MAC499 - Trabalho de Formatura Supervisionado

Recuperação de Informações em Bancos de Dados Textuais

IME-USP

Aluna: Marcela Ortega Garcia

Orientador: Prof. Dr. João Eduardo Ferreira

Introdução

Um dos principais objetivos da área de banco de dados é armazenar e recuperar dados de maneira eficiente. O crescimento do uso de bancos para dados sem estrutura definida e o aumento do volume de dados desencadeou a necessidade de novas técnicas.

A Recuperação de Informação (RI) é uma área que trata da representação, armazenamento, organização e acesso a informações [1]. No caso de bancos de dados textuais, uma técnica muito utilizada é chamada de busca indexada. Nesse método, índices que representam os documentos são previamente extraídos dos textos e consultados no momento da busca.

A proposta deste trabalho de formatura é estudar o processo de recuperação de informações em banco de dados textuais. Para atingir esse objetivo, integramos a biblioteca de indexação e busca *Ferret* [2] ao sistema do Centro de Estudos do Genoma Humano [3].

Fundamentos

Antes de iniciar um processo de recuperação é necessário definir a coleção de textos. Cada texto deve ser submetido a operações gerando uma *visão lógica* do mesmo, a partir da qual serão construidos os índices.

Tendo todos os índices prontos, podemos inciar o processo de RI. Para isso, o usuário deve descrever as palavras que serão utilizadas como parâmetros de busca e essa especificação também é submetida às operações textuais, criando uma consulta. Essa consulta pode ser reformulada pelo sistema visando melhores resultados ou enviada diretamente para o próximo passo. Com os termos obtidos, é realizada então a busca por documentos e posteriormente eles são ranqueados de acordo com sua relevância.

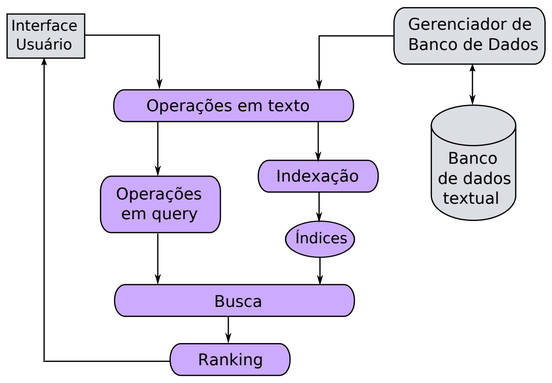

Figura 1: Processo de recuperação de informação

Operações em texto

Na linguagem escrita algumas palavras carregam mais significado e representam melhor o conteúdo de um texto que outras. Além de ser útil para encontrar possíveis textos relevantes, pré processar o texto melhora o desempenho da busca já que reduz o tamanho do vocabulário utilizado como índice.

- **Análise léxica:** identificação de palavras que serão utilizadas como índices.
- **Eliminação de *stopwords*:** palavras frequentes em quaisquer textos são consideradas inúteis como índices. *Exemplos:* o, a, um, uma, de, para.
- ***Stemming*:** aplicar operações em uma palavra para encontrar sua raíz gramatical. *Exemplo:* "recuperar" é raíz de "recuperação" e "recuperado".

Indexação

Com as palavras obtidas pelas operações descritas acima, podemos criar o índice da coleção. A estrutura mais utilizada é a de arquivos invertidos: um mecanismo de indexação orientado à palavra com estrutura composta pelo vocabulário e suas ocorrências [4].

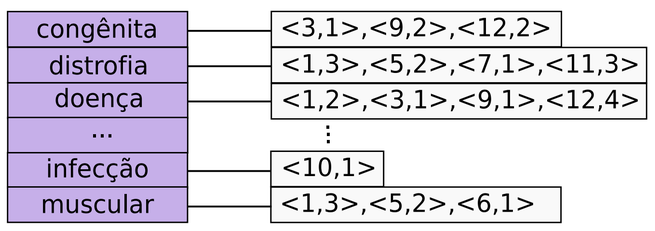

Figura 2: Arquivo invertido

As listas de ocorrências são sequências de pares $< d, f_{d,t} >$, nos quais t representa o termo, d o documento e $f_{d,t}$ a frequência do termo t no documento d.

Modelo Vetorial

A partir dos índices, é possivel calcular o grau de relevância dos documentos em relação à pesquisa do usuário e assim, ordená-los. Para o cálculo, atribui-se pesos a cada termo, isto é, um valor numérico $w_{i,j}$ que indica o grau de relevância do termo k_i no documento d_j. Caso o termo não esteja presente, $w_{i,j} = 0$.

