Átila Augusto Carvalho 2019006442

TP de Programação e desenvolvimento de dados 2

1) Introdução

Nesse TP o objetivo é implementar uma função de busca em um índice invertido usando o cosine ranking. Assim, tendo um grupo de arquivos, o usuário poderá fornecer uma querry (consulta) e a partir dessa consulta o programa deve retornar para o usuário qual dos arquivos consultados será mais semelhante ao querry.

Para realizar essa tarefa, será criado um índice invertido, que associa cada palavra a cada arquivo em que tal palavra é mencionada.

2) Implementação

2.1) Ideia

O índice invertido foi implementado na classe Index, como um mapa, e para cada palavra lida em um arquivo seria adicionado um nó no mapa, a chave desse nó é uma string armazenando a palavra lida. Cada nó com uma determinada chave está linkado a um array de números inteiros, cada posição n nesse vector está associada a um arquivo, e o número armazenado nessa posição é o número de vezes que essa chave aparece no arquivo associado.

Sempre que um arquivo é lido pelo programa, o nome desse arquivo é adicionado a um vector de strings, e se o nome desse arquivo está na posição c[n] nesse vector, a posição n do vector de inteiros está associada a esse arquivo.

Exemplo:

Dado os seguintes arquivos:

A1.txt: olha ele la

A2.txt: eu me recuso

A3.txt: Olha sim cara. E olha logo!

A4.txt: Lou-cu-ra

```
vector<string> c;//guarda os nomes dos arquivos
c ={"A1.txt", "A2.txt", "A3.txt", "A4.txt",};
map<string, int* > base_;
base_["olha"] == {1, 0, 2, 0};
base_["recuso"] == {0, 1, 0, 0};
```

Dessa forma uma palavra pode ser rapidamente associada aos arquivos que a contém e a quantas vezes ela aparece em cada arquivo.

2.2) Atributos

Todos os seguintes atributos usados na classe Index são privados.

base_: Uma variável do tipo map<string, int[50]> usada para guardar o mapa usado para implementar o índice.

t_: Uma variável do tipo int usada para guardar quantos arquivos foram lidos, só é inicializada após a leitura de todos os arquivos e serve para facilitar diversos cálculos em funções internas da classe. Útil em alguns cálculos.

keys_: Uma variável do tipo vector<string> usada para guardas as chaves de cada nó de base_. Útil na hora de iterar através das chaves do mapa.

2.3) Métodos

vector<string> Conv(string c): recebe uma string c com o nome de um arquivo, abre esse arquivo e retorna um vector<string> vc, onde cada posição de vc é ocupada por uma palavra do arquivo original.

void Form(vector<string>& vc): recebe um vector de strings como parâmetro e formata ele, passando todas as letras para minúsculas e retirando pontuação.

void Form(string c): recebe uma strings como parâmetro e formata ela, passando todas as letras para minúsculas. Não necessita retirar pontuação pois só é usada na querry, que tem como precondição ter apenas letras e números.

void AddFile(string c, int j): recebe uma string c com o nome de um arquivo e um inteiro j, que é o número associado ao arquivo. Após chamar Conv(string c) e Form(vector<string>& vc), adiciona todas as palavras do arquivo ao mapa.

void AddFile(vector<string> vc, int j): análoga a função acima, mas ela não precisa chamar Conv(string c), pois trabalha diretamente com a querry.

void SetSize(int i): coloca o valor de i dentro do atributo t_, único método privado de Index.

void Print(string c): exibe todos os valores do vector base_[c].

int filenum() const: retorna o valor de t_.

double idf(string c): retorna a importância da palavra de c para o índice.

double sim(vector<string> q, int i): retorna a similaridade entre a querry q e o arquivo i.

void run(): lê dados do teclado e organiza a ordem das funções

Para ler os arquivos do PC, eles devem estar dentro de uma pasta chamada “Resource”, que está no mesmo diretório dos arquivos, e digitar o nome dos arquivos com a extensão.

ex:D1.txt

Ao terminar de informar os arquivos que vão ser lidos, o usuário deve enviar a letra f.

Para informar a querry, o usuário deve apertar enter após cada palavra, e ao terminar deve enviar o caractere “.”. É pré-condição que a querry não tenha nenhuma pontuação.

3) Conclusão

A classe para armazenamento dos dados foi implementada com sucesso por meio do mapa descrito previamente e de algumas variáveis para facilitar cálculos.

A unidade de teste não foi incluída pois não consegui fazer o “doctest.h” rodar na minha IDE. //amanhã mesmo eu migro para o Linux

A função sim() (responsável pelo cálculo da semelhança) por algum motivo retorna valores inesperados quando ela lê o número de vezes que uma palavra aparece na querry, portanto a consulta por cosine ranking não foi adequadamente implementada.