Universidade Federal do ABC
Centro de Matemática, Computação e Cognição

Compressão de Dados e Entropia no Contexto Linguístico

Lucas Silva Amorim

Santo André - SP, Maio de 2022

Lucas Silva Amorim

Compressão de Dados e Entropia no Contexto Linguístico

Projeto de Graduação apresentado ao Centro de Matemática, Computação e Cognição, como parte dos requisitos necessários para a obtenção do Título de Bacharelado em Ciências da Computação.

Universidade Federal do ABC – UFABC

Centro de Matemática, Computação e Cognição

Bacharelado em Ciências da Computação.

Orientador: Prof.ª Dr.ª Cristiane M. Sato

Santo André - SP

Maio de 2022

Resumo

Segundo a ABNT, o resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. Umas 10 linhas (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto.

Palavras-chaves: latex. abntex. editoração de texto.

Abstract

This is the english abstract.

Keywords: latex. abntex. text editoration.

Sumário

Introdução

Este documento segue as normas estabelecidas pela **??**, 3.1-3.2).

Figuras

As normas da **??**, 3.1-3.2) especificam que o caption da figura deve vir abaixo da mesma.

A Figura 1 ilustra...

Figura 1 – Breve explicação sobre a figura. Deve vir abaixo da mesma.

Tabelas

A Tabela 1 apresenta os resultados...

Tabela 1 – Breve explicação sobre a tabela. Deve vir acima da mesma.

XX	FF	PP	YY	Yr	xY	Yx	ZZ
615	18	2558	0,9930	0,9930	0,9930	0,9930	0,9930
615	18	2558	0,9930	0,9930	0,9930	0,9930	0,9930
615	18	2558	0,9930	0,9930	0,9930	0,9930	0,9930
615	18	2558	0,9930	0,9930	0,9930	0,9930	0,9930
615	18	2558	0,9930	0,9930	0,9930	0,9930	0,9930

Motivação

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Aenean nonummy turpis id odio. Integer euismod imperdiet turpis. Ut nec leo nec diam imperdiet lacinia. Etiam eget lacus eget mi ultricies posuere. In placerat tristique tortor. Sed porta vestibulum metus. Nulla iaculis sollicitudin pede. Fusce luctus tellus in dolor. Curabitur auctor velit a sem. Morbi sapien. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Donec adipiscing urna vehicula nunc. Sed ornare leo in leo. In rhoncus leo ut dui. Aenean dolor quam, volutpat nec, fringilla id, consectetuer vel, pede.

Objetivos

Nulla malesuada risus ut urna. Aenean pretium velit sit amet metus. Duis iaculis. In hac habitasse platea dictumst. Nullam molestie turpis eget nisl. Duis a massa id pede dapibus ultricies. Sed eu leo. In at mauris sit amet tortor bibendum varius. Phasellus justo risus, posuere in, sagittis ac, varius vel, tortor. Quisque id enim. Phasellus consequat, libero pretium nonummy fringilla, tortor lacus vestibulum nunc, ut rhoncus ligula neque id justo. Nullam accumsan euismod nunc. Proin vitae ipsum ac metus dictum tempus. Nam ut wisi. Quisque tortor felis, interdum ac, sodales a, semper a, sem. Curabitur in velit sit amet dui tristique sodales. Vivamus mauris pede, lacinia eget, pellentesque quis, scelerisque eu, est. Aliquam risus. Quisque bibendum pede eu dolor.

Parte I

Fundamentação Teórica

1 Conceitos e definições fundamentais

Este capítulo apresenta algumas definições e conceitos fundametais para o entendimento das técnicas de compressão que serão discutidas em capítulos posteriores.

1.1 Código

Um **código** C mapeia uma **mensagem** $m \in M$ para uma cadeia de **palavras código** em W^+,onde M é chamado **alfabeto de origem** e W^+ **alfabeto de palavras código**. Vamos utilizar a notação A^+ para se referir ao conjunto que contém todas as cadeias de A, *i.e*, $A^+ = \bigcup_{i \geq 1} A^i : A^i = (a_1, ..., a_i), a \in A$. Deste modo, podemos representar um código como uma função $\boldsymbol{C : M \rightarrow W^+}$. O **comprimento** da palavra código w, definido pela função $l(w)$, representa o número de bits de w

Os elementos dos alfabetos de origem e de palavras código podem ter um comprimento fixo ou variável. Códigos nos quais os alfabetos possuem um comprimento fixo são chamados de **códigos de comprimento fixo**, enquanto os que possuem alfabetos de comprimento variáveis são chamados **códigos de comprimento variável**. Provavelmente o exemplo mais conhecido de código de comprimento fixo seja código ASCII, que mapeia 64 símbolos alfa-numéricos (ou 256 em sua versão estendida) para palavras código de 8 bits. Todavia, a compressão de dados utiliza apenas códigos de comprimento variável, mas especificamente códigos que variam o comprimento de acordo com a probabilidade associada à mensagem (o tema será melhor detalhado em seções posteriores).

