

IMPLEMENTAÇÃO DE UM SISTEMA DE RECUPERAÇÃO EM MEMÓRIA SEGUNDO O MODELO VETORIAL

Esse exercício está dimensionado para ser feito em Python com a biblioteca NLTK. Você pode escolher outra linguagem por sua conta e risco.

O exercício usará a base CysticFibrosis2, disponível no Moodle.

Você deve fazer um sistema de recuperação da informação dividido em módulos especificados a seguir. O sistema poderá funcionar totalmente em memória, usando arquivos para comunicação entre módulos.

Os módulos devem, em geral, seguir o princípio de processamento em batch:

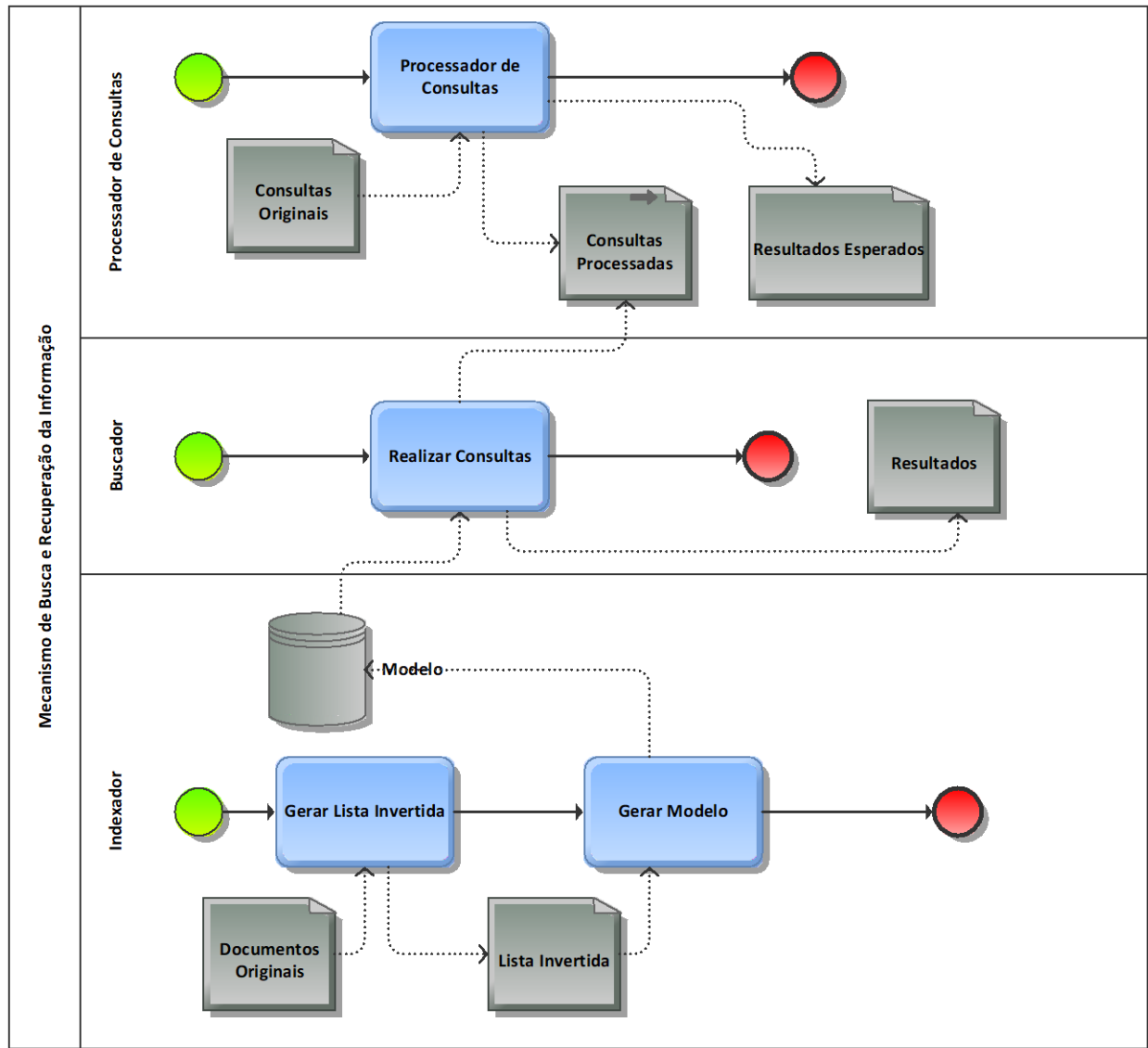
1. Ler todos os dados
2. Fazer todo o processamento
3. Salvar todos os dados

Alguns desses módulos podem ser reusados ou refeitos em exercícios posteriores.

