Modifikacija Lempel-Ziv-77 algoritma za kompresiju podataka

Miloš Stanimirović, Vojin Šenk, Ivan Stanojević

***Sadržaj* — Potreba za kompresijom podataka u specifičnim uslovima kao što je razmena podataka između mobilnog uređaja (sa ograničenim mogućnostima) i udaljenog servera, veoma je značajna u mobilnim telekomunikacijama. U ovom radu predstavljen je algoritam koji pokušava da odgovori ovim zahtevima, kodujući XML (Extensible Markup Language) podatke u cilju postizanja što većeg stepena kompresije. Predložene su dve modifikacije Lempel-Ziv 77 algoritma, koje povećavaju stepen kompresije originalnog algoritma, a da ipak obezbeđuju matematički dovoljno jednostavan koder i dekoder, kakav je jedan mobilni uređaj sa ograničenim procesorskim mogućnostima u stanju da obradi.**

***Ključne reči* — kompresija, LZ77, stepen kompresije, XML.**

I. Uvod

Pretpostavimo da imamo jednu (Java) klijent-server aplikaciju, u kojoj se na klijentskoj strani nalazi uređaj sa ograničenim procesorskim mogućnostima kao što je mobilni telefon. U takvoj aplikaciji razmena podataka između klijenta i servera je veoma učestala. Pretpostavimo da se komunikcija vrši preko HTTP protokola (Hypertext Transfer Protocol). Podaci koji se čitaju sa ili upisuju u serversku bazu podataka obično se prenose u određenoj konzistentnoj formi, kao što je XML (Extensible Markup Language). Količina ovakvih podataka ponekad može da bude veoma velika (pogotovo sa stanovišta klijenta). Iz ovih razloga javlja se potreba za kompresijom podataka.

Ovaj rad predstavlja pokušaj da se zadovolje zahtevi za kompresijom specifičnih podataka koji su sadržani u XML formi sa jedne strane, i zahteva za jednostavnim algoritmom, sa druge. Ovakav algoritam bi bez većih problema (u smislu vremena izvršavanja) mogao da obradi jedan procesorski i memorijski vrlo ograničen klijent.

Kada se XML podaci u ovakvoj aplikaciji posmatraju sa stanovišta sekvence (niza) karaktera, može se primetiti da u njoj ima relativno puno ponavljanja, što ostavlja mogućnost kompresije.

II. LZ77 algoritam

Lempel i Ziv su (A. Lempel, J. Ziv) 1977. godine u [1] prikazali algoritam (nazvan LZ77) koji je i dalje u širokoj upotrebi. Ovaj algoritam koristi ponavljanje podsekvenci kodujući zapravo mesto i dužinu tog ponavljanja. Operaciju poređenja kodujuće podsekvence sa već kodovanim simbolima LZ77 vrši nad klizećim prozorom veličine 2^n . Svi simboli (karakteri) koji se nađu van tog prozora ne uzimaju se u obzir. Na početku kodovanja prozor je prazan, a kako kodovanje odmiče, on se puni kodovanim simbolima. Kada se prozor napuni, on se pomera udesno odnosno „klizi“ tako da se u njemu uvek nalaze novi simboli, a simboli se levog kraja „ispadaju“. Čitav proces liči na Tjuringovu mašinu.

Kada se pojavi novi simbol (odnosno onaj koji se ni jednom ne pojavljuje u klizećem prozoru), on se koduje kao

$$[0, \mathbf{sym}], \tag{1}$$

gde je

- 0 kodovano pomoću jednog bita,
- **sym** kodovano pomoću m bita.

Simbol **sym** predstavlja binarnu predstavu karaktera koji se pojavio prvi put, a nulti bit pre njega označava da nema ponavljanja, odnosno da je narednih m bita rezervisano za kodovanje tok karaktera.

U slučaju ponavljanja odnosno poklapanja nove posekvence dužine b i podsekvence u prozoru čiji se prvi simbol nalazi na a-toj poziciji od kraja prozora, kodovanje se vrši pomoću:

$$[1, \mathbf{a}, \mathbf{b}], \tag{2}$$

gde je:

- 1 kodovano pomoću jednog bita,
- **a** kodovano pomoću n bita,
- **b** kodovano pomoću n bita.

