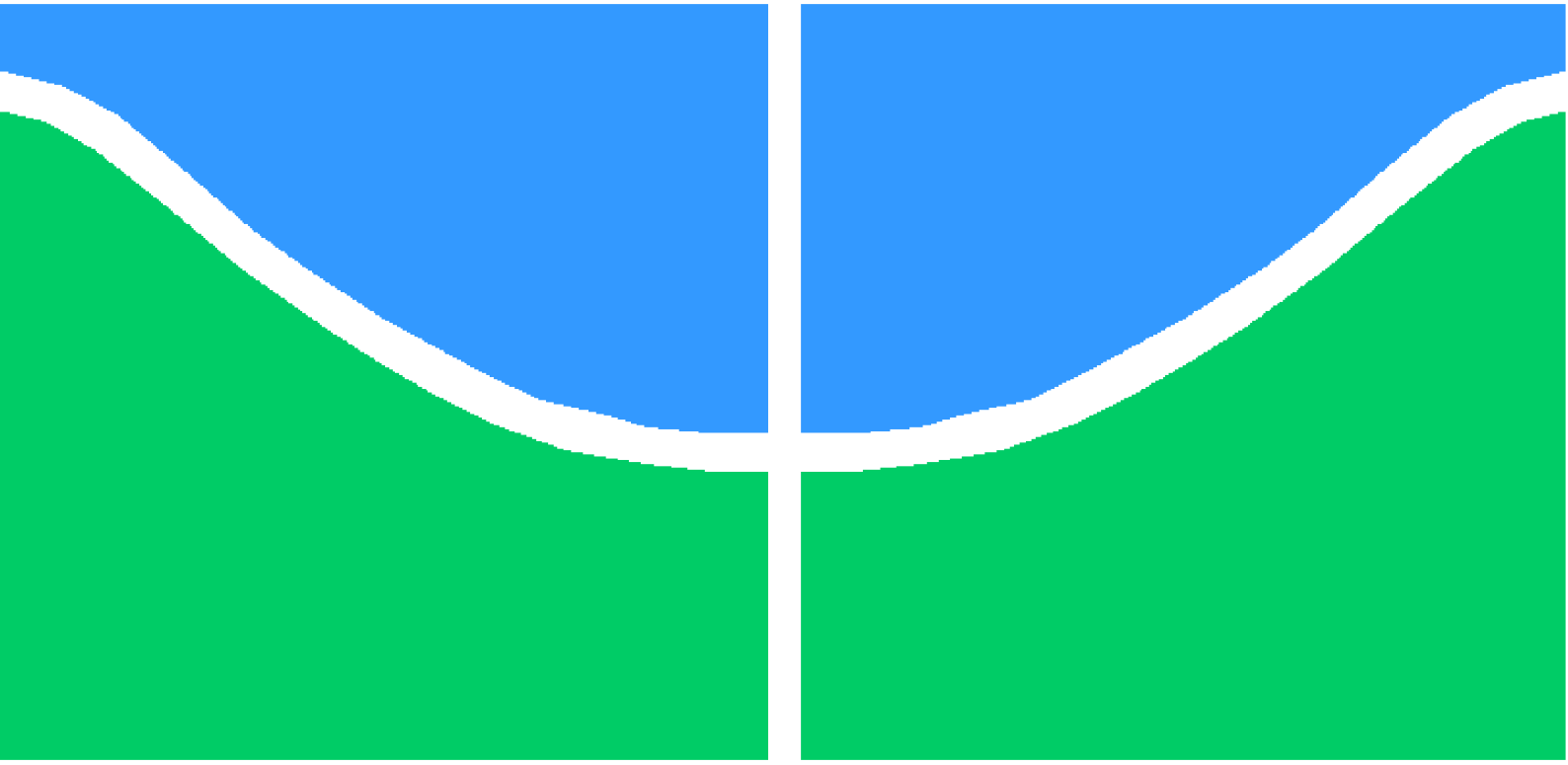

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

# AVALIAÇÃO DE ALGORITMOS PARA ORDENAÇÃO DE DOCUMENTOS DIGITAIS RÉCUPERADOS EM BUSCA

Autor: Greg Ouyama Martins

Orientador: Professor Doutor Sérgio Antônio Andrade de Freitas

Brasília, DF

2013

Greg Ouyama Martins

# AVALIAÇÃO DE ALGORITMOS PARA ORDENAÇÃO DE DOCUMENTOS DIGITAIS RECUPERADOS EM BUSCA

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Professor Doutor Sérgio Antônio Andrade de Freitas

