NI-VSM – 1.domácí úkol

Eliška Krátká (kratkeli), Ondřej Wrzecionko (wrzecond), Eliáš El Frem (elfreeli)

Obsah

1 Úvod

V této práci se věnujeme řešení 1. domácího úkolu z NI-VSM na téma entropie a kódování. Pracujeme se dvěma datovými soubory, respektive texty. Cílem práce je nalézt optimální binární instantní kód CC pro kódování znaků z prvního textu a porovnat, zdali je kód CC optimální i pro text druhý. Řešení jsme implementovali v programovacím jazyce Python.

1.1 Zadání úkolu

1. **(1b)** Z obou datových souborů načtěte texty k analýze. Pro každý text zvlášť odhadněte pravděpodobnosti znaků (symbolů včetně mezery), které se v textech vyskytují. Výsledné pravděpodobnosti graficky znázorněte.
2. **(1b)** Pro každý text zvlášť spočtěte entropii odhadnutého rozdělení znaků.
3. **(2b)** Nalezněte optimální binární instantní kód CC pro kódování znaků **prvního** z textů.
4. **(2b)** Pro každý text zvlášť spočtěte střední délku kódu CC a porovnejte ji s entropií rozdělení znaků. Je kód CC optimální i pro **druhý** text [1]?

1.2 Parametry úlohy

Reprezentantem skupiny je **Eliáš El Frem**. Parametry jsme vypočítali dle vzorce ze zadání úkolu [1]:

$$X = ((K \cdot L \cdot 23) \bmod 20) + 1,$$
$$Y = ((X + ((K \cdot 5 + L \cdot 7) \bmod 19)) \bmod 20) + 1,$$

kde K je den narození reprezentanta skupiny a L počet písmen v příjmení reprezentanta. Pro úlohu jsme na základě výpočtu parametrů použili datové soubory *005.txt* (první text) a *004.txt* (druhý text).

```
#!/usr/bin/env python3

K = 17
L = len("Frem")
fname1 = ((K*L*23) % (20)) + 1
fname2 = ((fname1 + ((K*5 + L*7) % (19))) % (20)) + 1
```

2 Postup řešení

V této sekci vysvětlujeme postup řešení domácího úkolu společně s klíčovými částmi zdrojového kódu, implementovaného v jazyce Python.

2.1 Výskyt znaků v textech

Začali jsme s odhadem pravděpodobností výskytu znaků v textech. Z datových souborů jsme si načetli oba soubory a zvlášť pro každý soubor a znak spočítali četnost. Pravděpodobnost výskytu znaku $P(X)$ odpovídá

$$P(X) = \frac{\text{četnost znaku}}{\text{celkový počet znaků}},$$

kde X je náhodná veličina – počet výskytů daného znaku v textu.

```
#!/usr/bin/env python3

file1 = open(f'hw1-source/00{fname1}.txt','r')
file2 = open(f'hw1-source/00{fname2}.txt','r')

file1_string = file1.read()
file2_string = file2.read()

# vypocet cetnosti
def countFreqs(to_cnt):
    char_cnt = {}
    for i in to_cnt.lower():
        if not char_cnt.get(i):
            char_cnt[i] = 1
        else:
            char_cnt[i] += 1
    return char_cnt

# odhad pravdepodobnosti
def countProbs(to_cnt):
    to_ret = {}
    char_cnt = countFreqs(to_cnt)
    for i in char_cnt.keys():
        to_ret[i] = char_cnt[i]/len(to_cnt)
    return to_ret

f1_freqs = countFreqs(file1_string)
f2_freqs = countFreqs(file2_string)
f1_probs = dict(sorted(countProbs(file1_string).items()))
f2_probs = dict(sorted(countProbs(file2_string).items()))
```