U praksi se pokazuje da vrednost **b**, koja teoretski može dostići vrednost 2^n-1 , retko kada ima veće vrednosti, pa je najčeće dovoljan i manji broj bita za predstavljanje **b**. Ovo dakle ukazuje na izvesnu redundantnost ovog algoritma.

Ovu redundantnost moguće je neutralisati korišćenjem Hafmanovog (D. A. Huffman) koda [2] za kodovanje vrednosti **a** i **b**, što naravno podrazumeva pamćenje $2^n \cdot 2$ vrednosti frekvencija pojave pojedinih vrednosti za **a** i **b** (određeno na velikom uzorku). U uslovima sa slabom procesorskom i memorijskom moći, kao što je navedeno u prethodnom odeljku, ovakvo rešenje nije najpovoljnije.

Varijanta gore opisanog algoritma, nazvana LZ-BW koja je opisana u [3], postiže veoma dobre stepene kompresije kao i njena bootstrap modifikacija [4], ali im je

M. Stanimirović, Fakultet Tehničkih Nauka, Trg D. Obradovića 6, 21000 Novi Sad, Srbija; (telefon: +381641887128; e-mail: milos.stanimirovic@yahoo.com).

V. Šenk, Fakultet Tehničkih Nauka, Trg D. Obradovića 6, 21000 Novi Sad, Srbija; (telefon: +381214750082 ; e-mail: ram_senk@uns.ns.ac.yu).

I. Stanojević, Fakultet Tehničkih Nauka, Trg D. Obradovića 6, 21000 Novi Sad, Srbija; (telefon: +381214750092; e-mail: cet_ivan@uns.ns.ac.yu).

složenost nešto veća, a takođe je potrebno korišćenje Hafmanovog kodovanja zbog redundantnosti.

III. Modifikacija 1

Kao što je u prethonom paragrafu objašnjeno, LZ77 algoritam je donekle redundantan, jer u slučaju ponavljanja neke podsekvence za kodovanje simbola **b** u najvećem broju slučajeva nije potrebno svih n bita (koliko je potrebno da bi se kodovala svaka vrednost iz opsega $1 \div 2^n - 1$, pri čemu je 2^n veličina prozora). Pretpostavimo dalje, da je za kodovanje simbola **b** u najvećem broju slučajeva dovoljno k bita (k<n). Postavlja se se pitanje kako kodovati vrednosti za **b** koje su veće od 2^k (a takve vrednosti su realne, mada retke)? Ovo se može rešiti statističkim kodovanjem, kao što je to objašnjeno u prethodnom paragrafu. Ali ako nam je korišćenje Hafmanovih frekvencija koje se nalaze u ponekad vrlo dugačkim tablicama suviše skupo (procesorski npr), potrebno nam je drugačije rešenje. Pošto se od svih 2^k podsekvenci dužine k samo sekvenca **0**, tj. $\underbrace{(00...0)}_{k\ puta}$ nikada ne može dogoditi kao dužina ponavljanja neke podsekvence, ona se može iskoristiti kao dodatni indikator, koji ukazuje da se sledećih n bita koriste kao binarni prikaz dužine ponovljene podsekvence. To znači da se, u slučaju da je ponavljana podsekvenca dužine l, $2^n > l > 2^k$, ona koduje sa

$$[1, \mathbf{a}, \mathbf{0}, \mathbf{b'}], \tag{3}$$

gde je

- 1 kodovano pomoću jednog bita,
- **a** kodovano pomoću n bita,
- **"0"** kodovano pomoću k bita (k < n),
- **b'** kodovano pomoću n bita.

Specijalni simbol **0** može imati bilo koju vrednost iz opsega $0 \div 2^k - 1$, a pošto se vrednost b=0 ne može nikad pojaviti (u klasičnom LZ77), nameće se baš ta vrednost.

