Trabalho Prático 5
Processador de consultas
Recuperação de Informação

Daniel Ferreira Abadi[1]
2018088062

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

1. Introdução

O problema abordado consiste em construir um processador de consultas sobre a coleção que nos foi passada em trabalhos anteriores, utilizando o modelo vetorial para gerar uma ordem entre as páginas.

2. Implementação

Para a realização deste trabalho foi necessário utilizar os dados gerados pelo trabalho anterior, como o arquivo invertido, o índice desse arquivo, um documento contendo os identificadores dos termos e um outro contendo os identificadores dos endereços web da coleção.

Inicialmente os conteúdos são passados para a memória principal, exceto o arquivo invertido completo. Todos são guardados em estruturas do tipo "Hash". Em especial o índice do arquivo invertido é guardado em um "Hash" cuja chave é o identificador do termo e o valor é uma estrutura "Tuple" com três elementos. Esses elementos são dois do tipo "long int" e um do tipo "float". Os dois primeiros são as posições iniciais e finais do arquivo invertido no qual àquela palavra se refere, em termos de bytes, e o outro é o "IDF" do termo associado.

Neste trabalho foi utilizado o peso "TF-IDF" juntamente com o modelo vetorial para gerar os resultados. Para o peso "TF" foi utilizada a variante "log normalization", cuja fórmula é:

$$1 + logf_{(i,j)}.$$

Já para "IDF", foi utilizada a variante "inverse frequency":

$$log\frac{N}{n_i}.$$

Para o cálculo da similaridade, requerida pelo modelo vetorial, foi utilizada a seguinte equação:

$$sim(d_j, q) = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} w_{i,q}^2}}$$

Onde $w_{i,q} = (1 + logf_{i,q}) \times log\frac{N}{n_i}$ é o peso do termo i na consulta e $w_{i,j} = (1 + logf_{i,h}) \times log\frac{N}{n_i}$ é o peso do termo i no documento j.

2.1. Calculando o "TF-IDF" da consulta

Quando a consulta é feita pelo usuário, ela é então dividida em palavras e colocadas em uma estrutura "Map", cuja chave é a palavra em si e o valor é um inteiro que representa a quantidade de vezes que a mesma apareceu na pesquisa. Com esse dado é então calculado o "TF" de cada termo da consulta, e então são buscados os "IDF's" no índice do arquivo invertido. Com esses valores é calculado o peso "TF-IDF" dos termos que são colocados em uma estrutura "Vector" do tipo "float".

2.2. Calculando o "TF-IDF" dos documentos

Enquanto se é calculado o "TF-IDF" da consulta, é também calculado para os documentos, por meio de uma função que recebe o termo e o procura na estrutura "Map", citada anteriormente, que guarda o índice do arquivo invertido. Já no arquivo invertido, é então utilizado um "Map" para guardar os documentos e a quantidade de vezes que uma palavra ocorreu no mesmo, sendo a palavra a chave e a quantidade de vezes um inteiro.

Com esses dados em mãos, é então calculado o "TF-IDF" e guardado em um "Map" cuja chave é o identificador do documento e o valor um "Vector" do tipo "float". Esse processo é feito palavra por palavra da pesquisa e, ao final, todos os documentos que possuem menos elementos em seu "Vector" do que a quantidade de palavras diferentes na consulta são descartados.

2.3. Calculando a similaridade

Tendo todos os pesos, é então calculada a similaridade entre a consulta e todos os documentos. Isso é feito durante uma iteração pelo "Map" que guarda o identificador da página e o "Vector" com os pesos.

Todos os resultados são então guardados em uma estrutura do tipo "Set" que armazena estrutura do tipo "Pair", sendo o primeiro elemento a similaridade e o segundo o identificador do documento. É então recuperado o endereço Web das cinco primeiras páginas com similaridade mais altas.

2.4. Detalhes adicionais

No início da execução do programa, os dados são carregados para a memória principal, como dito anteriormente. Esse carregamento dura em torno de oito segundos, sendo este tempo exibido em tempo de execução. Esse tempo ocorreu em um notebook com Intel(R) Core(TM) i5-5200U CPU @ 2.20 GHz.

Toda consulta é convertida para o formato no qual os dados da coleção foram transformados, a versão convertida é mostrada quando se faz a pesquisa. Para cada consulta é, também, exibido o tempo de busca dos resultados, assim como a quantidade de documentos encontrados, fora os cinco primeiros mais bem colocados com suas respectivas similaridades.

3. Consultas

Como nos foi especificado, precisaríamos prover 25 consultas com bons resultados em nosso buscador. Durante o processo de procurar bons resultados foi notado o quão imprecisa está a busca, não só devido à falta do "page rank", mas devido a não verificação

das posições do termo no documento. Em dezenas de consultas com mais de um termo, as palavras estavam espalhadas pelo documento, trazendo um péssimo resultado, mesmo com similaridade máxima.

Há, também, muitos problemas com consultas de um único termo, gerando similaridade máxima com vários documentos, retornando muitas páginas que nada tem a ver com a busca. Isso mostra que, com apenas o modelo vetorial, a consulta não é boa, necessitando de muito mais.

Um fato interessante é que como não há uma busca utilizando-se as posições do termo no documento, pesquisar sem utilizar preposições como de, da, do e artigos definidos e indefinidos é mais eficiente, pois retornam o mesmo resultado e com menos tempo de espera. Isso se deve ao fato de que preposições e artigos serem muito comuns em textos, então aparecem em quase todos os documentos, se não todos, o que gera uma leitura de um bloco muito grande no arquivo invertido, demorando consideravelmente mais.

4. Conclusão

Foi construído um processador de buscas utilizando-se dos dados gerados pelo trabalho anterior. O programa armazena vários dados em memória principal, como o índice do arquivo invertido, a relação de termos e seus identificadores, a relação de documentos e seus endereços. O buscador utiliza-se do modelo vetorial e similaridade para realizar o ranqueamento das páginas para as consultas.

Durante a tarefa de se conseguir 25 boas consultas foi notado que nosso processador, da forma como está, é ineficiente e dificilmente retorna algum resultado descente. Isso mostra que o modelo vetorial não funciona bem sozinho, precisando de complementos para ter uma boa performance.