Avaliação: Códigos de Huffman

SISTEMAS DE COMUNICAÇÃO II (COM029008)

Rhenzo Hideki Silva Kajikawa

27 de Janeiro de 2025

Sumário

1. Introdução

Este relatório tem como objetivo explorar a aplicação dos códigos de Huffman na compressão de dados, abordando tanto os aspectos teóricos quanto práticos. Serão apresentados cálculos de entropia, construção de códigos de Huffman e a implementação de um programa para compressão e descompressão de arquivos de texto. O estudo é dividido em duas questões principais: a primeira trata da teoria por trás dos códigos de Huffman, enquanto a segunda aborda a implementação prática de um compressor de arquivos.

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:

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

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

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:

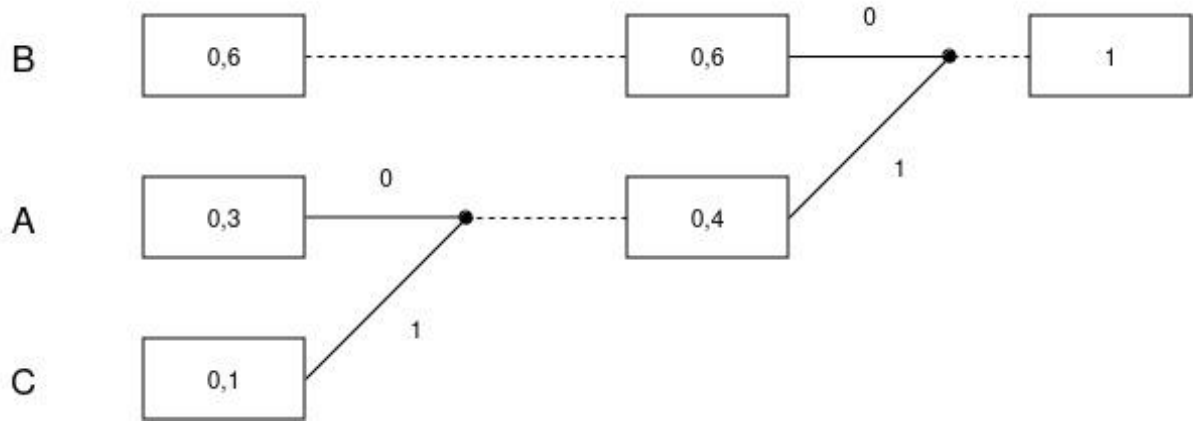

Figura 1: Diagrama de Huffman para a fonte.

Fonte: Elaborada pelo autor

A partir desse diagrama é possivel concluir :

$$\mathcal{X} = \{a, b, c\} \rightarrow [10, 0, 11] \tag{4}$$

Agora para calcular o comprimento temos :

$$l = 1 * 0,6 + 2 * 0,3 + 2 * 0,1 l = 1.4 \text{ bits/letras} \tag{5}$$

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 :

$$X^2 = \{\text{aa}, \text{ab}, \text{ac}, \text{bb}, \text{ba}, \text{bc}, \text{cc}, \text{ca}, \text{cb}\}$$

Com probabilidade:

$$P_x^2 = \{0.09, 0.18, 0.03, 0.36, 0.18, 0.06, 0.01, 0.03, 0.06\}$$

A entropia da extensão de seunda orderm é calculada da seguinte forma:

$$\begin{aligned} H(X^2) &= 0.09 \cdot \log_2(0.09) + 0.18 \cdot \log_2(0.18) + 0.03 \cdot \log_2(0.03) \\ &+0.36 \cdot \log_2(0.36) + 0.18 \cdot \log_2(0.18) + 0.06 \cdot \log_2(0.06) \\ &+0.01 \cdot \log_2(0.01) + 0.03 \cdot \log_2(0.03) + 0.06 \cdot \log_2(0.06) \end{aligned} \tag{6}$$

$$H(X^2) = 2.59 \text{ bits/superpalavras}$$

Ou desta forma:

$$H(X^2) = 2 * H(X) = 2 * 1,295 = 2,59 \text{ bits/superpalavra}$$

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

O diagrama abaixo ilustra o processo de construção do código de Huffman para a extensão de segunda ordem:

