Implementação e Comparação de Motores de Busca Utilizando Woosh e Elasticsearch na Base CRANFIELD

Maxwel Batalha da Silva Lopes

Instituto Multidisciplinar – Universidade Federal Rural do Rio de Janeiro (UFRRJ)

maxwellps@ufrrj.br

Abstract. *This study explores the implementation and comparison of search engines using the Woosh and Elasticsearch tools within the CRANFIELD database. Two fundamental stages are addressed in each search engine: indexing and the search phase. Through a comparative analysis of these tools, we evaluate the efficiency and accuracy of each in retrieving information within the specific context of the CRANFIELD database.*

Resumo. *Este trabalho explora a implementação e a comparação de motores de busca utilizando as ferramentas Woosh e Elasticsearch na base de dados CRANFIELD. Duas etapas fundamentais são abordadas em cada motor de busca: a indexação e a fase de busca. Por meio da análise comparativa dessas ferramentas, avaliamos a eficiência e a precisão de cada uma na recuperação de informações dentro do contexto específico da base de dados CRANFIELD.*

1. Introdução

A capacidade de recuperar informações relevantes de grandes conjuntos de dados é fundamental, especialmente no âmbito acadêmico e científico, onde a rápida localização de artigos é crucial para impulsionar a pesquisa. A criação de motores de busca eficazes desempenha um papel significativo nesse processo, sendo a escolha da ferramenta uma decisão crucial que afeta diretamente a precisão e eficiência dessa recuperação.

Neste estudo, busca-se avaliar e comparar dois motores de busca, Woosh e Elasticsearch, com o objetivo de entender sua eficácia na indexação e recuperação de informações na base de dados CRANFIELD. Woosh, uma biblioteca Python conhecida por seus recursos eficientes de indexação e busca de texto, será confrontada com Elasticsearch, um mecanismo de busca e análise de texto que opera em um ambiente de servidor distribuído.

A base de dados CRANFIELD, amplamente utilizada na área de recuperação de informações, oferece o ambiente propício para a realização dos testes e avaliações desses sistemas. Ao implementar e analisar esses motores de busca nesse contexto, este estudo visa fornecer uma comparação de suas capacidades, avaliando sua eficiência, precisão e desempenho.

2. Metodologia

A metodologia adotada neste estudo compreendeu a implementação de um mecanismo de busca utilizando as ferramentas Woosh e Elasticsearch na base de dados CRANFIELD.

Para aprimorar a precisão das consultas, considerou-se a remoção de stopwords, termos que possuem alta frequência na língua e usualmente não contribuem significativamente para a relevância de uma busca.

Dentre as stopwords selecionadas, como 'a', 'an', 'the', 'and', 'or', 'but', 'about', 'above', 'after', 'along', 'as', 'at', 'by', 'for', 'from', 'in', 'into', 'of', 'on', 'onto', 'over', 'to', 'up', 'with', 'is', 'are', 'was', 'were', optou-se por excluí-las durante o processamento das consultas através da função *remove_stopwords(query)*. Essa exclusão foi estrategicamente realizada para realçar termos-chave mais relevantes, reduzindo a interferência de palavras comuns que, embora frequentes, oferecem pouco valor descritivo na identificação de informações essenciais. Ao receber uma consulta, essa função segue um processo sequencial para remover as stopwords:

1. Inicialmente, a consulta é convertida para letras minúsculas, garantindo que a comparação com as stopwords seja feita sem distinção entre maiúsculas e minúsculas. Em seguida, a consulta é dividida em uma lista de palavras, onde cada palavra é identificada a partir dos espaços em branco como delimitadores.
2. O próximo passo é a remoção das stopwords da consulta. Utilizando a lista de stopwords predefinida, a função compara cada palavra da consulta, removendo aquelas que correspondem às stopwords da lista. Dessa forma, são mantidos apenas os termos considerados mais relevantes para a busca, destacando as palavras-chave que têm maior poder descritivo e contribuem de maneira mais significativa para a compreensão do conteúdo.
3. Por fim, as palavras restantes, após a exclusão das stopwords, são reunidas em uma única string, reconstruindo a consulta sem as palavras indesejadas.

