Construindo uma máquina de busca

Material de referência:

- [1] Ricardo Baeza-Yates e Berthier Ribeiro-Neto. “Recuperação de Informação: Conceitos e Tecnologia das Máquinas de Busca”, 2ª edição, Bookman, 2013.
- [2] Christopher D. Manning, Prabhakar Raghavan e Hinrich Schütze. “Introduction to Information Retrieval”, Cambridge University Press, 2008.

A utilidade de um sistema de busca de documentos é evidente: todos os dias utilizamos algum buscador web para localizar alguma página contendo informação que desejamos conhecer. Mas como esses serviços funcionam?

A área de conhecimento que lida com o desenvolvimento de máquinas de busca é chamada *recuperação de informação* (em inglês: *information retrieval*). De acordo com Baeza-Yates e Ribeiro-Neto [1],

> *“A Recuperação de Informação trata da representação, armazenamento, organização e acesso a itens de informação, como documentos, páginas Web, catálogos online, registros estruturados e semi-estruturados, objetos multimídia, etc. A representação e a organização dos itens de informação devem fornecer aos usuários facilidade de acesso às informações de seu interesse.”*

À lista de fonte de informações supracitadas podemos adicionar várias outras, como: mensagens de aplicativos de comunicação, emails, feeds de notícias, forums online, e demais fontes de texto. Além disso podemos considerar a miríade de fontes de documentos multimedia, como imagens, audios, vídeos, diagramas, partituras musicais, etc.

Qualquer massa substancial de informação rapidamente demanda alguma forma de organização para que o usuário localize aquilo que lhe interessa. Considere um livro: quantos artefatos de organização de informação estão presentes?

- A estruturação do texto em capítulos e seções;
- Capa descritiva;
- Números de página;
- Legendas de tabelas e figuras;
- Índice de capítulos;
- Índice remissivo (para busca por palavras-chave);
- Rótulos de destaque de informação (como painéis de destaque);
- O prefácio do livro geralmente apresenta uma visão geral deste;
- Referências bibliográficas;
- Um marcador de página para que nos recordemos de onde estávamos!
- E muitos outros artefatos.

Todos esses ítens não são informações extras do livro, mas sim informações acerca do livro, para facilitar a localização de informação.

O universo da informação digital é muito mais vasto do que um livro, e demanda uma solução compatível com sua natureza peculiar:

- Grande diversidade de documentos, de naturezas diferentes.
- Pouca estrutura em documentos.
- Principalmente documentos de texto.
- Documentos em mudança constante.

Máquinas de busca são uma alternativa de solução para o problema de organização da informação textual digital, e são a forma prefencial de localização de informação na Web. Outras alternativas existiram: por exemplo, o buscador web Yahoo começou como uma lista categorizada de páginas que compunham o *bookmark* de seu fundador! Mas rapidamente as máquinas de busca provaram-se superiores para tratar da complexidade crescente da Web.

Em um cenário de busca de informação bem-estruturada a solução mais direta para o problema de recuperação de informação é o uso de um sistema gerenciador de banco de dados tradicional. Por exemplo, para localizar os clientes de um banco que tem saldo maior que R$1.000,00 podemos consultar um banco de dados (hipotético) com a seguinte *query*:

```
SELECT idCliente FROM contas WHERE saldo > 1000;
```

Neste contexto sabemos exatamente como a informação está estruturada, e como localizar os números de identificação de clientes com os saldos especificados. Mas considere agora o contexto da Web, e as buscas:

"melhores modelos de berços de bebê"

"rio que cruza a Alemanha e a Austria"

"preço do dolar em 1/1/1990"

"meme da mulher e do gato"

E agora, como proceder? Como organizar a informação de modo a localizar as melhores páginas que discutem mobília para crianças, localizar elementos geográficos, consultar dados financeiros, e piadas da moda?

Anatomia de uma máquina de busca

Em uma máquina de busca podemos encontrar os seguintes elementos:

- Crawler
- Indexador
- Buscador
- Ranqueador
- Processador de query
- Interface de usuário

Crawler

Um crawler é um sistema que localiza e copia páginas da Web. Crawlers geralmente operam de modo recursivo:

- O crawler começa com uma lista de links a serem consultados
- Para cada link consultado, o crawler extrai os links existentes na página obtida e adiciona estes links ao conjunto de links à visitar, desde que estes não tenham sido visitados antes.

