Busca Termos: um sistema de leitura de arquivos

Ricardo C. Oliveira

Departamento de Ciências Exata Curso de Engenharia de Computação - Universidade Estadual de Feira de Santana (UEFS)
Avenida Transnordestina, s/n - Novo Horizonte - CEP 44036-900 - Feira de Santana - Bahia, Brasil

rickoliver001@hotmail.com

Abstract. *This report aims to describe the construction and development of a system for searching for terms from reading files on the user's computer. In addition, it reports what were the requirements of the project and how all the difficulties encountered in solving the problem were overcome, using the Python programming language for this purpose. Therefore, it was possible to construct the code by compliance with all the requirements imposed by the system.*

Resumo. *O presente relatório tem como objetivo descrever a construção e desenvolvimento de um sistema de busca de termos a partir da leitura de arquivos no computador do usuário. Além disso, relata quais foram os requisitos do projeto e a forma de como foram superadas todas as dificuldades encontradas na resolução da problemática, utilizando da linguagem de programação Python para tal finalidade. Diante disso, foi possível a construção do código acatando todos os requisitos impostos pelo sistema.*

1. Introdução

Atualmente, são utilizados variados sistemas de busca na internet, o mais conhecido deles é o Google. Esses programas utilizam métodos bastante otimizados e velozes para a realização das buscas dos usuários que os operam. Sua principal função é procurar, entre as inúmeras possibilidades disponíveis, tudo que está relacionado com o que o usuário digitar na internet, rede privada ou banco de dados.

Diante disso, buscando extrair a máxima eficiência desses tipos de sistemas em atividades contemporâneas, foi solicitado um sistema que realizasse a busca de termos presentes em arquivos de texto, como um “buscador pessoal”, em seu computador.

Assim, logo após a realização da leitura dos arquivos, torna-se possível a pesquisa e visualização dos termos procurados pelo usuário presentes no sistema. Tornando a busca mais rápida e eficiente.

2. Metodologia

Nas sessões tutoriais que ocorreram no período de construção do código, foram descobertos diversos desafios para a resolução do problema. Inicialmente, os integrantes das sessões tiveram dificuldade em entender o problema em si, tendo que relê-lo diversas vezes para o seu entendimento completo. Nesse contexto inicial, as ideias dos

integrantes foram sendo desenvolvidas de maneira superficial, pois nenhum deles detinha o conhecimento necessário para o solucionamento das questões iniciais.

Com o decorrer do tempo, as sessões tornaram-se muito mais produtivas com relação a qual lógica seria implementada para o solucionamento das questões encontradas. Com a contribuição de cada um nas sessões, o conhecimento acerca das "novas" funcionalidades foi expandido, que por sua vez, gerou como consequência uma variabilidade de ideias para a solução de cada parte do problema. Contudo, apesar das dificuldades iniciais, todos os integrantes conseguiram absorver bastante experiência e conhecimento das sessões tutoriais. Além do desenvolvimento individual, o entrosamento da equipe também foi aprimorado nesse trajeto, gerando uma maior sinergia para experiências futuras.

Ao final do período das sessões, todos já estavam bem encaminhados na resolução do problema. Dito isso, a evolução do conhecimento da equipe foi bastante proveitosa e construtiva.

Nesse contexto, é importante salientar quais foram os requisitos impostos pelo problema em pauta. O sistema de busca em questão consiste em uma alternativa rápida e eficiente de realizar a busca de termos presentes em documentos lidos pelo sistema. Para isso, o sistema deve utilizar um método de extrema eficiência chamado de índice invertido, que se trata de uma estrutura de dados bastante utilizada na Recuperação da Informação (*Information Retrieval*).

Basicamente, o sistema possui seis funcionalidades disponíveis ao usuário: leitura de arquivos de texto; atualização de arquivos já lidos; remoção de arquivos presentes no sistema; busca de termos que estão presentes nos arquivos lidos; visualização da estrutura de índice invertido presente no sistema; solicitação de ajuda de como operar o sistema.

A leitura de arquivos de texto (formato ".txt") deve ser realizada conforme o usuário indique um diretório, podendo sempre indicar algum outro para incrementar a sua lista de documentos posteriormente. Caso o usuário informe um diretório que já tenha sido indexado pelo sistema, o programa deve atualizar as informações presentes nele. Na funcionalidade referente à remoção, o usuário pode indicar um diretório ou arquivo para sua retirada do sistema, excluindo aqueles arquivos do banco de dados do programa. Na funcionalidade de busca, o sistema deve realizar a busca de palavras (termos) presentes nos documentos que já foram lidos pelo programa. Nessa busca, o sistema deve exibir em quais documentos contém aquele termo e informar quantas vezes ele ocorre no documento, além disso, apresentar a quantidade de documentos encontrados nos quais o termo está inserido. A exibição deve ser ordenada de maneira decrescente em função da frequência do termo nos documentos. O usuário também pode requisitar a visualização da estrutura de índice invertido, podendo observar quais termos estão presentes no sistema. Além dessas funcionalidades, é possível realizar a requisição de ajuda, obtendo a forma de como operar o programa.

