Introdução ao Sistema de Recuperação de Informação (Sistema RI)

O Sistema RI é uma poderosa ferramenta que permite a busca, recuperação e organização de informações a partir de uma grande base de dados. Esse sistema é composto por diversas classes responsáveis por diferentes etapas do processo, desde a coleta de documentos da web até a apresentação dos resultados de busca para o usuário. Nesta introdução, exploraremos as principais funcionalidades e a integração dessas classes no Sistema RI.

by Bruno Ferreira

Classe URL: Representação e Manipulação de URLs

A Classe URL é responsável por representar e manipular endereços web (URLs) de forma eficiente. Essa classe possui métodos para analisar a estrutura de uma URL, extrair informações como o domínio, o caminho, os parâmetros, entre outros. Além disso, a Classe URL também é responsável por gerar URLs válidas e normalizadas, garantindo que os dados coletados pela etapa seguinte do sistema sejam confiáveis e padronizados.

Principais Funcionalidades

- Análise e extração de informações de uma URL - Validação e normalização de URLs - Geração de URLs a partir de parâmetros

Importância no Sistema RI

A Classe URL é crucial para o funcionamento do Sistema RI, pois garante a integridade e a consistência dos dados coletados da web. Sem uma representação precisa das URLs, o sistema não seria capaz de acessar corretamente os documentos e realizar a indexação de forma eficiente.

Código do Url

URL.py

A classe Url representa uma página web.

Cada objeto da classe Url contém a URL da página web, o título da página, todos os subtítulos da página, todos os links da página, todos os parágrafos da página, todas as imagens da página, todas as listas da página e todas as tabelas da página.

Atributos:

- `url (str)` : Um URL da página web.
- `page (bs4.element.Tag)` : O título da página web.
- `titles (bs4.element.ResultSet)` : Todos os subtítulos da página web.
- `links (bs4.element.ResultSet)` : Todos os links da página web.
- `paragrafos (bs4.element.ResultSet)` : Todos os parágrafos da página web.
- `imagens (bs4.element.ResultSet)` : Todas as imagens da página web.
- `listas (bs4.element.ResultSet)` : Todas as listas da página web.
- `tabelas (bs4.element.ResultSet)` : Todas as tabelas da página web.

Exemplo:

> from bs4 import BeautifulSoup importar solicitações resposta = requests.get(" https://www.ifmg.edu.br ") sopa = BeautifulSoup(response.text, 'html.parser') url_obj = Url(" https://www .ifmg.edu.br ", sopa) print(url_obj.url) " https://www.ifmg.edu.br " print(url_obj.page.text) "IFMG - Instituto Federal de Minas Gerais" print(len(url_obj.titles)) 5 print(len(url_obj.links)) 128 print(len(url_obj.paragrafos)) 12 print(len(url_obj.imagens)) 20 print(len(url_obj.listas)) 10 print(len(url_obj.tabelas)) 2

Classe Coletor: Coleta de Documentos da Web

A Classe Coletor é responsável por realizar a coleta de documentos da web, acessando as URLs fornecidas e baixando o conteúdo dos documentos. Essa classe utiliza a Classe URL para gerar endereços válidos e acessa os documentos de forma automatizada. Após a coleta, os documentos são encaminhados para a próxima etapa do sistema, a indexação.

1

Acesso às URLs

A Classe Coletor utiliza a Classe URL para gerar endereços web válidos e acessar os documentos da internet de forma automatizada.

2

Baixa de Conteúdo

Após o acesso, a Classe Coletor baixa o conteúdo dos documentos web e os armazena em formato digital, preparando-os para a próxima etapa de indexação.

3

Encaminhamento

Os documentos coletados são então encaminhados para a Classe Indexador, que irá processá-los e adicionar à base de dados do Sistema RI.

Código do Coletor

Coletor.py

A classe Coletor é responsável pela coleta de páginas web.

Cada objeto da classe Coletor contém um código, um dicionário de URLs a serem coletadas e uma lista de objetos da classe Url que foram criados a partir das URLs coletadas.

