**1 - Principais desafios e soluções**:

* **Desafio**: Manipular e processar grandes volumes de dados textual, especialmente para a criação de índices invertidos eficazes. Indexação eficiente de documentos grandes para recuperação rápida.
* **Solução**: O uso de bibliotecas como NLTK para processamento de texto (stemming e remoção de stopwords) e JSON para serialização de dados permitiu eficiência no armazenamento e recuperação. Para o manuseio de caminhos e arquivos, a biblioteca os foi empregada, facilitando a organização dos dados.

**2 - Vantagens/desvantagens das soluções**:

* **Vantagens**:
  + **JSON**: Facilita a interoperabilidade dos dados entre diferentes sistemas e linguagens de programação.
  + **NLTK**: Proporciona ferramentas robustas de PLN que são especificamente boas para lidar com a língua portuguesa, facilitando a análise e manipulação de texto.
* **Desvantagens**:
  + **Performance de I/O**: A leitura e escrita frequente de arquivos pode se tornar um gargalo, especialmente com grandes volumes de dados.
  + **Uso de memória**: Carregar grandes datasets em memória pode ser custoso, exigindo máquinas com maior capacidade de memória.

**3 - Bibliotecas externas utilizadas**: As bibliotecas citadas na resposta anterior são todas utilizadas para diversos propósitos como manipulação de dados, interface de usuário, processamento de texto, e análise estatística.

 **json**: Usada para codificação e decodificação de dados no formato JSON, que é amplamente utilizado para armazenar e transmitir dados devido à sua simplicidade e legibilidade.

 **pdfkit**: Interface para o wkhtmltopdf, uma ferramenta que permite converter HTML para PDF. Utilizada para gerar relatórios formatados como PDF a partir de templates HTML.

 **os**: Fornece funções para interagir com o sistema operacional. Usada para manipular caminhos, diretórios e arquivos, como criar pastas, listar conteúdos de diretórios e gerenciar caminhos de arquivos.

 **jinja2 (Environment, FileSystemLoader)**: Motor de templates para Python que permite a criação de páginas HTML dinâmicas. Usada para renderizar HTML com base em dados dinâmicos, que depois podem ser convertidos em PDF com pdfkit.

 **streamlit (st)**: Framework para criação de aplicações web voltadas a dados e machine learning. Permite a criação rápida de interfaces de usuário para scripts Python.

 **time**: Usada para medir intervalos de tempo (como o tempo de execução de uma função) e realizar operações relacionadas ao tempo, como atrasos.

 **re**: Provê operações de correspondência de expressões regulares. Usada para manipulação de texto, como encontrar padrões específicos ou substituir caracteres.

 **collections (Counter)**: Fornece objetos de container adicionais, como Counter, que é uma subclasse de dicionário utilizada para contar objetos hashable.

 **sklearn.feature\_extraction.text (TfidfVectorizer)**: Parte do scikit-learn, usado para converter uma coleção de documentos brutos em uma matriz de características TF-IDF, que é usada para análise de texto.

 **tqdm**: Biblioteca que fornece uma barra de progresso que pode ser exibida durante loops longos. Usada para fornecer feedback visual sobre o progresso de processos demorados.

 **pandas (pd)**: Biblioteca de análise de dados que oferece estruturas de dados flexíveis e operações para manipular tabelas numéricas e séries temporais.

 **nltk**: Natural Language Toolkit, uma biblioteca para o processamento de linguagem natural (PLN) que fornece ferramentas para classificação de texto, tokenização, lematização, stemming, tag de parte de fala e muito mais.

 **logging**: Usada para registrar mensagens de erro e informações que podem ajudar no diagnóstico de problemas e na monitoração do comportamento de aplicações.

 **pyzabbix (ZabbixAPI)**: Cliente Python para interagir com o Zabbix, uma plataforma de monitoramento de rede. Permite automatizar tarefas administrativas do Zabbix através de scripts Python.

 **pymysql & mysql.connector**: Módulos que permitem a conexão com bancos de dados MySQL, executando comandos e gerenciando dados.

 **psycopg2**: Adaptador de banco de dados PostgreSQL para a linguagem Python, permitindo que aplicações Python se comuniquem com o PostgreSQL de maneira eficiente.

 **urllib.parse (urlparse)**: Usado para dividir URLs em seus componentes (ou reconstruí-los), ajudando na manipulação e análise de URLs.

**4 - Funcionamento das técnicas adotadas**:

* **TF-IDF**: Mede a importância de uma palavra em um documento em comparação com um corpus de documentos. O valor TF (Term Frequency) indica a frequência de uma palavra no documento, enquanto IDF (Inverse Document Frequency) diminui o peso de palavras que aparecem frequentemente em muitos documentos, considerando-as menos importantes.
* **Tokenização**: Divide strings em listas de substrings. Por exemplo, separar palavras de frases.
* **Stemming (RSLPStemmer)**: Reduz palavras ao seu radical básico, facilitando a indexação e pesquisa por agrupar variações da mesma palavra.
* **Chunking**: Cria pedaços de textos que podem ser tratados como unidades independentes, útil para processamento paralelo ou análise de documentos grandes.
* **Limpeza de Texto**: Envolve a remoção de caracteres especiais e pontuações, normalização de texto para minúsculas e remoção de palavras comuns (stopwords) que são irrelevantes para a análise.

**5 - Fórmulas usadas**:

* + **TF-IDF**: A fórmula é dada por:

* + Esta fórmula ajuda a identificar quais termos são importantes para um documento dentro de um conjunto de documentos, balanceando a frequência de termos com sua raridade no conjunto todo.

A organização dessas técnicas e bibliotecas permite desenvolver um sistema robusto de processamento e análise de texto, capaz de executar desde a tokenização e análise léxica até a geração de índices e pesquisa eficiente através de técnicas como TF-IDF.