Trabalho Prático: Máquina de Busca

Thiago Ferreira de Noronha

19 de setembro de 2019

1 RESUMO

O aluno deve usar a carga teórica e as habilidades em programação para desenvolver um sistema de consulta para armazenar dados e retornar informações requisitadas via queries. Para tanto, serão fornecidos conceitos necessários às buscas e as formas as quais os dados devem ser armazenados e apresentados.

2 INTRODUÇÃO

O processo de **Recuperação de informação** é o processo de conversão de dados, dispostos de forma estruturada ou não, em uma lista real de citações a documentos em um arcevo contendo informações válidas para a necessidade em questão [1]. Como mostra a Figura 2.1, uma chamada **Máquina de Busca** (MB) pode ser vista como um programa que recupera informações em uma base de dados. A MB recebe como entrada uma **expressão de busca** (*queries* ou consultas), processa a mesma e dá como saída os documentos os quais são, de acordo com a metodologia adotada, mais relevantes para a consulta fornecida.
De acordo com [2], documento é o termo genérico que designa os objetos portadores de informação. Um documento é todo artefato que representa ou expressa um objeto, uma ideia ou uma informação por meio de signos gráficos e icônicos (palavras, imagens, diagramas, mapas), sonoros e visuais (gravados em suporte de papel ou eletrônicos). Para que haja uma maior eficiência no processo de recuperação, eles precisam estar armazenados de uma forma que o retorno seja eficiente. Para tanto, utilizam-se tipos abstratos de dados para que a representação seja aquela que mais condiz com a utilidade do sistema.
O trabalho prático pede ao aluno a criação de um **índice invertido** para armazenar dados fornecidos na entrada e a implementação de uma MB para recuperar documentos associados à consultas, também fornecidas como entrada.

3 REPRESENTAÇÃO DE UM DOCUMENTO

Existem várias formas de armazenar documentos em um computador. Uma boa referência sobre como documentos são representados na memória secundária (um disco rígipo, exemplo) é o livro de Tharp [3].

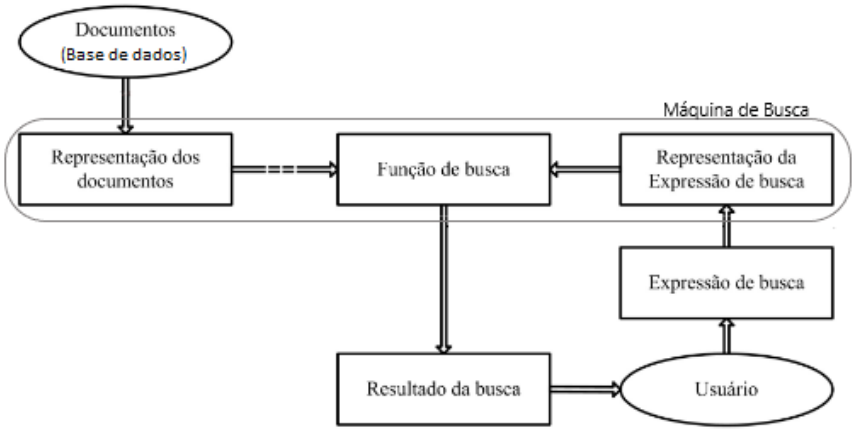

Figura 2.1: Fluxograma que representa uma consulta em uma maquina de busca

Em linhas gerais, um documento pode ser atualizado, inserido ou excluído de uma base de dados. Essas operações possuem custos computacionais, e portanto deve ser escolhida aquela estrutura de dados que melhor otimiza esses custos. O índice invertido é um tipo de estrutura que permite a busca da informação de forma muito rápida, ao passo de que é caro adicionar um novo documento ao mesmo.

3.1 Índice Invertido

Um índice invertido armazena os dados mapeando os termos às suas ocorrências nos documentos que compõem uma base de dados. Cada entrada de um índice invertido é uma determinada palavra e o valor associado a essa entrada é uma lista de documentos em que ela ocorre. Considerando os exemplos abaixos:
Documento "d1.txt":

```
Quem casa quer casa.  Porem ninguém casa.
Ninguém quer casa também.  Quer apartamento.
```

Documento "d2.txt":

```
Ninguém em casa.  Todos saíram.  Todos. Quer entrar?  Quem?  Quem?
```

Documento "d3.txt":

```
Quem esta no apartamento?  Ninguém, todos saíram.
```

O índice invertido seria o apresentado na Tabela 3.1.

Tabela 3.1: Índice invertido resultante.