Prilikom kodovanja, nakon što je vrednost **b** određena, potrebno je proveriti da li je ona veća od $2^k - 1$, pa ako jeste onda upisati **0** i odmah zatim dobijenu vrednost za **b** ali ovog puta kodovati sa n bita. U slučaju da je $b \leqslant 2^k - 1$ potrebno je kodovati kao

$$[1, \mathbf{a}, \mathbf{b''}], \tag{4}$$

gde je:

- 1 kodovano pomoću jednog bita,
- **a** kodovano pomoću n bita,
- **b''** kodovano pomoću k bita (k < n).

IV. Modifikacija 2

Kada je potrebno ispuniti zahteve poput onih navedenih u paragrafu II, zgodno bi bilo imati rečnik podsekvenci koje su unapred poznate i vrlo često se pojavljuju, kao što je recimo slučaj kod XML-a, gde se pojedina zaglavlja pojavljuju više puta (a najmanje dva puta). Neka je broj tih podsekvenci jednak **q**. Svaku od ovih „reči" možemo kodovati sa **p** bita, gde je $2^{p-1} < q \leqslant 2^p$.

Prilikom kodovanja „klasičan" LZ77 prozor treba proširiti rečnikom unapred određenih podsekvenci (reči) i ponavljanje (tj. najveću podsekvencu kod koje je utvrđeno da se pojavljuje još negde) prvo tražimo u tom proširenju, a samo u slučaju da tamo nije pronađena, tražimo je u neproširenom prozoru.

Slično modifikaciji 1, i ovde možemo koristiti poseban simbol za razlikovanje klasičnog LZ77 kodovanja i modifikovanog, tj.

$$[1, \mathbf{00}, \mathbf{R}], \tag{5}$$

gde je:

- 1 kodovano pomoću jednog bita,
- **00** kodovano pomoću n bita,
- **R** kodovano pomoću p bita.

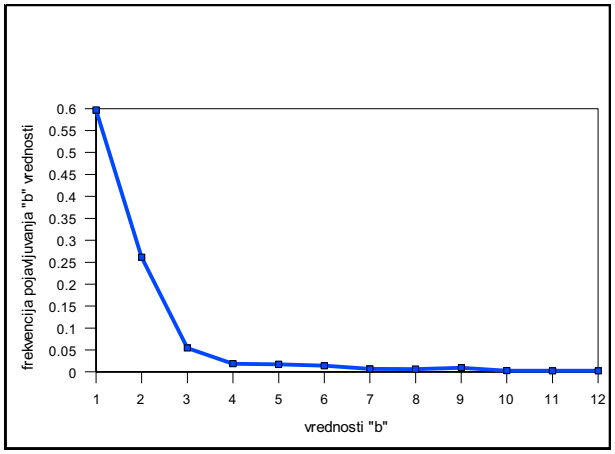
Simbol **00** je specijalni simbol koji označava da je ono što sledi "token" kodovan sa p bita, koji određuje reč iz unapred definisanog rečnika.

Jasno se primećuje nedostatak ove modifikacije jer se za redundantni simbol **00** koristi čitavih n bita, što dakle znači da bi ova modifikacija imala smisla samo za reči veće dužine, odnosno u rečnik mogu ući samo reči čija je dužina veća od **(**n+p)/(m+1) .

Korišćenjem modifikacije 1 i 2 na ulazu u dekoder mogu se pojaviti sledeće kombinacije kodovanih simbola (podsekvenci):

$$\begin{array}{l} [0, \mathbf{sym}], \\ [1, \mathbf{a}, \mathbf{b''}], \\ [1, \mathbf{a}, \mathbf{0}, \mathbf{b'}], \\ [1, \mathbf{00}, \mathbf{R}]. \end{array} \tag{6}$$

Da bismo se odlučili za određenu vrednost parametra k, od kojeg veoma zavisi stepen kompresije koji se može postići ovim algoritmom, potrebno je odrediti frekvencije pojavljivanja vrednosti parametra **b**, na većem broju sekvenci – uzoraka. Sl. 1. prikazuje frekvencije pojavljivanja pojedninih vrednosti **b** u sekvencama iz tabele 1.



