



uma lista de stop words da língua portuguesa, falada no Brasil. As pontuações **Não** estão presentes nessa lista.

<https://sites.google.com/site/kevinbouge/stopwords-lists>

## 4.1 O problema

A construção de tais índices de forma manual é custoso pois demanda que um especialista/autor leia o texto a marque os termos mais relevante e as suas ocorrências, para então o editor gerar o índice.

Por outro lado, é possível sim encontrar características em um texto que permita classificar uma palavra de acordo com a posição e frequência com que ela aparece no texto. Na Seção 4.7 apresenta-se o TF-IDF, que é uma métrica usada para medir a importância de uma palavra em uma coleção de documentos. O conceito TF-IDF pode ser transportado para o problema do índice remissivo com as páginas fazendo o papel da coleção de documentos.

A resolução dessa problema demandará a criação de índices em memória, que podem ser implementado usando dicionários. Por exemplo, um dicionário estático é a forma mais otimizada para responder se uma palavra que ocorre no texto é uma *stop word* ou não. Observe que as *stop words* da língua podem ser previamente ordenadas, e são poucas as alterações que elas devem sofrer ao longo do tempo.

### 4.1.1 O Estudo

Neste trabalho seu grupo deve implementar um programa que **gera o índice remissivo de um livro** que encontra-se em formato texto. Como mencionado anteriormente, a construção do índice implica no armazenamento das páginas associadas a uma determinada palavra, ou seja, para uma dada palavra deseja-se saber em que páginas ela ocorre. Para facilitar a sua implementação estabeleça o limite de no máximo 5(cinco) ocorrências que deverá estar associada a uma palavra.

Já está feito

**Diferente do índice remissivo tradicional que é impresso o seu índice ficará implementado em memória e deverá responder sempre que consultado sobre a ocorrência ou não de uma palavra.** Caso a palavra faça parte do índice deve-se apontar as páginas aonde ela ocorre, sempre privilegiando as ocorrências com maior significado associado.

## 4.2 Experimento (1-2 páginas)

### 4.2.1 Dicionários estáticos, semi-dinâmicos e dinâmicos

Na Seção 4.6 mostra-se uma métrica que avalia a qualidade da tabela hash construída a partir do agrupamento de chaves nas entradas. Use a métrica apresentada para redimensionar a sua tabela sempre que ela apresentar desempenho ruim. Contabilize o custo dessa operação contando as operações necessárias para reconstruir a sua tabela.

Explique o passo-a-passo para uso dos tipos abstrato implementado.

Ao final da construção do índice, apresente um mapa resumo da tabela, mostrando estatísticas que comprovem a eficiência da tabela construída. Baseie-se no Fator de Carga da sua tabela para derivar as estatísticas. Segue exemplos de estatísticas que podem ser coletadas:

- Taxa de ocupação da hash;
- Eficiência da tabela em relação ao fator de carga, percentagem das posições na tabela hash que obedecem o fator de carga;
- Caracterize a condição de cada entrada em relação ao fator de carga. Exemplos de caracterização inclui indicar a quantidade de entradas que armazenam:
  - Até 10% mais elementos que o fator de carga;
  - Entre 10% e 20% mais elementos que o fator de carga;
  - Entre 20% e 30% mais elementos que o fator de carga;
  - Até 10% menos elementos que o fator de carga;
  - Entre 10% e 20% menos elementos que o fator de carga;
  - Entre 20% e 30% menos elementos que o fator de carga;

#### 4.2.2 A configuração experimental

Mostre como o experimento foi realizado. Use figuras esquemáticas para mostrar os componentes da sua implementação, destacando como eles se conectam. Explique como os TADs devem ser utilizados.

### 4.3 Resultados e interpretações (2-3 páginas)

Mostra o gráfico com o número médio de comparações realizado para responder as consultas realizadas no índice. Isto é, um gráfico de linha que mostre a relação entre obra e o número médio de comparações. As cargas de trabalho serão montadas para explorar diferentes aspectos da implementação do dicionário. É preciso considerar que as comparações são geradas por buscas, inclusões e remoções, portanto é importante que o efeito dessas operações esteja claramente identificados.

### 4.4 Discussão (1/2-1 página)

Discuta os seus resultados. Compare os valores gerados por cada operação e em cada tipo de estrutura. Compare o seu resultado com o que diz a teoria. Em que cenário uma estrutura é mais adequada que outra?

### 4.5 Material e guia experimental

Um conjunto de arquivos texto será indicado para a realização dos estudos.

Veja o link no Colab para a base de dados.

## 4.6 Medindo o nível de agrupamento da tabela

Uma medida para o agrupamento observado na tabela hash é dado pela seguinte equação:

$$C = \sum_i (x_i^2) / n - \alpha \quad (4.1)$$

onde  $x_i$  é uma variável aleatória que captura a distribuição dos elementos nas entradas,  $n$  é o tamanho da instância que será mapeada para a tabela, e  $\alpha$  é o fator de carga esperado, quantos elementos serão mapeados para as entradas na média.

Um valor de  $C = 1.0$  significa que a função está distribuindo as chaves de maneira uniforme. Caso  $C > 1.0$  o agrupamento está segurando o desempenho da tabela. Para um  $C < 1.0$ , a hash está espalhando elementos de forma mais uniforme que uma distribuição Hash randômica. Este último cenário é um caso raro de acontecer, visto que as funções são gerais e operam sob conjuntos de chaves genéricas. A seguir, mostra-se a derivação da Equação 4.1.