Vocabulário	**Documentos**			**Vocabulário**	**Documentos**		
apartamento	d1.txt		d3.txt	no			d3.txt
casa	d1.txt	d2.txt		porem	d1.txt		
em		d2.txt		quem	d1.txt	d2.txt	d3.txt
entrar		d2.txt		quer	d1.txt	d2.txt	
esta			d3.txt	sairam		d2.txt	d3.txt
ninguem	d1.txt	d2.txt	d3.txt	tambem	d1.txt		
todos		d2.txt	d3.txt				

4 FUNÇÃO DE BUSCA

A função de busca tem um papel fundamental em atribuir pesos e ordenar os documentos de acordo com as suas respectivas relevâncias. Para este trabalho prático, foi adotado o **modelo vetorial** para representar os documentos. Nesse modelo, cada palavra (consequentemente consultas e documentos) possui um peso associado a ela, de tal forma que é possível representá-la através de um vetor geométrico. Assim, temos que o número de palavras distintas da coleção determina a dimensão do espaço que os documentos e consultas estão representados. Isso quer dizer que, se o vocabulário tem n palavras distintas, o número de dimensões é n ($\mathbb{R}^n$). Parte do trabalho prático é determinar as coordenadas de uma palavra P_x.

4.1 DETERMINAÇÃO DAS COORDENADAS DE UMA PALAVRA

Sejam tf e idf a quantidade de vezes (frequência) que uma palavra P_x aparece em d_j e a **importância** de P_x em d_j, respectivamente. O segundo fator é computado pela Equação 4.1.

$$idf(x) = \log\left(\frac{N}{n_x}\right), \tag{4.1}$$

onde N é o número de documentos e n_x é a quantidade de documentos que P_x apareceu. As coordenadas de P_x correspondente ao documento d_j é definida pela Equação 4.2:

$$W(d_j, P_x) = tf(d_j, P_x) \times idf(P_x), \tag{4.2}$$

Onde $W(d_j, P_x)$ é a coordenada do documento d_j no eixo P_t, $tf(d_j, P_t)$ é a frequência da palavra P_t no documento d_j e $idf(P_t)$ é a importância de P_t na coleção.
Uma questão ainda deve ser tratada: **dada duas palavras P_a e P_b pertencentes a um documento d_j, como definir quem é mais importante do que a outra em uma dada consulta?**

4.2 COSINE RANKING

O método de *Cosine Ranking* ou *Cosine Similarity* será utilizado para definir qual palavra vem primeiro na ordenação. Dada uma consulta q_i, o método nos permite determinar qual dos documentos d_j está mais próximo da consulta. A similaridade é dada pelo cosseno $cos(\theta)$ do par (d_j, q_i), descrita pela Equação 4.3:

$$sim(d_j, q) = \cos(\theta) = \frac{\sum_{i=1}^{t}(W(d_j, P_i) \times W(q, P_i))}{\sqrt{\sum_i W(d_j, P_i)^2} \times \sqrt{\sum_i W(q, P_i)^2}} \tag{4.3}$$

Suponha um vocabulário com apenas 3 palavras A, B e C e uma coleção com 4 documentos representados pela Figura 4.1. Os valores de $W(D_1, A)$, $W(D_1, B)$ e $W(D_1, C)$ resultam em $\vec{D}_1 = (0,84, 0,69, 0)$ (Figura 4.2). Agora, supondo que um usuário fez a consulta Q_1 composta por A e B, temos que $W(Q_1, A)$, $W(Q_1, B)$ e $W(Q_1, C)$, resultam nos valores apresentados na Figura 4.3. E portanto o vetor $\vec{Q}_1 = (0,28, 0,69, 0)$. Fazendo os mesmos cálculos para $W(D_2, A)$, $W(D_2, B)$, $W(D_2, C)$, $W(D_3, A)$, $W(D_3, B)$, $W(D_3, C)$, $W(D_4, A)$, $W(D_4, B)$ e $W(D_4, C)$, obtemos os resultados expressos na Figure 4.3
Em posse de todos os valores de $W(d_j, P_i)$ podemos aplicar a formula de similaridade (calculo do cosseno), para cada palavra e posteriormente somar os resultados encontrados.

$$sim(\vec{D}_1, A) = 0,24,\ sim(\vec{D}_2, A) = 0,16,\ sim(\vec{D}_3, A) = 0,16\ e\ sim(\vec{D}_4, A) = 0$$

D1	A A A B
D2	A A C
D3	A A
D4	B B

Figura 4.1: Documentos e suas respectivas palavras.

$$w(Dl,A) = idf(A) \times tf(Dl,A) = 0{,}28 \times 3 = 0{,}84$$
$$w(Dl,B) = idf(B) \times tf(Dl,B) = 0{,}69 \times 1 = 0{,}69$$
$$w(Dl,C) = idf(C) \times tf(Dl,C) = 1{,}38 \times 0 = 0$$

Figura 4.2: Valores das coordenadas.

$$w(Q,A) = idf(A) \times tf(Q,A) = 0{,}28 \times 1 = 0{,}28$$
$$w(Q,B) = idf(B) \times tf(Q,B) = 0{,}69 \times 1 = 0{,}69$$
$$w(Q,C) = idf(C) \times tf(Q,C) = 1{,}38 \times 0 = 0$$

Figura 4.3: Valores das coordenadas da consulta Q_1.