1.1.1 Códigos unicamente decodificáveis e livres de prefixo

Um código é **distinto** se pode ser representado como uma função **bijetora**, i.e, $\forall\ m_1, m_2 \in$ M, $C(m_1) \neq C(m_2)$. Um código é dito **unicamente decodificável** quando $C(m) = w \leftrightarrow C^{-1}(w) = m$, com $m \in M$ e $w \in W^+$.

Vamos definir C^+ como a **codificação** correspondente ao código C, tal que $C^+(m^n) = C(m_1)C(m_2)...C(m_n) : m^n = m_1m_2...m_n$, *i.e*, $C^+ : M^+ \rightarrow W^+$. A função de **decodificação** $D^+ : W^+ \rightarrow M^+$ se refere a operação inversa da codificação, de modo que dado um código **unicamente decodificável** C, $D^+(C^+(m^n)) = m^n$.

Um **código livre de prefixo** é um código C' em que $\nexists w_1, w_2 \in W^+ \mid w_1$ é **prefixo** de w_2, por exemplo, o conjunto de palavras código $W^+ := \{1, 01, 000, 001\}$ não possui nenhuma cadeia que é prefixo de outra dentro do conjunto. Códigos livres de prefixo podem ser *decodificados instantaneamente*, ou seja, podemos decodificar uma palavra código sem precisar verificar o início da seguinte.

Um código livre de prefixo pode ser modelado por uma arvore binária. Imagine que cada mensagem $m \in M$ é uma folha. A palavra código *C'(m)* é o caminho p da raiz até a folha correspondente a m, de maneira em que, para cada nó percorrido concatene um bit à p (0 quando o nó está à esquerda e 1 quando está à direita). Chamamos tal árvore de **árvore do código livre de prefixo**.

Teorema 1. *Todo código* ***livre de prefixo*** *é* ***unicamente decodificável****.*

Demonstração. Seja C um código livre de prefixo e $S_n = s_1...s_n$ uma mensagem codificada por C. Vamos provar por indução que o teorema é verdadeiro para todo n $\in \mathbb{Z}+$

Casos base: Quando n = 1, a mensagem S só possui uma palavra código, logo é unicamente decodificável. Se n = 2, então S possui uma palavra código s_1 que não pode ser prefixo de s_2 (pela própria definição de códigos livres de prefixo), o que claramente significa que S é unicamente decodificável.

Passo indutivo: Seja k $\in \mathbb{Z}+$, e suponha por hipótese de indução que o teorema vale para n $\leq$ k. Como S_{k+1} é livre de prefixo, existe um prefixo de S_{k+1}, $S_j = s_1...s_j$ (com j $\leq$ k + 1) que é unicamente decodificável (dado que ela não pode ser prefixo de nenhuma outra). a mensagem $S'_{k+1} = s_{j+1}...s_{k+1}$ ainda é uma concatenação decodificável e $|S'_{k+1}| \leq |S_{k+1}|$, o que significa que por hipótese de indução S'_{k+1} é unicamente decodificável. Como $S_{k+1} = S_j\ S'_{k+1}$, segue que S_{k+1} é unicamente decodificável. □

1.2 Relações fundamentais com a Teoria da Informação

A codificação é comumente divida em duas componentes diferentes: *modelo* e *codificador*. O *modelo* identifica a distribuição de probabilidade das mensagens baseado em sua semântica e estrutura. O *codificador* toma vantagem de um possível *bias* apontado pela modelagem, e usa uma estratégia gulosa em relação a probabilidade associada às mensagens para reduzir seu tamanho. (substituindo as mensagens que ocorrem com maior frequência por símbolos menores).

Desta forma, é evidente que os algoritmos de compressão sempre devem tomar vantagem de alguma distribuição de probabilidades "desbalanceada"sobre as mensagens para efetivamente reduzir o tamanho destas, portanto, a compressão é fortemente relacionada com a probabilidade. Nesta seção, vamos construir o embasamento teórico necessário para entender a relação entre as probabilidades associadas e o comprimento das mensagens, e consequentemente criar uma noção dos parâmetros que devem ser maximizados para alcançar uma codificação eficiente.

