Avaliação: Códigos de Huffman

SISTEMAS DE COMUNICAÇÃO II (COM029008)

Rhenzo Hideki Silva Kajikawa

27 de Janeiro de 2025

Sumário

1. Introdução

2. Desenvolvimento

2.1. Questão 1

Considere uma fonte discreta sem memória (DMS) com alfabeto dado por $\mathcal{X} = \{a, b, c\}$ e probabilidades respectivas dadas por $p_X = \left[\frac{3}{10}, \frac{6}{10}, \frac{1}{10}\right]$.

(a) Calcule a entropia da fonte.

(b) Determine um código de Huffman para a fonte. Qual o comprimento médio do código obtido?

(c) Calcule a entropia da extensão de segunda ordem da fonte.

(d) Determine um código de Huffman para a extensão de segunda ordem da fonte. Qual o comprimento médio do código obtido? Comente o resultado.

2.2. Calculo da entropia da fonte

Para calcular a entropia da fonte será utilizada a seguinte formula:

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x) \tag{1}$$

Logo para está fonte podemos:

$$H(X) = \frac{3}{10} \cdot \log_2\left(\frac{3}{10}\right) + \frac{6}{10} \cdot \log_2\left(\frac{6}{10}\right) + \frac{1}{10} \cdot \log_2\left(\frac{1}{10}\right) \tag{2}$$

$$H(X) = 0,521 + 0,442 + 0,332 = 1,295 \tag{3}$$

Código para validar:

```
import komm
import numpy as np

px = [3/10, 6/10, 1/10]

#Calculando entropia da fonte
dms = komm.DiscreteMemorylessSource(px)
print(dms.entropy())
```

2.3. Código Huffman da fonte e comprimento

Para fazer o código Huffman e o comprimento iremos fazer primeiro o diagrama:

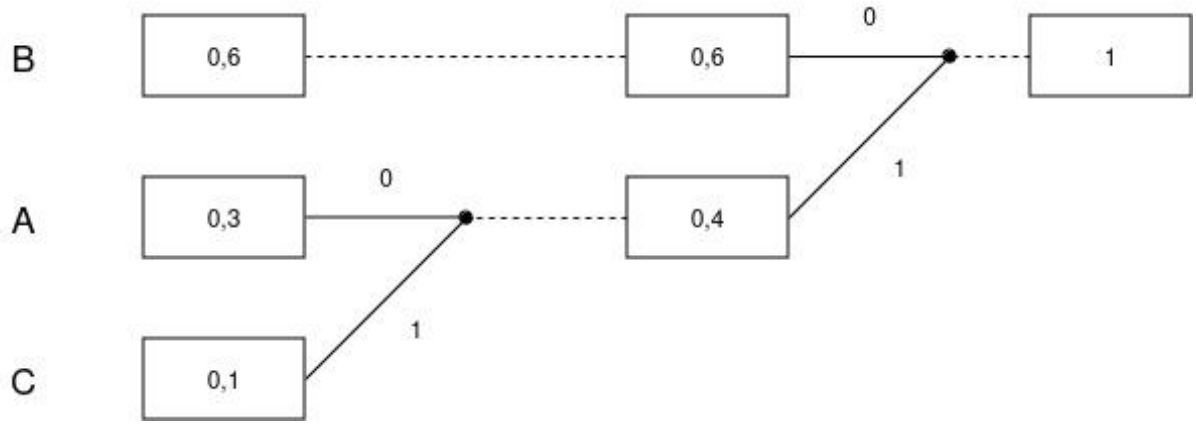

Figura 1: Fonte: Elaborada pelo autor

A partir desse diagrama é possivel achar que : dado por $\mathcal{X} = \{a, b, c\}$ temos $[10, 0, 11]$

Agora para calcular o comprimento temos :

$$l = 1 * 0,6 + 2 * 0,3 + 2 * 0,1 l = 1.4 \text{ bits/letras} \tag{4}$$

Código para validar:

```
huffman_code = komm.HuffmanCode(px)
print(huffman_code.codewords , huffman_code.rate(px))
```