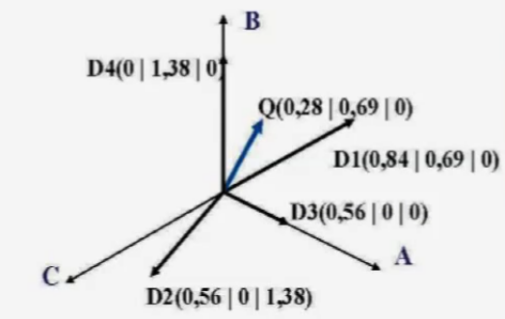

Figura 4.4: Plano contendo todos os vetores envolvidos.

$$sim(\vec{D}_1, B) = 0{,}47,\ sim(\vec{D}_2, B) = 0{,}0,\ sim(\vec{D}_3, B) = 0{,}0\ e\ sim(\vec{D}_4, B) = 0{,}95$$

Portanto, a similaridade dos documentos com $\vec{Q}_1$, são:

$$sim(\vec{D}_1, \vec{Q}_1) = 0{,}71,\ sim(D_2, \vec{Q}_1) = 0{,}0,\ sim(D_3, \vec{Q}_1) = 0{,}0\ e\ sim(D_4, \vec{Q}_1) = 0{,}95$$

E o ranking seria dado por: (Tabela 4.1)

Tabela 4.1: Ranking.

Ranking	
Posição	Documentos
1	D_4
2	D_1
3	D_2 e D_3

5 O QUE DEVE SER ENTREGUE

O trabalho para ser considerado completo deve conter no mínimo todas as partes descritas abaixo:

5.1 LEITURA DE ARQUIVOS (1 PONTO)

O sistema recebe como entrada um conjunto de arquivos de texto, que devem ser lidos, palavra após palavra, para construir o índice invertido. Detalhes:

- Você pode assumir que os arquivos contém apenas caracteres que são letras (a-z e A- Z), números (0-9), e caracteres de pontuação (",", ".", "?", etc.).
- Você pode assumir que o texto não tem acentos nem "ç".
- Após ler cada palavra, você deve:
  - i Transformar todas as letras maiúsculas em minúsculas;
  - ii Apagar todos os caracteres que não são letras ou números. Por exemplo, depois de ler "Guarda-Chuva?", você deve transformá-la em "guardachuva", antes de inseri-la no índice invertido. Desta forma, a mesma palavra apresentada com letras minúsculas ou maiúsculas, ou que estão adjacentes a pontuação, não serão diferenciadas.

5.2 ESTRUTURAS DE DADOS PARA ARMAZENAR AS COORDENADAS DOS DOCUMENTOS (6 PONTOS)

Um índice invertido deverá ser implementado para que as coordenadas de um documento sejam armazenadas. Aqui, a criatividade pode ser usada para implementarem utilizando a teoria e quaisquer ferramentas dispostas da linguagem adotada, desde que sejam estruturas apresentadas em sala de aula.

5.3 CONSULTAS ATRAVÉS DE UM RANKING (6 PONTOS)

Para as consultas, o aluno deve impementar o *cosine raking* de forma a ordenar as suas respectivas saídas. Para tanto, o programa desenvolvido deve imprimir em tela a lista ordenada dos K's documentos mais próximos das mesmas, tal qual explicado na subseção 4.2.

5.4 TESTES DE UNIDADE (8 PONTOS)

O aluno deve executar testes unitários para os métodos implementados. Os testes devem seguir as instruções dadas em sala de aula, devendo ter uma cobertura mínima de 75% do código. Portanto, é incentivado para que o máximo número de casos sejam cobertos.

5.5 DOCUMENTAÇÃO E ENTREGA (9 PONTOS)

A documentação não deverá exceder o limite de 10 páginas, sendo submetida no formato PDF juntamente com o código fonte via GitHub. Cada aluno deve criar seu próprio repositório público no site e enviar o link junto com a submissão do pdf contendo a documentação via moodle. A documentação deverá contemplar pelo menos os 3 tópicos:

- Introdução;

- Implementação: com uma explicação clara e objetiva de como o problema que foi resolvido, justificando os algoritmos e as estruturas de dados utilizadas. Para auxiliar nessa atividade utilize pseudocódigos, diagramas e demais figuras que achar conveniente. Não é necessário incluir trechos ou mostrar detalhes de código da sua implementação, exceto quando esses influenciam no seu algoritmo principal.

- Conclusão.

O trabalho deve ser entregue até o **dia 22 de novembro de 2019**.

Desejamos a todos um bom trabalho.

REFERÊNCIAS

[1] C. N. Mooers, "Zatocoding applied to mechanical organization of knowledge," *American documentation*, vol. 2, no. 1, pp. 20–32, 1951.

[2] Y.-F. LE COADIC and A. C. da Informação, "Brasília," 2004.

[3] A. L. Tharp, *File Organization & Processing*. John Wiley & Sons, 1988.