A operação do sistema deve ser realizada unicamente por argumentos de linha de comando, caso os argumentos não sejam adequados, o programa deve fornecer ajuda ao usuário.

Diante disso, é relevante apresentar a forma de como o programa foi construído e quais funções estão presentes nele. O código possui três arquivos para o seu funcionamento: o código principal, um módulo com as funções utilizadas e um arquivo para o armazenamento dos dados lidos pelo sistema (criado a partir da leitura de pelo menos um arquivo). No arquivo principal são encontradas sequências de chamadas de funções presentes no módulo.

Sob uma perspectiva geral, o código principal é dividido por blocos de acordo com a ação que o usuário deseja realizar. Diante disso, para cada processo disponível no sistema, existe uma sequência lógica de chamadas de funções.

Inicialmente, utilizando a biblioteca *sys,* através da função *argv,* o programa recebe os argumentos de linha de comando em formato de lista. A partir disso, os processos realizados partem dessa lista, a qual será a única entrada por parte do usuário. Diante disso, dependendo da atividade que o usuário deseja realizar, os argumentos presentes na lista serão responsáveis por toda a lógica do sistema.

Para todo processo presente no código (exceto a exibição da ajuda), é de extrema importância que haja, inicialmente, a leitura dos dados do arquivo de armazenamento, pois, assim, os dados lidos anteriormente não serão armazenados em vão. Então, antes de qualquer atividade, é realizada a leitura desse arquivo, caso ele não exista, será a primeira vez em que o usuário está operando o sistema. Nesse caso, o programa reconhece que é a primeira utilização e aguarda até que o primeiro arquivo de texto seja lido.

Adiante, seguindo com o fluxo do código, é encontrada a lógica de leitura. Caso o usuário deseje realizar tal processo especificando um diretório ou arquivo, o programa reconhece qual das duas opções ele escolheu. Caso a opção tenha sido ler um diretório, o sistema realiza a leitura de todos os arquivos de texto presentes nele. Porém, caso a opção tenha sido ler um arquivo, o sistema realiza a leitura somente do arquivo de texto especificado. A partir dessa leitura, o programa realiza o tratamento de todas as palavras existentes no arquivo, assim, elas são incrementadas na estrutura de índice invertido.

Utilizando a mesma ideia do processo de leitura, a remoção é feita seguindo os mesmos passos, porém, removendo os arquivos (ou arquivo) da estrutura de índice invertido. Assim, de acordo com o que o usuário informou como alvo de remoção (arquivo ou diretório), o sistema realiza o processo conforme solicitado.

Seguindo adiante, caso o utilizador do programa deseje atualizar os dados de um arquivo ou diretório, o programa realiza o processo de remoção seguido do comando de leitura. Em síntese, para a atualização dos documentos solicitados, é necessário sua remoção e sua posterior releitura, atualizando assim, todos os termos presentes no documento.

Diante disso, após esses três processos explicitados anteriormente, visando uma maior eficiência, o sistema de busca necessita do armazenamento dos dados lidos, removidos ou atualizados, para o seu posterior reaproveitamento. Assim, ao final de cada um desses processos, há o armazenamento da estrutura de índice invertido, contendo todos os termos lidos e suas relativas aparições. Essa estrutura foi construída utilizando um dicionário, contendo, como valor, uma matriz com suas aparições. Nelas,

estão contidas as seguintes informações: caminho absoluto do arquivo e a quantidade que o termo aparece nele.

No arquivo de armazenamento também está contido uma lista contendo o caminho absoluto de todos os arquivos que estão presentes no índice invertido. Por meio dessa lista, é possível realizar a validação do processo de remoção de documentos do índice, impossibilitando erros dessa natureza.

Seguindo adiante, os processos de busca de termos, visualização do índice e ajuda não necessitam de armazenagem como os anteriores. Para a busca de termos, há somente uma busca no dicionário que contém a lógica do índice invertido de acordo com o termo solicitado. Já para a visualização do índice, o sistema exibe todas as informações de maneira formatada para o utilizador, podendo ser visualizados todos os termos seguidos de suas aparições nos arquivos. Por fim, para a ação de ajuda, são exibidos *print's* contendo a forma de como operar o sistema de maneira adequada.