2.4. Calculo da extensão do código de Huffman para para segunda

Para extender o código para a segunda ordem, é seguida da seguinte forma :

$$
\begin{aligned}
X^2 &= \{\text{aa}, \text{ab}, \text{ac}, \text{bb}, \text{ba}, \text{bc}, \text{cc}, \text{ca}, \text{cb}\} \\
P_x^2 &= \{0.09, 0.18, 0.03, 0.36, 0.18, 0.06, 0.01, 0.03, 0.06\} \\
H(X^2) &= 0.09 \cdot \log_2(0.09) + 0.18 \cdot \log_2(0.18) + 0.03 \cdot \log_2(0.03) \\
&+0.36 \cdot \log_2(0.36) + 0.18 \cdot \log_2(0.18) + 0.06 \cdot \log_2(0.06) \\
&+0.01 \cdot \log_2(0.01) + 0.03 \cdot \log_2(0.03) + 0.06 \cdot \log_2(0.06) \\
H(X^2) &= 2.59 \text{ bits/superpalavras}
\end{aligned}
\tag{5}
$$

Código para validar:

```
H_X = dms.entropy()

# Entropia da extensão de segunda ordem
H_X2 = 2 * H_X

print(f"Entropia da fonte original (H(X)): {H_X:.4f} bits")
print(f"Entropia da extensão de segunda ordem (H(X^2)): {H_X2:.4f} bits")
```

2.5. Determinando a extensão de segunda ordem e comprimento médio

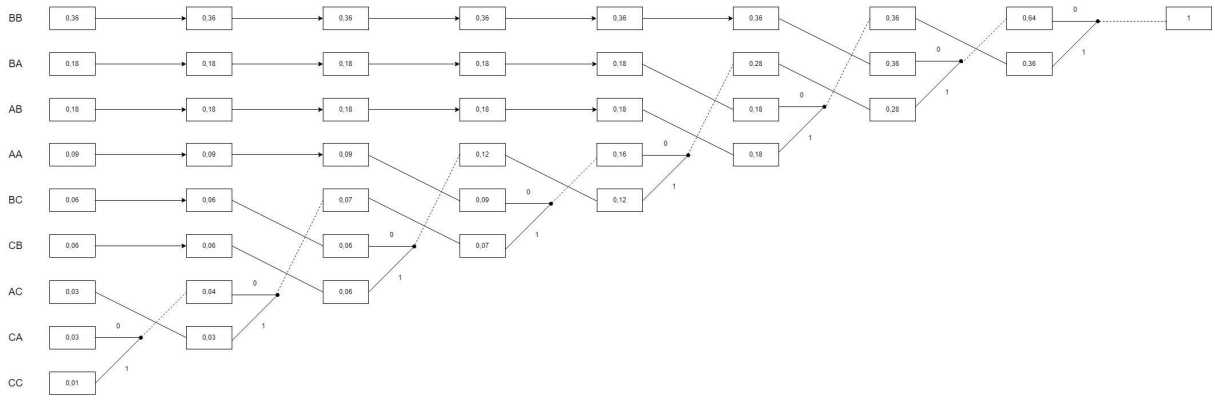

Figura 2: Fonte: Elaborada pelo autor

$$X^2 = \{\text{aa}, \text{ab}, \text{ac}, \text{bb}, \text{ba}, \text{bc}, \text{cc}, \text{ca}, \text{cb}\}$$
$$\text{Temos na sequência as palavras} \tag{6}$$
$$\{0100, 11, 010100, 10, 00, 0111, 01011, 0110, 010101\}$$

Para o comprimento médio temos:

$$\{\text{aa}, \text{ab}, \text{ac}, \text{bb}, \text{ba}, \text{bc}, \text{cc}, \text{ca}, \text{cb}\}$$
$$\{0.09, 0.18, 0.03, 0.36, 0.18, 0.06, 0.01, 0.03, 0.06\}$$
$$\{0100, 11, 010100, 00, 10, 0111, 010101, 01011, 0110\} \tag{7}$$
$$\{4, 2, 6, 2, 2, 4, 5, 4, 6\}$$
$$l = 0.09 * 4 + 0.18 * 2 + 0.03 * 6 + 0.36 * 2 + 0.18 * 2 + 0.06 * 4 + 0.01 * 6 + 0.03 * 5 + 0.06 * 4$$
$$l = 2.67$$

Código para validar:

```
# Símbolos e probabilidades da fonte original
symbols = ['a', 'b', 'c']
probabilities = [3/10, 6/10, 1/10]

# Criação dos símbolos da extensão de segunda ordem
symbols_second_order = [s1 + s2 for s1 in symbols for s2 in symbols]

# Cálculo das probabilidades da extensão de segunda ordem
probabilities_second_order = [p1 * p2 for p1 in probabilities for p2 in
probabilities]

# Criação do código de Huffman para a extensão de segunda ordem
huffman_code_second_order = komm.HuffmanCode(probabilities_second_order)

# Exibição do código de Huffman
print("Código de Huffman para a extensão de segunda ordem:")
for symbol, code in zip(symbols_second_order,
huffman_code_second_order.codewords):
    print(f"Símbolo: {symbol}, Código: {code}")

# Cálculo do comprimento médio do código
average_code_length = sum(len(code) * prob for code, prob in
zip(huffman_code_second_order.codewords, probabilities_second_order))
print(f"Comprimento médio do código: {average_code_length:.4f}")
```

2.6. Questão 2

Escreva um programa para comprimir e descomprimir arquivos de texto usando códigos de Huffman. Seu programa deve:

• Determinar a frequência de cada caractere do arquivo de entrada.