Atributos:

> - `codigo (str)` : Um código fornecido quando um objeto da classe Coletor é criado.
> - `urls (dict)` : Um dicionário que armazena as URLs que serão coletadas. As chaves são as URLs e os valores são booleanos que indicam se uma URL já foi coletada (True) ou não (False).
> - `objects_url (list)` : Uma lista que armazena objetos da classe Url que foram criados a partir das URLs coletadas.

Métodos:

> - `addUrl(url)` : Adiciona uma nova URL ao dicionário self.urls, se ela ainda não estiver presente.
> - `extrair_informacoes(params=None)` : Extrai informações das URLs que ainda não foram visitadas e que são permitidas pelo arquivo robots.txt.
> - `can_fetch(url)` : Verifique se a coleta é permitida para um URL especificado pelo arquivo robots.txt.
> - `extrair_em_profundidade(profundidade=0, params=None)` : Extrai informações das URLs até a profundidade especificada. Quando a profundidade é 0, este método extrai informações de todas as URLs do mesmo domínio que a primeira URL adicionada ao Coletor. Quando a profundidade é um valor diferente de 0, ele extrai informações das URLs até essa profundidade.

Exemplo:

> > coletor = Coletor("Root") coletor.addUrl(" https://www.ifmg.edu.br ") coletor.extrair_em_profundidade() print(len(coletor.objects_url)) 1

Classe Indexador: Indexação de Documentos Coletados

A Classe Indexador é responsável por processar os documentos coletados pela Classe Coletor e criar um índice invertido. Esse índice mapeia as palavras-chave presentes nos documentos e associa-as aos documentos em que elas aparecem. Essa estrutura de dados é crucial para permitir buscas rápidas e eficientes no Sistema RI.

1

Processamento de Documentos

A Classe Indexador recebe os documentos coletados e realiza o processamento de texto, como remoção de stopwords, stemming e tokenização.

2

Construção de Índice Invertido

Com as informações extraídas, a classe cria um índice invertido que associa cada termo aos documentos em que ele aparece.

3

Armazenamento do Índice

O índice invertido é então armazenado para ser utilizado posteriormente pela Classe Buscador durante as consultas dos usuários.

Código do Indexador

Indexador.py

A classe Indexador é responsável por gerar um índice invertido a partir das páginas da web coletadas por um objeto da classe Coletor.

Cada objeto da classe Indexador contém um objeto da classe Coletor, uma lista de títulos tokenizados das páginas da web coletadas, um conjunto de stopwords em português e um dicionário que armazena o índice invertido.

Atributos: `coletor (Coletor)` : O objeto Coletor que coletou os dados. `stop_words (set)` : Um conjunto de palavras de parada. `inverted_index (dict)` : O índice invertido gerado. `F (dict)` : Um dicionário que armazena a frequência de cada token.

Métodos: `__init__(self, coletor: Coletor)` : Inicializa um objeto Indexador com o Coletor especificado. `inverted_index_generator(self) -> None` : Gera o índice invertido. `update_F(self) -> None` : Atualiza a frequência de cada token sem índice invertido. `save_index(self) -> None` : salva o índice invertido em um arquivo JSON. `weight_tokenize(self) -> None` : Calcula o peso de cada token sem índice invertido. `add_attr_inverted_index(self, params=None) -> None` : Adicionados atributos ao índice invertido. `remove_key_stop_word(self) -> None` : Remova as palavras de parada do índice invertido.

Exemplo:

> coletor = Coletor("Root") coletor.addUrl(" https://www.ifmg.edu.br ") coletor.extrair_informacoes() indexador = Indexador(coletor) indexer.inverted_index_generator() indexer.weight_tokenize() indexer.add_attr_inverted_index () indexer.update_index() indexer.remove_key_stop_words() indexer.save_index() index-Root.json = 128

Classe Expressões: Cálculo da Modelagem Vetorial

A Classe Expressões é responsável por implementar a modelagem vetorial, uma técnica de recuperação de informação que representa documentos e consultas como vetores em um espaço multidimensional. Essa classe calcula os pesos dos termos presentes nos documentos e nas consultas, permitindo a comparação entre eles e a apresentação dos resultados mais relevantes.

