

Health Search – Sistema de busca

Amanda Lima Bezerra

Engenharia da Computação - Universidade Estadual de Feira de Santana (UEFS) Av.
Transnordestina, s/n, Novo Horizonte Feira de Santana – BA, Brasil – 44036-900

limaa9000@gmail.com

Resumo. Visando organizar a busca de documentos, um grande hospital de Feira de Santana contactou os alunos do curso de engenharia da computação da Universidade Estadual de Feira de Santana para criar um sistema de busca usando o índice invertido. O sistema foi feito na linguagem de programação Python, no sistema operacional Windows 11 e usando o editor de código fonte Visual Studio Code.

1. Introdução

Os arquivos de um hospital em Feira de Santana estão espalhados em diversos setores, arquivos esses com exames, laudos e prognósticos de pacientes. Por conta da grande quantidade de arquivos que estão em diversas instâncias do hospital, a busca por documentos não é simples e rápida como deveria ser. Em vista disso, o hospital entrou em contato com os alunos do primeiro semestre de engenharia da computação da Universidade Estadual de Feira de Santana para desenvolver um sistema de busca capaz de localizar palavras chaves contidas nos documentos. Esse software facilitará a localização dos documentos e poupará tempo aos funcionários do hospital.

Analisando o pedido do programa feito pelo hospital, os pré-requisitos propostos são ter o funcionamento baseado em linhas de comando e a criação de um índice no qual todas as palavras presentes nos arquivos devem estar armazenadas juntamente com as informações: em quais arquivos estão presentes e a quantidade da mesma em cada arquivo. Ademais, deve existir as opções de adicionar mais diretórios ao índice, atualizar diretórios já indexados, remover arquivos ou diretórios e por último mostrar o índice por completo.

O índice invertido é uma estrutura de dados usada para mapear e salvar algum conteúdo com a finalidade de uma busca rápida. No caso do projeto proposto o mapeamento é feito dentro do conteúdo dos arquivos, que são arquivos de texto, e o conteúdo que é guardado são as palavras presentes nos arquivos, onde elas estão presentes juntamente com a quantidade de vezes que a mesma aparece em cada arquivo.

Para auxiliar a construção do programa a biblioteca os foi uma sugestão que veio junto com o projeto. Essa biblioteca permite a manipulação de arquivos e diretórios e foi essencial para a construção do software. Por fim, o sistema necessariamente deve ser feito na linguagem de programação Python.

2. Metodologia

A construção do projeto foi feita por etapas, primeiramente foi construído as linhas de comando seguido da função de ajuda. A parte central do código foi construído baseado na organização do índice: criação, salvamento, busca, atualização ou adição e por último a remoção. A ordem do planejamento e execução do software foi feita assim para permitir uma testagem a cada etapa construída, simulando assim como o usuário vai manusear o programa.

2.1 Sessões do PBL

Durante as sessões da disciplina de MI Algoritmos o problema foi discutido a fim de chegar a soluções para os entraves que o projeto trazia. Inicialmente as metas foram direcionadas a entender melhor o que é o índice invertido e quais as funções presentes na biblioteca os do python. Ao longo das semanas os objetivos foram direcionados a manipulação de arquivos de texto e como obter seu conteúdo, alterar ou escrever. Saber como percorrer palavras por palavras de um texto e como filtrar os caracteres foi a base da criação do índice invertido.

Por último, foi durante as sessões que foi consolidado o conhecimento sobre como seria o procedimento para usar o programa, já que ele deve ser exclusivamente por linhas de comandos.

2.2 Ordenação e lógica do código

O primeiro passo da construção do programa foi definir os comandos que o programa aceitaria e a ajuda caso o usuário errasse ao colocar comandos inválidos. Usando a biblioteca sys do python o programa usa somente os dois primeiros argumentos da linha de comando, sendo que cada funcionalidade tem seus próprios argumentos válidos. Qualquer argumento não válido a função ajuda é chamada e apresenta todas as orientações.

A seguir foram definidas as funções de adição, busca, atualização, remoção e mostrar o índice como mostra a Figura 1, sendo que todas elas possuem validações usando *try* e *except* como forma de entregar uma mensagem ao usuário de operação concluída ou erro. Como forma de facilitar o entendimento a lógica por trás de cada código vai ser descrito em forma de pseudocódigo.



