Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
DCC024 – PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE II - (PDSII)

TRABALHO PRÁTICO
Máquina de Busca

Aluna: Priscilla Pereira Alves Michelle
Matrícula: 2018048540
Prof. Thiago Ferreira de Noronha

1.Introdução

Um *site* ou portal buscador é caracterizado pelo o funcionamento de seu motor de busca. Este, rastreia a informação disponível em arquivos diversos extraindo documentos, palavras ou termos que melhor representam a informação buscada, e então a armazena em uma gigantesca base de dados que pode ser consultada pelos usuários por meio de uma interface de busca. Assim, os buscadores funcionam coletando informação não organizando-as para o fluxo de acesso rápido e eficaz.[1]

A recuperação de uma informação relevante é diretamente influenciada pela consulta do usuário que deve traduzir o assunto em que está interessado a pesquisar numa linguagem fornecida pelo sistema. Na maioria das vezes isto implica em especificar um conjunto de palavras que resume um assunto que transmita a semântica da informação desejada. Um *termo indexado* é uma palavra cuja semântica ajuda a lembrar do tema principal do documento. Desta forma, termos são usados para indexar e sumarizar o conteúdo do documento. [2]

No modelo Booleano, os documentos e consultas são representados como um conjunto de termos indexados. As consultas são especificadas usando expressões booleanas. O critério do modelo Booleano é baseado no sistema binário, ou o termo da consulta está presente no conjunto de termos indexados ou o termo não está presente. No modelo Booleano, não existe um grau intermediário de relevância para o documento em relação à consulta, ele é relevante ou não.[2]

Neste trabalho foi implementada uma Máquina de busca que avalia a similaridade da palavra pesquisada com a palavras contidas nos arquivos de consulta. No presente caso os arquivos de consulta são trechos do livro *Dom Casmurro* de Machado de Assis.

2. Implementação

O sistema recebe como entrada um conjunto de arquivos de texto, que devem ser lidos, palavra após palavra, para construir o índice invertido. O programa lê essas entrada e normaliza todas as letras para minúsculas e retira todos os sinais gráficos e acentuação, mantendo assim um conjunto de palavras e números formatados. Um índice invertido armazena as que as coordenadas de um documento. As consultas, são via *cosine raking* que ordena as saídas ordenando a lista dos *K* 's documentos mais próximos das consultas.

Os arquivos d1.txt, d2.txt, d3.txt, d4.txt e d5.txt são os arquivos de texto que serão tratados no problema.

Os arquivos Leitor.h e Leitor.cpp são os códigos que tratam os textos, lendo os arquivos de texto como entrada e por meio da função "isalpha" e "lowercase" os textos são

normalizados como minúsculas e sem pontuação ou acentuação gráfica. Neste arquivo também há a função que aponta a última palavra e a função que apaga o arquivo palavra por palavra a partir da última. Isso é útil para gravar todas as palavras em um único arquivo e em ordem em que elas aparecem, por isso também o índice marca o arquivo de origem das palavras. Um contador também é implementado, tanto para quantidade total de palavras quanto para a quantidade de palavras distintas.

No arquivo ranking.h e ranking.cpp as funções estão relacionadas ao ranking das palavras. Nele todas as palavras são armazenadas em um único vetor e são implementadas funções que verificam se uma determinada palavra passada como parâmetro está no vetor, quantas vezes ela parece e qual seu arquivo de origem. A função acrescentar inclui palavras ao vetor de palavras a partir da última posição. Também são computadas as quantidades de palavras e de arquivos lidos. Um vetor vocabulário armazena as palavras e seu arquivo de origem.

De acordo com a própria proposta do trabalho, é necessário calcular a frequência, importância e distribuição das palavras nos arquivos. Por isso, uma vez armazenadas as palavras em um TAD é possível calcular quantas vezes ela aparece e sua coordenada e assim definir uma escala de importância. Com esses parâmetros também é possível calcular a similaridade entre palavras, e a partir disso a distância entre as coordenadas das mesmas.

3. Conclusão

Por meio das fórmulas estabelecidas no modelo booleano de indexação foi possível criar um código que avalia a frequência e a importância de uma palavra num conjunto de arquivos textos. Porém, elaborar o ranking e ordenar todas as palavras do arquivo se mostrou uma tarefa onerosa e não pode ser concluída.

4. Referências Bibliográficas

[1] ABDALA,CARMEN VERÔNICA MENDES; ANDRADE, VINÍCIUS ANTÔNIO DE. Recuperação de informação baseada em clusters.REVISTA USP, São Paulo, n.80, p. 54-61, dezembro/fevereiro 2008-2009

[2]Cerqueira, Alisson Gomes. Geração de Arquivo Invertido Utilizando Programação Paralela — MPI. Monografia de Graduação. Lavras, Minas Gerais - Brasil 2002.

[3] SAVITCH,Walter C++Absoluto.Pearson Education, Inc, Pearson Education do Brasil. São Paulo, 2002.