FACULDADE DE COMPUTAÇÃO

SISTEMAS DE INFORMAÇÃO

GUILHERME DA SILVA LIMA
KELE JUMI FORTUNATO

Geração de um Sistema de Organização e Recuperação da Informação

Monte Carmelo

2021

Geração de um Sistema de ORI

Primeiro trabalho prático da disciplina de Organização e Recuperação da Informação – GSI 512.

Carlos Cesar Mansur Tuma
Professor

Monte Carmelo

2021

Sumário

1. Introdução

A papelaria Difato possui um grande acervo de clientes, produtos e serviços tornando assim bem difícil executar de forma manual as tarefas diárias como os diversos tipos de consultas para efetuar a sua venda de produtos e serviços. O fundador e dono da Difato papelaria resolveu informatizar sua loja para diminuir o tempo de execução das tarefas que antes eram feitas de forma manual. As empresas informatizadas conseguem melhorar o seus processos internos com ganho de tempo e mais praticidade. Para isso a Difato papelaria precisa de um Banco de Dados para armazenar as diversas informações triviais para seu funcionamento.

Para o desenvolvimento deste projeto como um todo resolvemos dividi-lo em duas partes de execução. Na primeira parte fizemos a especificação de um modelo conceitual do banco de dados sendo as atividades:

- Levantamento dos requisitos do sistema e a elaboração de um diagrama de entidade e relacionamento de acordo com esses requisitos
- Aprimoramento do diagrama entidade relacionamento(DER)

Nas próximas seções iremos descreves o levantamento de requisitos e as atividades realizadas com alguns casos de uso com maior importância para o SGBD.

2. Desenvolvimento

Montagem dos Arquivos de Textos

A primeira etapa deste projeto consistiu na criação da pasta *‘textos”* com os textos disponibilizados pelo professor e a montagem manual dos arquivos de textos das stop words e separa-los na pasta *”stopwords*”, para que o programa possa saber o que é uma stop word e o que não é. As outras pastas com nomes sugestivos aos métodos de lematizar, diminutivo e aumentativo, tirar plural contém algumas palavras que serviram de base para comparar os resultados manuais do programa em desenvolvimento com os esperados pelo grupo.

2.1 Entrada

A entrada para este software constitui em 10 arquivos de textos agrupados em duas pastas:

- Textos: 5 arquivos correspondente os textos disponibilizados pelo professor, com o mesmo nome dos arquivos originais.
- Stopwords: 5 arquivos com as respectivas stop words dos arquivos logo acima citados.

2.2 Funções e Estrutura de Dados

Esta aplicação foi desenvolvida em um sistema operacional Windows 10 com as seguintes especificações: 8G ram, 240SSD, i59400F em parceria com notebook assus 8g de ram, 1tby, i5 da colaborado. A linguagem de programação utilizada foi Python 3.9.2 (64-bit) desenvolvida na IDE PyCharm 2020.3.4

2.2.1 Estrutura de Dados:

Esta aplicação utiliza listas de strings e variáveis inteiras.

2.2.2 Funções:

Esta aplicação conta com as seguintes funções:

- **Main:** Apenas Chama as demais funções da aplicação.

- **def leitura_stopwords(caminho):** Recebe o endereço do arquivo txt stop word por parâmetro, faz a leitura deste arquivo e retorna objeto lista, contendo uma lista de string de stop words.

- **def leitura_cancao(caminho):** Recebe o endereço do arquivo txt das canções por parâmetro, faz a leitura deste arquivo e retorna objeto lista, contendo uma lista das palavras da canção.

- **def remove_stopwords(doc,stopwords):** Recebe duas listas de string por parâmetro, e remove os valores da lista stopwords da lista doc através do método *set(list).differrence(list),* converte o retorno deste método numa lista de string novamente e a retorna.

- **def remove_plural(doc):** Recebe uma lista de string por parâmetro, e a percorre removendo as letras s do final da palavra usando o método [:-1] que acessa a ultima posição da string em python, exceto quando a penúltima e antepenúltima forem “i”

e “u” respectivamente pois correspondem a palavra “azuis”, e dela se remove os “is”. No arquivo de canções não foi encontrado nenhuma palavra terminada em s que não fosse um plural. Retorna uma lista sem plural e ordenada.

- **def lematizar(doc):** Recebe uma lista de string por parâmetro, esta função tem como finalidade percorrer a lista de string recebida por parâmetro e nela aplicar algumas regras da lematização. Cada if corresponde a uma regra, e de acordo com as letras finais de cada palavra, ele faz trocas para ajusta-las a regra da lematização. Retorna a lista alterada e ordenada.

- **def aumentativo_diminutivo(doc):** Recebe uma lista de string por parâmetro, esta função tem como finalidade percorrer a lista de string recebida por parâmetro e nela aplicar algumas regras do diminutivos e aumentativos. Cada if corresponde a uma regra, e de acordo com as letras finais de cada palavra, ele faz trocas para ajusta-las a regra da lematização. Retorna a lista alterada e ordenada.

- **def grava_doc(doc,stop,nome_arquivo):**Recebe duas lista de string por parâmetro e o nome do arquivo de texto a ser gravado, esta função diferente das outras acima faz chamada de outras funções (execeto a main por ser a função principal). Esta função grava o arquivo DOC de cada canção após cada alteração, por ela é feito a remoção de stop words com a chamada da função *remove_stopwords(doc,stop)*, e os demais métodos. Grava o arquivo de texto e retorna a lista de palavras tratadas.

- **def gera_dicionario (doc1,stop1,doc2,stop2, doc3 ,stop3 ,doc4 ,stop4 ,doc5 ,stop5 ,doc6,stop6,doc7,stop7):** Recebe todos as listas doc que correspondem as listas de palavras das canções e as listas stop que correspondem as stop words por parâmetro. Esta função tem como finalidade criar uma lista com toda as palavras já tratadas das sete canções sem repetições, essa função trata os arquivos lista recebidos por parâmetros com as chamadas de funções auxiliares que correspondem aos métodos de ORI, após é feita a inserção de palavra por palavra na nova lista, usando a expressão *not in* somente para caso ela não existir na lista para ser inserida. Retorna uma lista de string ordenada.

- **def posting(lista,doc1,doc2,doc3,doc4,doc5,doc6,doc7):** Recebe todos as listas doc que correspondem as listas de palavras das canções por parâmetro e a lista contendo todas as palavras unificadas dos sete docs feita pela função *gera_dicionario().* Percorre a a variável lista recebida por parâmetro e verifica todos os doc se possuem algum item igual através de comparações, variáveis flag e um contador para fazer escrita no arquivo. Escreve o arquivo de texto posting.

- **def dicionario(lista,doc1,doc2,doc3,doc4,doc5,doc6,doc7):** Recebe todos as listas doc que correspondem as listas de palavras das canções por parâmetro e a lista contendo todas as palavras unificadas dos sete docs feita pela função *gera_dicionario().* Percorre a a variável lista recebida por parâmetro e verifica todos os doc se possuem algum item igual através de comparações, variáveis flag para fazer escrita no arquivo. Escreve o arquivo de texto dicionário.