Brasília, DF

2013

Greg Ouyama Martins
AVALIAÇÃO DE ALGORITMOS PARA ORDENAÇÃO DE DOCUMENTOS DIGITAIS RECUPERADOS EM BUSCA/ Greg Ouyama Martins. – Brasília, DF, 2013-
111 p. : il. (algumas color.) ; 30 cm.

Orientador: Professor Doutor Sérgio Antônio Andrade de Freitas

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2013.

1. Palavra-chave01. 2. Palavra-chave02. I. Professor Doutor Sérgio Antônio Andrade de Freitas. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. AVALIAÇÃO DE ALGORITMOS PARA ORDENAÇÃO DE DOCUMENTOS DIGITAIS RECUPERADOS EM BUSCA

CDU 02:141:005.6

Greg Ouyama Martins

# AVALIAÇÃO DE ALGORITMOS PARA ORDENAÇÃO DE DOCUMENTOS DIGITAIS RECUPERADOS EM BUSCA

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 01 de junho de 2013:

---

**Professor Doutor Sérgio Antônio Andrade de Freitas**
Orientador

---

**Professor Doutor Edson Alves da Costa Júnior**
Convidado 1

---

**Professor Doutora Carla Silva Rocha Aguiar**
Convidado 2

Brasília, DF
2013

*Este trabalho é dedicado aos meus, pais, professores e amigos que me acompanharam durante meu tempo de graduação e contribuíram de forma positiva com meu crescimento e aprendizado.*

*"Depois de tudo que passamos.*
*Tudo que eu fiz.*
*Não pode ter sido em vão"*
*(Ellie, in The Last of Us)*

# Resumo

A busca de informação teve seu princípio através de bibliotecas, recuperando o que o usuário viria a necessitar através de consultas por meio de, por exemplo, cartões de catálogos, categorizando os livros por título, por autor, ano ou editora dos livros. Com o avanço da tecnologia, ocorreu a automação deste processo, fazendo com que esse tipo de tarefa fosse realizada através de um computador. Entretanto com o grande volume de informação disponível, nem sempre é fácil encontrar o que se procura com eficácia, tornando assim a atividade de busca cansativa e trabalhosa. Para tratar este problema existem estudos e implementação a respeito da ordenação de informação obtida através da recuperação de informação. É interessante também a adoção técnicas para realizar consultas personalizadas, de acordo com características pré-estabelecidas pelos usuários do motor. Através de estudos acerca de algoritmos dinâmicos, baseados em termos e estáticos de ordenação da recuperação da informação, o objetivo deste trabalho é analisar o que existe tratando de ordenação da recuperação de informação. Juntamente com a inserção de perfis para buscar um grau de personalização das consultas em conjunto com um motor de busca *open source*, será analisado qual dos algoritmos é mais preciso, através de métricas de precisão x *recall*, quais algoritmos permitem consulta com um grau de personalização, aceitando inserção de perfil das quatro engenharias da Universidade de Brasília – Faculdade do Gama, e como a engenharia de software pode contribuir com a ordenação da recuperação de informação.

