CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
RECUPERAÇÃO DE INFORMAÇÃO - MÁQUINAS DE BUSCA NA WEB

Última alteração: 2 de maio de 2017 — 1º semestre de 2017

Professores: Berthier Ribeiro-Neto e Nivio Ziviani
Monitor: Marlon da Silva Diasmailto:marlonsdias@gmail.com

Trabalho Prático 2: 17/04/2017

Data de Entrega: 21/05/2017
Penalização por Atrazo: 1 ponto até 28/05/2016 mais 1 ponto por dia útil a seguir

---

Arquivos Invertidos

O objetivo deste trabalho é projetar e implementar um sistema de programas para recuperar eficientemente informação em grandes arquivos armazenados em memória secundária, utilizando um tipo de índice conhecido como arquivo invertido. As principal referências para este trabalho são Witten, Moffat e Bell (1999), Neubert (2000) e Moffat e Bell (1995). Nesta primeira parte do trabalho o seu programa deverá ser capaz de construir o índice sem necessidade de ser *in situ*, e sem necessidade de usar compressão.

O conceito de arquivo invertido será apresentado a seguir. Considere um conjunto de documentos. A cada documento é atribuído um conjunto de palavras-chave ou atributos. Um *arquivo invertido* é constituído de uma lista ordenada de palavras-chave, onde cada palavra-chave tem uma lista de apontadores para os documentos que contêm aquela palavra-chave. Este é o tipo de índice utilizado pela maioria dos sistemas para recuperação em arquivos constituídos de texto.

A utilização de arquivo invertido aumenta a eficiência de pesquisa em várias ordens de magnitude, característica importante para aplicações que utilizam grandes arquivos constituídos de texto. O custo para se ter essa eficiência é a necessidade de armazenar uma estrutura de dados que pode ocupar entre 2% e 100% do tamanho do texto original, dependendo da quantidade de informação armazenada no índice, mais a necessidade de atualização do índice toda vez que a coleção de documentos sofre alguma alteração.

O que fazer

A estrutura de dados a ser implementada deverá ser constituída do vocabulário do texto, incluindo o número de documentos associados com cada palavra-chave e uma lista de ocorrências da palavra na coleção de documentos. Cada entrada da lista indica o número do documento onde a palavra ocorreu e o número de ocorrências. O formato da lista invertida deve ser o formato apresentado em Moffat e Bell (1995) e Witten, Moffat e Bell (1999). Para cada termo t devem ser armazenados todos as triplas $< d, f_{d,t}, p >$, onde d é o número do documento onde t ocorre, $f_{d,t}$ é a freqüência do termo t no documento d e p é a posição onde t ocorre dentro de d.

Pontos Extras

As listas de números de documentos dentro da lista invertida contém números inteiros positivos em ordem ascendente. Existem métodos de compressão específicos para conjuntos de números em ordem ascendente que levam a boas taxas de compressão. três destes métodos são codificação unária, Elias-γ e Elias-δ, todos descritos nas referências citadas abaixo. Este trabalho não exige o uso de compressão do índice. Entretanto, quem decidir usar compressão no momento de armazenar a lista invertida no disco terá 10% de *pontos extras*.

Processamento de Consultas

O sistema de programas recebe do usuário uma ou mais palavras e imprime todos os documentos que satisfaçam a consulta. No caso deste trabalho a linguagem de consulta utiliza o modelo booleano e assume um conector lógico `and` entre palavras. A saída será apenas a impressão dos documentos (seus identificadores) que satisfaçam a consulta.

Como fazer

1. A linguagem de programação pode ser C++ ou Java.
2. Utilizar a sua coleção obtida no TP1.
3. Você pode utilizar um parser público. Um que pode ser utilizado está em: `http://sourceforge.net/projects/htmlcxx` ou `https://jsoup.org/` para Java.
4. É proibido fazer tudo em memória principal, sendo necessário ir ao disco e fazer ordenação externa.
5. Preocupem com eficiência: façam medida do tempo de criação do índice. Experimentos em diferentes máquinas, ou mesmo comparando tecnologias de storage diferentes (como HD vs SSD) sao apreciados, ainda que não contem ponto extra.

Avaliação

O trabalho será avaliado a partir das listagens dos programas, da documentação entregue, da análise de complexidade das rotinas e do resultado da execução.

Apresente uma boa documentação do trabalho, contendo pelo menos os seguintes itens: saída legível mostrando o funcionamento do código, código bem estruturado, comentários explicativos sobre os algoritmos e estruturas de dados, análise da complexidade, resultados experimentais medindo tempos e análise dos resultados.

Referências

Ian H. Witten, Alistair Moffat e Timothy C. Bell: *Managing Gigabytes: Compressing and Indexing Documents and Images*, Morgan Kaufmann Publishers, 1999, second edition.

Justin Zobel e Alistair Moffat: Inverted Files for Text Search Engines, *ACM Computing Surveys 38*(2), 2006.

Alistair Moffat e Timothy C. Bell: In Situ Generation of Compressed Inverted Files, *Journal of the American Society for Information Science 46*(7), 1995: 537–550.

Ricardo Baeza-Yates and Berthier Ribeiro-Neto: *Modern Information Retrieval*, Pearson, 2011 (second edition), 913 pages.