• Utilizar essas frequências para construir o código de Huffman.

• Comprimir o arquivo de entrada .txt usando o código de Huffman, gerando um arquivo de saída com extensão .bin.

• Descomprimir o arquivo de saída .bin , gerando um arquivo .txt idêntico ao arquivo de entrada.

Por simplicidade, assuma que o código de Huffman seja conhecido tanto na compressão quanto na descompressão — na prática, o código deve ser armazenado no arquivo de saída para que o arquivo de entrada possa ser descomprimido.

Teste seu programa com o livro Alice's Adventures in Wonderland, de Lewis Carroll, disponível em https://www.gutenberg.org/files/11/11-0.txt. Para este caso, determine:

(a) A entropia da distribuição de frequências dos caracteres do livro.

(b) O comprimento médio do código de Huffman obtido.

(c) O tamanho (em bytes) e a taxa de compressão do arquivo comprimido

Compare com a tabela a seguir, que mostra o tamanho do arquivo original e dos arquivos comprimidos com diferentes formatos de compressão

Formato	Tamanho(bytes)	Taxa de compressão
original	154573	0.00%
zip	54176	64.95%
gz	54037	65.04%
zst	48789	68.44%
7z	48280	68.77%
xz	48232	68.80%
bz2	42779	72.32%
bz3	40362	73.89%

2.7. Entropia da distribuição e Comprimento médio

Para fazer o calculo de entropia é necessário fazer a abertura e leitura do arquivo e armazenar a quantia de caracteres. sendo assim foi criado uma função para fazer a contagem dos characteres, gerar e gerar a pmf

```
def contar_caracteres(arquivo):
    contador = {}
    pmf = {}

    # Ler o arquivo e contar caracteres
    with open(arquivo, 'rb') as f:  # Mudado para 'rb'
        texto = f.read().decode('utf-8')  # Decodifica após ler os bytes
        total_chars = len(texto)

        for char in texto:
            contador[char] = contador.get(char, 0) + 1

    # Calcular PMF
    for char, freq in contador.items():
        pmf[char] = freq / total_chars

    return contador, pmf, texto
arquivo = 'alice.txt'

# Contar caracteres e obter texto
contador, pmf, texto_original = contar_caracteres(arquivo)


# Exibir contagem de caracteres em ordem decrescente
print("Contagem de caracteres:")
# Ordenar por quantidade (valor) em ordem decrescente
for char, quantidade in sorted(contador.items(), key=lambda x: x[1],
reverse=True):
    print(f'{char}: {quantidade}')
```

Esta primeira parte apenas mostra os characteres e a sua respectiva quantidade no livro.

```
Contagem de caracteres:
 : 24633
e: 13552
t: 10345
a: 8239
...
```

Após apresentar os caracteres é mostrada a pmf , calculada a entropia e comprimento médio

```
# Ordenar a PMF em ordem decrescente
sorted_pmf = sorted(pmf.items(), key=lambda item: item[1], reverse=True)

# Criar código de Huffman
probs = [prob for _, prob in sorted_pmf]
```

```
print("\nPMF:")
for char, prob in sorted_pmf:
    print(f'{char}: {prob:.6f}')

huffman = komm.HuffmanCode(probs)

dms = komm.DiscreteMemorylessSource(probs)
print("A entropia da distribuição de frequências dos caracteres do livro:",
dms.entropy())
print("Comprimento médio do código de Huffman obtido:", huffman.rate(probs))
```

temos como saida:

```
PMF:
 : 0.166352
e: 0.091519
t: 0.069862
a: 0.055640
o: 0.054519
...
A entropia da distribuição de frequências dos caracteres do livro:
4.620542049477668
Comprimento médio do código de Huffman obtido: 4.661185321249611
```

Assim é obtido que a entropia obtida pelo livro é de aproximadamente 4.62 e o comprimento médio do código de huffman obtido é de 4.66

2.8. Tamanho (em bytes) e a taxa de compressão do arquivo comprimido

3. Conclusão