Estruturas de Dados Probabilísticas para Representação de Conjuntos

Juan P. A. Lopes[1], **Paulo E. D. Pinto**[1]*, **Fabiano de S. Oliveira**[1]*

[1]IME/DICC, Universidade do Estado do Rio de Janeiro (UERJ)
Av. São Francisco Xavier, 524, Maracanã – 20550-013 – Rio de Janeiro – RJ

me@juanlopes.net, pauloedp@ime.uerj.br, fabiano.oliveira@ime.uerj.br

Abstract. *This paper describes the most important aspects of the probabilistic data structures Bloom Filter, Count-Min Sketch, MinHash, and HyperLogLog, that allow, through the use of hash functions, to estimate some properties of sets and multisets without needing to keep them in memory. All data structures described in this paper are constructed using easily parallelizable online algorithms, making them especially important in the big data scenario where an approximate answer is acceptable.*

Resumo. *Este artigo descreve os aspectos mais importantes das estruturas de dados probabilísticas Filtro de Bloom, Count-Min Sketch, MinHash e HyperLogLog, que permitem, através do uso de funções de hash, estimar algumas propriedades sobre conjuntos e multiconjuntos sem a necessidade de mantê-los em memória. Todas as estruturas apresentadas neste artigo são construídas através de algoritmos online facilmente paralelizáveis, o que as torna especialmente importantes no cenário de grandes volumes de dados onde uma resposta aproximada é aceitável.*

1. Introdução

A emergência de sistemas que lidam com grandes volumes de dados exigiu que fossem elaboradas novas técnicas para processá-los. Muitas vezes, armazená-los para posterior processamento pode ser inviável. Este cenário levou à popularização de uma nova classe de aplicações que operam sobre fluxos de dados [Babcock et al. 2002]. Dado o volume e a natureza virtualmente infinita desses fluxos, é desejável que os algoritmos e estruturas de dados que os processem sejam capazes de computar resultados de forma contínua e em apenas uma passagem, utilizando o mínimo possível de recursos. Para muitos problemas, não são conhecidos algoritmos determinísticos que satisfaçam estas características. As técnicas probabilísticas podem endereçar estas dificuldades ao custo de fornecerem uma resposta correta com certa probabilidade [Gibbons and Matias 1999].

Nas próximas seções serão abordadas quatro estruturas de dados probabilísticas que, utilizando o *hash* dos elementos de conjuntos e multiconjuntos, são capazes de representar operações sobre eles.

*Projeto parcialmente financiado por FAPERJ.

2. Filtro de Bloom

Um *filtro de Bloom* representa um conjunto e permite verificar a pertinência de elementos com possibilidade de falsos positivos [Bloom 1970]. É uma representação bastante compacta: são necessários menos de 10 bits por elemento para uma probabilidade de 1% de falsos positivos [Bonomi et al. 2006]. A estrutura representa um conjunto S de cardinalidade n utilizando um vetor B de m bits e k funções de *hash* $h_i : S \to [1..m]$, com $i \in [1..k]$, cujas funções de dispersão possuem distribuição uniforme. Inicialmente, $B[i] = 0$ para todo $i \in [1..m]$. Para inserir um elemento x, é preciso atribuir $B[h_i(x)] \leftarrow 1$ para todo $i \in [1..k]$. Para verificar a pertinência de um elemento x, a estrutura responde afirmativamente precisamente quando $B[h_i(x)] \leftarrow 1$ para todo $i \in [1..k]$. Naturalmente, a condição é necessária mas não suficiente para garantir a pertinência, e portanto falsos positivos podem existir. Quanto mais bits forem usados no filtro de Bloom, menor a probabilidade de um falso positivo.

Em [Bonomi et al. 2006], mostra-se que a probabilidade de falso positivo é minimizada quando $k = q \ln 2$, onde $q = m/n$ (o número de bits alocados por elemento), e dada por

$$\Pr[\textsc{FalsoPositivo}] \approx ((1/2)^{\ln 2})^q.$$

Note que, com apenas 10 bits por elemento, é possível obter uma probabilidade de falsos positivos de aproximadamente 0.8%.

