Trabalho Prático: Máquina de Busca

Thiago Ferreira de Noronha
Lucas Victor Silva Pereira

02 de Abril de 2019

1 INTRODUÇÃO

Recuperação de informação - (information Retrieval) é o nome dado ao processo ou método pelo qual um potencial usuário de informação é capaz de converter a sua necessidade de informação em uma lista real de citações a documentos em um acervo contendo informações úteis para ele.[1]
Uma maquina de busca são robôs capazes de recuperar a informação dentro de uma base de dados especifica, dado uma consulta (*query*) feita por um usuário. O seu funcionamento é dado pela necessidade que o usuário tem de uma determinada informação. Ele então expressa essa necessidade para o sistema na maioria das vezes através de um conjunto de *palavras-chave*, que denominamos de consulta. Com a consulta montada o sistema de buscas vai até a base de dados de documentos e tenta recuperar as respostas que são mais relevantes para a consulta especificada. O fluxograma abaixo, que será explicado detalhadamente, representa a arquitetura de um sistema de busca explicado acima.

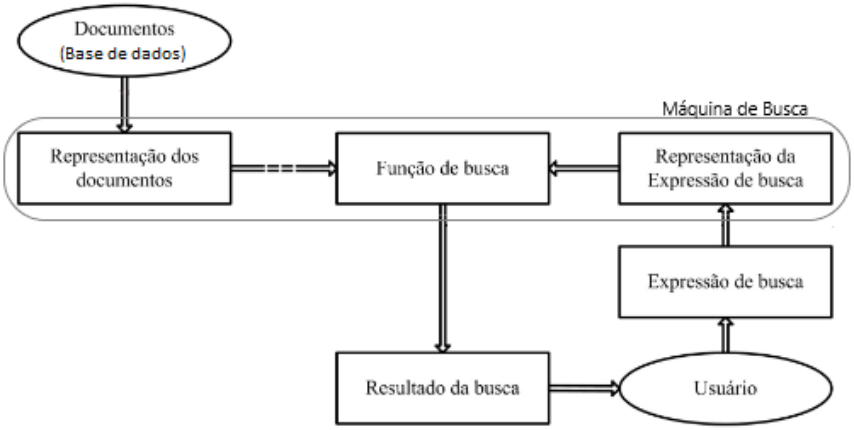


Figura 1.1: Fluxograma que representa uma consulta em uma maquina de busca

2 DOCUMENTOS

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, figuras, símbolos), sonoros e visuais (gravados em suporte de papel ou eletrônicos).[2]
Neste trabalho a maquina de busca a ser implementara dará suporte apenas a documentos textuais, que irão formar a *base de dados* onde o buscador realizará as consultas determinas pelos usuários.

2.1 REPRESENTAÇÃO DOS DOCUMENTOS E BUSCA

O modelo de representação a ser utilizado neste trabalho é o *modelo vetorial.* Este tipo de sistema vê seus componentes (documentos e consultas) como um junto de *palavras-chave,* mais especificamente como vetores, onde é salvo o peso de cada termo (cada palavra do vocabulário). Em um modelo vetorial as coordenadas de um vetor qualquer são determinadas pelas palavras que o descrevem[1].Desta forma, temos que o número de palavras distintas da coleção determina a dimensão do espaço onde os documentos e consultas serão representados, ou seja, se o vocabulário tem n palavras distintas o número de dimensões é n ($\mathbb{R}^n$).

2.1.1 ÍNDICE INVERTIDO

Um modo comum de representação dos documentos que facilita a vida os robôs de busca é a criação de um *índice invertido* para a coleção de documentos que representa a base de dados. Dada a coleção de documentos, um índice invertido é uma estrutura contendo uma entrada para cada palavra (termo) que aparece em pelo menos um documento. Para cada palavra, o índice invertido armazena a lista de documentos que a contém. Por exemplo, dados os seguintes documentos:
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:

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				

[1] Um vetor é sempre descrito pela sua norma, direção e sentido.

3 Função de Busca

Pela seção 2 sabemos que tanto os documentos quanto as consultas são representados por vetores que relacionam os termos aos seus respectivos pesos, ou seja, cada dimensão de um vetor representa o peso de uma das n palavras do vocabulário utilizado. Supondo um documento d_j qualquer, se uma determinada palavra P_x não esta contida nele, então no vetor que representa d_j na posição (dimensão) de P_x seu peso é zero, mas dado duas palavras P_A e P_B pertencentes ao documento d_j, como definir os pesos de P_A e P_B? Em outras palavras, como dizer que P_A é mais importante para representar d_j do que P_B ou o inverso?

3.1 Determinação das Coordenadas

Para determinar as coordendas de uma palavra P_x usaremos 2 termos semelhantes o **tf** e **idf**, que são as abreviações de *term frequency* e *inverse document frequency*, que em português significa *frequência do termo* e *inverso da frequência nos documentos*. Da própria definição do nome temos que *tf* é a quantidade de vezes que uma palavra P_x aparece dentro de um determinado documento d_j. Já *idf* expressa a importância de uma palavra P_x dentro da coleção de documentos (o quão significante é o fato de P_x ocorrer em d_j) e é calculada por:

$$idf(t) = \log(\frac{N}{n_t})$$

Onde, N = número total de documentos de uma coleção;
n_t = número de documentos onde a palavra P_t ocorreu;
Obs.: o termo log na expressão é apenas para suavizar o resultado. Se a ocorrência da palavra P_t for um número muito menor que a quantidade de documentos o idf seria um número muito grande.