Ponderação de Termos

A Classe Expressões utiliza técnicas como TF-IDF (Term Frequency-Inverse Document Frequency) para calcular o peso de cada termo nos documentos e nas consultas.

Cálculo de Similaridade

Com os pesos dos termos, a classe calcula a similaridade entre os vetores dos documentos e da consulta, utilizando métricas como o cosseno da similaridade.

Ordenação de Resultados

Com base nos cálculos de similaridade, a Classe Expressões ordena os documentos de forma decrescente, priorizando os mais relevantes para a consulta do usuário.

Código do Expressões

Expressões.py

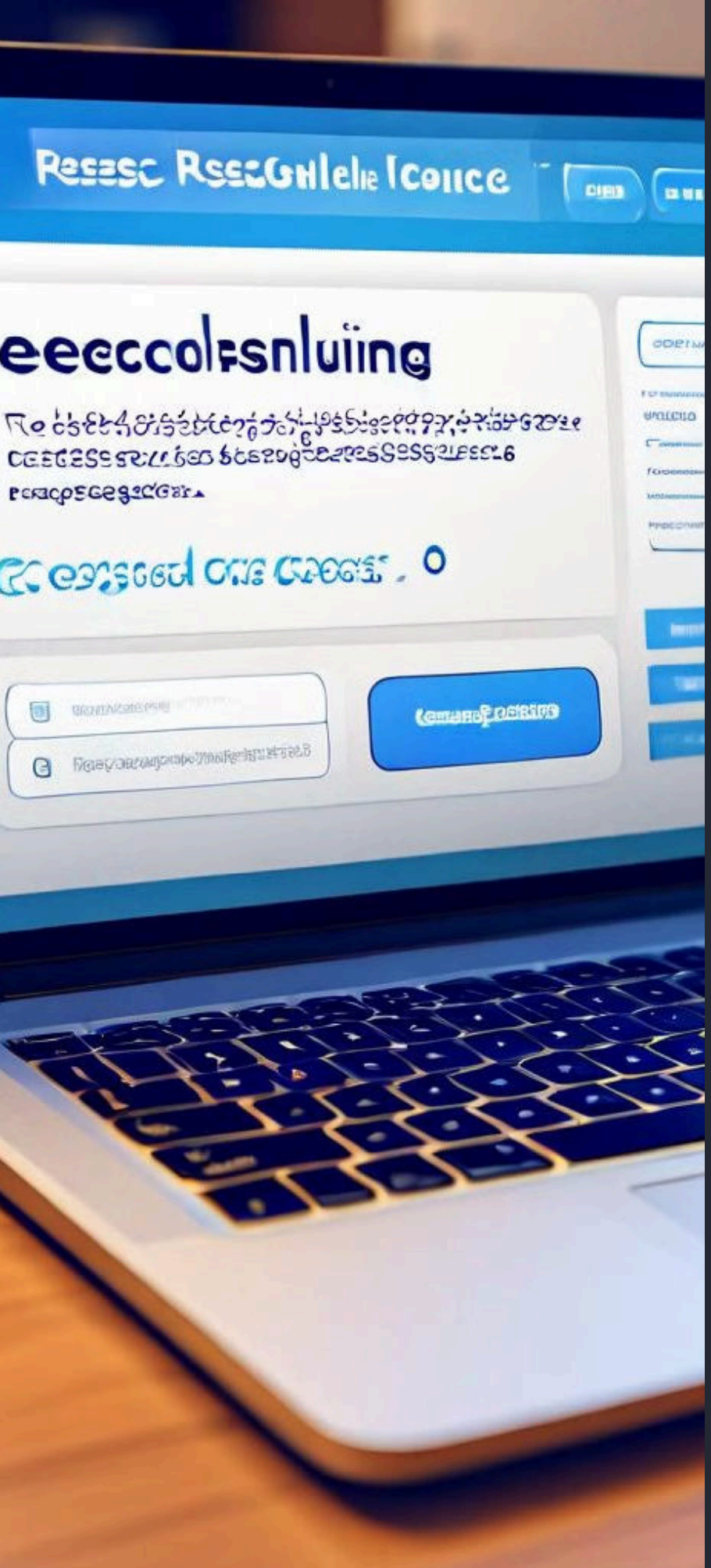
A classe Expressões contém métodos para calcular diferentes expressões matemáticas.

