Predavanja IV

Tags Done

Kada imamo izvor informacija i raspodelu pojavljivanja simbola u izvornom alfabetu, cilj je bio formirati kod tako da je sredja duzina kodne reci u alfabetu koda minimalna. Srednja duzina kodne reci je proporcijalna duzini poruke. Sto je manja duzina kodne reci, to znaci da ce biti manja kodirana osnovna poruka.

Eliminisanje zavisnosti je moguce pomocu grupisanja, teorijski je moguce, ali u praksi nije izvodljivo. Eksponencijalno se povecava ukupan broj simbola kojom bi formirali kodnu rec.

Da li moze da se konstuise drugi metod koji ce uzeti u obzir zavisnot i bice upotrebljiv za poruku normalne duzine.

LZ77 i LZ78 su osnovni algoritmi.

77 i 78 jesu godine u kojima su nastali.

Kod Huff algoritma, radili smo procenu postojanja simbola u kodnoj reci i za svaki simbol formiramo kodnu rec i poruku kodiramo tako sto menjamo kodne reci.

Sama konstrukcija koda je bila nezavisna od poruke.

Ovde ujedno konstruisemo kod, kako dolazi poruka. Poruku predstavljamo kao niz simbola alfabeta izvora. Ideja je da pocev od trenutne pozicije
i pokusavamo da pronadjemo u prethodnom delu poruke, da li se taj odredjeni segment ponovio. Od i do j, da li se ta grupa simbola k negde ponovila.

```
Za trenutnu poziciju i, algoritam
trazi poziciju j od koje se najvise
elemenata ponavlja.
Formalno,
tako j za koje je max k takvo da je
x_i+s = x_j+s za s = 0,1..k-1
Pozicija j trazi se na rastojanju w.
Ne vise od w se odaljimo od trenutne
pozicije i trazimo najduze ponavljanje
na izlazu pamtimo (1,j,k)
```

```
1 - nasli ponavljanje
j - trenutak od koga se desava ponaljanje
k - broj simbola koji se ponavlja

u koliko se na poziciji i nalazi novi
simbol koji se pre nije pojavio, tada
se to ponavljanje ne prihvata.

Kodiranje

A   B   B   A   B   B   A   B   B
B   A   A   B   A   B   A
(0,A) (0,B) (1,1,1) (1,3,6) (1,4,2)
(1,1,1) (1,3,2) (1,2,2)

Dekodiranje:

A B B A B B A B B B A ...

U konkretnoj implementaciji potrebno je
naci nacin na koji se formiraju torke,
staticko binarno kodiranje outputa.
Cilj LZ alogritama je predprocesiranje, a nakon toga
torke i podatke koje dobijemo za njih
konstruisemo odgovarajuce reci na alfabetu
koda koje ce biti zapamcene.
```

Nedostatak LZ77 algoritma je sirina prozora w, gledamo da eliminisemo zavisnost samo w simbola unazad. Za razliku od grupisanja simbola, posmatramo grupe od w simbola, ali samo one koje se ponavljaju i broj ponavljanja. Postojanje w je odredjeni nedostatak jer sva ponavljanja koja su dalja od sirine prozora ne uzimamo u obzir. Takodje, problme moze da bude i vremenska slozenost algoritma. U koliko ne radimo pretprocesiranje teksta, moramo za svako 2 da vidimo koliko je to najduze ponavljanje.

LZ78

Varijanta LZ77 ali bez ogranicene duzine prozora, pri cemu on zahteva u isto vreme generisanje pomocne strukture podataka (recnik).

Tokom obrade, formiramo recnik pronadjenih simbola. Ucitavamo karaktere sve dok se ne dobije rec koja ne postoji u recniku. Trazimo najduzu rec koja se nalazi u recniku *wi,* a *c* je novi simbol. N aoutputu saljemo *(i,c)*, saljemo indeks *i* iz recnika i simbol c. U recnik ubacujemo *wi* kada se nadoveze *c.*

Recnik efikasno mozemo da rekonstruisemo i u procesu dekodiranja.

```
Kodiranje

A B B C B C A B A B C A A B C A A B
korak   pos   izlaz   recnik
    1      1     (0,A)    A
    2      2     (0,B)    B
    3      3     (2,C)    BC
    4      5     (3,A)    BCA
    5      5     (2,A)    BA
    6      10    (4,A)    BCAA
    7      14    (6,B)    BCAAB

Dekodiranje

<0,A> <0,B> <2,C> <3,A> <2,A> <4,A> <6,B>
A B B C B C A B A B C A A B C A A B
```

Recnik ne mora da bude rekonstuisam u istom fajlu, vec moze biti sahranjen u memoriji.

Prva osobina recnika je da ima kompaktnu reprezentaciju u memoriji. Ako pamtimo kao niz stringova, to zahteva dosta memorije, redudantnosti u memoriji. I sama pretraga je bitna zbog same vremenske slozenosti.

Niz istih simbola, samo A, u svakom koraku imacemo samo slovo A.. k-puta, dobijamo kvadratnu slozenost, k provera pri cemu je prvi string duzine 1 + 2.. + k. Dobijamo kvadratnu slozenost, gde u dugim porukama ona nije dobra. Duzina cekanja algoritma je tada neprihvatljiva.

Zbog toga su stabla bitna za predstavljanje recnika. Samo kodiranje je linearno

LZW

Varijavnta LZ78 koja ispravlja nedostatak, a to je format izlaza (indeks i naredni karakter). Bez tog karaktera mi ne bismo mogli da rekontruisemo tu kodnu rec i on ide bez kompresije, direknto ide u output. To se postize LZW algoritmo. Postoji problem na koji nacin osmisliti strukturu kodiranja za parove.

Prva stavka koja se razlikuje od LZ78, jeste to da se ubacuju svi simboli. U principu, kodiranje se postize na siti nacin, gledanjem najduzeg ponavljanja.

```
Kodiranje

a b c a b a b c
1 a
2 b
3 c
4 d
5 a b -> 6 b c
            -> 7 c a
            -> 8 a b a
            -> 9 a b c

Dekodiranje

1 -> a
2 -> b
3 -> c
5 -> a b

Rec koju ubacujemo u recnik ona
kasni jedna korak
a inicijanno ubacujemo sve
simbole u recnik

Prednost je sto u svakoj iteraciji
pamtimo index.
```

An Explanation of the `Deflate' Algorithm

It's important before trying to understand DEFLATE to understand the other two compression strategies that make it up -- Huffman coding and LZ77 compression.

https://www.zlib.net/feldspar.html