Tópicos em Recuperação de Informação

Nivio Ziviani

Conjunto de transparências elaborado por Nivio Ziviani, Patrícia Correia e Fabiano C. Botelho.

Compressão de Índices

- Arquivos invertidos são amplamente usados para indexar grandes arquivos de texto.
- O tamanho de um arquivo invertido pode ser reduzido usando compressão nas listas invertidas.
- Números dos documentos em ordem ascendente: seqüência de *d-gaps* (a diferença entre d_{k+1} e d_k) entre os números de documentos.
- Processamento é seqüencial desde início da lista: números originais dos documentos podem ser obtidos através do cálculo da soma dos *d-gaps*.

  Ex.: Suponha que um termo ocorra em 8 documentos:

  $$<8;\ 3,\ 5,\ 20,\ 21,\ 23,\ 76,\ 77,\ 78>$$

  Calculando-se os *gaps* entre os documentos:

  $$<8;\ 3,\ 2,\ 15,\ 1,\ 2,\ 53,\ 1,\ 1>$$

Compressão de Índices

- Ambos os métodos de representação das listas necessitam de $\lceil \log N \rceil$ bits por apontador, onde N é o número de documentos da coleção.
- Entretanto, *d-gaps* podem ser codificados em muito menos do que $\lceil \log N \rceil$ bits.
- Os *gaps* são pequenos para palavras freqüentes e grandes para palavras infreqüentes, a compressão pode ser obtida codificando-se pequenos valores com códigos menores
  - Por exemplo, o código unário, no qual um inteiro x é codificado como $x - 1$ bits 1 e seguido de um bit 0.
    Ex.: O código unário para o inteiro 3 é 110.

Métodos para Compressão de Índices

Métodos locais

Modelo de compressão para cada lista associada a um termo é ajustado de acordo com algum parâmetro local. Por exemplo, a freqüência do termo.

Métodos globais

Listas são comprimidas usando um modelo comum de compressão

- Não-parametrizados:
  - Bitwise: Unário, Elias-δ e Elias-γ.
  - Bytewise: Elias-γ com um sufixo múltiplo de 7.
- Parametrizados: Golomb
  - envolve algum parâmetro que pode ser ajustado à distribuição dos *d-gaps*
  - exige duas passadas no texto.

Métodos para Compressão de Índices

Método	**Referência**
Métodos Globais	
Não parametrizados	
Unário	
Binário	
γ	Elias (1975); Bentley and Yao (1976)
δ	Elias (1975); Bentley and Yao (1976)
Parametrizados	
Bernoulli	Golomb(1966); Gallager and Van Voorhis (1975)
Observed frequency	
Métodos Locais	
Bernoulli	Witten, Bell, and Nevill (1992); Bookstein, Klein, and Raita (1992)
Skewed Bernoulli	Teuhola (1978); Moffat and Zobel (1992)
Hyperbolic	Schuegraf (1976)
Observed frequency	
Batched frequency	Moffat and Zobel (1992)
Interpolative	Moffat and Zobel (1996)

Métodos para Compressão de Índices

1. Código unário é equivalente a atribuir uma probabilidade de $Pr[x] = 2^{-x}$ para intervalos de comprimento x, o que representa um valor muito pequeno. Entretanto, precisa de até $N * n$ bits, n igual ao tamanho do vocabulário (WMB99, p.117).
2. Código binário puro assume distribuição uniforme (N documentos precisam de $\log N$ bits para cada apontador) no intervalo $1 \ldots N$ (não reflete a realidade).
3. Muitos códigos usam algo entre uniforme (codificação binária) e o decaimento exponencial binário da codificação unária. Um deles é o código γ.

Relacionamento entre Probabilidades e Códigos

- Claude Shannon (1948) em seu teorema:
  - Em um esquema de codificação ótimo, um símbolo que é esperado para ocorrer com probabilidade p deve ser atribuído um código de tamanho $\log_2 \frac{1}{p}$ bits.
- O número de bits no qual o símbolo é melhor codificado representa o conteúdo informacional do símbolo.
- A quantidade média de informação por símbolo sobre o alfabeto todo é chamado de *entropia* da distribuição de probabilidade, que é dada por:

$$E = \sum p_i \log_2 \frac{1}{p_i}$$

Medida em bits por símbolo e representa um limite que nunca pode ser batido em um dado modelo.

Codificação Unária

Para a codificação de um inteiro x usando o método unário repete-se $(x-1)$ bits com valor 1, em seguida acrescenta-se um bit com o valor 0.

1	0
2	10
3	110
4	1110
5	11110
6	111110
7	1111110
8	11111110
9	111111110
10	1111111110

Codificação de Elias-γ

O número é dividido em duas partes:

- Primeira parte:
  - código unário de $1 + \lfloor \log x \rfloor$ bits.
- Segunda parte:
  - tamanho do código em bits: $\lfloor \log x \rfloor$ bits.
  - codificando o número $x - 2^{\lfloor \log x \rfloor}$ em binário.

