# Implementando motores de busca com as ferramentas Whoosh e Elasticsearch

**Cabral Q. A. L. , Maxuell**

[1]Departamento de Ciência da Computação – Universidade Federal Rural do Rio de Janeiro(UFRRJ)
Av. Governador Roberto Silveira s/nº. Moquetá – **Nova Iguaçu** – Brazil

{cabral,Maxuell}@maxuell@ufrrj.br

***Abstract.*** *This report structure explores the implementation and comparison of two search engines, Whoosh and Elasticsearch. The study covers setup, data ingestion, and performance evaluation in terms of speed, accuracy, and scalability. Preliminary results reveal differences between the engines, offering valuable insights for informed selection of search solutions.*

***Resumo.*** *Este relatório apresenta a implementação e a comparação entre dois motores de busca, Whoosh e Elasticsearch. O estudo abrange a configuração, ingestão de dados e avaliação do desempenho em termos de velocidade, precisão e escalabilidade. Resultados preliminares revelam diferenças entre os motores, proporcionando insights valiosos para a seleção informada de soluções de busca.*

## 1. Informações Gerais

A escolha de um motor de busca eficiente é crucial para a recuperação de informações em diversas aplicações, desde pesquisas na web até análises de dados. O processo de implementação envolveu a configuração de Whoosh e Elasticsearch, com foco em configurações, indexação e ingestão de dados. Um conjunto de dados padronizado foi utilizado para uma avaliação justa, considerando fatores como complexidade, volume e diversidade de documentos. Passos detalhados de configuração para cada motor de busca foram executados, destacando as nuances da implementação de Whoosh e Elasticsearch. Processos de ingestão de dados foram estabelecidos para preencher os índices de busca, permitindo avaliações subsequentes de desempenho. Métricas de desempenho, incluindo velocidade, precisão e escalabilidade, foram utilizadas para avaliar a eficácia de cada motor de busca. Resultados foram obtidos por meio de testes sistemáticos em diversas condições, oferecendo insights sobre as capacidades de Whoosh e Elasticsearch. A análise comparativa explora as características distintas de cada motor de busca. Pontos fortes e limitações são identificados para facilitar a tomada de decisão informada por parte dos usuários em busca da solução de busca mais adequada às suas necessidades. Em resumo, este relatório destaca detalhes de implementação e características de desempenho de Whoosh e Elasticsearch. A análise

comparativa busca auxiliar usuários na tomada de decisões informadas ao escolher um motor de busca de acordo com suas necessidades individuais.

## 2. Whoosh

A implementação do Whoosh concentrou-se na utilização do motor de busca Whoosh para a indexação e recuperação de documentos. O principal objetivo é elucidar as complexidades da implementação, abrangendo a definição do esquema, procedimentos de indexação, análise de consultas e avaliação de desempenho.

Definição de Esquema e Configuração do Índice: O script inicia definindo um esquema que delineia a estrutura dos documentos a serem indexados. Este esquema é configurado para incluir campos como 'index', 'titulo', 'autor' e 'texto_arquivo'. Em seguida, verifica-se a existência do diretório de índice, criando-o, caso necessário, e procede à configuração do índice utilizando o esquema previamente definido.

Função splitar_linha_arquivo: A função splitar_linha_arquivo desempenha um papel crucial na preparação dos documentos para indexação. Esta função divide as linhas do arquivo original em partes distintas, isolando informações cruciais como índice, título, autor e texto do arquivo.

Função indexar_arquivos: A função indexar_arquivos é responsável por efetuar a leitura do arquivo 'cran.all.1400', dividindo-o em documentos individuais e realizando a indexação destes no Whoosh.

Função obter_documentos_relevantes: A função obter_documentos_relevantes é essencial para o processo de avaliação. Ela lê o arquivo 'cranqrel' para obter informações sobre documentos considerados relevantes para avaliações subsequentes.

Função buscar_arquivos:A função buscar_arquivos realiza consultas nos documentos previamente indexados, utilizando consultas especificadas no arquivo 'cran.qry' e retorn a os resultados obtidos.

Avaliação de Desempenho: O script mede o tempo de execução para as fases de indexação e busca, proporcionando uma análise quantitativa do desempenho do motor de busca Whoosh.