Seja  $x_i$  o número de elementos na entrada  $i$ . Para cada um dos  $n$  elementos, pode-se considerar uma variável aleatória  $e_j$ , que receberá um valor 1(um) se tal elemento for mapeado para a entrada  $i$ , com probabilidade  $1/m$ , onde  $m$  é o tamanho da tabela. E  $e_j$  será igual a 0(zero) se o elemento Não for mapeado para a entrada  $i$ . O tamanho da entrada  $x_i$  é uma variável aleatória que é a soma de todas aquelas variáveis aleatórias:

$$x_i = \sum_{j \in 1 \dots n} e_j \quad (4.2)$$

Seja  $\langle x \rangle$  o valor esperado (média) da variável  $x$ , e  $Var(x)$  a variância de  $x$ , que é igual a  $\langle (x - \langle x \rangle)^2 \rangle = \langle x^2 \rangle - \langle x \rangle^2$ . Assim tem-se que:

$$\begin{aligned} \langle e_j \rangle &= 1/m \\ \langle e_j^2 \rangle &= 1/m \\ Var(e_j) &= 1/m - 1/m^2 \\ \langle x_i \rangle &= n \langle e_j \rangle = \alpha \end{aligned}$$

A variância da soma de variáveis randômicas independentes é a soma de suas variâncias. Assumindo-se que  $e_j$  são variáveis randômicas independentes, tem-se então que:

$$\begin{aligned} Var(x_i) &= n Var(e_j) \\ &= \alpha - \alpha^2/m \\ &= \langle x_i^2 \rangle - \langle x_i \rangle^2 \\ \langle x_i^2 \rangle &= Var(x_i) + \langle x_i \rangle^2 \\ &= \alpha(1 - 1/m) + \alpha^2 \end{aligned}$$

Agora, se todas as  $m$  variáveis  $x_i$  forem somadas e divididas por  $n$ , como na Equação 4.1, deve-se efetivamente dividir tal soma por  $\alpha$ :

$$\begin{aligned} (1/n) \sum_i \langle x_i^2 \rangle &= (1/\alpha) \langle x_i^2 \rangle \\ &= 1 - 1/m + \alpha \end{aligned}$$

Subtraindo  $\alpha$ , tem-se  $1 - 1/m$ , que é um valor próximo de 1 se  $m$  for grande, independente de  $n$  ou  $\alpha$ .

Agora, supondo que a função de hash direcionasse os mapeamentos para um em cada  $c$  entradas da tabela. Nesse caso, para as entradas não vazias, teria-se

$$\begin{aligned} \langle e_j \rangle &= \langle e_j^2 \rangle \\ &= c/m \\ \langle x_i \rangle &= \alpha c \\ (1/n) \langle \sum_i x_i^2 \rangle - \alpha &= (1/n)(m/c)(\text{Var}(x_i) + \langle x_i \rangle^2) \\ &= 1 - c/m + \alpha c \\ &= 1 - c/m + \alpha(c - 1) \end{aligned}$$

Se a medida de agrupamento apresenta um valor maior que 01(hum), é provável que a função de hash esteja ignorando uma fração significativa de entradas.

## 4.7 Medindo a importância de uma palavra em um coleção de documentos

TF-IDF é as iniciais de “*Term Frequency, Inverse Document Frequency*”, e é uma maneira de pontuar a importância de palavras (ou “termos”) em um documento baseado em quão frequente ele aparece em diversos documentos. De forma intuitiva...

- Se uma palavra aparece frequentemente em um documento, ela é importante. Atribui-se a palavra uma pontuação alta.
- Mas se a palavra aparece em muitos documentos, ela não é um identificador único. Atribui-se a palavra uma pontuação baixa.

Assim, palavras comuns tal qual “ao” e “para”, que aparecem em muitos documentos, irão ter pontuação reduzida. Palavras que aparecem com frequência em um único documento irão ter pontuação majorada.

As funções que precisam ser implementadas para definir o TF-IDF dos termos são estas:

- `tf(palavra, doc)` computa “frequência do termo” que é o número de vezes que a *palavra* aparece no *doc*, dividido pelo número total de palavras no *doc*. É preciso arrumar uma maneira eficiente de contar as palavras em um documento.
- `n_containing(palavra, docs)` retorna o número de documento que contém a *palavra*.

- $idf(palavra, docs)$  computa “inverse document frequency” que mede quão comum uma palavra é entre todos os documentos em  $docs$ . Quanto mais comum for a palavra mais baixo será o seu  $idf$ . Toma-se a razão entre o número total de documentos e o número de documentos que tem a  $palavra$ , e em seguida toma-se o log dessa razão. Adiciona-se 1 ao divisor para evitar a divisão por zero.
- $tfidf(palavra, doc, docs)$  computa o TF-IDF, que é o produto do  $tf$  e  $idf$ .

## 4.8 Relatório

O relato dos seus resultados deve ser feito usando um arquivo `.md`, que possui marcadores especiais para construção de arquivos textos que podem ser compilados para melhor visualização. No google, use as seguintes palavras para encontrar a documentação que explica como escrever arquivos `.md`: *github markdown portuguese*