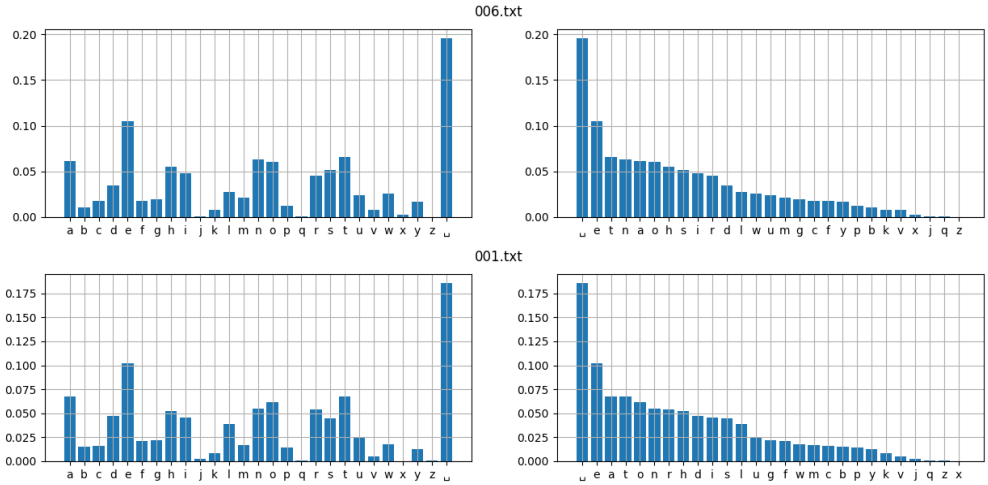
NI-VSM – Úloha 1.

Kristýna Janovská, Jakub Rigoci, Jorge Zuñiga

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.

Porovnávali jsme soubory 001.txt a 006.txt. Použitým jazykem byl Python, pro vizualizaci byla použita knihovna matplotlib.
Pravděpodobnost znaků byla odhadnuta jako poměr počtu výskytů daného znaku a celkového počtu znaků. Výstupem funkce `char_probability`, která tyto pravděpodobnosti počítá je slovník, kde klíč je znak abecedy a item je pravděpodobnost výskytu.

K = 20
L = len(Janovská)



Z grafů vyplývá, že se pravděpodobnosti jednotlivých znaků v souborech podobají, nejčastějšími znaky byly v obou případech mezera a e, nejméně častými pak q, z, j a x.

2. **(1b)** Pro každý text zvlášť spočtěte entropii odhadnutého rozdělení znaků.

Entropie byla vypočtena na základě vzorce

$$H(\boldsymbol{X}) = -\sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x}) \log p(\boldsymbol{x}).$$

kde X je množina znaků. V kódu slouží k výpočtu entropie funkce `get_entropy`.
Výsledné entropie:

001.txt: 4.0946897287085235
006.txt: 4.085023448817994

3. **(2b)** Nalezněte optimální binární instantní kód C pro kódování znaků **prvního** z textů.

Použit byl Huffmanův kód, neboť podle věty 6.17 je tento kód optimální. Byl použit algoritmus z kapitoly 6.6, kdy se postupně spojí dvě nejméně pravděpodobné hodnoty, dokud nezůstane jedna poslední, reverzním chodem se pak konstruuje kódové slovo. Pro hodnotu x, která vznikla spojením hodnot u a v se vytvoří kódové slovo méně pravděpodobné hodnoty připojením 1 za C(x) a kódové

slovo více pravděpodobné hodnoty připojením 0 za C(x). Je tak sestrojen binární Huffmanův kód.
V kódu probíhá výpočet Huffmanova kódu pomocí funkce `binary_code`.
Optimální kód sestrojen pro text 001.txt:

[('a', '1011'), ('b', '100010'), ('c', '100011'), ('d', '0010'), ('e', '010'), ('f', '110101'), ('g', '00010'), ('h', '0011'), ('i', '0000'), ('j', '110001101'), ('k', '1100010'), ('l', '11001'), ('m', '110000'), ('n', '0111'), ('o', '1001'), ('p', '100001'), ('q', '11000110011'), ('r', '0110'), ('s', '11011'), ('t', '1010'), ('u', '00011'), ('v', '11000111'), ('w', '110100'), ('x', '11000110010'), ('y', '100000'), ('z', '1100011000'), (' ␣', '111')]

4. (2b) Pro každý text zvlášť spočtěte střední délku kódu C a porovnejte ji s entropií rozdělení znaků. Je kód C optimální i pro druhý text?

Střední délka byla vypočtena na základě vzorce

$$L(C) = \sum_{x \in \mathcal{X}} \ell(x) p(x)$$

kde X je množina znaků a l(x) délka kódového slova příslušejícího prvku x. Výpočet střední délky je zajištěn funkcí `encoding_mean`, a následná kontrola optimality kódu pak funkcí `is_code_optimal`.
Výsledné střední délky:

001.txt: 4.137503871167543
006.txt: 4.1560782967032965

Dle věty 6.14 pro optimální instantní D-ární kód C* diskrétní náhodné veličiny X platí

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

Podle této věty jsme zjistili, že kód je optimální jak pro text 001.txt, tak i pro text 006.txt

Závěr

Náplní 1. úlohy bylo analyzovat a porovnávat dva texty, v našem případě 001.txt a 006.txt. Byly zjištěny pravděpodobnosti výskytů jednotlivých znaků, spočítány entropie pro jednotlivé texty, vytvořen optimální kód pro text 001.txt a na základě tohoto kódu spočteny střední délky kódu pro oba texty. Na závěr byla testována optimalita tohoto kódu, přičemž se ukázalo, že kód je optimální pro oba dva texty.