**Palavras-chaves**: busca. consulta. recuperação de informação. motor de busca. index. ordenação

# Abstract

The information retrieval had its beginning through libraries, seeking what the user would need by searching through, for example, cards catalogs, categorizing the books by their title, author, year or their publishing house. With the technological progress, this searching process was automated making this kind of searching was made through some computer. However, with the large volume of information available, it is not always easy to find what you are looking effectively, thus making the search activity tiresome and hard task. To address this problem, there are researches and implementation about ranking the information obtained from the information retrieval. It is also interesting to adopt techniques for performing custom queries in accordance with predetermined characteristics by users of the engine. Through studies about dynamics, term based, and statics ranking algorithms, the main objective of this work is to analyze what exists about ranking the retrieval information. Along with the inclusion of profiles to find a degree of retrieval customization in conjunction with a search engine open source, which will be analyzed is more accurate algorithms, through metrics of precision and recall, algorithms which allow consultation with a degree of customization, accepting insertion profile of the four engineering University of Brasilia - Faculty of Gama's graduation courses, and how software engineering can contribute to the ordering of information retrieval.

**Key-words**: Query, Information Retrieval, Search Engine, Index, Ranking.

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| DFR | *Divergence From Randomness* |
| BM25 | *Best Match* |
| DFI | *Divergence From Independence Model* |
| DFRee | *Divergence From Randomness free from parameters* |
| DLH | DFR *model based on Hypergeometric and Laplace Normalization* |
| PL2 | *Poisson model and Euclidean Norm* |
| HITS | *Hypertext Induced Topic Search* |
| SALSA | *The Stochastic Approach for Link-Structure Analysis* |
| FGA | *Faculdade do Gama* |

# Sumário

# 1 Introdução

A quantidade de informação que pode ser obtida por meio da internet ou bibliotecas é muito grande, quando comparado com tempos em que o nível tecnológico não era muito avançado. Atualmente é muito fácil encontrar alguma informação que esteja procurando na internet.

Um motor de busca é o sistema, que com a união de seus componentes, realiza o trabalho de recuperar a informação de uma determinada consulta do usuário. Entretanto, mesmo com um motor de busca, algumas vezes o resultado de uma consulta não é o que o usuário realmente espera. É necessário garantir que a informação recuperada seja aquela que o usuário realmente espera, ou que pelo menos seja uma solução próxima para a sua consulta por meio do componente de ordenação. Além disso, a maioria dos motores de busca existentes não oferecem um meio de personalização da busca, e quando oferecem, é por meio de dados armazenados de consultas previamente realizadas.

Através deste trabalho, será realizado a avaliação de performance de alguns algoritmos de ordenação, analisando quais deles possuem resultados mais precisos e quais aderem à implementação de perfis, para oferecer um nível de personalização nas consultas, sendo possível realizar a seleção prévia do perfil que se deseja.

A metodologia utilizada nesta primeira etapa do trabalho foi estudar uma bibliografia considerada clássica no meio (YATES; NETO et al., 1999), entender o que é a recuperação de informação, como funciona um motor de busca para em seguida realizar o estudo a respeito dos componentes que fazem parte do motor de busca: o rastreador web, o index e a ordenação. Após a etapa de estudos iniciais, foi decidido que o componente de ordenação seria o alvo da segunda etapa do trabalho.

Na segunda etapa do trabalho foram realizados testes com alguns dos algoritmos por meio de uma ferramenta de busca já implementada, e utilizada como *framework*, tendo apenas o componente de ordenação substituído de acordo com a necessidade.

Estes testes foram realizados para analisar a precisão de uma consulta entre os algoritmos de ordenação e verificar se aceitavam ou não a implementação de perfis. Para fazer os testes foi necessário implementar classes de perfis dentro da ferramenta, escolhendo perfis consistentes e reais para que a experimentação possua dados reais, que são as engenharias da Universidade de Brasília – Faculdade do Gama, através da análise curricular de cada curso. Implementados os perfis, foi necessário estabelecer pesos diferentes para cada uma das áreas de conhecimento dos currículos distintos. Assim foi possível determinar os termos que seriam utilizados para as três consultas, podendo abranger diferentes perfis, realizadas com os diferentes pesos.