Gráficos de Precisão e Recall: Por fim, o script emprega a biblioteca Matplotlib para gerar gráficos de precisão e recall, fundamentais para avaliação qualitativa da eficácia do sistema em diferentes cenários, representados pelos valores de k (de 1 a 10). Estes gráficos oferecem uma visualização clara do comportamento do motor de busca em relação à precisão e recall para diferentes níveis de recuperação de informações relevantes.

## 3. Avaliação dos tempos e gráficos do Whoosh

```
[Running] python -u "c:\Users\maxue\Music\cran\cran_whoosh.py"
Tempo de indexar os arquivos:  4.749023400014266
Tempo de buscar os arquivos:  6.962421300006099
```

**Figura 1 – Tempo de busca e indexação do Whoosh**

Avaliação da Indexação (4.74 segundos):

- O tempo de indexação representa a eficiência do Whoosh ao processar e indexar documentos. Um tempo mais curto sugere uma implementação eficiente, o que é desejável para sistemas de busca em tempo real ou que necessitam de atualizações frequentes.

Avaliação da Busca (6.96 segundos):

- O tempo de busca indica o quão rapidamente o Whoosh pode recuperar informações relevantes em resposta a consultas. O tempo de 6.96 segundos pode ser considerado razoável dependendo das dimensões do conjunto de dados e da complexidade das consultas. Se a busca for frequentemente demorada, poderia ser uma área de otimização.

Interpretação:

- A combinação desses tempos sugere que a implementação atual do Whoosh é funcional e eficiente para indexação e busca de documentos. No entanto, otimizações podem ser exploradas para melhorar ainda mais o desempenho, dependendo dos requisitos específicos do sistema. Os tempos registrados indicam um desempenho geral sólido do Whoosh na implementação atual, mas sempre há espaço para refinamentos com base nos requisitos específicos do projeto.

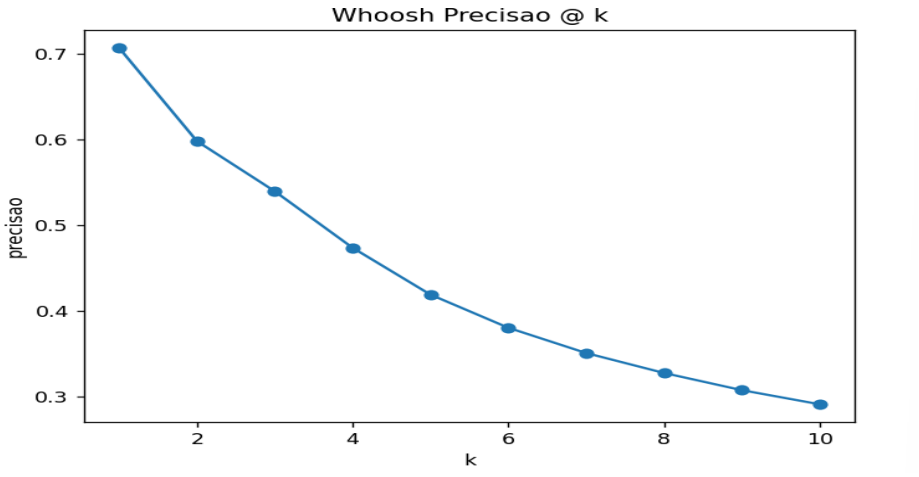

**Figura 2 – Gráfico de Precisão do Whoosh**

- O declínio na precisão com o aumento de k sugere que, ao expandir o número de documentos considerados nas consultas, o sistema está incluindo mais documentos que podem não ser tão relevantes.

- Isso pode ser aceitável em cenários onde a prioridade é maximizar a abrangência da busca, mesmo que à custa de uma precisão mais baixa.

