Como funções de similaridade atuam em diferentes domínios: um estudo de casos

Com o passar do tempo os sistemas de busca foram se popularizando e aprimorando, com respostas que facilitam a tomada de decisão do usuário, fornecendo um conjunto de opções supostamente mais relevantes. Este processo é conhecido como Recuperação de Informação.

“Recuperação de Informação é encontrar materiais (normalmente documentos) de natureza não-estruturada (normalmente texto) que satisfaçam uma informação necessária em grandes coleções (normalmente armazenadas em computadores).” (Manning, Raghavan, Schütze, 2008, p.1)

Por décadas, utilizou-se sistemas de recuperação de informação em ambientes restritos com usuários específicos utilizando alguma linguagem de consulta. No começo da década de 90, coincidentemente com o advento da Internet, sistemas com consultas em linguagem natural tiveram maior destaque.

Não seria prático executar tais buscas em todos os documentos toda vez que uma informação for requerida, tornando-se um processo inviável dependendo do tamanho da coleção, por exemplo, com o número de páginas na Internet este processo seria extremamente lento. Portanto, criar um índice dos documentos de antemão permite executar a busca mais rapidamente, ao custo do armazenamento e manutenção deste.

Uma forma de fazer tais buscas é por meio do Modelo de Recuperação Booleana (do inglês, *Boolean Retrieval Model*), o qual permite consultas com expressões booleanas, utilizando termos e operadores lógicos como AND, OR e NOT para conectá-los. Com esse modelo, o usuário consegue encontrar precisamente que documentos dentro da coleção satisfazem a sua busca.

Cada documento possui um conjunto de termos nele presente, que o distingue de outros, e para cada termo que se busca, existem documentos em que esse está presente. Assim, a forma mais natural de criar esse índice é uma matriz de incidência, no qual cada posição pode ser 1 ou 0, se o documento contém o termo ou não. Essa matriz, portanto, é criada levando-se em conta todos os termos presentes na coleção. Para saber quais documentos contém certo termo, consulta-se a linha da matriz correspondente ao termo.

Para uma determinada coleção de documentos, o conjunto de termos presentes é chamado de vocabulário ou léxico e a estrutura de dados que guarda o vocabulário é chamada de dicionário.

Existe aqui uma diferença entre a informação necessária, que é aquilo que o usuário visa encontrar de fato, e consulta, que é o que o usuário fornece ao sistema. Um documento é dito relevante se contém informações que o usuário julgue compatíveis com o que lhe é necessário. Podemos medir a eficiência do sistema quanto a uma busca de duas formas: sua

precisão, ou quantos dos documentos retornados são relevantes; e seu *recall*¹, ou que parcela dos documentos relevantes do sistema foi retornada.

Tendo uma consulta de dois termos “ t_1 AND t_2 ”, o modelo de recuperação booleana considera as linhas da matriz de incidência como dois números binários, e aplica o operador. Portanto, nesse modelo, temos um baixo *recall*, pois o sistema filtra os documentos a partir da *consulta*, não tendo flexibilidade para incluir outros que poderiam ser relevantes. É um modelo muito limitado, tendo em vista as necessidades dos usuários.

A matriz de incidência dos termos é, em geral, extremamente esparsa. Se tivermos documentos de 1000 palavras, mas um dicionário com 10^6 termos, teremos cada linha da matriz com no máximo 0,1% das posições com 1.

Entretanto, existe uma estrutura mais compacta chamada Lista Invertida (do inglês, *inverted index*), que indica para cada termo do dicionário os documentos em que esse está presente. Cada registro nesta lista é chamado de postagem (do inglês, *posting*). Para cada termo temos uma lista de postagens.

Para gerar o índice temos os seguintes passos:

1. Reunir os documentos que farão parte do índice.
2. Criar *tokens*² para as palavras criando uma lista para cada documento.
3. Normalizar estes *tokens* por meio de um processo linguístico. Os *tokens* normalizados serão os termos indexados (o dicionário).
4. Atribuir uma identificação (número inteiro em série, por exemplo) para cada documento e atribuí-los às listas dos termos que neles aparecem.

Depois, ordena-se o dicionário em ordem alfabética, ocorrências repetidas de um mesmo termo em um documento são mescladas e entradas repetidas de um mesmo termo no dicionário são agrupadas. O produto final é composto pelo dicionário de termos e suas postagens. O dicionário pode ser armazenado na memória, dependendo de seu tamanho, enquanto as postagens serão lidas do disco.

Essa estrutura de dados pode ser usada para guardar estatísticas, como o número de documentos em que um termo aparece (que é também o comprimento da lista de postagens). Essas estatísticas podem ser usadas para ranquear os resultados de uma busca de forma mais eficiente.

As listas de postagens podem ser feitas com diferentes estruturas de dados, como vetores e listas ligadas. Vetores de tamanho fixo são pouco eficientes, pois podemos ter listas de postagens de tamanhos muito discrepantes, o que resultaria em desperdício de espaço. Com vetores de tamanho variável, perde-se tempo apenas no redimensionamento, logo, se o índice não for muito atualizado, este pode ser útil. Por sua vez, listas ligadas precisam de mais espaço por causa dos ponteiros.

