Integrantes:

- Lucca Augusto
- Matheus Marchesotti
- Tiago Denicoli

Link do Github: https://github.com/LuccaAug/GoogleHumilde/

Introdução

Os sistemas de buscas são ferramentas extremamente úteis quando consideramos a quantidade enorme de informação presente na internet, por isso são acessados diariamente por grande parte da população mundial. Além disso, a Google, uma das maiores empresas do mundo, se estabeleceu com a criação de um buscador de sites, sendo hoje provedora de um uma quantidade enorme de serviços variados.

A ideia básica do sistema de buscas desse trabalho é retornar um subconjunto de documentos que correspondem a uma busca feita, sendo que o conjunto é ordenado para melhor responder os interesses do usuário. O código foi feito utilizando principalmente a linguagem C++, com a utilização de Python em apenas um script.

Para executar o programa, basta rodar a seguinte linha:

g++ -std=gnu++14 main.cpp dataset.cpp indice_invertido.cpp ranking.cpp dataset.teste.cpp indice_invertido.teste.cpp ranking.teste.cpp

Os documentos a serem utilizados estão no diretório */documentos/* e dentro dessa pasta temos os arquivos para busca que é um subconjunto dos baixados do dataset fornecido no Moodle, em */documentos/* e os para teste, em */teste/*.

Implementação

Primeiramente, o programa cria uma lista do nome dos documentos de buscano diretório */documentos/documentos/* e um para os de teste em */documentos/teste/*, a partir do script de python “cria_arquivo.py”.

Depois, é instanciado um Dataset que possui um método que lê cada uma das linhas desse documento e chama outro método que lê o documento relativo ao nome escrito na lista. Então as palavras presentes no documento são adicionadas a um vector de strings , apenas uma vez por palavra, após submetidas ao tratamento de strings por um método do dataset. E paralelamente um vector de vector de pairs é responsável por armazenar o id da palavra (índice do vector externo) documentos que possuem aquela palavra (índice do vector interno), id do documento que possui esta palavra (pair->first) e quantas vezes este documento a possui (pair->second).

Após estes passos é instanciada uma variável da classe Indice_Invertido, e se chama um método desta classe que tem como objetivo criar um arquivo .txt que é nada mais que uma matriz, composta por linhas de documentos e colunas de coordenadas da palavra daquele índice no documento daquela linha.

E por fim, é instanciada uma variável da classe Ranking e é chamado um método dela que tem por objetivo criar um vetor de coordenadas para a query assim como o índice invertido faz para cada documento, calcular o cosine ranking, e mostrar os top 10 documentos relacionados aquela pesquisa.

Como nos testes tivemos que modificar muito os arquivos para que fosse um programa testável, durante o processo de criação dos testes foi documentado o jeito encontrado pela equipe, explicando um pouco melhor cada função, classe e método.

Testes

O jeito encontrado para testar o programa foi calcular o índice invertido manualmente para um dataset pequeno, composto de apenas dois documentos apresentados nesta tabela:

Título do Documento	Texto do Documento
A.txt	Documento teste! PDS, Pds
B.txt	2: Teste docu-mento teste

Assim teremos um dataset cujo o vector relacao_ da classe dataset seria assim:

Índice da Palavra	Índice das suas ocorrências	Id do Documento	Quantidade de vezes que aparece neste documento
Vector (externo)	Vector (Interno)	Pair→first	Pair→second
0	0	1	1
0	1	2	1
1	0	1	1
1	1	2	2
2	0	1	2
3	0	2	1

E o vector palavras_ também da classe dataset estariam assim:

Índice da Palavra	String da palavra em si
0	documento
1	teste
2	pds
3	2

Com isso em mente, sabemos os seguintes dados:

- O método Le_lista() itera sobre um documento auxiliar que contém os nomes de todos os arquivos utilizados e chama o método Le_documento() para cada um
- O método Le_documento() faz o dataset propriamente dito.
- O método Formata_palavra(string palavra) retorna uma string sem qualquer caractere especial, além de converter as letras maiúsculas em minúsculas.
- O método Contar_palavras() deve retornar 2.
- O método Numero_de_Documentos deve retornar 2.
- O método Quantos_documentos_possuem_essa_palavra(int idPx) deve retornar 2 para o valor 0, 2 para o valor 1, e 0 para qualquer outro valor.
- O método Quantas_vezes_dj_possui_Px(int dj, int idPx) deve retornar os seguintes valores:

int dj	int idPx	retorno
1	0	1
1	1	1
1	2	2
1	3	0
2	0	1
2	1	2
2	2	0
2	3	1
Valor que não está na tabela	Valor que não está na tabela	0

E assim podemos efetuar o teste da classe dataset e utilizar tais valores para as classes dependentes dela.

Já no cálculo do Índice Invertido deste dataset temos que os vetores de coordenadas dos documentos são o seguinte:

Palavra	documento	teste	pds	2
A.txt	0	0	2*log(2)	0
B.txt	0	0	0	log(2)

Com isso em mente, sabemos os seguintes dados:

- O método Arquivo_Indice_Invertido() cria um txt cujo cada linha é um documento, já na ordem dos ids da struct Documento, e cada coluna é a coordenada daquela palavra.
- O método W(int dj, int idPx) recebe o id do documento e o id da palavra e calcula a coordenada, neste caso só retorna valores diferentes de 0 para (dj = 1, idPx = 2) que é 2*log(2) já que a palavra tem importância log(2) e aparece 2x no documento, e (dj = 2, idPx = 3) que é log(2) já que a palavra aparece apenas uma vez e tem importância log(2)
- O método idf(int idPx) recebe o id da palavra e calcula a importância dela naqueles documentos, como “documento” e “teste” aparecem em todos os documentos, tais strings tem importância 0, e “pds” e “2” tem importância log(2)
- O método Calc_coordenadas_consulta() calcula os W da query, como se ela fosse um documento, por isso os documentos começam no id 1, já que o id 0 foi reservado para a query e retorna o ponteiro de um vetor com estas coordenadas.

Conclusão

Esse projeto se apresentou como um desafio para todos os integrantes do grupo e, justamente por isso, serviu de grande aprendizado. Após as aulas, qualquer impressão de que já estávamos sabendo o conteúdo apresentado e como utilizar as ferramentas foi derrubada rapidamente, logo quando surgiram os primeiros problemas de sintaxe, testes, seguidos pelos problemas relacionados ao Github e de depuração, mostrando que tínhamos muito mais dúvidas do que respostas.

Após passar por todos os desafios apresentados, é gratificante observar que aprendemos como resolver diversos problemas que provavelmente aparecerão novamente durante nossa jornada na ciência da computação, além de ter fixado bem o conteúdo de testes, POO, STL, modularização e etc. Além disso, foram desenvolvidas outras habilidades imprescindíveis para um profissional, principalmente o trabalho em equipe e a capacidade de pensar em como resolver bugs e onde procurar ajuda, seja em algum fórum ou outro colega. Particularmente, foi interessante aprender que todas pessoas podem acrescentar alguma coisa, sendo ela um conhecimento específico, um atalho que facilita programar ou um ponto de vista diferente sobre o código e a lógica.