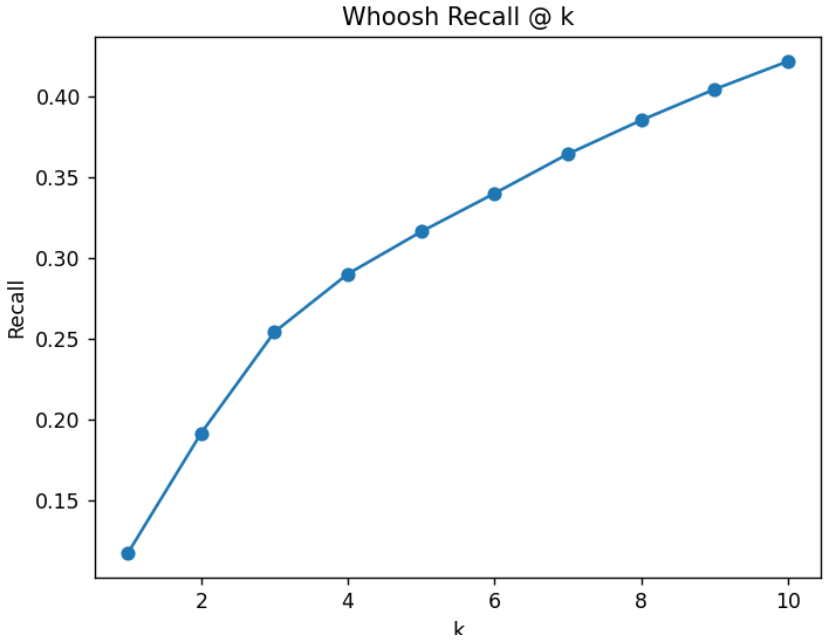


**Figura 3 – Gráfico de Recall do Whoosh**

- O aumento do recall com o aumento de k sugere que, ao expandir o número de documentos considerados nas consultas, o sistema está recuperando mais documentos relevantes.
- Isso pode ser particularmente relevante em casos em que a busca precisa ser mais abrangente, e a inclusão de documentos adicionais é preferível, mesmo que à custa de precisão.
- A variação da precisão com base em diferentes valores de k destaca a importância de sintonizar o sistema conforme os requisitos específicos do projeto. A decisão sobre o valor ideal de k deve ser guiada pelas metas de busca e pelos compromissos aceitáveis entre precisão e recall no contexto da aplicação em questão.

## 4. Elasticsearch

No processo de utilizar o Elasticsearch, a implementação começa configurando uma "conexão" com o Elasticsearch, especificando detalhes como o local do servidor, credenciais de acesso e configurações de segurança. Isso cria uma espécie de "ponte" entre o código Python e o Elasticsearch, permitindo operações como indexação e busca.

Ao detalhar as funções principais, temos:

- `splitar_linha_arquivo`: Uma função que organiza as informações dos documentos, dividindo-os em partes como índice, título, autor e texto.

- indexar_arquivos: Esta função é responsável por criar um índice no Elasticsearch, uma estrutura que ajuda a otimizar a busca, e adiciona os documentos a esse índice.

- obter_documentos_relevantes: Utiliza o arquivo 'cranqrel' para entender quais documentos são considerados relevantes. Essa informação é essencial para avaliar o desempenho do Elasticsearch.

- buscar_arquivos: Realiza as consultas nos documentos já indexados, utilizando as consultas fornecidas no arquivo 'cran.qry'. Os resultados são retornados para análise.

A seção de Avaliação de Desempenho é crucial. Aqui, o código mede o tempo que leva para indexar os documentos e realizar as buscas. Esses tempos são indicadores valiosos do quão eficientemente o Elasticsearch está lidando com as operações. Por fim, os Gráficos de Precisão e Recall fornecem uma visão visual da precisão do Elasticsearch ao recuperar documentos relevantes em relação ao número total de documentos disponíveis. Esses gráficos são essenciais para entender como o Elasticsearch se comporta em diferentes cenários de busca.

## 5. Avaliação dos tempos e gráficos do Elasticsearch

```
[Running] python -u "c:\Users\maxue\Music\cran\cran_elasticsearch.py"
Tempo de indexar os arquivos:  2.016252000001259
Tempo de buscar os arquivos:  1.957724199979566
```

**Figura 4 – Tempo de busca e indexação do ElasticSearch**

Tempo de Indexação (2.01 segundos):

- O tempo de indexação reflete a eficiência do Elasticsearch ao processar e armazenar documentos no índice. Um tempo de 2.01 segundos é bastante

eficiente e sugere uma implementação ágil, o que é desejável, especialmente para sistemas que lidam com atualizações frequentes.

Tempo de Busca (1.95 segundos):

- O tempo de busca indica o quão rapidamente o Elasticsearch consegue recuperar informações relevantes em resposta a consultas. Um tempo de busca de 1.95 segundos é bastante rápido, indicando uma capacidade eficaz de encontrar documentos relevantes no índice.