A base de dados CRANFIELD é um conjunto clássico e fundamental na área de recuperação de informação. Ela é amplamente utilizada para avaliar e testar sistemas de busca, sendo composta por três arquivos principais: *cran.all.1400*, *cran.qry* e *cranqrel*.

4. *cran.all.1400*: Armazena informações completas dos documentos na base de dados. Cada registro representa um artigo ou documento relevante para a pesquisa. É utilizado para indexação e referência durante a implementação e teste dos motores de busca.
5. *cran.qry*: Contém as consultas utilizadas para avaliar os motores de busca. Cada consulta representa uma pergunta ou solicitação de informação. Essas consultas são essenciais para testar a capacidade dos motores de busca em encontrar informações relevantes.
6. *cranqrel*: Lista os documentos relevantes para cada consulta no arquivo cran.qry. Essa relação de relevância serve como referência para avaliar a precisão dos resultados retornados pelos motores de busca em relação ao que é considerado relevante para cada consulta.

Para lidar com as consultas do arquivo cran.qry, foi implementada a função *extract_queries_from_cranqry(file_path).* Essa função desempenha um papel crucial no processamento das consultas da base de dados CRANFIELD. O procedimento é feito da seguinte maneira:

1. Leitura do arquivo e Extração das Consultas: A função é responsável por ler o arquivo "cran.qry" especificado pelo caminho file_path e extrair as consultas contidas nele. Cada consulta está associada a um ID único.
2. Aplicação da função *remove_stopwords(query)*: Após a leitura das consultas, é aplicada a função em cada uma das consultas extraídas. Isso é crucial para aprimorar a precisão e relevância das buscas, eliminando palavras de baixa relevância que não contribuem significativamente para os resultados.
3. Armazenamento em Lista Identificável:: As consultas, após passarem pelo processo de remoção de stopwords, são armazenadas em uma lista que é retornada no fim da função. É importante mencionar que cada consulta na lista é identificável pelo ID associado a ela dentro de cada documento no arquivo "cran.qry".

Para processar o arquivo cranqrel, responsável por listar os documentos relevantes para cada consulta, foi desenvolvida a função *extract_relevant_documents(file_path)*. Esta função converte cada consulta em uma lista contendo os IDs dos documentos relevantes, ordenados por sua relevância para a respectiva consulta. Cada lista é identificada pelo ID correspondente à consulta.

Para os comparativos entre os dois buscadores, Whoosh e Elastic, foi utilizado o tempo de indexação e o tempo de busca em cada motor através da função *time()* da biblioteca *time.* Além do tempo, foi utilizado a métrica de precisão para comparar o percentual de acerto entre os dois motores. Em cada consulta é calculada a precisão e no final é feito uma média entre as precisões para obter um panorama geral.

2.1. Indexação

A indexação é a etapa onde os dados são organizados e estruturados para facilitar a recuperação eficiente de informações posteriormente. Os documentos ou dados são processados e analisados para extrair palavras-chave, conceitos ou outros elementos relevantes que os representem. Esses elementos são então armazenados em uma estrutura otimizada chamada de índice.

O índice é uma espécie de mapa ou catálogo que aponta para a localização dos documentos que contêm informações específicas. Essencialmente, ele permite uma busca rápida e eficiente. Cada termo, palavra ou conceito extraído durante a indexação torna-se uma entrada nesse índice, apontando para os documentos onde esses termos são encontrados.

Durante a busca, o sistema consulta esse índice para identificar rapidamente quais documentos contêm os termos procurados, acelerando significativamente o processo de recuperação de informações.

Os documentos do arquivo *cran.all.1400* é onde fazemos a busca de uma determinada consulta. Os documentos são divididos em 4 partes e cada parte possui um marcador:

1. **.I**: Além de indicar o início de um novo documento, indica o ID do documento com o número que vem logo após o identificador.
2. **.A**: Identifica o autor do documento.
3. **.B**: Identifica a bibliografia do documento.
4. **.W**: Identifica o conteúdo do documento.