Todos os módulos devem possuir um LOG que permitam pelo menos a um programa posterior, usando o módulo logging de Python:

1. Identificar quando iniciaram suas operações
2. Identificar quando iniciam cada parte de seu processamento
 - a. Ler arquivo de configuração
 - b. Ler arquivo de dados
3. Identificar quantos dados foram lidos
4. Identificar quando terminaram os processamentos
5. Calcular os tempos médios de processamento de consultas, documento e palavras, de acordo com o programa sendo usado
6. Identificar erros no processamento, caso aconteçam

O modelo do sistema a ser implementado é o seguinte:



PROCESSADOR DE CONSULTAS

O objetivo desse módulo é transformar o arquivo de consultas fornecido ao padrão de palavras que estamos utilizando.

- 1) O Processador de Consultas deverá ler um arquivo de configuração
 - a. O arquivo é criado por vocês
 - b. O nome do arquivo é PC.CFG
 - c. Ele contém dois tipos de instruções
 - i. LEIA=<nome de arquivo>
 - ii. CONSULTAS=<nome de arquivo>

- iii. **ESPERADOS**¹=<nome de arquivo>
 - iv. As instruções são obrigatórias, aparecem uma única vez e nessa ordem.
- 2) O Processador de Consultas deverá ler um arquivo em formato XML
 - a. O arquivo a ser lido será indicado pela instrução LEIA no arquivo de configuração
 - i. O formato é descrito pelo arquivo “cfc2-query.dtd”.
 - ii. O arquivo a ser lido é “cfquery.xml”.
- 3) O Processador de Consultas deverá gerar dois arquivos
 - a. Os arquivos deverão ser no formato cvs
 - i. O caractere de separação será o “;”, ponto e vírgula
 - 1. Todos os caracteres “;” que aparecerem no arquivo original devem ser eliminados
 - ii. A primeira linha do arquivo cvs deve ser o cabeçalho com o nome dos campos
 - b. O primeiro arquivo a ser gerado será indicado na instrução CONSULTAS do arquivo de configuração
 - i. Cada linha representará uma consulta
 - 1. O primeiro campo de cada linha conterá o número da consulta
 - a. Campo QueryNumber
 - 2. O segundo campo de cada linha conterá uma consulta processada em letras maiúsculas, sem acento
 - a. A partir do campo QueryText
 - 3. Cada aluno poderá escolher como criar sua consulta
 - c. O segundo arquivo a ser gerado será indicado na instrução ESPERADOS
 - i. Cada linha representará uma consulta
 - 1. O primeiro campo de cada linha conterá o número da consulta
 - a. Campo QueryNumber
 - 2. O segundo campo conterá um documento
 - a. Campo DocNumber
 - 3. O terceiro campo conterá o número de votos do documento
 - a. Campo DocVotes
 - 4. Uma consulta poderá aparecer em várias linhas, pois podem possuir vários documentos como resposta
 - 5. As linhas de uma consulta devem ser consecutivas no arquivo
 - 6. Essas contas devem ser feitas a partir dos campos Records, Item e do atributo Score de Item
 - a. Considerar qualquer coisa diferente de zero como um voto

GERADOR LISTA INVERTIDA

A função desse módulo é criar as listas invertidas simples.

- 1) O Gerador Lista Invertida deverá ler um arquivo de configuração
 - a. O nome do arquivo é GLI.CFG
 - b. Ele contém dois tipos de instruções
 - i. LEIA=<nome de arquivo>
 - ii. ESCRIVA=<nome de arquivo>
 - iii. Podem ser uma ou mais instruções LEIA
 - iv. Deve haver uma e apenas uma instrução ESCRIVA

¹ Na versão anterior do exercício estava “RESULTADOS”, porém mais tarde a palavra ESPERADOS é usada.