1.2.1 Distribuição de Probabilidade e Esperança

Dado um experimento e um espaço amostral Ω, uma **variável aleatória** X associa um número real a cada um dos possíveis resultados em Ω. Em outras palavras, X é uma função que mapeia os elementos do espaço amostral para números reais. Quando a imagem de X pode assumir um número finito de valores, dizemos que X é uma **variável aleatória discreta**.

Podemos descrever melhor uma variável aleatória, atribuindo probabilidades sobre os valores que esta pode assumir. Esses valores são atribuídos pela **função de densidade de probabilidade**, denotada por p_X. Portanto, a probabilidade do evento $\{\ X = x\ \}$ é a função de distribuição de probabilidade aplicada a x, *i.e*, $p_X(x)$.

$$p_X(x) = P(\{X = x\}) \tag{1.1}$$

Note que, a variável aleatória pode assumir qualquer um dos valores no espaço amostral que possuem uma probabilidade $P > 0$, portanto

$$\sum_x p_X(x) = 1. \tag{1.2}$$

O **valor esperado** (ou **esperança**) da variável aleatória X é definido como

$$\mathbf{E}[\ X] \ = \sum_x x p_X(x). \tag{1.3}$$

1.2.2 Comprimento médio do código

Seja p a distribuição de probabilidade associada ao alfabeto de origem M. Assuma que C é um código tal que $C(m) = w$, definimos o **tamanho médio** de C como:

$$l_a(C) = \sum_{m \in M, w \in W^+} p(m) l(w) \tag{1.4}$$

Um código C livre de prefixo é **ótimo** se $l_a(C)$ é mínimo, isto é, para qualquer código livre de prefixo C' temos que

$$l_a(C) \leq l_a(C') \tag{1.5}$$

1.2.3 Entropia

A **Entropia de Shannon** aplica as noções de Entropia física (que representa a aleatoriedade de um sistema) à Teoria da Informação. Dado um sistema S e a função p sendo a distribuição de probabilidade associada a S, definimos **Entropia** como:

$$H(S) = \sum_{s \in S} p(s) \log_2 \frac{1}{p(s)} \tag{1.6}$$

Por esta definição temos que quanto menor o *bias* da função de distribuição de probabilidade relacionada ao sistema, maior a sua entropia. Em outras palavras, a entropia de um sistema esta intimamente ligada a sua "desordem".

Shannon (incluir referencia papper do shannon) aplica o mesmo conceito de entropia no contexto da teoria da informação, "substituindo"o conjunto de estados S pelo conjunto de mensagem M, isto é, M é interpretado como um conjunto de possíveis mensagens, tendo como $p(m)$ a probabilidade de $m \in M$. Baseado na mesma premissa, Shannon mede a informação contida em uma mensagem da seguinte forma:

$$i(s) = \log_2 \frac{1}{p(s)}. \tag{1.7}$$

1.2.4 Comprimento de Código e Entropia

Nas secções anteriores, o comprimento médio de um código foi definido em função da distribuição de probabilidade associada ao seu alfabeto de origem. Da mesma forma, as noções de **Entropia** relacionada a um conjunto de mensagens, tem ligação direta com as probabilidades associadas a estas. A seguir, será mostrado como podemos relacionar o comprimento médio de um código a sua entropia através da **Desigualdade de Kraft-McMillan** , e por consequência estabelecer uma relação direta entre **a Entropia de um conjunto de mensagens e a otimalidade do código associada a estas mensagens**.

Lema 2 (Desigualdade de Kraft-McMillan)**.** ***McMillan*** *. Para todo código binário unicamente decodificável* $\mathbf{C}$ *:* $\mathbf{M} \rightarrow \mathbf{W}^+$.

$$\sum_{w \in W^+} 2^{-l(w)} \leq 1.$$

Kraft*. Para qualquer conjunto* L *de comprimento códigos que satisfaça:*

$$\sum_{l \in L} 2^{-l} \leq 1.$$

$\exists C$ *binário livre de prefixo :* $|C(w_i)| = l_i$ *,* $\forall w \in W^+$*.*

Demonstração.