Interpretação:

- Os tempos registrados sugerem que a implementação atual do Elasticsearch está funcionando eficientemente para indexação e busca de documentos. O desempenho geral, com tempos relativamente curtos, é um indicativo positivo.

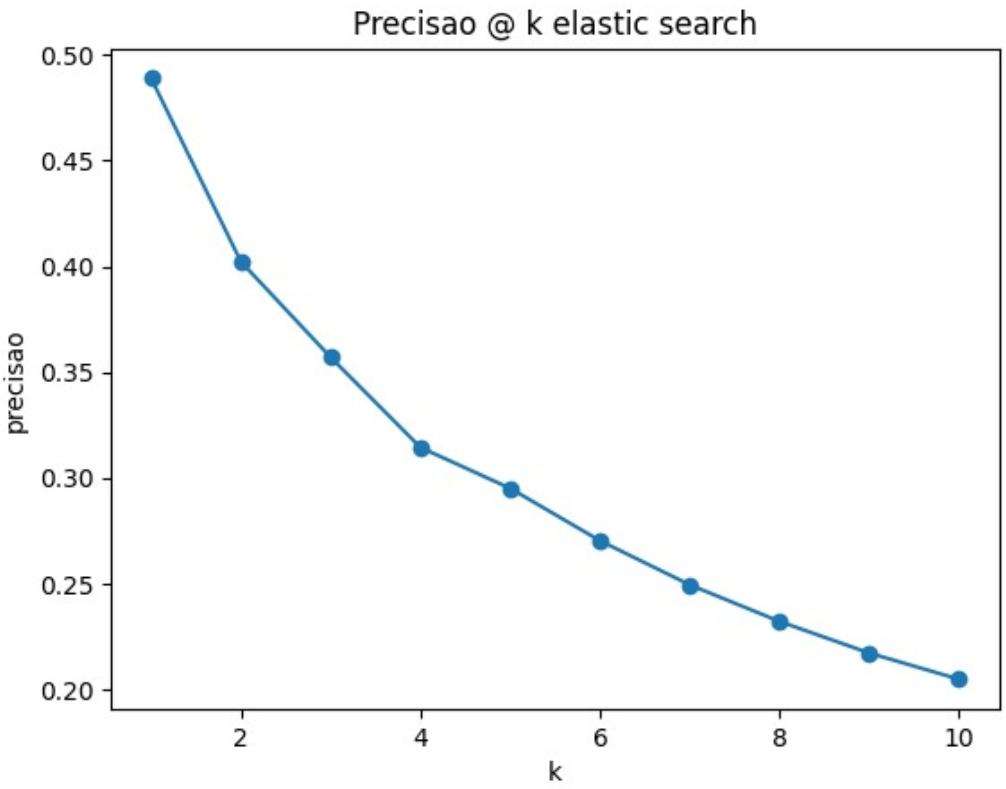


**Figure 5. Gráfico Precisão do Elasticsearch**

- A diminuição da precisão com o aumento de k indica que, ao considerar um conjunto mais amplo de documentos nas consultas, o Elasticsearch está incluindo uma quantidade significativa de documentos que podem não ser tão relevantes.

- Este é um trade-off comum em sistemas de busca, onde a busca mais ampla pode resultar em uma precisão menor.