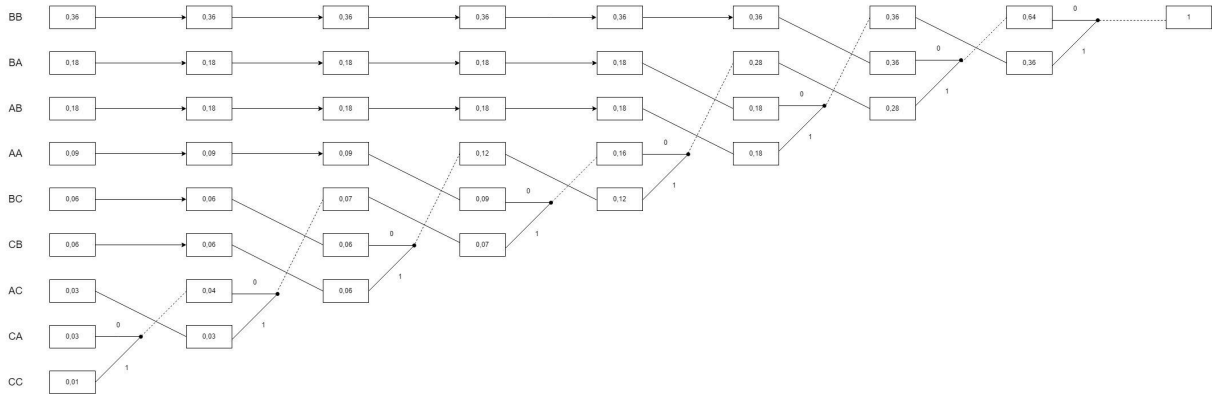

Figura 2: Fonte: Elaborada pelo autor

$$
\begin{gathered}
X^2 = \{\mathrm{aa}, \mathrm{ab}, \mathrm{ac}, \mathrm{bb}, \mathrm{ba}, \mathrm{bc}, \mathrm{cc}, \mathrm{ca}, \mathrm{cb}\} \\
\text{Temos na sequência as palavras?} \\
\{0100, 11, 010100, 10, 00, 0111, 01011, 0110, 010101\}
\end{gathered} \tag{7}
$$

O código a seguir valida o código de Huffman e o comprimento médio:

$$
\begin{gathered}
\{\mathrm{aa}, \mathrm{ab}, \mathrm{ac}, \mathrm{bb}, \mathrm{ba}, \mathrm{bc}, \mathrm{cc}, \mathrm{ca}, \mathrm{cb}\} \\
\{0.09, 0.18, 0.03, 0.36, 0.18, 0.06, 0.01, 0.03, 0.06\} \\
\{0100, 11, 010100, 00, 10, 0111, 010101, 01011, 0110\} \\
\{4, 2, 6, 2, 2, 4, 5, 4, 6\}
\end{gathered} \tag{8}
$$

$$
l = 0.09 * 4 + 0.18 * 2 + 0.03 * 6 + 0.36 * 2 + 0.18 * 2 + 0.06 * 4 + 0.01 * 6 + 0.03 * 5 + 0.06 * 4
$$

$$
l = 2.67 \text{ bits/superpalavra}
$$

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

A entropia da distribuição de frequências dos caracteres do livro foi calculada:

$$H(X) = 4.62 \text{ bits} \tag{9}$$

O comprimento médio do código de Huffman obtido foi:

$$l = 4.66 \text{ bits/caractere} \tag{10}$$

O código a seguir valida esses cálculos

```
# Contar caracteres e calcular PMF
contador, pmf, texto_original = contar_caracteres('alice.txt')

# Criar código de Huffman
huffman, codigos = criar_codigo_huffman(pmf)

# Calcular entropia e comprimento médio
dms = komm.DiscreteMemorylessSource(list(pmf.values()))
print("Entropia:", dms.entropy())
print("Comprimento médio:", huffman.rate(list(pmf.values())))
```

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

O arquivo original possui 154,573 bytes. Após a compressão, o tamanho do arquivo foi reduzido para 86,278 bytes, resultando em uma taxa de compressão de 44.18%.

A tabela abaixo compara o tamanho do arquivo original com o tamanho do arquivo comprimido:

Formato	Tamanho (bytes)	Taxa de Compressão
Original	154,573	0.00%
Huffman	86,278	44.18%

3. Conclusão

Este relatório demonstrou a eficácia dos códigos de Huffman na compressão de dados. Na primeira parte, foram realizados cálculos teóricos de entropia e construção de códigos de Huffman para uma fonte simples e sua extensão de segunda ordem. Na segunda parte, foi implementado um programa para compressão e descompressão de arquivos de texto, aplicado ao livro Alice's Adventures in Wonderland. Os resultados mostraram uma taxa de compressão de 44.18%, evidenciando a utilidade dos códigos de Huffman em aplicações práticas.