Os marcadores são fundamentais para o processo de indexação dos documentos na base de dados CRANFIELD. Durante o processo de indexação, o arquivo cran.all.1400 foi lido linha por linha, e a cada ocorrência do marcador '.I', iniciava-se o processo de indexação de um novo documento.

Inicialmente, o ID do documento era extraído. Em seguida, o título era identificado usando o marcador '.T'. Antes de ser indexado, o título passava pela função de remoção de stop words. Esse mesmo procedimento era repetido para todas as seções do documento até a identificação de um novo documento através do marcador '.I'. Foi decidido que a bibliografia e autor não tinham necessidade de passar pelo processo de remoção de stop words.

Esse processo de indexação é repetido da mesma forma para os dois buscadores, com exceção de como esse os dados são salvos em cada índice. Cada motor de busca segue uma estrutura diferente e a forma de armazenar os índices precisou ser adaptada para cada motor:

- **Whoosh**: É uma biblioteca Python utilizada para indexação e busca de texto. Durante o processo de indexação no Whoosh:
    - Estrutura de Dados: Os documentos são representados como dicionários do Python, onde cada chave corresponde a um campo (como título, autor, conteúdo) e seus respectivos valores representam o conteúdo desses campos.
    - Definição do Esquema: É criado um esquema que especifica como os documentos serão indexados. Esse esquema define os campos, os tipos de análise de texto a serem aplicados (tokenização e etc.) e outras configurações relacionadas à estrutura do índice.
    - Processamento e Indexação dos Documentos: Cada documento é processado de acordo com o esquema definido. Durante esse processo, os textos são analisados, tokens são gerados e outras operações de processamento são aplicadas.
    - Armazenamento no Índice: As informações extraídas dos documentos são armazenadas no índice Whoosh, seguindo a estrutura definida pelo esquema. Isso inclui a criação de estruturas otimizadas para recuperação de informações.

- **Elasticsearch**: É um mecanismo de busca e análise de texto distribuído. Durante o processo de indexação no Elasticsearch:
    - Estrutura de Dados: Os documentos são representados no formato JSON (JavaScript Object Notation), onde cada documento é um objeto JSON com campos e seus respectivos valores.
    - Definição de Mapeamento: É definido um mapeamento que especifica como os documentos serão indexados. O mapeamento define os campos, os tipos de dados, análises de texto e outras configurações.
    - Análise e Indexação de Documentos: Cada documento é processado de acordo com o mapeamento definido. Durante esse processo, os textos são analisados, tokenizados e transformados em estruturas otimizadas para armazenamento e busca, conforme as configurações do mapeamento.
    - Armazenamento no Índice: As informações extraídas dos documentos são armazenadas nos índices Elasticsearch. Os índices contêm estruturas otimizadas para recuperação rápida de informações durante as buscas.

Tanto no Whoosh quanto no Elasticsearch, a indexação de documentos é realizada por meio de campos extraídos de marcadores, como mencionado anteriormente. Marcadores como ID, título, autor, bibliografia e conteúdo representam seções específicas dos documentos a serem indexados. Esses campos são definidos no esquema do Whoosh ou no mapeamento do Elasticsearch, proporcionando uma estrutura organizada para mapear e catalogar informações dos documentos. Essa organização facilita tanto a indexação quanto a busca, garantindo uma localização eficaz dos dados nos respectivos motores de busca.

3.2. Busca

Nos dois motores (Whoosh e Elasticsearch), a função de busca recebe consultas do arquivo cran.qry. Essas consultas são pesquisadas em todos os parâmetros do índice, incluindo campos como "content", "title", "author" e "bibliography". É importante notar que cada mecanismo de busca utiliza uma estrutura diferente para realizar essa busca, aplicando algoritmos e métodos distintos para recuperar e retornar os resultados das consultas dentro desses campos do índice.