A análise dos algoritmos foi realizada através de tabelas e gráficos gerados por consultas pré-determinadas de acordo com a métrica de precisão x *recall*. Assim de acordo com a quantidade de curvas, variação das curvas e dados de precisão, foi possível observar a performance dos perfis dentro de cada um dos algoritmos selecionados.

O trabalho está estruturado com segundo capítulo abordando como o motor de busca funciona e seus componentes, descrevendo o que é um rastreador web, três operações de texto que se mostram interessantes para o contexto de motor de busca e a descrição da ordenação de recuperação de informação. Em seguida, no terceiro capítulo, serão apresentados alguns algoritmos de ordenação que possuem uma base consolidada dentre os algoritmos de busca existentes. Posteriormente, no capítulo quatro, haverá o desenvolvimento do experimento, juntamente com a análise dos algoritmos. E por fim, no último capítulo, serão apresentados os algoritmos com melhor desempenho.

# 2 Motor de Busca

## 2.1 Introdução

Há relatos de que o ser humano guarda e busca informação há cerca de 4000 anos (YATES; NETO et al., 1999). No decorrer do tempo as informações vão se acumulando e consequentemente tornando-se volumosas. Com essa grande quantidade de informação, torna-se cada vez mais difícil encontrar a informação que se procura com exatidão e eficiência.

Atualmente as bibliotecas não são exatamente a primeira fonte de informação que as pessoas buscam, seja pela possível dificuldade de locomoção ou pela dificuldade de encontrar informação buscada quando se compara com outros meios que se encontram à disposição. As bibliotecas foram um dos primeiros lugares a possuir um gigantesco arsenal de informação e junto com esse arsenal, a difícil tarefa de organizar e localizar os livros. Para tentar contornar esse tipo de problema, era feita catalogação bibliográfica, que é um registro com todos os itens bibliográficos de uma ou várias bibliotecas, e dentro dessa catalogação bibliográfica, haviam itens físicos individuais nomeados cartões catálogos. Os cartões catálogos eram fichas que continham o nome do autor, título da obra, edição, editora, ano de publicação e o ISBN da obra.

Obviamente, com o decorrer do tempo, a quantidade de informação e de livros vem aumentando, tornando cada vez maior o número de livros nas bibliotecas, e assim, maior a quantia de cartões catálogo. Mesmo com organização, a tarefa de buscar o cartão catálogo desejado demandava tempo e esforço, com o advento da tecnologia, desnecessários. Assim a biblioteca tornou-se uma das primeiras instituições a fazer uso da tecnologia para realização de buscas através de um motor de busca. Inicialmente os motores de busca utilizados em bibliotecas eram elaborados por universidades, mas posteriormente eram feitos por empresas (YATES; NETO et al., 1999).

As buscas em bibliotecas são marcadas por três gerações. A primeira geração consiste basicamente na automação dos cartões catálogo, porém com buscas muito limitadas (apenas no nome do autor e título do livro). Já na segunda geração, preocupou-se com a implementação de novas funcionalidades de buscas, permitindo que seja realizada a busca por keywords, assunto do título, entre outros. E por fim, a terceira e atual geração está ainda em desenvolvimento e tem como objetivo a melhoria de interfaces gráficas, formas eletrônicas, características em hypertextos e arquitetura aberta (YATES; NETO et al., 1999) (SALTON; MCGILL, 1986).

Juntamente com o desenvolvimento da tecnologia e com o surgimento da web, as

informações, que antes eram tão procuradas em bibliotecas, podem ser acessadas por meio da internet nas mais diversas fontes. Sendo assim é possível levar em consideração que na internet há muito mais quantidade de informação, não necessariamente aproveitável ou verídica, do que nas bibliotecas. Há basicamente três fatores que juntamente ao avanço da computação contribuíram para tornar a internet uma conhecida fonte e para o avanço nas buscas na internet. O primeiro fator é que se tornou mais barato ter acesso a várias fontes de informações, sejam elas por meio de jornais, revistas, radio ou mesmo na biblioteca e na internet. O segundo fator é o avanço em todos os tipos de comunicação digital, atualmente é possível encontrar dispositivos com preços acessíveis e com acesso à rede, mostrando que é possível encontrar a informação que se deseja onde quer que a pessoa esteja. E por fim, o terceiro e último fator foi a liberdade de publicar qualquer informação, como já citado, seja ela aproveitável ou não, verídica ou não, aumentando significativamente a quantidade de informação e a popularidade da web (YATES; NETO et al., 1999).

