1. domača naloga: Stiskanje podatkov

Tokrat se bomo posvetili problemu brezizgubnega stiskanja podatkov [1]. Izdelali bomo lasten program za stiskanje datotek, ki bo deloval na podoben način kot vam zagotovo znane aplikacije (WinZip, WinRAR, gzip, 7-Zip, compress, bzip2, pkzip, ...).
Poznamo veliko postopkov, kako učinkovito zapisati nek niz znakov. Na predavanjih ste jih spoznali nekaj, ki temeljijo predvsem na informacijski teoriji: Shannonovo, Fanojevo, aritmetično in Huffmanovo kodiranje. Ogledali ste si tudi postopke, ki kodirajo na podlagi slovarja (LZW, LZ77) in algoritem DEFLATE, ki v grobem združuje LZ77 in Huffmanovo kodiranje. Obstaja seveda še veliko drugih pristopov (Rice, Golomb, Burrows-Wheeler, LZMA, ...). Pri tej nalogi se bomo osredotočili na postopek kodiranja bajtnih parov (angl. Byte Pair Encoding – BPE).

Algoritem kodiranja bajtnih parov

Algoritem kodiranja bajtnih parov (BPE) [2] je brezizgubni postopek za stiskanje podatkov. Prvotno je bil zasnovan za stiskanje datotek, vendar se je uveljavil tudi v drugih aplikacijah, npr. pri strojnem prevajanju [3]. Izvedenke algoritma BPE so postale ključnega pomena tudi pri segmentaciji besedil v žetone, ki jih uporabljajo veliki jezikovni modeli, kot so GPT-2, GPT-3 in BERT.

Algoritem BPE odlikuje preprosta implementacija in učinkovitost. Deluje na principu iterativnega združevanja najpogostejših parov znakov v nizu. V vsakem koraku algoritma poiščemo najpogostejši par znakov in ga zamenjamo z indeksom znakov v slovarju. Postopek se ponavlja, dokler v nizu ne najdemo več parov, ki jih lahko združimo, ali dokler ne presežemo določenega števila vnosov v slovarju.

BPE - kodiranje

```
inicializiraj začetni seznam vrednosti znakov S (nabor ASCII)
znake ASCII v vhodnem nizu zamenjaj z indeksi iz S
ponavljaj:
    preštej pojavitve vseh parov znakov [z1,z2] v vhodnem nizu
    poišči najpogostejši par znakov Z=najpogostejši([z1,z2])
    če je frekvenca(Z) < 2 ali je presežena dovoljena dolžina S:
        prekini
    na konec seznama S dodaj Z
    k=indeks(Z)
    zamenjaj vse pare znakov Z v vhodnem nizu s k
```

BPE - dekodiranje

```
preberi seznam znakov S
i=dolzina(S)-1
ponavljaj dokler i>=256:
    zamenjaj vse pojavitve i v vhodnem nizu s S[i]
    i=i-1
indekse v izhodnem nizu zamenjaj z znaki ASCII
```

Naloga

V programskem jeziku Python napišite funkcijo z imenom `naloga1` . Funkcija mora implementirati kodiranje in dekodiranje sporočil na osnovi algoritma za kodiranje bajtnih parov. Največje število vnosov v seznamu znakov je omejeno na $2^{12} = 4096$; indekse torej kodiramo z enakomernim 12-bitnim kodom. Kot vhodni argument funkcija sprejme `vhod`, ki predstavlja vhodno sporočilo, in seznam znakov oziroma parov indeksov `vhodS`. Če je `vhodS` prazen, pomeni, da je potrebno `vhod` kodirati, sicer pa je potrebno `vhod` dekodirati. Izhodni argumenti funkcije so trije: zakodirano/odkodirano sporočilo `izhod`, seznam `izhodS` in kompresijsko razmerje `R`. Glede na to ali vhod kodiramo ali dekodiramo, se izhodni argumenti določijo na naslednji način:

- Če kodiramo: 8-bitne znake v sporočilu `vhod` zakodiramo v `izhod` z algoritmom BPE. Kompresijsko razmerje `R` se izračuna kot $\frac{|vhod|*8}{|\text{izhod}|*12}$, kjer je $|x|$ število znakov v x. Seznam `izhodS` vsebuje seznam vrednosti vseh znakov ASCII, ki mu sledijo vsi pari indeksov, v istem vrstem redu, kot so bili dodani pri kodiranju. Kod ni enoznačen, saj se lahko zgodi, da je več parov znakov z isto, najvišjo frekvenco, kar pomeni, da imamo na voljo več možnosti za združevanje parov znakov. Da dosežemo enoznačno kodiranje, vedno izberemo tisti par, ki se na vhodu pojavi prvi, če sporočilo beremo od leve proti desni. Poleg tega pare znakov vedno zamenjujemo v vrstnem redu kot se pojavljajo na vhodu, če sporočilo beremo od leve proti desni.

- Če dekodiramo: `vhod` vsebuje zakodirano sporočilo, ki ga zdaj sestavljajo indeksi: števila med 0 in 4095. `vhodS` vsebuje seznam vrednosti vseh znakov ASCII, ki mu sledijo vsi pari indeksov, v istem vrstem redu, kot so bili dodani pri kodiranju. Sporočilo odkodiramo v `izhod`, ki je seznam znakov ASCII. Kompresijsko razmerje `R` se izračuna kot $\frac{|izhod|*8}{|\text{vhod}|*12}$, kjer je $|x|$ število znakov v x. `izhodS` naj bo prazen.

Prototip funkcije:

```
def naloga1(vhod: list, vhodS: list) -> tuple[list, list, float]:
  """
  Izvedemo kodiranje ali dekodiranje z algoritmom BPE.
  Najvecja dolzina vhodS je 4096.

  Parameters
  ----------
  vhod : list
    Seznam vhodnih znakov: bodisi znaki abecede (ASCII)
    (ko kodiramo) bodisi indeksi
    (ko dekodiramo).
  vhodS : list
    Seznam ASCII kod in parov indeksov
    ce je []: kodiramo vhod
    sicer: dekodiramo vhod

  Returns
  -------
  (izhod, izhodS, R) : tuple[list, list, float]
      izhod : list
          Ce kodiramo: "izhod" je kodiran "vhod",
          Ce dekodiramo: "izhod" je dekodiran "vhod"
      izhodS : list
          Ce kodiramo: seznam ASCII kod in parov indeksov
          Ce dekodiramo: []
      R : float
          Kompresijsko razmerje
  """

  izhod = []
  izhodS = []
  R = float('nan')
  return (izhod, izhodS, R)
```

Testni primeri

Na učilnici se nahaja arhiv `tis-naloga1.zip`, ki vsebuje tri testne primere sporočil, za katere imate podane vhode in izhode. Primeri so podani v obliki datotek `.json`. Priloženo imate tudi funkcijo `test_naloga1`, ki jo lahko uporabite za preverjanje pravilnosti rezultatov, ki jih vrača vaša funkcija. Pri testiranju vaše funkcije upoštevajte, da je rezultat pravilen, če se `izhod` in `izhodS` popolnoma ujemata z rešitvami; dovoljeno odstopanje za `R` je 10^{-6}. Izvajanje vaše funkcije je časovno omejeno na 30 sekund.

Vaš program lahko uporablja samo tiste pakete, ki so del standardne knjižnice Python 3.12 (https://docs.python.org/3.12/library/) in paket `numpy`. Na sistemu za preverjanje vaših rešitev drugi paketi niso nameščeni.

Namigi

Python ima že vgrajeno podatkovno strukturo slovar (*dictionary*), ki vam bo prišla zelo prav pri reševanju naloge. Prav tako vam bo lahko prav prišla funkcija `Counter`, ki je del modula `collections`.

Literatura

[1] D.G. Luenberger: Information Science, Princeton University, pogl. 4, 2006.

[2] Philip Gage. 1994. A new algorithm for data compression. C Users J. 12, 2 (Feb. 1994), 23–38.

[3] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units,"in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 2016.