Teoricamente chegará o momento em que todos os links da Web foram visitados e o crawler pode finalmente descansar! Na prática, toda vez que o crawler completa um conjunto completo de links, o número de novos links descobertos supera o número inicial de links!

O crawler envia requisições para os sites da Web, recebe as várias páginas e as armazena de modo bruto em um armazém de páginas. É importante para os sites que estes estejam abertos para a investigação do crawler, senão estes sites não irão aparecer na página do serviço de busca. Porém alguns sites não estão interessados em aparecer nos buscadores, ou podem recear o volume de comunicação (e consequente cobrança financeira do operador de telecomunicações): neste caso o autor do site pode configurar suas preferências em um arquivo intitulado robots.txt, que deve estar acessível para o crawler. Veja https://support.google.com/webmasters/answer/6062608?hl=en para maiores informações.

O diagrama abaixo ilustra um crawler:

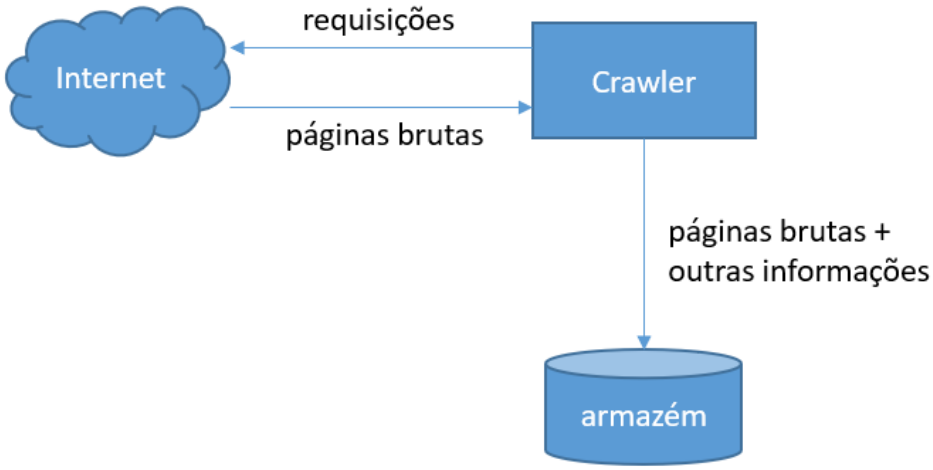


Figura 1: Diagrama ilustrativo de um crawler.

Indexador

O indexador processa as páginas do armazém e armazena o resultado em um repositório de páginas pré-processadas. Mais ainda, cria um índice para localização eficiente de páginas. O diagrama a seguir ilustra o funcionamento de um indexador:

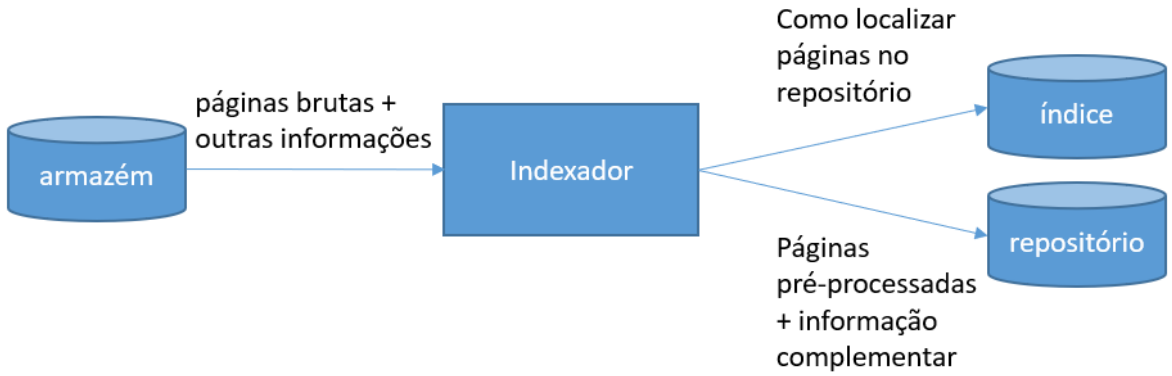


Figura 2: Diagrama esquemático de um indexador.