2.2 Entropie odhadnutého rozdělení znaků

Pro každý text a jemu odpovídající odhadnuté rozdělení znaků jsme spočítali entropii dle vzorce

$$H(X) = -\sum_{x \in X} p(x) \log p(x),$$

kde $p(x)$ je pravděpodobnostní funkce veličiny X v bodě $x \in \mathbb{R}$ a log značí dvojkový logaritmus o základu 2 [2].

```
#!/usr/bin/env python3

# vypocet entropie
probs1 = np.fromiter(f1_probs.values(),float)
probs2 = np.fromiter(f2_probs.values(),float)
entropy1 = -probs1.T.dot(np.log2(probs1))
entropy2 = -probs2.T.dot(np.log2(probs2))
```

2.3 Optimální binární instantní kód CC

Sestrojili jsme optimální binární instantní kód CC pro kódování znaků obsažených v prvním textu. Optimální kód je kód s nejmenší možnou střední délkou. Kód je instantní právě tehdy, když žádné kódové slovo není prefixem jiného. Tyto vlastnosti splňuje Huffmanův kód, který jsme implementovali dle formulace v 6. přednášce. Algoritmus opakovaně spojuje dva nejméně pravděpodobné znaky do jednoho symbolu a vytváří tak binární strom, ze kterého se pak zpětným chodem zkonstruují kódová slova pro všechny původní hodnoty [3].

```
#!/usr/bin/env python3

# list stromu
class node:
    def __init__(self, freq, char=None):
        self.freq = freq
        self.char = char
        self.left = None
        self.right = None
        self.code = ''
    def __lt__(self, sec):
        return self.freq < sec.freq

# prioritni fronta
Q = PriorityQueue()
for i in ([node(v, k) for k, v in f1_freqs.items()]):
    print(i.char)
    Q.put(i)
while (Q.qsize() > 1):
    left = Q.get()
    right = Q.get()
    tmp = node(left.freq + right.freq)
    tmp.left = left
    tmp.right = right
    Q.put(tmp)
top = Q.get()

# sestrojeni Huffmanova kodu
def construct_code(root):
    Q = PriorityQueue()
    Q.put(root)
    coding = {}
    while(not Q.empty()):
        curr = Q.get()
        right = curr.right
        left = curr.left
        if(right):
            right.code = curr.code + '0'
            Q.put(right)
        if(left):
            left.code = curr.code + '1'
            Q.put(left)
        if(curr.char):
            coding[curr.char] = curr.code
    return coding
```

2.4 Střední délka kódu CC

V závěru jsem pro oba texty spočetli střední délku kódu CC odpovídající

$$L(CC) = \sum_{x \in X} l(x)p(x),$$

kde $l(x)$ je délka kódového slova příslušejícího prvku $x \in X$ [3]. K posouzení, zdali je kód CC optimální i pro druhý text jsme využili větu z 6. přednášky, která dává do souvislosti entropie a střední délku kódu.