Outro ponto importante para o entendimento do desenvolvimento do código é sua ordem de codificação. Em primeiro plano, a atenção foi voltada para como implementar os argumentos de linha de comando no sistema. Após o conhecimento desse modo de entrada no código, o foco foi redirecionado para como realizar a leitura de arquivos e diretórios, assim como, diferenciar quais deles poderiam ser lidos pelo programa. Após a realização dessa etapa, o tratamento das palavras lidas nos arquivos de texto foi o principal foco, pois, após esse tratamento, seria possível a implementação do índice invertido.

Após o tratamento, foi importante dar ênfase em como seria implementado o índice invertido. Após vários estudos e pesquisas, foi possível a construção de tal estrutura no código em questão. Seguindo adiante, o processo de leitura teve de ser desenvolvido com mais afinco, pois continha alguns erros pequenos. Logo após isso, a lógica de remoção de documentos foi desenvolvida e aplicada ao sistema.

Mais adiante, foi desenvolvido a lógica de busca de termos no código, incluindo suas exibições ao usuário de forma decrescente. Nesse mesmo período, a visualização do índice foi implementada logo em seguida, tornando possível sua realização conforme requisição do usuário. A exibição de ajuda foi implementada logo adiante.

Por último, foram feitas as lógicas de atualização do índice, assim como, validações referentes a erros ocasionais no programa, prevenindo-o de possíveis erros.

Para o desenvolvimento das soluções implementadas no programa, foi utilizado o sistema operacional Windows 10, juntamente com a linguagem de programação Python manipulada através do editor de código-fonte Visual Studio Code.

3. Resultados e discussões

O programa deve ser utilizado de acordo com as opções que são apresentadas aos seus usuários. Sendo assim, o sistema possui uma lógica de operação para cada escolha do utilizador. Dependendo da operação a ser realizada, a formatação da entrada feita pelo usuário por meio dos argumentos se modifica. Portanto, o entendimento do manual de uso do programa e suas entradas devem ser especificadas.

Antes de tudo, qualquer processo solicitado antes da leitura de pelo menos um arquivo, além do comando de ajuda, não será possível, pois o banco de dados do sistema não existe ainda, ou não possui nenhuma informação. Portanto, além do comando de ajuda, qualquer operação deve ser realizada somente após a leitura de pelo menos um arquivo.

Para o processo de leitura, o usuário deve inserir como argumento, o comando "LER" seguido logo após do arquivo ou diretório o qual deseja realizar a leitura. Caso a opção de leitura escolhida tenha sido um arquivo, deve-se colocar a extensão ".txt" ao final. Caso o usuário insira um arquivo ou diretório previamente indexado, o sistema informa que é necessário o comando de atualização.

Para os processos de remoção e atualização, o usuário deve inserir os argumentos na mesma formatação do processo de leitura. Porém, com os referidos comando "REMOVER" para a remoção e "ATT" para a atualização.

Já para o processo de busca, o usuário deve pôr o comando "BUSCAR" e logo em seguida, o termo que procura no sistema como argumentos.

Seguindo adiante, para o processo de visualização do índice e ajuda, são necessários somente os comandos "VER" e "HELP", respectivamente, como argumentos. Vale ressaltar que, para qualquer comando irregular, o sistema imprime automaticamente na tela a mensagem de ajuda para o usuário.

Atribuindo a mesma importância das entradas, as saídas possuem papel fundamental no entendimento do sistema. Essencialmente, é possível distinguir os tipos de saídas: as provenientes dos comandos de visualização (visualizar o índice, ajuda e busca); confirmações de processos de leitura, remoção e atualização; validações referentes a prevenção de erros.

As saídas referentes aos comandos de visualização são as requisitadas pelo problema. Na visualização do índice é possível observar e identificar quais termos estão presentes na estrutura e em quais arquivos eles aparecem, assim como, qual a sua recorrência no arquivo. Na busca, a saída se configura nas aparições daquele termo nos arquivos contidos no sistema, ordenadas de forma decrescente. Na ajuda, as saídas são basicamente exibições de como operar o sistema da forma correta.

Seguindo adiante, as saídas referentes às confirmações de processos estão relacionadas a conclusão de tal processo. Essas saídas são por exemplo: "Arquivo lido com sucesso!" ou "Arquivo removido com sucesso!". Nessa mesma ideia, há também confirmação de quando não foi possível a realização do processo requisitado.

Por fim, há também mensagens de prevenção de erros, como "Não foi possível abrir o arquivo" e afins.

