Universidade Federal de Minas Gerais

Trabalho Prático de Programação e Desenvolvimento de Software II

Professores: Thiago Ferreira de Noronha e Lucas Victor Silva Pereira

Aluno: Felipe Freitas Vieira Campidelli

Belo Horizonte
Junho de 2019

1 Introdução

A recuperação de informação é um processo que possibilita a transformação de uma busca pelo usuário em uma série de citações e documentos contendo dados relevantes, dessa forma simplificando a procura em um acervo de informações ao minimizar a necessidade de procura pelo documento mais relevante.

No trabalho foi proposto a implementação de uma maquina de busca responsável pela recuperação de informação pertinente a partir de uma busca do usuário, apresentando os resultados pelos nomes dos documentos mais relevantes em ordem decrescente de significância.

Na implementação do código os arquivos que representam o acervo são contados em um diretório separado e transformados em vetores de strings onde cada string representa uma palavra, quando o usuário entra com a busca, a mesma é transformada em um vetor de string e a partir daí cada arquivo e a busca é transformado num vetor peso onde cada coordenada representa o peso de uma palavra. Finalmente é calculado a similaridade do cosseno entre a busca e cada documento, onde o resultado com maior valor simboliza o documento mais relevante à busca, por fim é impresso ao usuário o nome de cada documento em ordem decrescente de relevância.

2 Implementação

O trabalho foi implementado em c++ e compilado utilizando a IDE Code::Blocks 17.12 e dividido nos arquivos metodos.cpp, metodos.h e main.cpp. Foi utilizada somente uma biblioteca externa do c++, o doctest.h para a implementação dos testes de unidade.

As bibliotecas string e vector foram utilizadas para o armazenamento e tratamento das informações, enquanto fstream, sstream, bits/stdc++.h, stdio.h, sys/types.h e dirent.h foram utilizadas para facilitar o fluxo, tratamento e leitura de informações, possibilitando a quebra de uma string num vector de string, a contagem de arquivos em um diretório e a passagem de informação de um arquivo de texto para um vector de string. Por fim as bibliotecas iostream e cmath foram usadas para possibilitar a entrada e saída de informações e as operações matemáticas raiz quadrada e ln, respectivamente.

Apesar da sugestão do tratamento dos dados em um set foi utilizado um vector como substituto devido a sua familiaridade, sendo usado para armazenar tanto strings representando palavras quanto doubles representando os vetores de coordenadas pesos, foi também utilizado um map para armazenar cada documento e sua respectiva relevância para a busca.

Funções implementadas:

A função Count_File() abre a pasta Dataset, conta quantos arquivos e retorna a quantidade. Faz uso da biblioteca dirent.h.

A função Read_Files() incrementa uma string que é passada como nome do arquivo a ser aberto e insere as palavras do arquivo num vector de string, dessa forma ela abre todos os arquivos e insere todas as palavras no vector depois de transforma-las em minúsculo em seguida o retorna. Faz uso das bibliotecas string, vector e fstream.

A função Read_File(string) se comporta exatamente como Read_Files() para um arquivo específico, dessa forma ela retorna um vector de string de somente um arquivo.

A função Count_Files_Word(string) recebe uma palavra e checa quantos documentos possuem está palavra, em seguida retorna está quantidade. Faz uso das bibliotecas string, bits/stdc++.h e fstream.

A função Modulo(vector<double>) recebe um vetor de peso, calcula seu módulo ou norma e o retorna, posteriormente utilizada no calculo da similaridade de cosseno. Faz uso das bibliotecas vector e cmath.

A função Somatorio(vector<double>, vector<double>) recebe dois vetores de peso e calcula o produto escalar entre eles e o retorna, posteriormente utilizada no calculo da similaridade de cosseno. Faz uso das bibliotecas cmath e vector.

A função Cosine_Similarity(vector<double>, vector<double>) recebe dois vetores de peso, calcula a similaridade dos cossenos e a retorna. Similaridade dos cossenos é o equivalente de calcular o cosseno entre os vetores.

A função idf(string) recebe uma palavra e calcula a sua frequência inversa na documentação que consiste em tirar o ln do número de documentos divido pelo numero de documentos com ocorrência da palavra. Faz uso da biblioteca cmath

A função tf(string, string) recebe um arquivo e uma palavra e percorre o vetor de string correspondente ao arquivo checando o numero de ocorrências da palavra, ao fim do vetor retorna a quantidade. Faz uso da biblioteca vector.

A função Vetor_Documento(string, vector<string>) recebe um arquivo e um vetor palavra e calcula o seu vetor peso por meioda inserção do produto de cada idf e tf. Por fim retorna o vetor peso. Faz uso da biblioteca vector.

A função make_sentence(string) recebe a busca feita pelo usuário, insere cada palavra em um vetor de string e o retorna. Faz uso das bibliotecas string e sstream.

A função Cosine_Ranking(string) recebe a busca feita pelo usuário e insere em cada posição de um vector de double (em ordem crescente de nome do documento) a relevância de um documento para a busca e retorna o vetor de relevância.

Por fim, na main é pedido ao usuário para fazer a busca, em seguida é criado o vetor de relevância e inserido num map em conjunto com seu respectivo documento e finalizando os documentos no map são impressor em ordem decrescente de relevância, dessa forma o documento mais relevante é o primeiro resultado da busca, o segundo mais relevante é o segundo resultado e por aí em diante.

3 Conclusão

Podemos perceber que a simplificação proporcionada pela utilização de uma maquina de busca em uma consulta num acervo de informação é enorme, acabando com a necessidade de procurar em todo o acervo ao retornar os documentos mais relevantes.

A divisão do programa em arquivos simplificou a compreensão e implementação do código como um todo, em decorrência economizando horas do trabalho. Enquanto a implementação dos testes contribuiu para o mais fácil entendimento do programa e a convicção do seu funcionamento.

Sobretudo não foi implementado uma classe devido a dificuldade em identificar qual seria o objeto, seus atributos e a simplificação que isto acarretaria no trabalho, também não foi implementada uma função hash como foi originalmente proposto tendo em vista que o ranking cosseno já seria responsável por retornar os documentos em ordem decrescente de relevância, satisfazendo assim o principal objetivo do programa.

Os arquivos estão disponíveis no GitHub https://github.com/FelipeFVCampidelli, o dataset disponibilizado está na pasta Dataset_P e para utiliza-lo com o programa é necessário a copia dos dados para o diretorio do programa. Para a compilação é possível criando um projeto console application pela IDE Code::Blocks com os arquivos e o compilando. Por padrão o programa lerá cinco arquivos exemplo contidos na pasta Dataset e no diretório do código, para alterar os arquivos a serem lidos deve-se alterar o nome das strings nas respectivas funções.