Ex.: $x = 5$

$$
\begin{array}{ll}
1 + \lfloor \log 5 \rfloor = 3 & \text{(110 em unário)} \\
5 - 2^{\lfloor \log 5 \rfloor} = 5 - 2^2 = 1 & \text{(01 em binário de tamanho 2)}
\end{array}
$$

portanto o código de 5 é 11001.

Codificação de Elias-γ

Decodificação::

1. Extrai código unário C_u
2. Trata próximos $C_u - 1$ bits como código binário C_b

Logo $x = 2^{C_u - 1} + C_b$

Para código 1110 001, $C_u = 4$ e $C_b = 1$

Assim $x = 2^3 + 1 = 9$

Método preferido para codificar gaps:

- Representa gap x em aproximadamente $lx \simeq 1 + 2\log x$ bits.
- A probabilidade associada é:

$$Pr[x] = 2^{-lx} \simeq 2^{-(1+2\log x)} = \frac{1}{2x^2}$$

Codificação de Elias-δ

O número é dividido em duas partes:

- Primeira parte:
  - código Elias-γ de $1 + \lfloor \log x \rfloor$ bits.
- Segunda parte:
  - tamanho do código em bits: $\lfloor \log x \rfloor$ bits.
  - codificando o número $x - 2^{\lfloor \log x \rfloor}$ em binário.

Ex.: $x = 5$

$$1 + \lfloor \log 5 \rfloor = 3 \qquad \text{(101 em Elias-}\gamma\text{)}$$
$$5 - 2^{\lfloor \log 5 \rfloor} = 5 - 2^2 = 1 \qquad \text{(01 em binário de tamanho 2)}$$

portanto o código de 5 é 10101.

Codificação de Elias-δ

$$
\begin{aligned}
lx &= 1 + 2\lfloor \log(1 + \lfloor \log x \rfloor) \rfloor + \lfloor \log x \rfloor \\
&= 1 + 2\lfloor \log \log 2x \rfloor + \lfloor \log x \rfloor
\end{aligned}
$$

$$
\begin{aligned}
Pr[x] &= 2^{-lx} = 2^{-(1+2\lfloor \log \log 2x \rfloor + \lfloor \log x \rfloor)} \\
&= \frac{1}{2x(\log x)^2} \qquad (1)
\end{aligned}
$$

Observação: código δ é menor para valores maiores de x.

Exemplos

Número	Unário	Gama	Delta
1	0	0:	0::
2	10	10:0	10:0:0
4	1110	110:00	10:1:00
7	1111110	110:11	10:1:11
1044	$1^{1043}0$	11111111110:0000010100	1110:010:0000010100
$2, 9, 3$	$10, 111111110, 110$	10:0, 1110:001, 10:1	10:0:0, 110:00:001, 10:0:1

Note que os dois pontos e vírgulas são ilustrativos. Eles não aparecem no arquivo compactado.

Comprimento dos códigos

Qual códificação é melhor? Isto depende da distribuição de probabilidade $Pr[x]$ que governa os valores de x que estão sendo codificados.

valor de x	Unário	Gama	Delta
1	1	1	1
2	2	3	4
4	4	5	5
8	8	7	8
16	16	9	9
$1,000$	$1,000$	19	15
$1,000,000$	$1,000,000$	39	28
x	x	$1 + 2\lfloor \log x \rfloor$	$1 + 2\lfloor \log \log 2x \rfloor + \lfloor \log x \rfloor$

Codificação Baseada em Bytes

Problema:

- Computador opera com granularidade de bytes ou palavras.
- Operações com máscaras de bits tornam a codificação e decodificação bem mais lentas.

Solução:

- Representar cada código como uma seqüência de bytes ou vários códigos em uma palavra de forma que não haja nenhum código quebrado em mais de uma palavra.

Codificação Baseada em Bytes

- Se $x \leq 128$, então um único byte é usado para armazenar $x - 1$ em binário sendo o bit mais significativo igual a 0.
- Caso contrário:
  - Empacote os 7 bits menos significativos de $x - 1$ em um byte com o bit mais significativo igual a 1.
  - Codifique o número $x/128$ recursivamente da mesma forma.

Codificação Bytewise

/* Para codificar $x \geq 1$: */
$x = x - 1$
while ($x \geq 128$)
  (a) writeByte($128 + x \bmod 128$)
  (b) $x = x/128 - 1$
writeByte(x)

Exemplos:

x	$x - 1$ (Binário)	Código Bytewise
2	1	00000001
4	11	00000011
8	111	00000111
127	1111111	01111111
1044	10000010011	10010011 : 00000111

Decodificacão Bytewise

/* Para decodificar $x \geq 1$: */
$b =$ readByte()
$x = 0$
$p = 1$
while ($b \geq 128$)
    (a) $x = x + (b - 127) \times p$
    (b) $p = p \times 128$
    (c) $b =$ readByte()
$x = x + (b + 1) \times p$.