*Sl. 1. Frekvencija pojavljivanja prvih 12 vrednosti parametra **b** na uzorku podataka.*

Na osnovu ovog grafika koji je dobijen kodovanjem podataka (mahom različitih XML sadržaja), sa prozorom veličine 255 (odnosno n=8), može se zaključiti da je za kodovanje parametra **b** u najvećem broju slučajeva dovoljno 3 bita (tj. vrednosti 1-7 se pojavljuju mnogo češće od ostalih). Na sl. 1. je zbog preglednosti prikazano samo prvih 12 vrednosti (sve ostale su manje od 0.0005).

V. Realizacija

Program koji omogućava primenu kodera i dekodera na osnovu gore opisanog algoritma je zbog potreba aplikacije pomenute u prvom paragrafu napisan i testiran u programskom jeziku Java, ali autori preporučuju korišćenje programskog jezika C++ koji omogućava bolje performarnse, bar što se tiče brzine izvršavanja.

VI. Rezultati algoritma

U tabeli 1 prikazani su stepeni kompresije za različite izvore XML-a koji se mogu pronaći na navedenim adresama. Vrednost parametra k je 3. Stepen kompresije je ovde izračunat na osnovu

$$a = \frac{L}{L_N} \ , \qquad (7)$$

pri čemu je L dužina u bajtovima komprimovanog fajla a L_N nekomprimovanog.

Tabela 1: Stepen kompresije za različite xml sadržaje (k=3).

Izvor podataka	*Stepen kompresije*
archives.seul.org/seul/edu/Nov-1998/msg00108.html	0.52
www.xml.com/pub/a/1999/01/namespaces.html	0.57
www.xml.com/pub/a/1999/01/namespaces.html	0.59
www.xml.com/pub/a/1999/01/namespaces.html	0.63
wap.b92.net	0.50
wap.yahoo.com	0.50
www.b92.net/info/rss/vesti.xml	0.83
www.nytimes.com/services/xml/rss/nyt/Business.xml	0.80
www.nytimes.com/services/xml/rss/nyt/Health.xml	0.81
www.nytimes.com/services/xml/rss/nyt/Music.xml	0.81

U dobijanju ovih rezultata korišćena je samo modifikacija 1. Kod korišćenja modifikacije 2 poželjno je da reči koje se koriste u rečniku budu što duže, i da ih bude što manje, pa se na taj način mogu postići još bolji rezultati.

VII. Zaključak

Modifikacije LZ77 algoritma, predstavljene u ovom radu, omogućavaju realizaciju veoma jednostavnog a time i relativno brzog kodera/dekodera za kompresiju podataka koji je moguće koristiti u prenosu XML sadržaja. Jednostavnost algoritma, kao primarni zahtev, omogućava njegovu primenu u mobilnoj telefoniji, kao i kod drugih slično ograničenih uređaja.

Dalja istraživanja ovog algoritma moguće je sprovesti razvojem modifikacije 2, koja je specifična za svaku vrstu podataka, pa se stoga može proizvoljno menjati.

Literatura

[1] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, IT-23(3):337–343, 1977.

[2] V. Senk. Lecture notes from the information theory and coding course., 1994. unpublished.

[3] P. Bender and J. Wolf. New asymptotic bounds and improvements on the lempel-ziv data compression algorithm. IEE Transactions on Information Theory, 37(3):721–729, 1991.

[4] M. M. Dalkilic, W. T. Clark, J. C. Costello, P. Radivojac, "Using Compression to Identify Classes of Inauthentic Texts" , School of Informatics, Indiana University, Bloomington, IN 47408.

Abstract

Need for data compression under specific conditions such as in data interchange between a mobile device (with limited capabilities) and a remote server is very important in mobile communications. This paper presents an algorithm that tries to address these requirements, by coding XML (Extensible Markup Language) data in order to achieve higher level of compression. We suggest two modifications of the original Lempel Ziv 77 algorithm, which increase its level of compression for this kind of sequences, and still provide mathematically simple encoder and decoder, that mobile devices with limited capabilities are able to process.

MODIFICATION OF LEMPEL-ZIV-77 ALGORITHM FOR DATA COMPRESSION

Miloš Stanimirović, Vojin Šenk, Ivan Stanojević