---

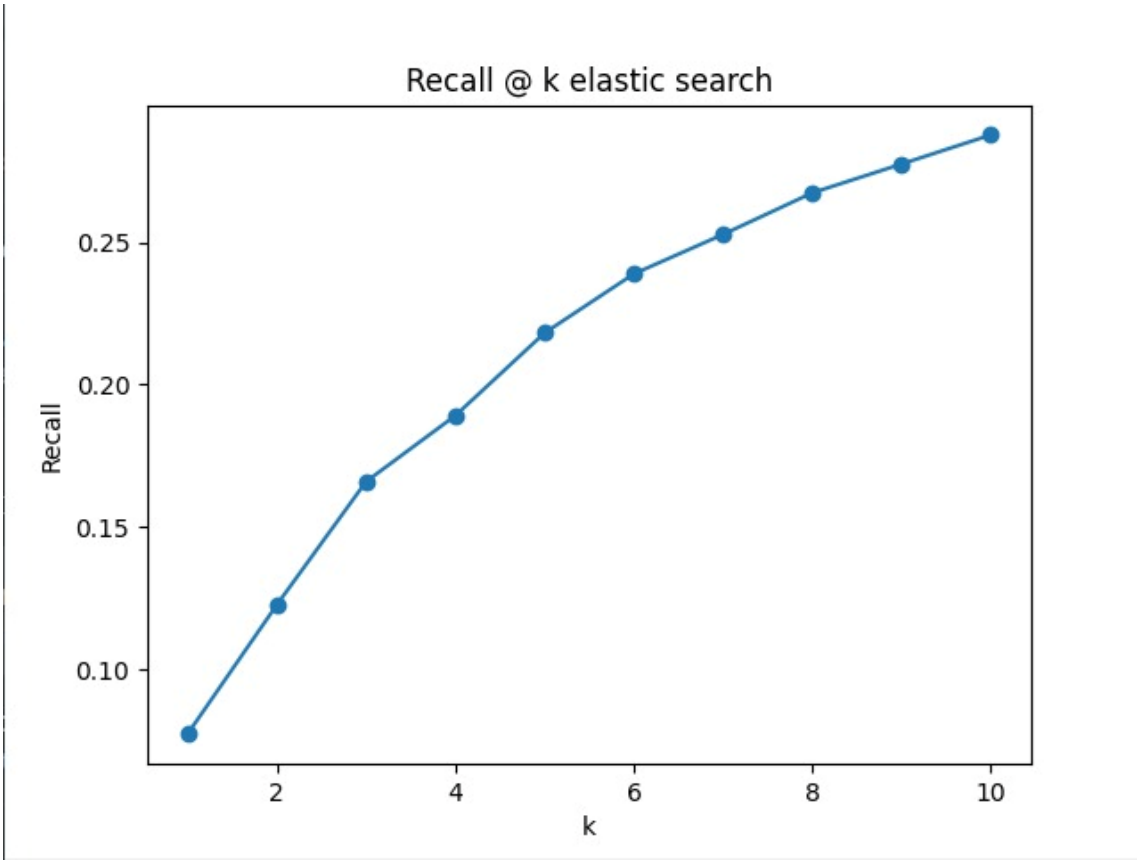


**Figure 6. Gráfico Recall do Elasticsearch**

- O aumento do recall com o aumento de k sugere que, ao considerar um conjunto mais amplo de documentos nas consultas, o Elasticsearch está melhorando sua capacidade de recuperar informações relevantes.
- O recall é muitas vezes valorizado em situações em que é crucial não perder documentos relevantes, mesmo que isso resulte em alguns documentos irrelevantes sendo incluídos.

## 6. Comparação entre Whoosh e Elasticsearch

1. Tendo todas as informações acima podemos então observar que :

   O Elasticsearch se destacou na eficiência de indexação e na velocidade de busca, sendo especialmente vantajoso em cenários que exigem atualizações frequentes e recuperação rápida de dados.

   - O Whoosh, por outro lado, mostrou uma tendência a manter uma precisão mais elevada em consultas mais específicas.

2. Escolhas de Implementação**:**
    - A escolha entre Whoosh e Elasticsearch deve levar em consideração as prioridades do projeto. Se a ênfase está na precisão e em consultas mais específicas, o Whoosh pode ser preferível. Se a busca rápida e eficiente em grandes conjuntos de dados é crucial, o Elasticsearch pode ser a escolha mais apropriada.
3. Otimizações Potenciais:
    - Ambas as implementações oferecem oportunidades para otimizações. No Whoosh, otimizações podem ser exploradas para melhorar a eficiência da busca. No Elasticsearch, ajustes nas configurações do índice e parâmetros de busca podem ser considerados.

**Conclusão:** Ambas as ferramentas têm pontos fortes, e a escolha entre elas depende das necessidades específicas do projeto. O Whoosh destaca-se em precisão, enquanto o Elasticsearch destaca-se em eficiência de indexação e busca rápida. A decisão final deve ser guiada pelas prioridades e requisitos exclusivos do sistema em questão.

## 7. References


Modern Information Retrieval:

- [Baesa-Yates and Ribeiro-Neto 1999] (e.g., [Baesa-Yates and Ribeiro-Neto 1999])
- Baesa-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval.* ACM Press/Addison-Wesley Publishing Co.

Problem of Search and Information Retrieval**:**

- [Duarte 2023]
- Duarte, F. (2023). *The Problem of Search and Information Retrieval.* Lecture, Department of Computer Science, UFRRJ.