Montando Busca TF - IDF

PRÉ-PROCESSAMENTO

PARALELISMO DE DADOS

ARQUIVO .db SQLITE 3
com texto em ascii

⇓ #include <sqlite3.h>

Importar texto p/ programa

⇓

Ler do arquivo "separadores.txt"
e usar os separadores p/ converter o
texto em vetor
Parecido com .split(...)

solução: strtok

⇓

Ler do arquivo "stopwords.txt"
e remover as stopwords.
Filtrá-las do vetor anterior.

⇓ #include <libstemmer.h>

Aplicar a função sb_stemmer_stem()
do SnowballStemmer em cada palavra
do vetor anterior.

BARREIRA →

⇓

Montar a matriz de frequência ou
índice de arquivo invertido.

⇓

Calcular o idf: $\left(\frac{\text{<número de docs>}}{\text{<número de docs com a palavra>}}\right)$

"<word>" : $\log_2 \left(\frac{\text{<número de docs que <word> aparece>}}{} \right)$

⇓

Calcular os vetores dos documentos:

"<docid>" : [tf(<word>, <docid>) * idf[<word>] for word in vocab]

⇓

Calcular as normas dos vetores anteriores

← SALVAR ESTRUTURA RESULTANTE NUM ARQ. BINÁRIO

CONSULTA

Consulta do usuário como argumento do programa:

cat consulta.txt | ./a.out

ou

./a.out "<consulta do usuario>"

⇓

Repetir as etapas de pré processamento. } PARALELISMO DE DADOS

⇓

Calcular similaridade:

PARALELISMO DE DADOS ↓

Para cada docid:

$$sim = \frac{\text{<vetor do docid>} * \text{<vetor da consulta>}}{\text{<norma do docid>} * \text{<norma da consulta>}}$$

Retornar resultados em ordem decrescente de similaridade.