FACOM- UFU
Professor: Wendel Melo

Primeiro trabalho de Organização e Recuperação da Informação 2019-02

Descrição

Este trabalho é subdivido em duas etapas:

1. Implementação de uma rotina para a geração de um índice invertido para uma base de documentos no contexto de um sistema de Recuperação da Informação (RI);
2. Implementação do modelo booleano de RI que deverá utilizar o índice invertido gerado na etapa 1 para responder a uma consulta.

Deve ser entregue apenas um **único** programa desenvolvido em Python que realize as duas tarefas descritas. O trabalho deve ser feito em grupo de **um ou dois alunos,** e o código gerado deve ser entregue por e-mail ao professor (wendelmelo@ufu.br) até a data 22/10/2019.

Aviso importante: se for detectado cópia ou qualquer tipo de trapaça entre diferentes grupos, todos os grupos serão punidos com a nota zero. Portanto, pense bem antes de pedir para copiar o trabalho do seu coleguinha, pois ele poderá ser punido também!

É obrigatório o uso do pacote nltk para a extração de radicais dos termos do vocabulário e obtenção de uma lista válida de *stopwords*. Os detalhes sobre a geração do índice invertido e o processamento das consultas são descritos a seguir. **É importante ler com atenção e seguir todos os detalhes da especificação sob pena de perda de pontos na nota do trabalho!**

A base de documentos

A base de documentos é composta por um conjunto arbitrário de arquivos de texto puro. Assuma que nesses arquivos texto, palavras são separadas por um ou mais dos seguintes caracteres: espaço em branco ( ), ponto (.), reticências(...) vírgula (,), exclamação (!), interrogação (?) ou enter (\n). Seu programa deve tratar caracteres maiúsculos e minúsculos como sendo equivalentes.

As *stopwords*

As *stopwords* são termos que, tomados isoladamente, não contribuem para o entendimento do significado de um documento. Note então que, as *stopwords* **não** devem ser levadas em conta na geração do índice invertido! Seu programa deve considerar a lista de stopwords para a língua portuguesa disponível no pacote nltk, conforme visto em aula. (veja os exemplos da aula do dia 24/09/2019).

As consultas

As consultas a serem respondidas pelo sistema são compostas por termos conectados pelos operadores & (AND), | (OR) e ! (NOT). Assim, o sistema deve ser capaz de responder consultas como as seguintes:

- cão & gato — Leia-se: cão AND gato
- forno | fogão & cozinha — Leia-se: forno OR fogão AND cozinha
- avião | carro & !charrete | navio — Leia-se: avião OR carro AND NOT charrete OR navio

De modo a tornar a sua vida um pouco menos difícil, assuma que as consultas não podem conter parênteses e que o operador ! (NOT) tem precedência sobre & (AND), que por sua vez tem precedência sobre | (OR). Assim, você pode assumir que a consulta sempre é recebida já na forma normal disjuntiva. Portanto, a consulta:

avião | carro & !charrete | navio

pode ser respondida através de três tipos de documentos:

1. Documentos que contenham o termo “avião”;
2. Documentos que contenham o termo “carro” mas não contenham o termo “charrete”;
3. Documentos que contenham o termo “navio”.

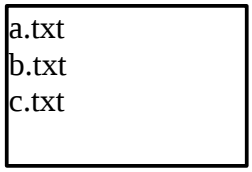
A entrada do programa

Seu programa deverá receber dois argumentos como entrada **pela linha de comando**. O primeiro argumento especifica o caminho de um arquivo texto que contém os caminhos de todos os arquivos que compõem a base, cada um em uma linha. O segundo argumento especifica o caminho de um arquivo texto que traz uma consulta a ser respondida.

Exemplo: Vamos supor que nossa base é composta pelos arquivos *a.txt*, *b.txt* e *c.txt*. Vamos supor também que nosso programa se chama *modelo_booleano.py*. Assim, chamaríamos nosso programa pela linha de comando fazendo:

> python modelo_booleano.py base.txt consulta.txt

onde o arquivo *base.txt* contém os caminhos para os arquivos que compõem a base de documentos (ressalta-se que o arquivo *base.txt* pode conter um número arbitrário de caminhos para os arquivos que compõem a base de documentos, não necessariamente 3), conforme a seguir:



base.txt

, e o arquivo *consulta.txt* possui uma consulta a ser respondida pelo sistema de RI, escrita em uma única linha no formato especificado anteriormente.

```
casa & amor | casa & !mora
```

consulta.txt

Vamos supor que a nossa lista de *stopwords*, obtida do pacote nltk, contenha os seguintes termos:

```
o
de
na
em
não
uma
```

lista de stopwords

A saída do programa

O programa deverá gerar dois arquivos de saída, com nomes e conteúdo exatamente como a seguir:

- *indice.txt* : arquivo que contem o índice invertido gerado a partir dos documentos da base
- *resposta.txt* : arquivo com os nomes dos documentos que atendem a consulta do usuário

O arquivo *indice.txt*:

O programa deve gerar um arquivo chamado *indice.txt*, que contem o índice invertido gerado a partir dos documentos da base. Para a geração do índice invertido, é preciso considerar cada palavra **não *stopword*** que apareça em algum dos documentos da base e **extrair seu radical usando o pacote nltk**. É obrigatório gerar e manter o índice invertido em memória usando alguma estrutura de dados (por exemplo, dicionários).

Para cada um desses radicais no indice, é preciso apontar o número do arquivo em que o mesmo aparece, e a quantidade de vezes em que a mesmo aparece no arquivo. Os arquivos são numerados segundo a ordem em que aparecem no arquivo que indica os documentos da base, que, para o nosso exemplo, foi denominado como *base.txt*. Assim, o arquivo *a.txt* é o arquivo 1, o arquivo *b.txt* é o arquivo 2 e, por fim, o arquivo *c.txt* é o arquivo 3. Suponha que estes arquivos estejam preenchidos conforme abaixo:

```
Era uma CASA muito
engracada. Não tinha teto,
não tinha nada.
```

a.txt

```
quem casa quer casa.
QUEM não mora em
casa, também quer casa!
```

b.txt

```
quer casar comigo, amor?
quer casar comigo,
faça o favor!
mora na minha casa!
```

c.txt

Apenas para facilitar o entendimento, desconsiderando a extração dos radiciais, o arquivo *indice.txt* com o índice gerado seria composto por:

```
amor: 3,1
casa: 1,1 2,4 3,1
casar: 3,2
comigo: 3,2
engracada: 1,1
faca: 3,1
favor: 3,1
mora: 2,1 3,1
nada: 1,1
quem: 2,2
quer: 2,2 3,2
teto: 1,1
tinha: 1,2
```

indice.txt (sem extração de radicais)

Observe que, para cada palavra no índice invertido, temos uma lista de pares *a,q* onde *a* é o número do arquivo em que a palavra aparece, e *q* é a quantidade de vezes em que a palavra aparece no arquivo. Assim, para a palavra casa, por exemplo, temos o par 1,1, indicando que no arquivo 1, este termo aparece uma vez. Em seguida, temos o par 2,4, indicando que no arquivo 2 este termo apareceu 4 vezes. Por fim, temos o par 3,1, indicando que, no arquivo 3, este termo aparece uma vez. **Note que as *stopwords* não devem entrar no índice invertido!** Não é estritamente necessário que o índice invertido seja gerado em ordem alfabética. **É importante que o índice gerado siga exatamente o padrão aqui indicado!** Lembre-se, todavia, que o seu programa deve gerar o índice a partir dos radicais dos termos do vocabulário. Assim, considerando a extração dos radicais, o índice seria composto por:

```
am: 3,1
cas: 1,1 2,4 3,3
comig: 3,2
engraç: 1,1
faç: 3,1
favor: 3,1
mor: 2,1 3,1
nad: 1,1
qu: 2,2 3,2
tet: 1,1
tinh: 1,2
```

indice.txt (com extração de radicais)

Note que palavras que possuem o mesmo radical serão contadas como sendo equivalentes, como no caso de casa e casar.

O arquivo *resposta.txt*

O arquivo *resposta.txt* contém a resposta à consulta contida no arquivo de consulta, no nosso exemplo, *consulta.txt*. A primeira linha desse arquivo deve conter a quantidade de documentos que satisfazem a consulta. As demais linhas contém os arquivos da base que atendem a consulta, um por linha, conforme o exemplo a seguir, onde respondemos a consulta especificada em nosso arquivo *consulta.txt*. Note que será preciso considerar todas as disjunções da consulta, além de desconsiderar *stopwords* presentes na mesma e também extrair os radiciais de seus termos:

```
2
a.txt
c.txt

```

resposta.txt

Para construir a resposta à consulta, seu programa deve se basear exclusivamente no conteúdo do índice invertido gerado! Isto quer dizer que, após a construção do índice invertido, o conteúdo dos documentos da base não deve mais ser diretamente utilizado! Para impressionar seu professor e mostrar que compreendeu a utilidade do índice invertido, procure usar a estrutura do índice gerado em memória para obter o conjunto de documentos que atendem a consulta.