Whoosh: A função *busca(search_string)* que implementa a busca utilizando a biblioteca Whoosh. Ele realiza os seguintes passos:

1. Abertura do Índice: Abre o índice dos documentos previamente criado pelo Whoosh.
2. Definição dos Campos: Define os campos nos quais deseja realizar a pesquisa, como "content", "title", "author" e "bibliography".

3. Parser de Consulta Multifield: Cria um parser de consulta capaz de pesquisar em múltiplos campos especificados.
4. Remoção de Stopwords: Remove as stopwords da consulta, se necessário.
5. Conversão da Consulta: Substitui espaços por " OR " na consulta. Essa operação é realizada porque por padrão o Whoosh executa uma busca com operador "AND", ou seja, tenta encontrar documentos que contenham exatamente todas as palavras fornecidas na consulta. Ao utilizar "OR" entre as palavras, a busca se torna mais flexível, permitindo encontrar documentos que contenham pelo menos uma das palavras na consulta, o que amplia os resultados potenciais da busca.
6. Realização da Busca: Realiza a busca utilizando o parser de consulta e o índice aberto. Itera pelos resultados encontrados, recuperando os IDs dos documentos correspondentes e os armazena em uma lista para retorno.

Elasticsearch: A função *busca(search_string)* que utiliza o Elasticsearch para a busca dos documentos. Ele segue os passos a seguir:

1. Especificação dos Campos: Define os campos onde a busca será realizada, como "content", "title", "author" e "bibliography".
2. Execução da Consulta: Realiza a consulta no Elasticsearch, buscando nos campos especificados.
3. Processamento dos Resultados: Itera pelos resultados encontrados, adicionando os IDs únicos dos documentos encontrados a um conjunto para evitar duplicatas. Retorna uma lista dos IDs únicos dos documentos encontrados.

Essencialmente, ambas as funções busca realizam buscas nos campos especificados, porém, utilizando métodos e estruturas diferentes fornecidas pelas respectivas bibliotecas (Whoosh e Elasticsearch) para realizar as consultas e processar os resultados.

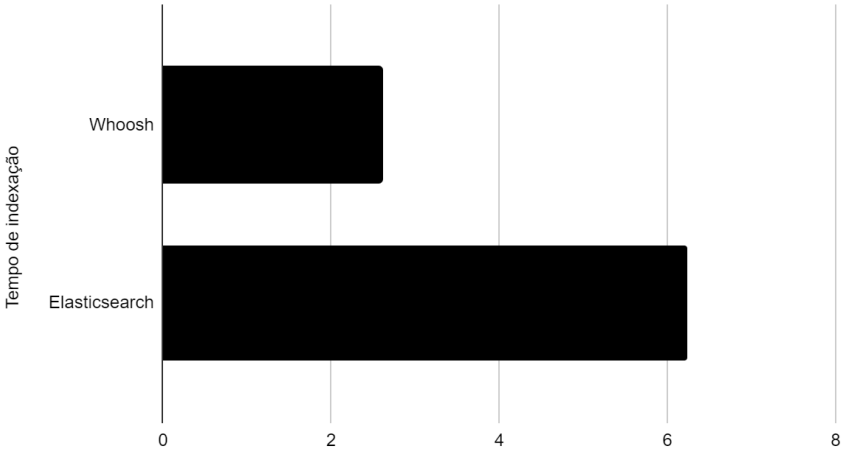
4. Resultado

Como como métrica de resultados foi utilizado o tempo de execução da indexação de cada motor, o tempo de busca e precisão.