O que está expresso na formula matemática de *idf* é que o peso de uma palavra P_x é inversamente proporcional a quantidade de vezes que ela aparece na coleção de documento. Isso porque quanto mais vezes uma palavras aparece, menos importância ela terá para definir o assunto tratado na consulta.
Dado *tf* e *idf* podemos determinar as coordenadas de um vetor correspondente a um documento da base de dados utilizamos a seguinte equação:

$$W(d_j, P_t) = tf(d_j, P_t) \times idf(P_t)$$

Onde, $w(d_j, P_t)$ 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;
$idf(P_t)$ é a importância de P_t na coleção.
Obs.: Geralmente tf é um termo normalizado, onde a frequência do termo P_t no documento d_j é dividido por $max_{1<j<m}\ tf(d_j, P_t)$, onde m é o número de documento contidos na coleção, ou seja, $tf(d_j, P_t)$ é dividido pela máxima frequência de qualquer termo dentro de um documento.

3.2 Cosine Ranking

Para *rankear* os documentos da base de dados com relação a consulta especificada por um usuário, podemos utilizar o método de *cosine ranking* já que o sistema de busca é baseado em vetores o documento mais relevante dada uma consulta q_i é o documento que esta mais próximo do vetor q_i. Sendo assim podemos calcular a similaridade dos vetores d_j com relação ao vetor q_i calculando o *cosseno* entre eles.

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

Suponha um vocabulário com apenas 3 palavras A, B e C e uma coleção com 4 documentos, sendo eles:

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

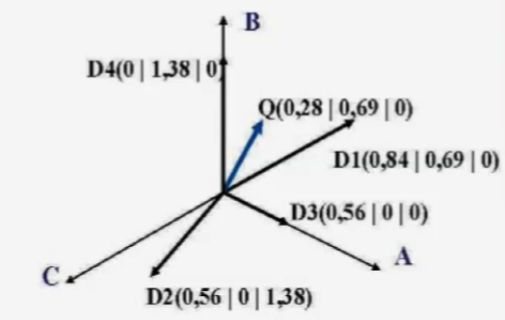
Os valores de $W(D_1, A)$, $W(D_1, B)$ e $W(D_1, C)$, são:

$$w(D1, A) = idf(A) \times tf(D1, A) = 0{,}28 \times 3 = 0{,}84$$
$$w(D1, B) = idf(B) \times tf(D1, B) = 0{,}69 \times 1 = 0{,}69$$
$$w(D1, C) = idf(C) \times tf(D1, C) = 1{,}38 \times 0 = 0$$

Com isso sabemos que o vetor $\vec{D}_1 = (0{,}84, 0{,}69, 0)$
Supondo que um usuário qualquer fez a consulta Q_1 composta por A e B, temos que $W(Q_1, A)$, $W(Q_1, B)$ e $W(Q_1, C)$, são:

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

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 $W(D_4, C)$, temos:



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.

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

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

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:

Ranking	
Posição	Documentos
1	D_4
2	D_1
3	D_2 e D_3

Note que a consulta Q pode ter varias palavras, mas não pode ser uma bigrama[2] ou trigrama, como lua cheia e banco de dados por exemplo. Caso fosse permitido consultas com essas propriedades o modelo de ranqueamento seria muito mais complexo.
Obs.: O mapa de coordenadas pode ser escrito como uma *função hash perfeita* (livre de conflitos). Imagine o mapa de coordenas como o índice invertido, ou seja, como uma tabela onde a linhas são os documentos e consultas e as colunas são as palavras. Desta forma a função hash perfeita é

$$h(X, Y) = X * largura_{max} + Y$$

Onde, X é posição da palavra P_x;
Y é a posição do documento d_y;
$largura_{max}$ é o número de palavras que o vocabulário utilizado tem.

4 O QUE DEVE SER ENTREGUE

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

4.1 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.

[2] Sequência de duas palavras consecutivas que possuem um significado agregado.

4.2 Estruturas de dados para armazenar as coordenadas dos documentos (6 pontos)

Para implementar a estrutura que armazena as coordenadas dos documentos contidos na base de dados, que neste nosso caso equivale à implementação do índice invertido, vocês devem utilizar uma estrutura de dados do tipo dicionário (map), onde a função hash perfeita é dada pela formula apresenta na sesão 3.2. Dicionários são estruturas associativas que armazenam elementos combinando uma chave com um valor. A chave é usada exclusivamente para identificar o elemento, que contém um valor mapeado. Podemos fazer uma analogia com a lista telefônica, em que o nome da pessoa é a chave, e o telefone da pessoa é o valor associado ao nome. Neste trabalho, as chaves são todas as palavras contidas nos documentos e o valor associado a cada chave é o conjunto (set) com os nomes dos documentos.

4.3 Consultas através de um Ranking (6 pontos)

Nesta parte do trabalho você utilizará o **Ranking Cosseno** (*Cosine Ranking*) que faz uso da **estrutura que armazena as coordenadas dos documentos** para rankear consultas utilizando uma base de dados textual. Dada uma consulta (*query*) especifica criada por um usuário qualquer, o seu buscador deve imprimir em tela a lista ordenada dos $K's$ documentos mais próximos desta *query* (do documento mais similar para o menos similar), assim como foi explicado na subseção 3.2

4.4 Testes de unidade (8 pontos)

Desenvolvam testes unitários para os métodos que forem criados seguindo as instruções dadas em sala. Tentem cobrir todos os casos que vocês imaginarem. A taxa de cobertura mínima de código deve ser de 75%.

4.5 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 grupo 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 de código da sua implementação e nem mostrar detalhes da sua implementação, exceto quando esses influenciam no seu algoritmo principal.
- Conclusão.

O trabalho deve ser entregue até o **dia 13 de Junho 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.