Documentação Trabalho Prático: Máquina de Busca

Universidade Federal de Minas Gerais

Programação e desenvolvimento de Software II

Luiz Henrique da Silva Gonçalves

Hilário Corrêa da Silva Neto

Matheus Irias

22 de novembro de 2019

1 Introdução:

O trabalho tem como objetivo a implementação e desenvolvimento de um sistema de consulta para armazenar dados e retornar informações requisitadas via queries. Para tal tarefa foi implementado um índice invertido para armazenar dados fornecidos na entrada e também a criação de uma máquina de busca para recuperar os documentos associados a consultas.

1.1 Base de Documentos:

A base de documentos é composta por um conjunto arbitrário de arquivos de texto, formado por palavras separadas por espaço ou outro tipo de caractere especial, podendo ser ainda palavras maiúscula ou minúscula. Para facilirar a verificação dos resultados, optou-se por usar os mesmos arquivos dos enunciados no PDF do Trabalho Prático.

1.2 Índice invertido:

O índice invertido implementado considera um conjunto de documentos (base de dados). A cada documento é atribuído um conjunto de palavras-chave, onde cada palavra-chave tem uma lista de apontadores para os documentos que contêm tais palavras.

1.3 Máquina de Busca:

Criada para recuperar informações na base de dados, recebe como entrada uma consulta e dá como saída os documentos mais relevantes. Nesse modelo os documentos e as consultas são tratados como vetores e a relevância dos documentos é calculada a partir da similaridade, ordenando os documentos.

2 Implementação

O TP foi implementado em apenas um arquivo .cpp, com funções de leitura de arquivo e conversão das palavras, também uma função para o índice invertido e por último uma função para aplicação da similaridade. Na função main foi implementada a máquina de Busca, com a leitura dos documentos do diretório e o processamento de ranqueamento e relevância dos documentos com base na pesquisa do usuário.

2.1 Fluxograma da implementação do trabalho

1.Bibliotecas utilizadas:

```cpp
#include <iostream>
#include <string>
#include <fstream>
#include <set>
#include <map>
#include <vector>
#include <cmath>
#include <dirent.h>

using namespace std;
```

2.Funções:

String converte(string palavra)

Função responsável por converter as palavras lidas dos documentos em minúsculo e também retirar os caracteres especiais.

Nessa função recebemos uma string como entrada e percorremos cada caractere da palavra por um loop e convertemos as letras para minúsculo e, se o caractere for especial, o retiramos da palavra.

```cpp
string converte(string palavra){
    int caract=1;
    while(caract){
        caract=0;
        for(int i=0;i<palavra.size();i++){
            palavra[i]=towlower(palavra[i]);
            if(palavra[i]=='-'  || palavra[i]=='.' || palavra[i]=='?' || palavra[i]==' ' || palavra[
                caract=1;
                for(int j=i;j<palavra.size();j++){
                    palavra[j]=palavra[j+1];
                }
                //palavra.pop_back();

            }

        }
    }
    return palavra;

}
```

Void Inv(vector<string> &palavras, string filename)

Função Inv tem por objetivo abrir os arquivos de acordo com o filename que é passado como parâmetro e ir indexando cada palavra ao seu respectivo arquivo.

Nessa função recebemos como parâmetro um vector de string palavras e uma string do nome do arquivo. Primeiro lemos o diretório onde se encontra os arquivos e associamos cada arquivo ao filename e depois lemos cada arquivo e inserimos as

palavras dos arquivos no vector palavras, indexando dessa forma cada palavra ao seu documento.

```
49  void Inv(vector<string> &palavras,string filename){
50  try{
51      fstream file;
52      filename = "./docs/" + filename;
53      file.open(filename.c_str(), fstream::in | fstream::out);
54      string palavra;
55      if(file.is_open()){
56          while(file >> palavra){
57              palavra=converte(palavra);
58              palavras.push_back(palavra);
59          }
60          file.close();
61      }
62      else{
63          throw 0;
64      }
65
66  }catch(int v){
67      cout<<"Arquivo nao encontrado!"<<endl;
```

*Double simililar(double*x, double *y, int tam)*

Aplica a formula da similaridade. Recebe dois arrays, o primeiro matematicamente correspondente ao vetor gerado pelas coordenadas w de um (e apenas um) documento, com cada dimensão correspondente a uma palavra do vocabulário. O segundo corresponde ao vetor com as coordenadas w da query, com, também, cada dimensão representando uma das palavras do vocabulário e sua respectiva comparação com a query.

```
33  double similar(double* x, double* y, int tam) {
34      double num = 0;
35      double denom1 = 0;
36      double denom2 = 0;
37      int i;
38      for(i=0;i<tam;i++) {
39          num = num + (x[i] * y[i]);
40          denom1 = denom1 + (x[i] * x[i]);
41          denom2 = denom2 + (y[i] * y[i]);
42      }
43      denom1 = sqrt(denom1);
44      denom2 = sqrt(denom2);
45      return num/(denom1 * denom2);
46  }
```