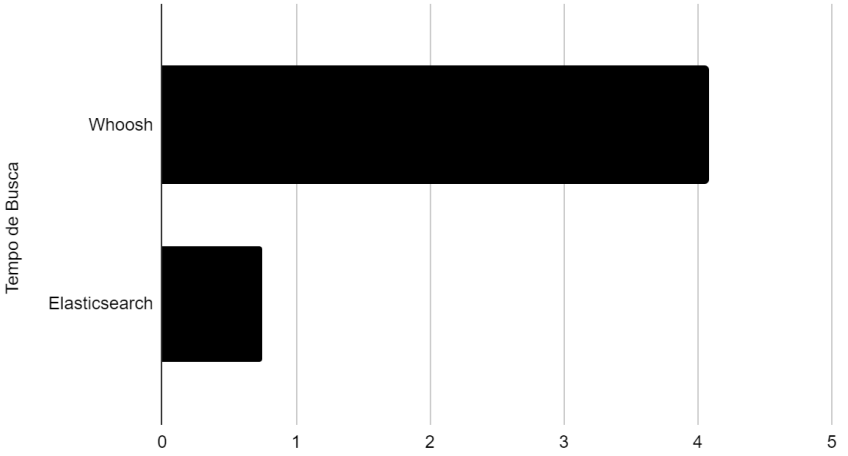
O Whoosh demorou 2.6020 segundos para indexar a base de dados e 4.0752 segundos com as buscas. No total foram 6,6772 segundos de execução.

O Elasticsearch demorou 6.2420 segundos para indexar a base de dados e 0.7490 segundos com as buscas. No total foram 6,991 segundos de execução.