Figura 1 - Funções principais do código.

I. Função adição

Parâmetros: diretório

Essa variável é o caminho para o diretório

Criar um dicionário com o nome de cada arquivo de texto como chave e como valor o caminho para esse arquivo

Estrutura do dicionário {'Nome do arquivo.txt': 'Caminho do arquivo'}

Criar o índice invertido em um dicionário

Criado em uma função a parte que será detalhada neste relatório

Salvar o índice em um arquivo

II. Função busca

Parâmetros: palavra buscada

Variável obtida pela linha de comando

Buscar o índice dentro do arquivo

Se a palavra estiver no índice salvar o valor da chave em uma variável chamada matriz:

 Salvar o tamanho da matriz em uma variável

O tamanho da matriz é igual a quantidade de documentos que tem essa palavra

Usando a lógica do *selection sort* para organizar a apresentação do resultado da busca

 Mostrar ao usuário o resultado da busca

Se não:

 Mostrar ao usuário que a palavra não está presente no índice

III. Função atualização

Parâmetros: diretório

Essa variável é o caminho para o diretório

Criar um dicionário auxiliar com o nome de cada arquivo de texto como chave e como valor o caminho para esse arquivo.

Estrutura do dicionário {'Nome do arquivo.txt': 'Caminho do arquivo'}

Criar o índice invertido auxiliar em um dicionário

Buscar o índice que está desatualizado no arquivo

Filtrar qualquer referência aos arquivos a serem atualizados do índice desatualizado

Comparar o índice desatualizado e o índice auxiliar
Adicionar o índice atualizado no arquivo

IV. Função remoção

Parâmetros: diretório ou arquivo

Essa variável é o caminho para o diretório ou arquivo

Salvar a extensão do arquivo em uma variável

Se a extensão for igual a .txt:

Nesse caso é um único arquivo.

Criar um dicionário auxiliar com o nome de cada arquivo de texto como chave e como valor o caminho para esse arquivo. Mas nesse caso o dicionário teria um único elemento.

Se não:

Nesse caso é um diretório.

Criar um dicionário auxiliar com o nome de cada arquivo de texto como chave e como valor o caminho para esse arquivo.

Filtrar qualquer referência aos arquivos a serem removidos do índice.

Adicionar o índice atualizado no arquivo.

V. Função mostrar o índice

Pegar o índice de dentro do arquivo

Percorrer o índice mostrando na tela a chave e o valor de cada elemento no dicionário.

VI. Função ajuda

Mostrar na tela os comandos válidos e orientações de como colocar os diretórios.

As outras funções, Figura 2, dentro do programa auxiliam as funções principais, funções que são chamadas diretamente pelos comandos, mas que são necessárias para o funcionamento correto do software. Todas essas funções são usadas em diferentes momentos no código, por isso não podem ficar restritas dentro das funções principais. A mais importante delas é a criação do próprio índice.



```
def buscando_indice():...  
  
def listagem_arquivos(diretorio):...  
  
def indice_invertido(dicionario_caminhos):...  
  
def filtrar_caracteres(lista_palavras):...  
  
def adicionar_indice_arquivo(dicionario_indice_invertido):...  
  
def comparar_dicionarios(dicionario_indice_invertido_aux, dicionario_indice_invertido):...  
  
def filtrar_arquivos(dicionario_caminhos):...
```

Figura 2 - Funções secundárias.

I. Função índice invertido

Parâmetro: dicionário dos arquivos

Dicionário com os caminhos de cada arquivo

Criar um dicionário vazio para o índice

Estrutura do dicionário para o índice {'Palavra': [['Nome do arquivo', Número da # palavra no texto]...]}

Percorrer o dicionário dos arquivos

Abrir cada um dos arquivos

Filtrar possíveis linhas vazias

Separar as palavras em listas de palavras

Filtrar todos os caracteres

Percorrer a lista de palavras

Se a palavra existe no dicionário do índice:

 Se é de um novo arquivo:

 Colocar as informações em uma nova lista

 Se não:

 Somar no contador

Se não:

 Adicionar a nova palavra no dicionário

2.3 Linhas de comando

A obrigação de ser um código manipulado através de linhas de comando trouxe uma dificuldade inicial ao projeto, visto que foi algo inédito, juntamente com a falta de familiaridade com o prompt de comando, dificuldade essa que foi superada ao longo da produção do sistema. Somente com o suporte da biblioteca sys foi possível organizar e definir todos os comandos válidos no projeto que podem ser vistos na Figura 3.