No modelo de recuperação booleana, tendo essas listas ordenadas pelas identificações dos documentos, é fácil de processar buscas. No caso do operador AND, basta recuperar as listas de dos dois termos e selecionar os termos presentes nas duas

¹ Não foi encontrada nenhuma tradução para esse termo na literatura da área.

² Esse termo não foi traduzido por ser mais comumente utilizado.

listas. Já que as listas estão ordenadas podemos avançar intercaladamente entre elas. Sendo n_i o tamanho de cada uma das listas, esta operação tem $O(n_1+n_2)$ comparações. Para processar a consulta inteira, pode-se fazer uma operação de cada vez. Assim, para uma consulta genérica, a complexidade da busca é $\Theta(N)$, sendo N o tamanho do dicionário, que na prática é uma constante imensa.

Uma forma de otimizar o processamento da consulta é mudar a ordem em que as operações são feitas, ordenando os termos da consulta pelo tamanho de suas listas de postagens. Nenhum resultado parcial será maior do que a menor das listas utilizadas até ali, logo, começando pelas operações com as listas menores, o tamanho dos resultados parciais será sempre menor ou igual a menor das listas utilizadas na consulta.

Existem outros modelos de recuperação de informação, como modelos de recuperação ranqueada (do inglês, *ranked retrieval models*), no qual a consulta submetida pelo usuário assume formato livre e o sistema deve responder a essas consultas. Um exemplo é o modelo de espaço vetorial contrastando com o uso de operadores.

Para ranquear os resultados de uma busca é necessário algum sistema de pontuação. Um documento que menciona um termo da consulta mais vezes é mais interessante, logo, deve receber um pontuação maior. Essa pontuação de um termo t em um documento d será chamado de peso de t em d . Pensando em consultas de texto livre, que é uma forma mais comum na *web*, seria fácil computar a pontuação de um documento, sendo a soma dos pesos dos termos da consulta presentes no documento. A forma mais natural de atribuir este peso é pela frequência do termo, denotada $tf_{t,d}$, mas pode-se usar qualquer função de ponderação. Essa forma é conhecida como modelo saco-de-palavras (do inglês, *bag of words model*) que leva em consideração apenas o número de ocorrências dos termos, mas não a ordem destes. Mesmo que a ordem das palavras divirja entre dois documentos, se as frequências dos termos forem similares, esses devem ser similares.

Alguns termos podem ser muito frequentes na coleção, logo, não terão muita influência no ranqueamento dos documentos, sendo necessária algum tipo de atenuação. Utilizando a frequência do termo na coleção, um termo muito presente em poucos documentos e um termo pouco presente em muitos documentos teriam a mesma frequência, então, a atenuação não surtiria efeito. Portanto, é comum usar a frequência de documentos (df_t), ou seja, o número de documentos da coleção em que o termo está presente. Assim, definimos frequência inversa do documento (idf):

$$idf_t = \log \frac{N}{df_t}$$

Um termo que está em todos os documentos terá $idf = 0$. Quanto mais raro o termo na coleção, maior será o seu idf . Multiplicando $tf_{t,d}$ por idf_t temos um balanceamento entre a presença de um termo em um documento e a sua raridade na

coleção. Um documento que diz muito sobre um assunto que aparece pouco na coleção deve ser mais relevante para este assunto. Definimos então o balanceamento tf-idf:

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \cdot \text{idf}_t$$

Trabalhando com o documento como um vetor de pesos para os termos do dicionário, temos a pontuação de um documento d em relação à uma consulta q :

$$\text{Score}(q,d) = \sum_{t \in q} \text{tf-idf}_{t,d}$$

Esta representação de documentos como vetores em um espaço vetorial é conhecida como modelo vetorial (do inglês, *vector space model*) e pode ser usado em outras operações de recuperação de informação. Além disso, outras funções de ponderação podem ser usadas no lugar de tf-idf. Para cada documento d temos, então, um vetor $\overline{V}(d)$ com componentes sendo as ponderações. Assim, o espaço vetorial terá um eixo para cada termo.

Dois documentos serão similares se tiverem vetores com componentes proporcionalmente parecidas, mesmo que as componentes de um documento sejam maiores do que a do outro. Para computar esta similaridade, é comum o uso do cosseno entre os dois vetores:

$$\text{sim}(d_1, d_2) = \overline{V}(d_1) \cdot \overline{V}(d_2) / |\overline{V}(d_1)| \cdot |\overline{V}(d_2)|$$

Temos a divisão do produto vetorial entre os vetores sobre o produto de seus comprimentos euclidianos. Esta divisão é necessária para normalizar os resultados. Se os vetores já estiverem normalizados, apenas o produto vetorial é necessário.

A coleção será representada pelo conjunto destes vetores: uma matriz $M \times N$, sendo M o número de termos do dicionário e N o número de documentos. Da mesma forma que um documento, uma consulta pode ser representada por um vetor. Sendo assim, a pontuação de um documento em relação a uma consulta será:

$$\text{score}(q,d) = \overline{V}(q) \cdot \overline{V}(d) / |\overline{V}(q)| \cdot |\overline{V}(d)|$$

Além do uso do cosseno para o cálculo da similaridade, outras funções podem ser usadas. Busca-se, hoje, determinar de maneira mais formal a influência destas funções em diferentes domínios.