabilitățile cumulate se recalculează în momentul în care cea mai mare dintre probabilități atinge o valoare prag *Prg*, și nu după fiecare simbol generat de o sursă de informație. Acest lucru duce la o scădere a timpului necesar unui program de compresie pentru a codifica un șir de simboluri

Claudiu Soroiu este redactor al GInfo și Poate fi contactat prin e-mail la adresa csoroiu@yahoo.com.