O pré-processamento das páginas contém as seguintes etapas:

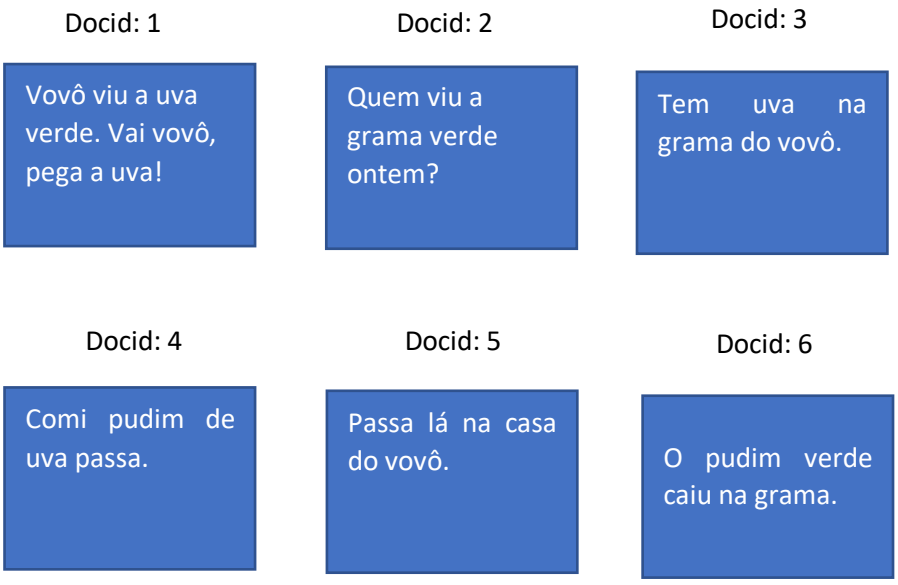
- Tokenização: divide o texto em suas unidades constituintes. Não basta separar por espaço: tem que lidar com pontuação, representação de números, abreviações, etc.
- Eliminação de *stopwords*: algumas palavras não são muito úteis para a tarefa de recuperação de informação, como por exemplo os artigos, preposições, conjunções, etc. Uma palavra que aparece em uma grande maioria de documentos acaba não sendo útil para localizar nenhum documento. Eliminar essas *stopwords* ajuda a manter o tamanho do repositório e do índice menores.
    - Contudo, a remoção de *stopwords* pode ser prejudicial. Considere o exemplo: "ser ou não ser". Claramente o usuário procura algo sobre Shakespeare. Mas com a remoção de *stopwords*, o documento passaria a conter apenas a palavra "ser", removendo quase completamente o significado do texto.
    - Uma alternativa à definição de *stopwords* como palavras de certas categorias sintáticas é definí-las através de suas frequências de ocorrência df_i (número de documentos no qual a i-ésima palavra aparece).
- Normalização de palavras: nesta etapa vamos transformar cada palavra em uma versão padronizada desta. Esta atividade inclui:
    - Tratamento de caixa: converter tudo para minúsculas, por exemplo.
    - *Stemming*: nesta etapa eliminamos os sufixos das palavras, para que a busca não seja sensível à variações sintáticas da mesma palavra, como conjugação verbal ou a formas singulares/plurais. Em inglês, um dos algoritmos mais utilizados para *stemming* é o algoritmo de Porter.
- Identificação de palavras-chave: Nem todas as palavras tem a mesma relevância para a tarefa de recuperação de informação. Alguns sistemas preferem adotar uma seleção de palavras mais relevantes para diminuir o tamanho do repositório e do indice, e acelerar a tarefa de busca. Outros preferem indexar todas as palavras sem distinção.
    - Substantivos normalmente carregam mais informação útil para a busca de documentos do que verbos, adjetivos ou advérbios.
    - Podemos também nos limitar a anotar a categoria verbal de cada palavra e deixar ao módulo ranqueador que use a informação gramatical para melhorar a qualidade do *ranking*. Neste caso não temos nenhum ganho de tamanho no repositório, pelo contrário: mais informação acaba sendo armazenada.

O indexador adiciona as informações resultantes em dois novos armazéns:

- Repositório: contém a parte mais importante do conteúdo de cada uma das páginas fornecidas pelo crawler, assim como informações adicionais como URL da página, lista de links, etc. As páginas são armazenadas no repositório e recebem uma chave de identificação, conhecida como *document id*.
- Indice: permite localizar todas as páginas associadas a cada uma das palavras do vocabulário.