No modelo vetorial, o peso do termo k_i em relação ao documento d_j é calculado por meio da fórmula

$$w_{i,j} = f_{i,j} \times \log \frac{N}{n_i}$$

na qual, $f_{i,j}$ é a frequência normalizada do termo k_i no documento d_j, N é a quantidade de documentos da coleção e n_i o número de documentos em que o termo k_i aparece.

Já o peso em relação à consulta q é definido por:

$$w_{i,q} = (0.5 + 0.5 f_{i,q}) \times log \frac{N}{n_i}$$

A partir dos pesos, a similaridade de um documento d_i em relação a uma consulta q é calculada como a correlação entre os vetores $\vec{d_j}$ e $\vec{q}$ definidos abaixo:

$$\vec{d_j} = (w_{1,j}, w_{2,j}, ..., w_{t,j})$$
$$\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{t,q})$$

$$sim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \times |\vec{q}|}$$

Estudo de caso: CEGH

O Centro de Estudos do Genoma Humano possui um sistema *Web* desenvolvido na linguagem *Ruby* por meio do *framework Rails*. Existem dados, tais como observações sobre um paciente e anotações realizadas durante as consultas, que são registrados em campos do tipo texto e não possuem estrutura definida. A partir deles, surgiu a necessidade da busca indexada.

Utilizando a biblioteca *Ferret*, uma *search engine* baseada no *Lucene* e escrita em *Ruby*, obtivemos resultados significativos.

Resultados obtidos

Seguem abaixo dois *screenshots* do sistema. No primeiro, um exemplo na página de busca e, no segundo, o resultado para a consulta "fendas AND parciais".

Figura 3: Página de busca do sistema do CEGH.

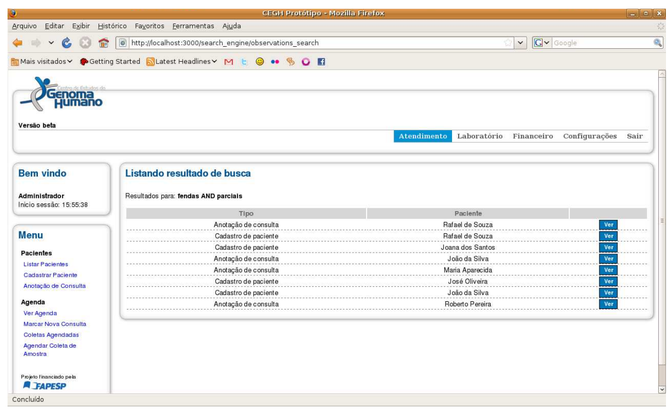

Figura 4: Resultados obtidos na busca.

Com a integração da busca indexada ao sistema do CEGH, realizamos dois tipos de testes comparativos a consultas SQL. Os resultados podem ser visualizados na figura 5.

1. Busca por palavra rara na coleção;
2. Busca por palavra frequente na coleção;

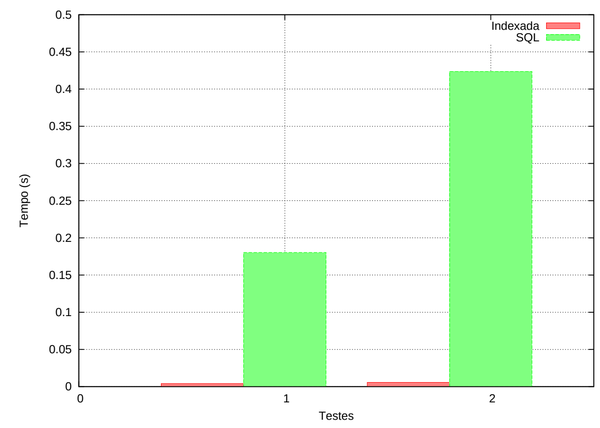

Figura 5: Indexação x SQL

Conclusão

Com este trabalho, concluimos que a utilização de índices é imprescindível para a busca textual. Na figura 5 podemos notar a significativa melhora de desempenho. Podemos observar também que quanto mais frequente a palavra, mais demorada será a consulta SQL.

Além da avaliação de desempenho, é importante salientar que utilizando os conceitos de RI é possível calcular o grau de relevância de documentos. Esse cálculo não está naturalmente disponível com a utilização de SQL.

Referências

[1] R. Baeza-Yates, B. Ribeiro-Neto, *Modern information retrieval.*
[2] Ferret `http:www.davebalmain.com`
[3] CEGH `http:www.genoma.ib.usp.br`
[4] J. Zobel and A. Moffat. *Inverted files for text search engines.*