Uvažujme optimální D-nární kód C^* diskrétní náhodné veličiny X. Potom platí

$$H_D(X) \leq L(C^*) < H_D(X) + 1 \tag{1}$$

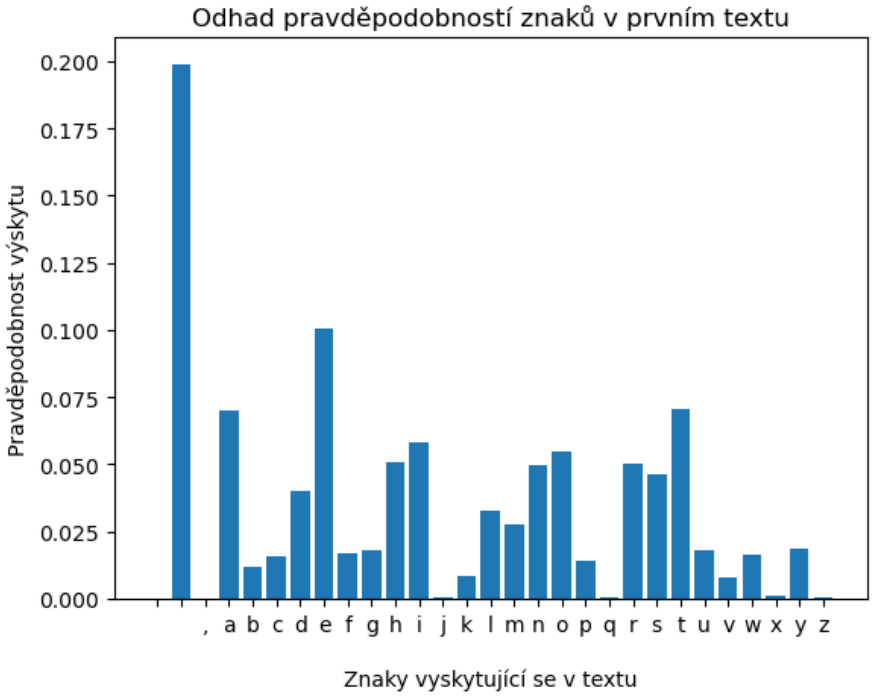
```
#!/usr/bin/env python3

# vypocet stredni delky kodu
codeLens = np.array([len(v) for _, v in coding.items()])
Lc1 = codeLens.T.dot(probs1)
Lc2 = codeLens.T.dot(probs2)
```

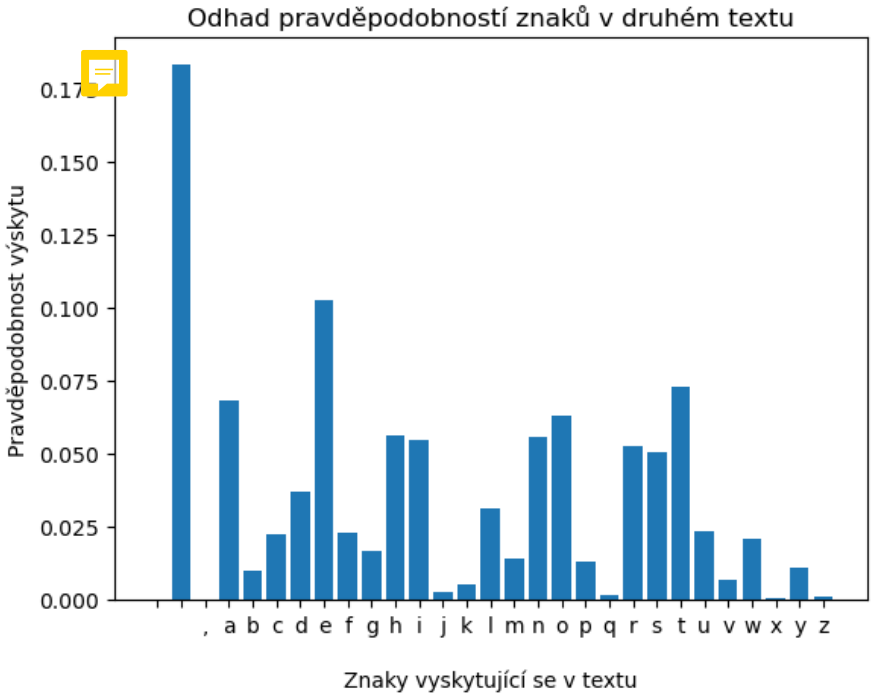
3 Výsledky

V domácím úkolu jsme analyzovali dva anglické texty. Datový soubor *005.txt* obsahuje první text a *004.txt* druhý text.

3.1 Odhad pravděpodobností znaků

Z grafů odhadnutých pravděpodobností znaků pozorujeme, že v obou textech se po mezeře nejčastěji vyskytují znaky e, t, a a, která jsou nejčetnějšími písmeny v běžném anglickém textu [4].





3.2 Entropie odhadnutého rozdělení znaků

Entropie odhadnutého rozdělení znaků prvního textu je o pár setin menší než textu druhého. V tabulce ji uvádíme zaokrouhlenou na tři desetinná místa.