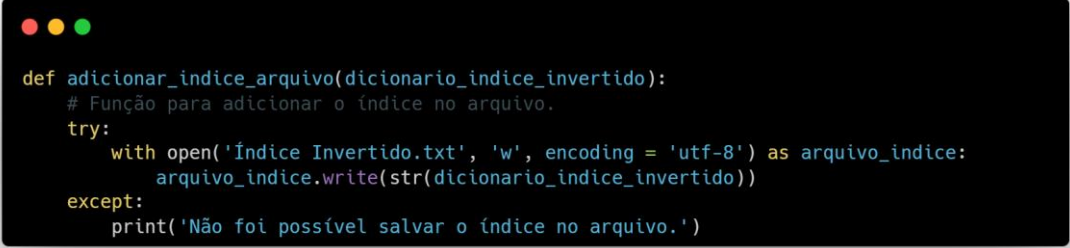
Figura 3 - Comandos válidos.

2.4 Manipulação dos arquivos

O conhecimento de saber manipular arquivos foi a base desse projeto, visto que existiu a necessidade de obter o conteúdo de cada arquivo indicado pelo usuário para a produção do índice. Além disso houve a necessidade de guardar de forma permanente a base de dados da busca.

A principal estrutura usada para todos esses fins, *with*, está na Figura 4. A facilidade em usa-la está justamente na não necessidade de fechar o arquivo depois de

manipulado, isso é feito automaticamente. Os parâmetros dentro da função *open* dependem do objetivo ao codificar, seja somente ler o arquivo, escrever ou acrescentar algo no mesmo. No caso do exemplo na Figura 4 a intenção for escrever por cima, ou seja, independente do que existia antes no arquivo a função vai ignorar e colocar outra informação. Por último *encoding* é usado para fazer com que o computador entenda todas os acentos presentes na língua portuguesa.



```
def adicionar_indice_arquivo(dicionario_indice_invertido):  
    # Função para adicionar o índice no arquivo.  
    try:  
        with open('Índice Invertido.txt', 'w', encoding = 'utf-8') as arquivo_indice:  
            arquivo_indice.write(str(dicionario_indice_invertido))  
    except:  
        print('Não foi possível salvar o índice no arquivo.')
```

Figura 4 – Função adicionar índice.

3. Resultado e discussões

Após concluído a construção do programa houve a testagem e a verificações dos erros, além do mais as análises das entradas válidas. Todas as descrições necessárias para o bom funcionamento do programa serão feitas nesse tópico.

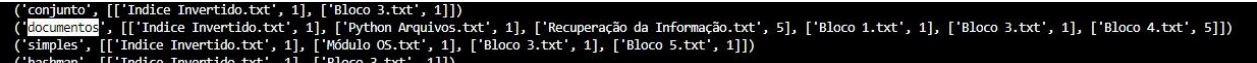
3.1 Dados de entrada e saída

Por ser um programa baseado em linhas de comando nenhuma entrada testada fez o código parar de funcionar, mas as entradas que fazem o programa funcionar estão limitadas aos comandos de ações: adicionar, atualizar, remover, buscar e mostrar. Juntamente com esses comandos de ações deve ser colocado outro parâmetro como o caminho do diretório, a palavra buscada ou índice. Qualquer comando errado as orientações de ajuda aparecem na tela.

Os dados de saída estão limitados ao próprio resultado da busca de uma palavra ou o índice inteiro, fora isso as únicas mensagens que aparecem são mensagens de confirmação, caso a operação tenha sido um sucesso, ou de negação, caso o processo tenha falhado em alguma etapa. A negação mais comum de se aparecer é a inserção de um caminho de um diretório inexistente.

