Recuperação de Informação - Máquinas de Busca na Web: Trabalho Prático 2

Professor: Berthier Ribeiro-Neto
Monitor: João Mateus de Freitas Veneroso

April 9, 2018

1 Arquivos Invertidos

O objetivo deste trabalho é projetar e implementar um sistema para recuperar informação eficientemente a partir de arquivos grandes armazenados em memória secundária utilizando um tipo de índice conhecido como arquivo invertido. As principais referências para este trabalho são Witten, Moffat e Bell (1999) [1] e Moffat e Bell (1995) [3]. O seu programa deverá ser capaz de construir o índice invertido, sem a necessidade de ser *in situ* (permutação in-place), e sem a necessidade de usar compressão.

A utilização do arquivo invertido aumenta a eficiência da 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.

2 O que fazer

A estrutura de dados a ser implementada deve constituir do vocabulário do texto, incluindo o número de documentos associados a 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 Witten, Moffat e Bell (1999) [1] e Moffat e Bell (1995) [3]. Para cada termo t devem ser armazenadas todos as triplas $< d, f_{d,t}, p >$, onde d é o número do documento onde t ocorre, $f_{d,t}$ é a frequência do termo t no documento d e p é a posição onde t ocorre dentro de d.

3 Pontos Extras

As triplas armazenadas na lista invertida contém somente números inteiros positivos em ordem ascendente. Existem métodos de compressão para conjuntos de inteiros em ordem ascendente que possuem 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.

4 Processamento de Consultas

O programa deve receber do usuário uma ou mais palavras e imprimir 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 deve ser apenas a impressão dos documentos (suas urls) que satisfaçam a consulta.

No vocabulário, além de armazenar o termo, você pode armazer a posição no arquivo invertido onde começam as triplas daquele termo. Dessa forma, ao iniciar a máquina de busca, basta carregar o vocabulário com esses apontadores em memória principal para responder consultas em termo de milisegundos.

5 Como fazer

- A linguagem de programação deve ser C++.
- O código deve compilar em Linux.
- Utilizar a coleção compilada do TP 1. Vocês receberão essa coleção assim que eu tiver recebido a maior parte das coleções do TP 1. Por enquanto, vocês podem fazer os testes com o que vocês coletaram no TP 1 ou com a coleção de teste disponibilizada no link:
  https://drive.google.com/open?id=1OWg_92nSjjAq1ezuvkdbi_rBT5DLIEiB
- Você pode utilizar um parser público. Algumas possibilidades são:
  - http://sourceforge.net/projects/htmlcxx
  - https://github.com/google/gumbo-parser
- É proibido fazer tudo em memória principal, sendo necessário ir ao disco e fazer ordenação externa.
- Preocupem com a eficiência: façam medida do tempo de criação do índice e outras métricas relevantes.

6 Avaliação

O trabalho será avaliado a partir do código, 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 e análise dos resultados.

References

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

[2] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. Inverted files for text search engines. *ACM Computing Surveys*, 38(2), 2006.

[3] Alistair Moffat and Timothy C. Bell. In situ generation of compressed inverted files. *Journal of the American Society for Information Science*, 46(7):537–550, 1995.

[4] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Pearson, second edition, 2011.