Text	**Entropie**
první (*005.txt*)	4.069
druhý (*004.txt*)	4.084

3.3 Optimální binární instantní kód CC

Optimální binární instantní kód CC pro **první** text každému znaku přiřazuje kódové slovo různé délky. Častěji vyskytujícím se znakům odpovídají kratší kódová slova a naopak.

Znak	**Kódové slovo**
mezera	11
\n	000111010110
,	000111010111
a	0100
b	0001101
c	011001
d	00100
e	0000
f	010101
g	010100
h	1001
i	0111
j	00011101010
k	0001111
l	01011
m	01101
n	1011
o	1000
p	0001100
q	0001110110
r	1010
s	00010
t	0011
u	001011
v	00011100
w	011000
x	0001110100
y	001010
z	0001110111

3.4 Střední délka kódu CC

Když srovnáme entropii rozdělení znaků se střední délkou kódu CC dle věty (1), tak pozorujeme, že pro oba texty je tato podmínka splněna. Tímto srovnáním jsme ověřili vztah střední délky kódu a entropie dle věty (1). Nicméně podmínka ve větě (1) je pouze postačující, ne nutná.[1] Z toho důvodu nemůžeme pouze na základě střední délky kódu rozhodnout o tom, zdali je CC optimální pro druhý text. Pro druhý text jsme spočítali Huffmanův kód CC2 (optimální instatní kód) a jeho střední délku, kterou jsme porovnali se střední délkou kódu CC. Střední délka kódu CC je větší než střední délka CC2, z čehož plyne, že kód CC **není optimální** pro druhý text. V tabulce opět uvádíme hodnoty zaokrouhleny na tři desetinná místa.

[1] *kód CC je optimální* $\Rightarrow$ *platí nerovnovnost v (1)*

Text	Entropie	Střední délka kódu CC	Střední délka kódu CC2
první (*005.txt*)	4.069	4.114	-
druhý (*004.txt*)	4.084	4.151	4.122

3.5 Optimální binární instantní kód CC2

Pro úplnost přikládáme i kód CC2.

Znak	Kódové slovo
mezera	000
\n	111110000110
,	111110000111
a	0100
b	0101010
c	001000
d	00101
e	110
f	11110
g	010100
h	0111
i	1001
j	11111001
k	1111101
l	01011
m	111000
n	1000
o	0110
p	111001
q	111110001
r	1010
s	1011
t	0011
u	11101
v	0101011
w	001001
x	11111000010
y	111111
z	1111100000

4 Závěr

Zanalyzovali jsme dva datové soubory, text **005.txt** a **004.txt**. Pro oba texty jsme nalezli odhad pravděpodobností výskytu znaků a spočítali jejich entropii. Pro první text jsme nalezli optimální instantní kód CC a ověřili jsme, že platí vztah entropie a střední délky kódu dle nerovnosti ve větě (1). Pro druhý text jsme pouze na základě výpočtu střední délky kódu nemohli ověřit, zdali je optimální nebo ne. Nalezli jsme optimální instantní kód CC2 pro druhý text, spočítali jeho střední délku, která je menší než střední délka kódu CC. Kód CC proto není optimální instantní kód pro druhý text.

Reference

[1] P. Hrabák. Domácí úkol 1. https://courses.fit.cvut.cz/MI-SPI/homework/hw1/index.html.

[2] P. Hrabák, P. Novák, D.Vašata. Entropie. https://courses.fit.cvut.cz/MI-SPI/lectures/files/NI-VSM-Lec-05-Slides.pdf.

[3] P. Hrabák, P. Novák, D.Vašata. Teorie informace. https://courses.fit.cvut.cz/MI-SPI/lectures/files/NI-VSM-Lec-06-Slides.pdf.

[4] R. Lórencz. Základní pojmy v kryptologii, substituční šifry. blokové, transpoziční šifry. https://courses.fit.cvut.cz/BI-BEZ/media/lectures/bez1.pdf.