3. *Count-Min Sketch*

Count-Min Sketch [Cormode and Muthukrishnan 2005] permite estimar os elementos de um vetor A sem guardá-los todos em memória. O vetor $A[1..n]$ é definido incrementalmente através de operações de atualização na forma de pares ordenados (i, c), representando um acréscimo de c unidades na i-ésima posição do vetor (i.e., $A[i] \leftarrow A[i] + c$). Além disso, permite estimar o produto escalar de dois vetores representados por tal estrutura. A estrutura é composta por uma matriz $M[1..k, 1..m]$, atualizada de forma análoga a um filtro de Bloom, porém cada função de hash mapeia uma linha distinta. A atualização de um par ordenado (i, c) se dá através de $M[j, h_j(i)] \leftarrow M[j, h_j(i)] + c$ para todo $j \in [1..k]$.

A estimativa $\widehat{A[i]}$ é definida por $\min_{j=1}^{k} M[j, h_j(i)]$. É possível mostrar que para uma matriz com $m = \lceil e/\epsilon \rceil$ e $k = \lceil \ln(1/\delta) \rceil$, $A[i] \leq \widehat{A[i]} \leq A[i] + \epsilon \|A\|_1$ com probabilidade $1 - \delta$, onde $\|A\|_1 = \sum_{i=1}^{n} A[i]$. Também é possível estimar o produto escalar $A \cdot B$ de dois vetores, estimado por $\widehat{A \cdot B} = \min_{j=1}^{k} (\sum_{r=1}^{m} M_A[j, r] \cdot M_B[j, r])$. Mostra-se que $A \cdot B \leq \widehat{A \cdot B} \leq A \cdot B + \epsilon \|A\|_1 \|B\|_1$, com probabilidade $1 - \delta$.

4. *MinHash*

MinHash [Broder 1997] permite estimar a semelhança entre conjuntos através da aproximação do coeficiente $J(A, B)$ de similaridade de *Jaccard*, definido para dois conjuntos A e B, como [Real and Vargas 1996]

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

Dada uma função de *hash* h, definimos $h_{\text{min}}(A)$ como o menor valor de h aplicado aos elementos de A, isto é, $h_{\text{min}}(A) = min_{x \in A} h(x)$. Definimos a assinatura ***MinHash*** de A associado às funções de hash $h_1, \ldots, h_k$ como sendo a tupla $(h_{1,\text{min}}(A), \ldots, h_{k,\text{min}}(A))$. É possível mostrar que $\Pr[h_{\text{min}}(A) = h_{\text{min}}(B)] = J(A, B)$. Assim, a comparação de cada posição nas assinaturas ***MinHash*** dos conjuntos A e B (que resulta em 1 se as respectivas posições são iguais e 0 caso contrário) resulta em k variáveis de Bernoulli que podem ser utilizadas como um estimador de $J(A, B)$. O erro do estimador pode ser calculado aplicando o limite de Chernoff onde, para haver erro menor que θ, com confiança de $1 - \delta$, é preciso escolher k tal que

$$k \geq \frac{2 + \theta}{\theta^2} \times \ln(2/\delta).$$

Como ilustração, para $k = 512$, o erro deve ser menor que 12.37%, com 95% de confiança.

5. *HyperLogLog*

HyperLogLog [Flajolet and Martin 1985] permite estimar o número de elementos distintos em um fluxo de dados, utilizando memória sublinear. Mais especificamente, é possível estimar a cardinalidade de elementos distintos em um conjunto com vários bilhões de elementos, com 2% de erro padrão, utilizando menos de 1.5KB de memória. A estrutura baseia-se na observação do padrão de bits do *hash* dos elementos do conjunto. Note que a probabilidade do hash iniciar com um certo número de bits zero é dada por

$$\begin{aligned}
\Pr[h(x) = 1...] &= 2^{-1} \\
\Pr[h(x) = 01...] &= 2^{-2} \\
\Pr[h(x) = 001...] &= 2^{-3} \\
&\vdots \\
\Pr[h(x) = 0^{p-1}1...] &= 2^{-p}
\end{aligned}$$

O algoritmo consiste em particionar o fluxo em m subfluxos disjuntos e, para cada um, observar o maior prefixo $0^{p-1}1$, indicativo de que a cardinalidade naquele fluxo é, com alta probabilidade, da ordem de 2^p. Quanto maior o valor de m, mais precisa se torna a estimativa. De fato, em [Flajolet and Martin 1985], mostra-se que o erro relativo padrão do estimador é igual a $1.04/\sqrt{m}$. Para $m = 2048$ (aproximadamente 1.3KB, usando 5 bits para registrar o maior prefixo de cada fluxo), o erro padrão esperado é de 2.3%.