Desigualdade de Kraft Sem perda de generalidade, suponha que os elementos de L estão ordenados de maneira em que:

$$l_1 \leq l_2 \leq ... \leq l_n$$

Agora vamos construir um código livre de prefixo em uma ordem crescente de tamanho, de maneira em que $l(w_i) = l_i$. Sabemos que, um código é livre de prefixo se e somente se ,existe uma palavra código w_j que não contém nenhuma das palavras código anteriores

(w_1, w_2, w_{j-1}) como prefixo. Sem as restrições de prefixo, uma palavra código de tamanho l_j poderia ser construída de 2^{l_j} maneiras diferentes. Com a restrição apresentada anteriormente, considerando uma palavra w_k anterior a w_j (i.e, $k < j$), existem $2^{l_j - l_k}$ possíveis palavras código seriam prefixo de w_k, e que portanto não podem pertencer ao código. Vale notar que o conjunto de palavras código "proibidas"(devido a restrição apresentada anteriormente) são excludentes entre si, pois se duas palavras código menores que j fossem prefixo da mesma palavra código, elas seriam prefixos entre si.

A partir dessas premissas, sabemos que o total de palavras código de tamanho j que possuiriam prefixo é:

$$\sum_{i=1}^{j-1} 2^{l_j - l_i}$$

A construção do código livre de prefixo é possível se e somente se, existir uma palavra código de tamanho $j > 1$ que não está contida no conjunto das que possuem prefixo:

$$2^{l_j} > \sum_{i=1}^{j-1} 2^{l_j - l_i}$$

Levando em consideração a primeira palavra código j_1:

$$2^{l_j} \leq \sum_{i=1}^{j-1} 2^{l_j - l_i} + 1 = 2^{l_j} \leq \sum_{i=1}^{j} 2^{l_j - l_i}$$
$$= 1 \geq \sum_{i=1}^{j} 2^{l_j - l_i} = \sum_{i=1}^{j} 2^{l_j - l_i} \leq 1$$

Se aplicarmos essa lógica para a maior palavra código, isto é j = n, chegamos em:

$$\sum_{l \in L} 2^{-l} \leq 1.$$

Note que os argumentos utilizados para a construção da prova possuem dupla-equivalência, portando concluem a prova nos dois sentidos.

Desigualdade de McMillan

□

Lema 3 (Entropia como limite inferior para o comprimento médio)**.** *Dado um conjunto de mensagens* M *associado a uma distribuição de probabilidades* p *e um código unicamente decodificável* C*.*

$$H(M) \leq l_a(C)$$

Demonstração. Queremos provar que $H(M) - l_a(C) \leq 0$, dado que $H(M) \leq l_a(C) \Leftrightarrow H(M) - l_a(C) \leq 0$.

Substituindo a equação 1.6 em *H(M)* e 1.4 em $l_a(C)$, temos:

$$\begin{aligned} H(M) - l_a(C) &= \sum_{m \in M} p(s) \log_2 \frac{1}{p(m)} - \sum_{m \in M, w \in W^+} p(m) l(w) \\ &= \sum_{m \in M, w \in W^+} p(m) \left(\log_2 \frac{1}{p(m)} - l(w) \right) \\ &= \sum_{m \in M, w \in W^+} p(m) \left(\log_2 \frac{1}{p(m)} - \log_2 2^{l(w)} \right) \\ &= \sum_{m \in M, w \in W^+} p(m) \log_2 \frac{2^{-l(w)}}{p(m)} \end{aligned}$$

A **Desigualdade de Jansen** afirma que se uma função *f(x)* é côncava, então $\sum_i p_i \ f(x_i) \leq f(\sum_i p_i \ x_i)$. Como a função $\log_2$ é côncava, podemos aplicar a Desigualdade de Jansen ao resultado obtido anteriormente.

$$\sum_{m \in M, w \in W^+} p(m) \log_2 \frac{2^{-l(w)}}{p(m)} \leq \log_2 (\sum_{m \in M, w \in W^+} 2^{-l(w)})$$

Agora aplicamos a desigualdade de Kraft-McMillan, e concluímos que:

$$H(M) - l_a(C) \leq \log_2 (\sum_{m \in M, w \in W^+} 2^{-l(w)}) \Rightarrow H(M) - l_a(C) \leq 0.$$

□

Lema 4 (Entropia como limite superior para o comprimento médio de um código livre de prefixo ótimo)**.** *Dado um conjunto de mensagens* M *associado a uma distribuição de probabilidades* p *e um código livre de prefixo ótimo* C*.*

$$l_a(C) \leq H(M) + 1$$

Demonstração. Sem perda de generalidade, para cada mensagem $m \in M$ faça $l(m) = \left\lceil \log_2 \frac{1}{p(m)} \right\rceil$. Temos que:

$$\begin{aligned} \sum_{m \in M} 2^{-l(m)} &= \sum_{m \in M} 2^{-\left\lceil \log_2 \frac{1}{p(m)} \right\rceil} \\ &\leq \sum_{m \in M} 2^{-\log_2 \frac{1}{p(m)}} \\ &= \sum_{m \in M} p(s) \\ &= 1 \end{aligned}$$