A seguir será apresentado como é estruturado um motor de busca, seus componentes, alguns dados curiosos a respeito dos motores de busca atuais, alguns que já foram criados há algum tempo, que já não fazem mais parte dos motores de busca ativos, e uma descrição breve da importância e do funcionamento de cada um dos componentes do motor de busca.

## 2.2 Estrutura de um Motor de Busca

Com o aumento da popularidade da internet juntamente com o aumento de informação, se viu mais que necessário uma forma eficiente e eficaz de recuperar a informação que se deseja. Para isso é necessário a utilização de um motor de busca.

Um motor de busca é um sistema de recuperação de informação. Ele construído especificamente para encontrar dados, locais ou na rede, de acordo com o desejo e a intenção do usuário por meio de critérios específicos programados dentro de um dos algoritmos que o compõem, de forma rápida e organizada. Há diferenças entre um motor de busca padrão e um motor de busca web. Uma das principais diferenças entre as duas é que no motor de busca web, todas as consultas (*queries*) devem ser executadas sem acessar o próprio texto, acessados apenas pelo index, que logo será explicado. Se não fosse o index o responsável por acessar o texto, seria necessário armazenar todas as páginas web no seu próprio computador para realizar a consulta, o que é completamente inviável e desnecessário, ou fazer com que a pesquisa fosse realizada em todas as páginas web durante o tempo que foi realizado a consulta, sem que houvesse um index pronto, ocasionando em uma lentidão intensa (YATES; NETO et al., 1999). Os motores de busca web utilizam-se da técnica de indexação muito similar as usadas nas bibliotecas do século passado (YATES; NETO et al., 1999) (SALTON; MCGILL, 1986), mas mesmo assim são necessários três

componentes operando simultaneamente para que seu funcionamento seja harmônico. A Figura 1 mostra como é estruturado um motor de busca web.