6. Análises Práticas e Conclusão

Para verificar os limites teóricos previstos, implementamos e testamos as estruturas apresentadas utilizando todas as obras de Shakespeare como conjunto de dados. Para cada estrutura foram utilizados parâmetros compatíveis com as operações. A Figura 1 mostra os resultados encontrados em contraste com o erro previsto pela teoria.

Este resultado ajuda a mostrar a utilidade destas estruturas. Há grande valor teórico e prático em poder estimar propriedades de conjuntos sem mantê-los em memória. Por exemplo, filtros de Bloom são usados desde a década de 70 para economizar operações de E/S; *HyperLogLog* tem aplicação direta na contagem de eventos únicos em aplicações web (usuários únicos, geolocalização, etc.); existem aplicações para *MinHash*

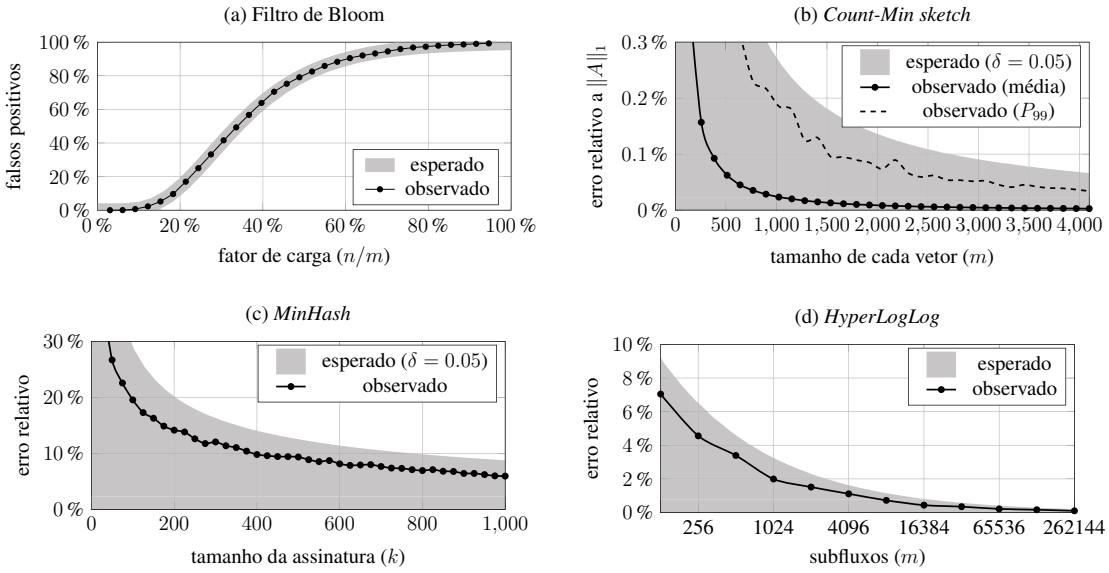

Figura 1. Erro observado ao variar parâmetros das estruturas

e *Count-Min sketch* para computar propriedades grafos muito grandes, que não caberiam em memória para serem utilizados com algoritmos determinísticos. Embora estes algoritmos sejam bastante recentes, percebe-se que há grande potencial para desenvolvimento de novas técnicas que permitam estimar a resposta para problemas clássicos com uso mais eficiente de recursos.

Referências

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16. ACM.

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426.

Bonomi, F., Mitzenmacher, M., Panigrahy, R., Singh, S., and Varghese, G. (2006). An improved construction for counting bloom filters. In *Algorithms–ESA 2006*, pages 684–695. Springer.

Broder, A. Z. (1997). On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE.

Cormode, G. and Muthukrishnan, S. (2005). An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75.

Flajolet, P. and Martin, G. N. (1985). Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209.

Gibbons, P. B. and Matias, Y. (1999). Synopsis data structures for massive data sets. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 909–910. Society for Industrial and Applied Mathematics.

Real, R. and Vargas, J. M. (1996). The probabilistic basis of jaccard's index of similarity. *Systematic biology*, pages 380–385.