- v. A instrução ESCREVA aparece depois de todas as instruções LEIA
- 2) O Gerador Lista Invertida deverá ler um conjunto de arquivos em formato XML
 - a. Os arquivos a serem lidos serão indicados pela instrução LEIA no arquivo de configuração
 - b. O formato é descrito pelo arquivo cfc2.dtd.
 - c. O conjunto de arquivos será definido por um arquivo de configuração
 - d. Os arquivos a serem lidos são os fornecidos na coleção
- 3) Só serão usados os campos RECORDNUM, que contém identificador do texto e ABSTRACT, que contém o texto a ser classificado
 - a. Atenção: Se o registro não contiver o campo ABSTRACT deverá ser usado o campo EXTRACT
- 4) O Gerador Lista Invertida deverá gerar um arquivo
 - a. O arquivo a ser gerado será indicado na instrução ESCREVA do arquivo de configuração
 - b. O arquivo deverá ser no formato cvs
 - i. O caractere de separação será o “;”, ponto e vírgula
 - c. Cada linha representará uma palavra
 - d. O primeiro campo de cada linha conterá a palavra em letras maiúsculas, sem acento
 - e. O segundo campo de cada linha apresentará uma lista (Python) de identificadores de documentos onde a palavra aparece
 - f. Se uma palavra aparece mais de uma vez em um documento, o número do documento aparecerá o mesmo número de vezes na lista
 - g. Exemplo de uma linha
 - i. FIBROSIS ; [1,2,2,3,4,5,10,15,21,21,21]

INDEXADOR

A função desse módulo é criar o modelo vetorial, dadas as listas invertidas simples.

- 1) O indexador será configurado por um arquivo INDEX.CFG
 - a. O arquivo conterá apenas uma linha LEIA, que terá o formato
 - i. LEIA=<nome de arquivo>
 - b. O arquivo conterá apenas uma linha ESCREVA, que terá o formato
 - i. ESCREVA=<nome de arquivo>
- 2) O Indexador deverá implementar um indexador segundo o Modelo Vetorial
 - a. O Indexador deverá utilizar o tf/idf padrão
 - i. O tf pode ser normalizado como proposto na equação 2.1 do Cap. 2 do *Modern Information Retrieval*
 - b. O indexador deverá permitir a alteração dessa medida de maneira simples
 - c. O Indexador deverá possuir uma estrutura de memória deve de alguma forma representar a matriz termo documento
 - d. O Indexador deverá classificar toda uma base transformando as palavras apenas da seguinte forma:
 - i. Apenas palavras de 2 letras ou mais
 - ii. Apenas palavras com apenas letras
 - iii. Todas as letras convertidas para os caracteres ASCII de A até Z, ou seja, só letras maiúsculas e nenhum outro símbolo
 - e. A base a ser indexada estará na instrução LEIA do arquivo de configuração
- 3) O sistema deverá salvar toda essa estrutura do Modelo Vetorial para utilização posterior

BUSCADOR

O objetivo desse módulo é obter os resultados de um conjunto de buscas em um modelo salvo.

- 1) O Buscador deverá ler o arquivo de consultas e o arquivo do modelo vetorial e realizar cada consulta, escrevendo outro arquivo com a resposta encontrada para cada consulta.
- 2) Para isso, usará o arquivo de configuração BUSCA.CFG, que possuirá duas instruções
 - a. MODELO=<nome de arquivo>
 - b. CONSULTAS=<nome de arquivo>
 - c. RESULTADOS=<nome de arquivo>
- 3) A busca deverá ser feita usando modelo vetorial
- 4) Cada palavra na consulta terá o peso 1
- 5) O arquivo de resultados deverá
 - a. Ser no formato .csv
 - b. Separar os campos por “;”, ponto e vírgula
 - c. Cada uma de suas linhas terá dois campos
 - i. O primeiro contendo o identificador da consulta
 - ii. O segundo contendo uma lista Python de ternos ordenados
 1. O primeiro elemento é a posição do documento no ranking
 2. O segundo elemento é o número do documento
 3. O terceiro elemento é a distância do elemento para a consulta

ENTREGA

Os alunos devem entregar em um arquivo ZIP com o nome do aluno (formato <nomedoaluno>.zip”:

1. Todo o código fonte
2. Um arquivo README.TXT com qualquer instrução adicional para uso do código entregue
3. Um arquivo MODELO.(DOC ou TXT) com a descrição do formato do modelo.
4. Todos os arquivos criados por sua execução.
5. O arquivo RESULTADOS.csv