Exemplo: decodificação de $x = 1044$

Comandos	x	b	p
1 – 3	0	$10010011 = 147$	1
4(a) – 4(c)	20	$00000111 = 7$	128
5	1044	$-$	$-$

Veja que $1044 = 128 \times (7 + 1) + 147 - 127$

Codificação de Golomb

- Parametrizado, usa densidade dos apontadores no arquivo invertido.
- Exige duas passadas no arquivo invertido.
- Se o número de apontadores f é conhecido então $f/(N \times n)$ dá a probabilidade de que qualquer documento selecionado randomicamente contenha qualquer termo selecionado randomicamente.
- Logo, ocorrências de apontadores podem ser modeladas como um processo de Bernoulli:
  - Chance de um gap de tamanho x é a probabilidade de ter $x - 1$ não ocorrências de x, cada com probabilidade $1 - p$, seguida por uma ocorrência de probabilidade p que é $Pr[x] = (1 - p)^{x-1} p$ (distribuição geométrica).
  - Estas probabilidades podem ser representadas por *Huffman-style code*!

Codificação de Golomb

Solomon Golomb, USC (1966):

- Definir o valor do parâmetro b.
- $(q+1)$ em unário, onde $q = \lfloor (x-1)/b \rfloor$.
- $r = (x-1) - q \times b$ codificado em binário com $\lfloor \log b \rfloor$ ou $\lceil \log b \rceil$ bits.
- Valores de r com $\lfloor \log b \rfloor$ bits começam com 0.
- Valores de r com $\lceil \log b \rceil$ bits começam com 1.

Ex.: $b = 3$ e $x = 7$

$$\begin{aligned} q &= \lfloor (7-1)/3 \rfloor &&= 2 \\ r &= (7-1) - 2 \times 3 &&= 0 \end{aligned}$$

Então $(q+1)$ em unário é 110

representação de 7 é 1100.

Codificação de Golomb

- Se b é escolhido para satisfazer (Gallager e Van Voorhis (1975))

$$(1-p)^b + (1-p)^{b+1} \leq 1 < (1-p)^{b-1} + (1-p)^b$$

- O método de codificação gera um código ótimo para a distribuição geométrica

$$b = \left\lceil \frac{\log(2-p)}{-\log(1-p)} \right\rceil$$

para $p = f/(N \times n)$

$$b \simeq \frac{\ln 2}{p} \simeq 0.69 \times \frac{N \times n}{f}$$

Distribuição para cada código

- Unário possui redundância mínima se $Pr[x] = 2^{-x}$ (Código de Huffman).
- Gama e Delta possuem redundância mínima se $Pr[x] \simeq 1/(2x^2)$ e $Pr[x] \simeq 1/(2x \log^2 x)$ respectivamente.
- Golomb é o melhor de todos. Golomb possui redundância mínima se

$$Pr[x] \simeq (1-p)^{x-1}p$$

  proporcionando

$$b = \left\lceil \frac{\log(2-p)}{-\log(1-p)} \right\rceil \simeq 0.69 \times \frac{1}{p},$$

  onde p é parâmetro da distribuição geométrica (a probabilidade de sucesso em uma sequêcia de Bernoulli).

Exemplos de Códigos para Inteiros

Gap x	Unary	Elias-γ	Elias-δ	Golomb ($b = 3$)
1	0	0	0	00
2	10	100	1000	010
3	110	101	1001	011
4	1110	11000	10100	100
5	11110	11001	10101	1010
6	111110	11010	10110	1011
7	1111110	11011	10111	1100
8	11111110	1110000	11000000	11010
9	111111110	1110001	11000001	11011
10	1111111110	1110010	11000010	11100

- O código Elias-δ para um inteiro arbitrário x requer $1 + 2\lfloor \log \log 2x \rfloor + \lfloor \log x \rfloor$ bits.
- Para pequenos valores de x os códigos de Elias-γ são menores que os de Elias-δ. Entretanto, no limite, como x torna-se grande, a situação é revertida.

Bitwise versus Bytewise

- Codificação bytewise comprime menos, mas a decodificação é mais rápida.
- Codificação bytewise permite se fazer saltos facilmente para encontrar o k-ésimo código subsequente.
- Não é preciso decodicar todos os símbolos para se decodificar o k-ésimo código subsequente.
- Efetividade da compressão para *stop words*:
  - Bitwise: 12 bits por entrada $< d, f_{d,t} >$
  - Bytewise: 16–20 bits por entrada $< d, f_{d,t} >$
- Efetividade da compressão para todo índice:
  - Bitwise: 8 bits por entrada $< d, f_{d,t} >$
  - Bytewise: 12–16 bits por entrada $< d, f_{d,t} >$