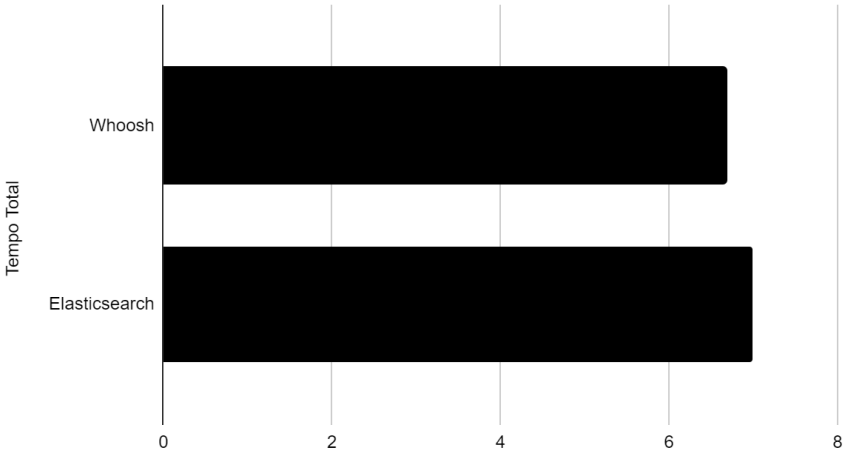
Tempo de indexação



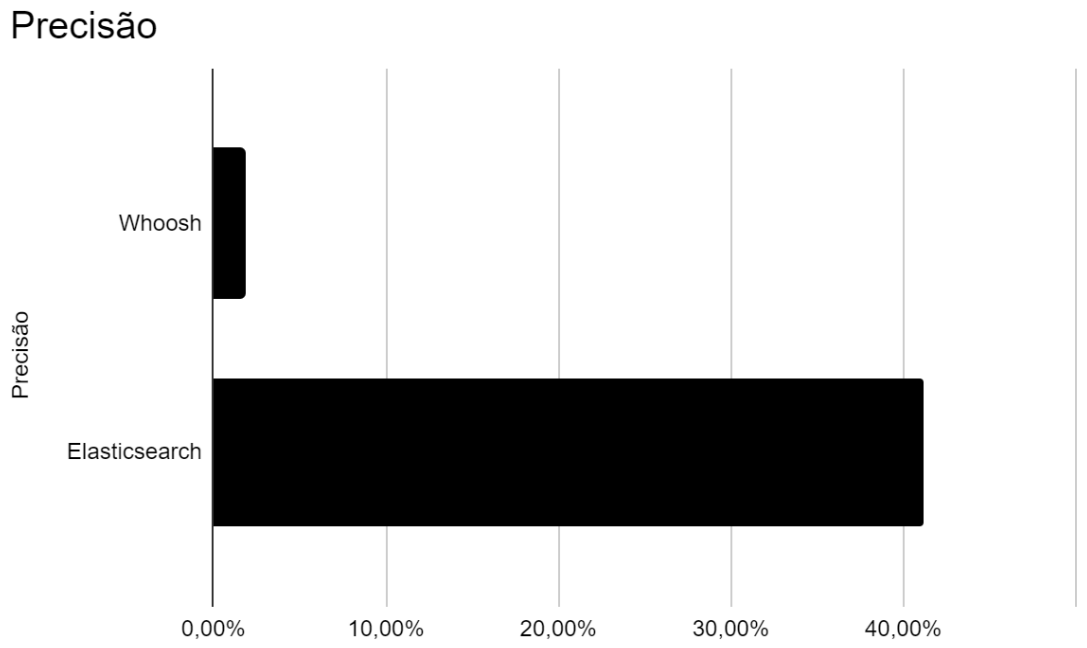
Tempo de Busca



Tempo Total



Para calcular a precisão pegamos a total de documentos que o motor acertou que seria relevante para a consulta e dividimos pelo total que ele retornou. O Whoosh apresentou uma precisão de 1,48% e o Elasticsearch 41%



5. Conclusão

Um ponto importante em comparação aos motores é que o Elasticsearch é mais complexo em termos de implementação quando comparado ao Whoosh. A necessidade de instalar, configurar e gerenciar um servidor para o Elasticsearch pode tornar sua utilização inicial mais desafiadora em comparação com o Whoosh, que é uma biblioteca Python mais simples de usar, não exigindo um servidor dedicado.

Considerando os dados coletados sobre o desempenho do Whoosh e do Elasticsearch, é possível traçar algumas conclusões:

Tempo de Indexação e Busca:

- O Whoosh apresentou um tempo de indexação mais rápido em comparação com o Elasticsearch. Por outro lado, o tempo de busca do Whoosh foi significativamente maior.
- Apesar do tempo de busca mais longo no Whoosh, o tempo total de execução foi ligeiramente menor. Isso indica que o impacto do tempo de indexação no Elasticsearch foi mais significativo na execução total.

Desempenho na Precisão dos Resultados:

- O Whoosh teve uma taxa de acerto extremamente baixa, acertando menos de 2% dos documentos relevantes. Isso indica um desempenho de precisão muito ruim na identificação de documentos relevantes para as consultas realizadas.
- Por outro lado, o Elasticsearch teve um desempenho mais sólido, com uma taxa de acerto de 41%. Isso sugere que o Elasticsearch foi capaz de encontrar uma porcentagem significativa de documentos relevantes nas consultas.

Considerações Finais:

- Embora o Whoosh tenha tido vantagens em termos de tempo de indexação mais rápido e tempo total de execução um pouco menor, seu desempenho na precisão dos resultados foi extremamente fraco.
- O Elasticsearch, apesar de ter um tempo de indexação mais longo, mostrou-se significativamente mais preciso na identificação de documentos relevantes durante as consultas.

Além da precisão superior do Elasticsearch, vale ressaltar que, mesmo com um tempo total de execução atualmente maior em comparação com o Whoosh, à medida que o número de consultas ou buscas aumenta em dados já indexados, o Elasticsearch pode mostrar uma vantagem considerável no tempo total de execução. Isso ocorre porque sua busca do Elasticsearch se mostrou 5 vezes mais rápida que a do whoosh. Com a precisão consideravelmente maior e uma capacidade escalável para lidar com um volume maior de buscas, o Elasticsearch tende a se destacar não apenas em termos de precisão, mas também em eficiência quando o número de consultas aumenta.

Referências

Whoosh. Whoosh: Uma biblioteca de busca em texto para Python. Disponível em: https://whoosh.readthedocs.io/en/latest/. Acesso em: 15 de dezembro de 2023.

Elastic NV. Elasticsearch: Uma ferramenta de busca e análise distribuída. Disponível em: https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html. Acesso em: 16 de dezembro de 2023.