Cada método da classe Expressões calcula uma expressão matemática específica. As expressões são definidas como (1 + log2(i)) * log2(n / ni), onde 'i' é um elemento de um conjunto 'j' ou de uma consulta 'q', 'n' é um número de documentos na coleção e 'ni' é um número de documentos em que Ki ocorre.

Métodos: `calcular_wij(i: int or float, n: int or float, ni: int or float) -> float or str` : Calcula o valor de peso dos termos (TF x IDF) no documento. `calcular_wiq(i: int or float, n: int or float, ni: int or float) -> float or str` : Calcula o valor de peso dos termos (TF x IDF) na consulta.

Exemplo:

> i = 2 n = 10 ni = 3 print(f"O peso de 'i' no conjunto 'j' é {Expressoes.calcular_wij(i, n, ni)}") "O peso de 'i' no conjunto 'j' ' é 3.4594316186372978" print(f"O peso de 'i' na consulta 'q' é {Expressoes.calcular_wiq(i, n, ni)}") "O peso de 'i' na consulta 'q' é 3.4594316186372978"



Classe Buscador: Busca e Recuperação de Informações

A Classe Buscador é o ponto de entrada do Sistema RI para os usuários. Essa classe recebe as consultas dos usuários, utiliza as outras classes do sistema para processar a busca e apresenta os resultados ordenados por relevância. A Classe Buscador é crucial para fornecer a consulta e o resultado da consulta para os usuários finais do Sistema RI.

1

Recebimento de Consulta

A Classe Buscador recebe a consulta do usuário, que é o ponto de partida do processo de recuperação de informação.

2

Processamento da Consulta

A consulta é então processada utilizando a Classe Expressões para calcular a similaridade entre a consulta e os documentos indexados.

3

Apresentação de Resultados

Finalmente, a Classe Buscador apresenta os documentos mais relevantes para o usuário, ordenados de acordo com os cálculos de similaridade.

Código do Buscador

Buscador.py

A classe Buscador realiza buscas em um índice invertido.

Atributos: `inverted_index (dict)` : O índice invertido onde a busca será realizada.

Métodos: `__init__(self, index_file: str)` : Inicializa um objeto Buscador com o índice invertido contido no arquivo especificado. `deep_search(self, query: str) -> set` : Realiza uma busca em profundidade no índice invertido. `width_search(self, query: str) -> set` : Realiza uma busca em largura no índice invertido. `rank(self, query: str) -> list` : Realiza um ranqueamento de documentos com base em consulta de pesquisa.

Exemplo de uso:

```
buscador = Buscador("index_file.json") result = buscador.deep_search("consulta de pesquisa")
print(result)
```

Integração entre as Classes no Sistema RI

O Sistema RI é composto por um conjunto de classes que trabalham de forma integrada para fornecer uma solução completa de recuperação de informações. A Classe URL garante a integridade dos dados coletados, a Classe Coletor realiza a coleta dos documentos, a Classe Indexador cria o índice invertido, a Classe Expressões calcula a relevância dos documentos e a Classe Buscador apresenta os resultados aos usuários. Essa integração permite que o sistema funcione de forma eficiente e forneça resultados de busca precisos e relevantes.