De acordo com a desigualdade de Kraft-McMillan existe um código livre de prefixo *C'* como palavras código de tamanho *l(m)*, portanto:

$$\begin{aligned} l_a(C') &= \sum_{m\in M', w\in W'^+} p(m)l(w) \\ &= \sum_{m\in M', w\in W'^+} p(m)\left\lceil \log_2 \frac{1}{p(m)} \right\rceil \\ &\leq \sum_{m\in M', w\in W'^+} p(m)(1+\log_2 \frac{1}{p(m)}) \\ &= 1 + \sum_{m\in M', w\in W'^+} p(m)\log_2 \frac{1}{p(m)} \\ &= 1 + H(M) \end{aligned}$$

Pela definição de código livre de prefixo ótimo, $l_a(C) \leq l_a(C')$, isto é:

$$l_a(C) \leq H(M) + 1$$

□

1.3 Códigos de Huffman

O **algoritmo de Huffman** (desenvolvido por David Huffman em 1952) é um dos componentes mais utilizados em algoritmos de compressão sem perda, servindo como base para algoritmos como o GZIP (utilizado amplamente na web). Os códigos gerados a partir do algoritmos de Huffman são chamados **Códigos de Huffman**.

O código de Huffman é descrito em termos de como ele gera uma árvore de código livre de prefixo. Considere o conjunto de mensagens M, com p_i sendo a probabilidade associada a m_i

- Crie uma floresta de árvores uma para cada mensagem do código. Faça com que o peso de cada vértice seja $w_i = p_i$ (onde o p_i representa a probabilidade associada à m_i).
- Faça até que a floresta possua uma única árvore
  - Selecione duas árvores com os pesos w_1 e w_2, de maneira que estes sejam os menores pesos.
  - Crie uma nova árvore a partir das duas selecionadas anteriormente, onde a raiz é um novo nó com o peso $w_1 + w_2$ e faça das duas árvore selecionadas os seus filhos. (por convenção o nó com menor peso fica à esquerda)

Algorithm 1 Algoritmo de Huffman

$Forest \leftarrow []$

for all $m_i \in M$ **do** ▷ Inicializando floresta
  $T \leftarrow newTree()$
  $node \leftarrow newNode()$
  $node.weight \leftarrow p_i$ ▷ $w_i = p_i$
  $T.root \leftarrow node$
  $Forest.append(T)$
end for

while $Forest.size > 1$ **do**
  $T1 \leftarrow ExtractMin(Forest)$ ▷ Extrai a árvore cujo o peso da raiz é mínimo
  $T2 \leftarrow ExtractMin(Forest)$
  $HTree \leftarrow newTree()$
  $HTree.root \leftarrow newNode()$

  $HTree.root.left \leftarrow T1.root$
  $HTree.root.right \leftarrow T2.root$
  $HTree.root.weight \leftarrow T1.root.weight + T2.root.weight$
  Forrest.append(HTree)
end while

1.3.1 Análise Assintótica

Seja n o tamanho do conjunto de mensagens M. Para que o algoritmo percorra toda a floresta, formada por uma árvore para cada $m \in M$, serão necessárias n interações. Considerando que as funções *ExtractMin()* e *.append()* foram construídas a partir de uma fila de prioridades de **heap**, o algoritmo será executado em $O(n \log_2 n)$.

1.3.2 Correctude

O teorema a seguir (escrito por Huffman) mostra a principal propriedade do Algoritmo de Huffman, os códigos de Huffman são códigos ótimos e livres de prefixo.

Teorema 5. *O algoritmo de Huffman gera um código ótimo livre de prefixo.*

Demonstração. □

Referências

HIRSCHBERG, D.S; LELEWER D.A; *Data compression,* Computing Surveys 19.3, 1987. Nenhuma citação no texto.

BLELLOCH G.E; *Introduction to Data Compression,* Carnegie Mellon, 2013 Nenhuma citação no texto.

BERTSEKAS D.P; TSITSIKLIS J.N; *Introduction to Probability* M.I.T, Lecture Notes Course 6.041-6.431, 2000 Nenhuma citação no texto.