Podemos imaginar que o repositório é um dicionário que mapeia um *document id* para uma lista de palavras. O índice é um dicionário que cumpre a missão oposta: mapeia uma palavra para um conjunto de *document ids*.

Por exemplo: suponha que o crawler buscou os seguintes documentos:



Após a tokenização temos a seguinte estrutura:

```
{
      1: ['Vovô', 'viu', 'a', 'uva', 'verde', '.', 'Vai', 'vovô', ',', 'pega',
'a', 'uva', '!'],
      2: ['Quem', 'viu', 'a', 'grama', 'verde', 'ontem', '?'],
      3: ['Tem', 'uva', 'na', 'grama', 'do', 'vovô', '.'],
      4: ['Comi', 'pudim', 'de', 'uva', 'passa', '.'],
      5: ['Passa', 'lá', 'na', 'casa', 'do', 'vovô', '.'],
      6: ['O', 'pudim', 'verde', 'caiu', 'na', 'grama', '.'],
}
```

Removendo stopwords e pontuação e normalizando o texto temos o seguinte:

```
{
      1: ['vovô', 'uva', 'verde', 'vovô', 'uva'],
      2: ['grama', 'verde', 'ontem'],
      3: ['uva', 'grama','vovô'],
      4: ['pudim', 'uva', 'passa'],
```

```
      5: ['passa', 'casa', 'vovô'],
      6: ['pudim', 'verde', 'caiu', 'grama'],
}
```

O índice, também chamado de *índice reverso*, inverte o dicionário anterior:

```
{
      'caiu': [6],
      'casa': [5],
      'grama': [2, 3, 6],
      'ontem': [2],
      'passa': [4, 5],
      'pudim': [4, 6],
      'uva': [1, 3, 4],
      'verde': [1, 2, 6],
      'vovô': [1, 3, 5],
}
```

Máquina de busca

A máquina de busca recebe uma requisição de busca (uma *query* ou consulta) e localiza os documentos mais interessantes do repositório para esta consulta. Na figura abaixo temos um exemplo de aplicação construída sobre uma máquina de busca.

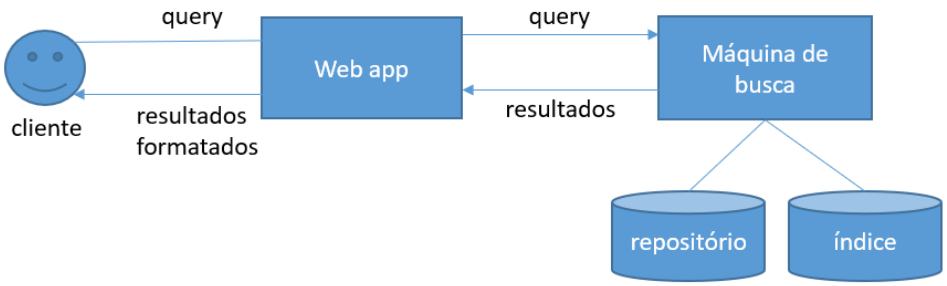


Figura 3: Diagrama esquemático simplificado da operação de uma máquina de busca.

Internamente, a máquina de busca consiste em três módulos:

- Entendimento de query
- Buscador
- Ranking

A figura abaixo ilustra os detalhes internos da máquina de busca:

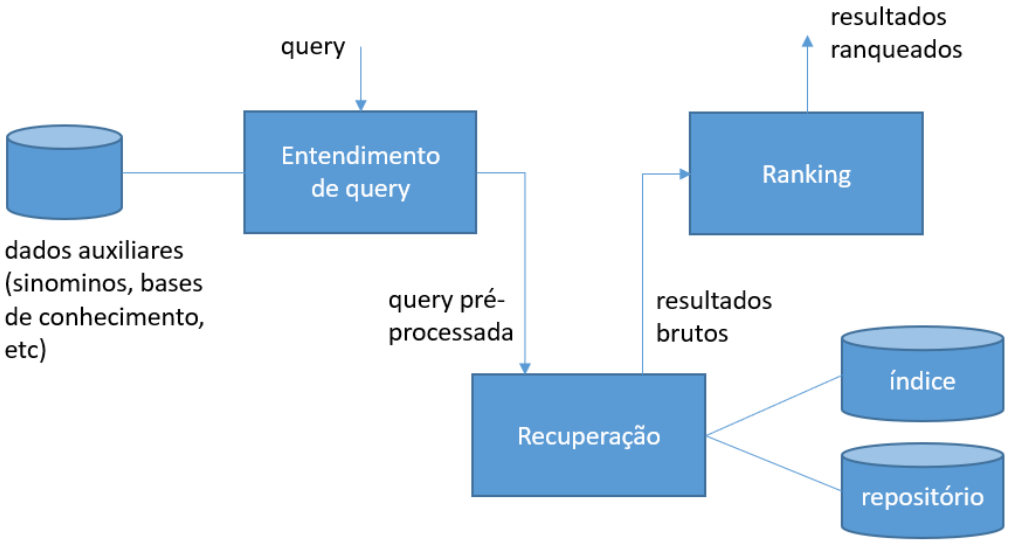


Figura 4: Vista detalhada de uma máquina de busca

Entendimento de query

Nesta etapa a query é modificada para adicionar informação relevante para a recuperação de documentos e para o ranqueamento.

Buscador (Recuperação)

Para uma lista de palavras definindo a busca a ser realizada, esta etapa recupera os documentos que importam, sem ordená-los por importância.

Por exemplo, suponha que a query seja “pudim verde”, e que o indice é como mostrado na seção anterior (repetido aqui por conveniência):

```
{
      'caiu': [6],
      'casa': [5],
      'grama': [2, 3, 6],
      'ontem': [2],
      'passa': [4, 5],
      'pudim': [4, 6],
      'uva': [1, 3, 4],
      'verde': [1, 2, 6],
      'vovô': [1, 3, 5],
}
```

Quais documentos satisfazem essa query? A palavra “pudim” aparece nos documentos 4 e 6. Já a palavra “verde” aparece nos documentos 1, 2 e 6. Fazendo a intersecção destes conjuntos de *document ids*, vemos que o único documento que satisfaz essa busca é o documento 6.

Já se a busca for mais permissiva “pudim OU verde” todos os documentos que contenham alguma destas palavras devem ser considerados – neste caso os documentos 1, 2, 4 e 6.

Ranking

Neste módulo devemos ordenar os documentos por relevância, do mais relevante para o menos relevante.

Uma estratégia clássica de determinação de relevância para uma busca é a estratégia *TF-IDF* (*term frequency – inverse document frequency*).

- *Term frequency*: é a frequência (contagem) de ocorrência de um termo em um dado documento. Por exemplo, no documento "Vovô viu a uva verde. Vai vovô, pega a uva!", a frequência do termo "vovô" é 2 (após normalização), e do termo "verde" é 1.
    - Quanto maior o *term frequency*, mais importante é o termo para o respectivo documento.
    - Na prática, costumamos usar o valor $1 + \log_2 f_{i,j}$ onde $f_{i,j}$ é a frequência de ocorrência da palavra i no documento j.
- *Document frequency*: é o número de documentos no qual uma palavra apareceu, dividido pelo número total de documentos. Por exemplo: no corpus de exemplo acima, a palavra "verde" aparece em 3/6 documentos, e a palavra "pudim" aparece em 2/6 documentos.
    - Quanto maior o *document frequency* de uma palavra, menos discriminante ela é para atividades de recuperação de informação.
- *Inverse document frequency*: posto que o document frequency está positivamente associado à irrelevância de uma palavra para a tarefa de buscar documentos, então o inverso da frequência de documentos está associada à relevância desta em recuperação de informação. Na prática não usamos diretamente o inverso do *document frequency*, mas sim o logaritmo deste.

Combinando estes conceitos, chegamos à ponderação TF-IDF:

$$\text{TF-IDF}_j = \sum_{i \in (\text{query} \cap d_j)} \left(1 + \log_2 f_{i,j}\right) \log_2 \left(\frac{N}{n_i}\right)$$

onde:

- $f_{i,j}$ é a frequência de ocorrência do termo i no documento j;
- n_i é o número de documentos em que o termo i aparece;
- N é o número total de documentos
- d_j é o conjunto de palavras do documento j