Classe URL

Responsável pela representação e manipulação de URLs

Classe Coletor

Responsável pela coleta de documentos da web

Classe Indexador

Responsável pela indexação dos documentos coletados

Classe Expressões

Responsável pelo cálculo da modelagem vetorial

Classe Buscador

Responsável pela busca e recuperação de informações

Código do Sistema de Recuperação de Informação

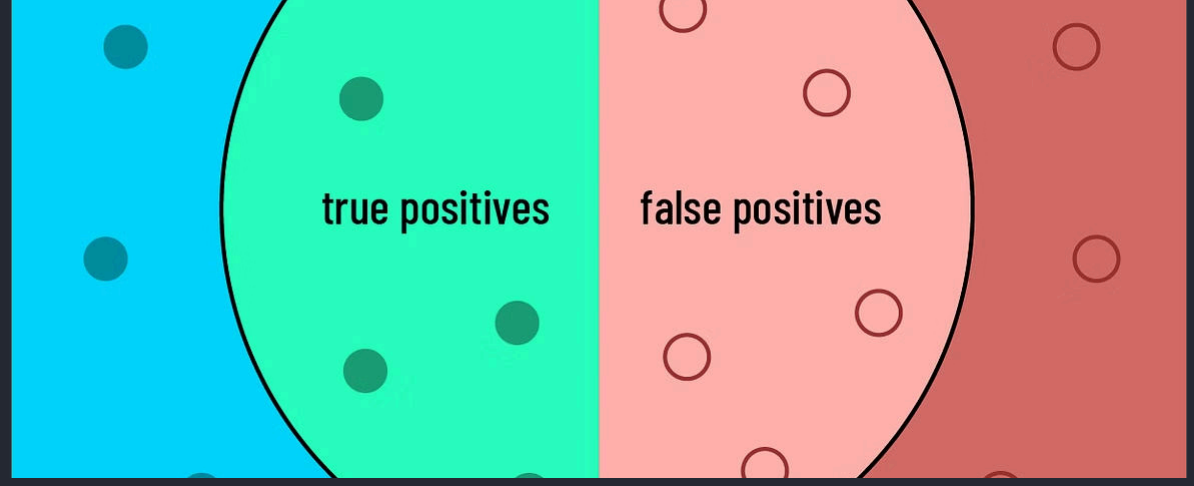
SistemaRI.py

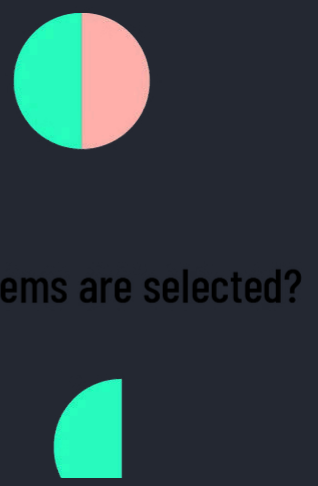
Este script realiza as seguintes etapas:

1. Cria um objeto da classe Coletor.
2. Adiciona URLs ao coletor.
3. Extrai as informações das URLs adicionadas ao coletor.
4. Cria um objeto da classe Indexador com o coletor.
5. Gera um índice invertido.
6. Salva o índice invertido em um arquivo JSON.
7. Solicite ao usuário que insira uma consulta de pesquisa.
8. Cria um objeto da classe Buscador com o nome do arquivo do índice invertido.
9. Pesquise a consulta de pesquisa sem índice invertido.
10. Imprima os resultados da pesquisa.

Exemplo:

> python SistemaRI.py O que você deseja pesquisar? IFMG https://www.ifmg.edu.br/portal





Considerações Finais sobre o Sistema RI

O Sistema RI é uma ferramenta poderosa e fundamental para a recuperação eficiente de informações em grandes bases de dados. Através da integração das suas classes, o sistema é capaz de coletar, indexar e apresentar os resultados mais relevantes para as consultas dos usuários. Eentretanto, não há um cálculo para a precisão e a revocação do resultado das buscas. Como deveria ter e funcionar tais métodos no sistema?

Precisão	A precisão do Sistema RI refere-se à capacidade de apresentar apenas os documentos mais relevantes para a consulta do usuário. Isso é alcançado através da modelagem vetorial e dos cálculos de similaridade realizados pela Classe Expressões.
Revocação	A revocação do Sistema RI representa a capacidade de recuperar a maior parte dos documentos relevantes para a consulta do usuário. Isso depende da eficiência da coleta e indexação realizadas pelas Classes Coletor e Indexador.