3.2 Testes

Os testes foram limitados ao sistema operacional Windows, inicialmente com arquivos de texto pequenos aonde era possível contar quantas palavras existiam em cada arquivo. Os resultados desses testes estão nas Figura 5 e Figura 6.



```
{  
    'conjunto': [['Índice Invertido.txt', 1], ['Bloco 3.txt', 1]],  
    'documentos': [['Índice Invertido.txt', 1], ['Python Arquivos.txt', 1], ['Recuperação da Informação.txt', 5], ['Bloco 1.txt', 1], ['Bloco 3.txt', 1], ['Bloco 4.txt', 5]],  
    'simples': [['Índice Invertido.txt', 1], ['Módulo OS.txt', 1], ['Bloco 3.txt', 1], ['Bloco 5.txt', 1]],  
    'hashman': [['Índice Invertido.txt', 1], ['Bloco 3.txt', 1]]  
}
```

Figura 5 – Dicionário índice.

```
PS D:\UEFS - 2022.1\MI - Algoritmos\Documentos para PBL 3> python "MI Algoritmos - Health Search.py" buscar documentos
----- PALAVRA BUSCADA -----
ARQUIVOS.....QUANTIDADE
Indice Invertido.txt          1
Python Arquivos.txt          1
Bloco 1.txt                   1
Bloco 3.txt                   1
Recuperação da Informação.txt 5
Bloco 4.txt                   5
```

Figura 6 – Resultado da busca.

3.3 Erros

A conclusão a partir de todos os testes feitos no programa é que ele está funcionando corretamente excetuando a remoção de um único arquivo. Por não conseguir separar o nome do arquivo do endereço do mesmo o programa nomeia o arquivo que deve ser retirado como “Arquivo X”. Acontece que se o usuário decidir retirar somente um arquivo o programa vai analisar os dados da busca procurando pelo nome de Arquivo X.

A única solução viável para o funcionamento da remoção de um arquivo seria renomear o arquivo a deletar do índice com o mesmo nome que o programa salva. O trecho do erro no código está na Figura 7.

```
def remocao(diretorio_arquivo):
    try:
        # Verificando se é uma pasta ou um único arquivo.
        root, ext = os.path.splitext(diretorio_arquivo)
        if ext == '.txt':
            # É um arquivo
            # Como eu não consegui separar o nome do arquivo do caminho eu coloquei como nome
do arquivo 'ArquivoX'
            dicionario_caminhos['ArquivoX'] = diretorio_arquivo
        elif ext == '':
            # É uma pasta
            dicionario_caminhos = listagem_arquivos(diretorio_arquivo)
            # O mesmo procedimento para ambas as situações.
            dicionario_indice_invertido = filtrar_arquivos(dicionario_caminhos)
            adicionar_indice_arquivo(dicionario_indice_invertido)
            return True
    except:
        return False
```

Figura 7 – Trecho do erro.

3.4 Manual de uso

Para um funcionamento correto do programa é necessário ter o arquivo do programa. O programa deve ser aberto pelo prompt de comando do computador em questão, uma vez aberta a pasta que o arquivo do programa está deve ser usado o comando python “MI Algoritmos - Health Search.py” e o comando desejado. É aconselhável que o usuário abra o programa sem nenhum comando para abrir a barra de ajuda e ir se familiarizando com os comandos válidos. Além do mais todos os endereços de diretórios devem ser colocados

entre aspas. Ao cadastrar todas as pastas o arquivo do índice irá aparecer juntamente com o arquivo do programa. Por último é necessário ressaltar que o programa só foi testado no sistema operacional Windows, sendo assim executar o software em qualquer um outro sistema operacional pode resultar em um mal funcionamento.

4. Conclusão

Em resumo, o programa foi finalizado, mas de fato poderia ter sido melhor. Porém o objetivo principal do trabalho que era aprender a trabalhar com arquivos e linhas de comando foi concluído.

5. Referências

“Acervo Lima – Índice invertido” (2019) <https://acervolima.com/indice-invertido/>, acessado em 01 de Junho de 2022.

“os — Diversas interfaces de sistema operacional” <https://docs.python.org/pt-br/3/library/os.html> acessado em 19 de Junho de 2022.

“Como usar o sys.argv para entrada de dados” (2017) <https://pt.stackoverflow.com/questions/172854/como-usar-o-sys-arg-para-entrada-de-dados> acessado em 20 de Junho de 2022.