Para compreender o desenvolvimento do código no decorrer do período de sua confecção, é de suma importância conhecer quais foram os testes efetuados nele. De início, os testes realizados foram referentes a manipulação dos argumentos de linha de comando, possibilitando um maior entendimento de seu funcionamento. Logo após esses testes, foi necessário a realização de testes relacionados à leitura de arquivos de texto. No início, os resultados referentes a esses testes retornavam somente uma linha de

cada arquivo, assim, o foco dos testes se baseou em como obter a solução para essa situação.

Logo após, os testes foram concentrados no tratamento das palavras lidas nos arquivos de texto, retirando caracteres especiais e afins. Assim, com a finalização dessa fase de testagem do tratamento de palavras, os testes foram redirecionados para a construção do índice invertido, observando quais casos especiais apareciam na sua formação e quais erros resultavam deles. Com o solucionamento e prevenção dos casos especiais dos testes anteriores, foram feitas diversas leituras de vários arquivos de texto para a confirmação de que o processo de leitura estava ocorrendo da maneira correta.

Seguindo adiante com os testes realizados, foi necessário realizar a testagem do processo de remoção, que consistia em remover um diretório ou um arquivo do índice. Logo após, a realização dos testes com relação a atualização do índice foram efetuados, eles se configuravam em observar se o processo estava de fato ocorrendo, conforme ia-se mudando o conteúdo dos arquivos de texto que seriam atualizados.

Por fim, os testes foram realizados em cima das entradas do usuário, como a existência de um arquivo ou não, comandos irregulares, e, além disso, a abertura do arquivo de armazenamento que o sistema utiliza.

Durante esse trajeto, houveram diversos erros na construção do código, por isso, é importante apresentá-los para uma melhor percepção do seu desenvolvimento e otimização.

Inicialmente, o código apresentava diversos erros relacionados a manipulação dos argumentos de linha de comando, assim foram necessários a realização de testes para o reconhecimento desses erros. Com o solucionamento desses erros iniciais, os próximos encontrados se localizaram na remoção de termos da estrutura de índice invertido. O sistema apresentava o erro de *"RuntimeError",* que significava que o dicionário mudava de tamanho durante a remoção de algum termo. Assim, foi preciso utilizar uma cópia do dicionário e manipulá-la para poder modificá-lo.

Logo após o solucionamento desses erros, foram encontrados erros relacionados a atualização do índice, pois quando havia a remoção de um arquivo, o programa não retirava o arquivo removido da lista de arquivos presentes no sistema. Assim, o tratamento deste erro foi de extrema importância para a conclusão do código. Em síntese, os problemas foram sanados para um melhor funcionamento do código, solucionando assim, os erros encontrados no período de confecção do mesmo.

4. Conclusão

Dado o presente problema, foi possível desenvolver e aprender variadas ferramentas e funcionalidades da linguagem Python. Todos os requisitos foram cumpridos adequadamente ao decorrer do desenvolvimento do projeto.

Em síntese, as áreas que podem receber melhora no código são as partes visuais apresentadas no terminal, e também a otimização do programa em si, promovendo uma melhor rapidez no processamento de dados realizado. Funcionalidades e implementações a mais realizadas no código foram: possibilidade de leitura de arquivos específicos (não somente diretórios). Essa funcionalidade foi implementada no código

para torná-lo mais completo ao usuário. Vale ressaltar que, o código utiliza padrões do sistema operacional windows, por isso, podem ocorrer alguns erros relacionados aos caminhos absolutos de outro sistema, os quais podem utilizar outra formatação.

O trajeto realizado tanto individualmente quanto pela equipe foi totalmente proveitoso para o conhecimento de novas funções presentes na linguagem de programação utilizada. Tornando assim, a experiência construtiva e elucidativa.

5. Referências

Curso de Python para iniciantes #12 - Manipulando arquivos, Refatorando, publicado em 19 de janeiro de 2021 na plataforma YouTube. Disponível em: Curso de Python para iniciantes #12 - Manipulando Arquivos - YouTube

04 14 Índices invertidos, Marcos André Silvera Kutova, publicado em 17 de setembro de 2016 na plataforma YouTube. Disponível em: 04 14 Índices invertidos - YouTube

Aulas Python - 109 - Ferramentas de Sistema V: Argumentos no Terminal, Ignorância Zero, publicado em 15 de dezembro de 2014 na plataforma YouTube. Disponível em: Aulas Python - 109 - Ferramentas de Sistema V: Argumentos do Terminal - YouTube

Como utilizar sys.args em Python, Only a programmer, publicado em 16 de outubro de 2019 na plataforma YouTube. Disponível em: Como utilizar sys.args em Python - YouTube