FACOM- UFU
Professor: Wendel Melo

Quarto trabalho de Organização e Recuperação da Informação 2020-01

Descrição

Este trabalho consiste na implementação do modelo de RI vetorial que deverá utilizar a ponderação TF-IDF (com logaritmo na base 10) calculada no trabalho anterior.

Deve ser entregue apenas um **único** programa desenvolvido em Python 3 que realize as duas tarefas descritas. O programa deve usar apenas as bibliotecas padrão Python 3, isto é, as bibliotecas que já vem com a instalação padrão do interpretador da linguagem, com exceção da biblioteca nltk, que deve ser utilizada para remoção de *stopwords* e extração de radicais.

O trabalho deve ser feito **individualmente** e o código gerado deve ser entregue por e-mail (através de arquivo anexado) ao professor (wendelmelo@ufu.br) até a 29/05/2021.

Aviso importante: se for detectado cópia ou qualquer tipo de trapaça entre trabalhos, todos os envolvidos 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 ponderação e o modelo 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!**

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 ou no processamento de consultas! Seu programa deve considerar a remoção de stopwords conforme epecificado no trabalho 1.

As consultas

As consultas a serem respondidas pelo sistema são compostas por termos conectados apenas pelo operador & (AND). Assim, o sistema deve ser capaz de responder consultas como as seguintes:

- samba — Leia-se: samba
- cão & gato — Leia-se: cão AND gato
- carro & roda & motor — Leia-se: (carro AND roda AND motor)

É importante ressaltar que o modelo vetorial prevê ranqueamento de documentos com base na similaridade entre os mesmos e a consultas. Assim, seu programa deve apresentar os documentos na ordem correta, considerando a similaridade entre o documento e a respectiva subconsulta ao qual o mesmo atende.

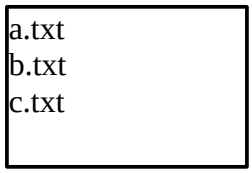
A entrada do programa

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_vetorial.py*. Assim, chamaríamos nosso programa pela linha de comando fazendo:

> python modelo_vetorial.py base.txt consulta.txt

onde o arquivo *base.txt* contém os caminhos para os arquivos que compõe a base de documentos, conforme a seguir:



base.txt

consulta.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.

A saída do programa

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

- *pesos.txt* : arquivo que contém ponderação TF-IDF de cada documento
- *resposta.txt* : arquivo com os nomes dos documentos que atendem a consulta do usuário

O arquivo pesos.txt

O programa deve gerar um arquivo texto chamado *pesos.txt* que contém os pesos de cada termo em cada documento segundo a ponderação TF-IDF (com cálculo de logaritmo na base 10), exatamente como especificado no trabalho anterior. Cada linha desse arquivo deve conter os pesos não nulos dos termos de um dos documentos da base. Por exemplo, considere o exemplo visto em aula, onde supomos que W, X e Y são os termos do nosso vocabulário, já após a remoção de *stopwords* e extração de radicais.

A base de documentos possui o seguinte conteúdo:

Documento	Conteúdo
doc1	W W W X
doc2	W W Y
doc3	W W
doc4	X X

Conforme calculado em aula, os vetores de pesos TF-IDF dos documentos são dados por:

Documento	Vetor de pesos
doc1.txt	(0.1845, 0.3010, 0)
doc2.txt	(0.1625, 0, 0.6021)
doc3.txt	(0.1625, 0, 0)
doc4.txt	(0, 0.3916, 0)

Assim, o conteúdo do arquivo *pesos.txt* será dado por (observe novamente que apenas os pesos diferentes de zero devem ser representados):

```
doc1.txt:  W, 0.1845    X, 0.3010
doc2.txt:  W, 0.1625    Y, 0.6021
doc3.txt:  W, 0.1625
doc4.txt:  X, 0.3916
```

pesos.txt

Note que, para cada documento da base, temos uma lista de pares t,q onde t é o termo, e q é o seu respectivo peso segundo a ponderação adotada. Assim, para o doc2.txt, temos o par W,0.1625. indicando que o termo W tem peso 0.1625 nesse documento, e o par Y,0.6021 indicando que o termo Y tem peso 0.6021 nesse documento. **Ressalta-se que apenas os pesos diferentes de zero devem ser representado no arquivo *pesos.txt.*** Você deve utilizar a ponderação TF-IDF para o cálculo das similaridades no modelo vetorial. Todavia, seu programa deve apenas gravar o arquivo *pesos.txt* sem jamais abri-lo para leitura. Após o cálculo dos pesos, mantenha-os em memória para os cálculos das similaridades. **Os logaritmos devem ser calculados na base 10.**

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, com seu respectivo grau de similaridade, um por linha, conforme o exemplo a seguir, onde respondemos a consulta *"W & Y"*. Note que será preciso desconsiderar *stopwords* presentes na mesma e também extrair os radiciais de seus termos:

```
3
doc2.txt  0.9983
doc3.txt  0.2031
doc1.txt  0.1061
```

resposta.txt

Considere que apenas os documentos com similaridade maior ou igual a 0.001 podem atender à consulta, isto é, 0.001 é o patamar mínimo para um documento ser considerado relevante. (**Não se esqueça de que, junto com cada documento na resposta, é preciso representar sua similaridade**)

Exemplos

Observe os exemplos com termos reais no site. Não deixe de usar o corretor automático para conferir seu código. Assumindo que seu arquivo fonte se chama modelo_vetorial.*py*, basta usar na linha de comando:

```
> python3   waxm_corretor_vetorial.pyc   base.txt  consulta.txt  modelo_vetorial.py
```

ou

```
> py   waxm_corretor_vetorial.pyc   base.txt  consulta.txt  modelo_vetorial.py
```