Double calcula_tf(vector<string> &palavras, string word)

Recebe o vetor com as palavras e um string (que representa uma palavra do vocabulário em específico ou a query, dependendo do contexto) e calcula o número de ocorrências da palavra passada por referência em cada doc.

```
double calcula_tf(vector<string> &palavras, string word) {
    int i = 0;
    vector<string>:: iterator it;
    for(it=palavras.begin();it!=palavras.end();it++) {
        if(word==*it)
            i++;
    }
    return i;
}
```

Int main ()

Faz a entrada de dados da query, além da declaração e alocação de todas as estruturas de dados que serão tratadas (palavras, vocabulário, lista de documentos), iteradores, variáveis auxiliares e índices, além de instruções para a abertura dos arquivos localizados na pasta docs.

3 Utilização

A interface do programa é muito simples. Ao abrir o executável, o usuário irá se deparar com uma mensagem o pedindo para inserir os strings a serem pesquisados. Cada string será guardado após o usuário teclar enter. Quando o ponto final(.) for inserido, termina a alocação da query.
Ao fim, será imprimida na tela uma lista com os nomes dos respectivos documentos e os respectivos cossenos de seus vetores com o da query, ordenada do documento mais similar para o mais distante.

4 Testes

Teste da função converte

```
55  TEST_SUITE("FUNCOES_MB"){
56  TEST_CASE("converte()"){
57      string word="ABACAXI";
58      string word1="GUARDA-CHUVA";
59      string word2="Ab!?-C";
60      CHECK(converte(word)=="abacaxi");
61      CHECK(converte(word1)=="guardachuva");
62      CHECK(converte(word2)=="abc");
63
64  }
```

Word checa se há conversão de maiúsculas para minúsculas, word1 e word2 checam a retirada de caracteres especiais.

Teste da função Inv

```
TEST_CASE("Inv(vector<string> &palavras,string filename)"){
    vector<string> palavras;
    vector<string>::iterator it=palavras.begin();
    string filename;
    string word1,word2;
    filename="arquivo.txt";
    CHECK(Inv(palavras,filename)=="Arquivo nao encontrado!");
    filename="d1.txt";
    ifstream file;
    file.open(filename.c_str());
    while(file >> word1){
            word2=converte(word1);
            palavras.insert(word2);
}
    CHECK(*it=="a");
    it++;
    CHECK(*it=="a");
    it++;
    CHECK(*it=="a");
    it++;
    CHECK(*it=="b")
}
```

O primeiro check veda a possibilidade de que o programa tente abrir um arquivo inexistente, os quatro seguintes checam se, dado o documento em questão, os strings são identificados corretamente.

Teste da função similaridade

```
TEST_CASE("similar(double* x, double* y, int tam)"){
    double num = 0;
    double denom1 = 0;
    double denom2 = 0;
    int i;
    double x[tam];
    double y [tam];
    for(i=0;i<tam;i++)
        x[i] = 1;
        y[i] = 1;
    CHECK(similaridade(x, y, tam) == 1);
    x[0] = x[2] = 0;
    y[0] = y[1] = 0;
    CHECK(similaridade(x,y,tam) == 0;
}
}
```

O primeiro CHECK verifica o cosseno entre dois vetores paralelos, que deve ser 1, dado que o ângulo entre eles é, por definição, zero. O segundo checa se o cosseno entre dois vetores unitários distintos é 0, o que indica que o ângulo entre eles é, também por definição, reto.

Teste da função calcula_tf

```
TEST_CASE("double calcula_tf(vector<string> &palavras, string word)"){
    int i = 0;
    vector<string> palavras;
    vector<string>::iterator it=palavras.begin();
    string word = "aa"
    for(it==palavras.begin();it!=palavras.end();it++)
        palavras.insert(aa);
    CHECK(calcula_tf(palavras, word) == vocabulario.size())
}
}
```

Um vector de strings é preenchido com um número de cópias de uma palavra igual ao tamanho do vocabulário. Verifica-se se o número de ocorrências da mesma palavra é igual ao tamanho esperado.

5 Conclusão

O grupo considerou que o trabalho foi de grande valia em relação a aprendizagem do conteúdo abordado durante todo o semestre, tendo sido abordado no TP desde os conteúdos mais simples vistos em aula até os mais complexos, nos possibilitando, dessa forma, ter uma visão ampla e pratica do desenvolvimento de uma aplicação tão importante nos dias de hoje. Certo de que o trabalho tinha por objetivo o desenvolvimento das capacidades técnicas em relação a programação de algoritmos e estruturas de dados, pode-se certificar que toda essa abordagem técnica foi aplicada na prática, concluindo com êxito assim o objetivo do projeto prático.