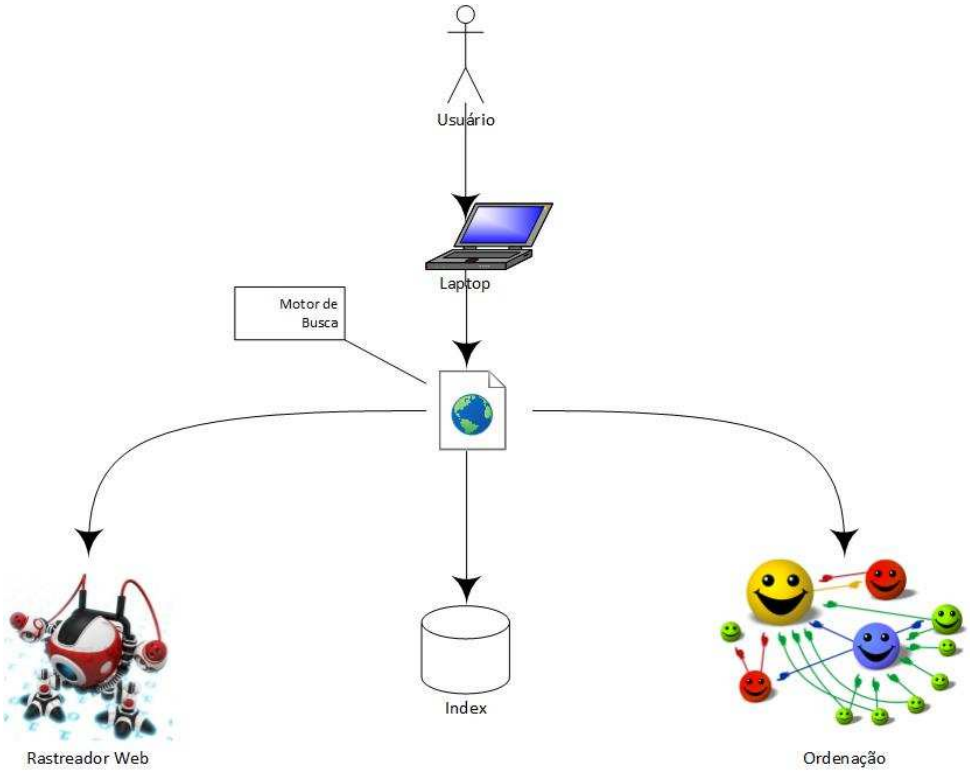


Figura 1 – Motor de Busca

Esses componentes são o web *crawler* (rastreador web), que é responsável por rastrear as informações pela web, o index, que é uma estrutura de dados que permite a realização da busca de forma mais rápida (YATES; NETO et al., 1999) e o *page ranking* (ordenação de página ou apenas categorização) responsável por sequenciar e julgar a importância da informação consultada pelo usuário.

Nos últimos vinte anos, diversos motores de busca web foram implementados e disponibilizados para a utilização do público na web, tais como Altavista, Yahoo!, Ask, AOL, Google, além de vários outros motores não tão grandes e que são comprados logo que começam a ganhar certo destaque, como Inktomi, Go.com ou o próprio Altavista.

Muito se especula sobre o que leva um motor de busca ser mais utilizado do que outro. Dados de 2011 que afirmam que o motor de busca mais utilizado no mundo era o Google, sendo utilizado por 82.80% dos usuários, enquanto o segundo lugar, ocupado pelo Yahoo!, contando com apenas 6,42% dos usuários e o terceiro lugar ocupado por um motor de busca chamado Baidu, contabilizando 4,89% dos usuários em busca de recuperação de informação (SHARMA; SHARMA, ). Porém, o mais interessante desses dados, é que na República Popular da China o motor de busca mais usado não era o Google e tampouco o Yahoo!, mas sim o Baidu possuindo 61,6% da parcela de mercado em busca web em

julho de 2009 (MARKET, 2009), justificando o porquê de ser a terceira colocada entre os motores de busca mais utilizados, mesmo sendo um motor de busca não tão conhecido mundialmente. Mas não é apenas na China em que o Google não é o primeiro colocado entre os motores de busca web: na Rússia um motor de busca chamado Yandex é quem ocupa a primeira posição, com 60% da parcela de mercado em abril de 2012 (TOP, 2013).

Não é apenas o fator popularidade que pode fazer um motor de busca ser o mais usado entre os usuários. Como já citado, o funcionamento harmônico entre os três componentes de um motor de busca é um fator válido para o aumento da utilização de determinados motores busca, assim como o destaque individual de um dos componentes do motor de busca, como, por exemplo, ir mais a fundo através dos links com o rastreador web, ter uma pré-busca melhor definida e dividida através do index ou mesmo trazer em melhor sequência os resultados para o usuário através da ordenação. Vale lembrar também que em alguns países, o uso de determinados motores de busca são restritos ou proibidos, sendo necessário recorrer a outros motores de busca.

## 2.3 Rastreador Web

Um rastreador web, também conhecido como aranha ou robô, é responsável por encontrar páginas para o seu motor de busca através dos links presentes em cada uma das páginas e esse rastreador pode ser implementado de forma que ele não percorra as maquinas remotas, mas sim um sistema local e mandando as requisições para servidores web remotos (YATES; NETO et al., 1999). Há diversas técnicas de implementação de comportamento do rastreador web, porém a mais simples é a partir de um endereço web (URL ou *uniform resource locator*) extrair outros endereços web recursivamente. Por causa desse tipo de implementação, é permitido que o usuário envie sites que ele julgue importante para determinada consulta, que não são imediatamente rastreados, pois são rastreados apenas depois de alguns dias ou semanas, que terão um conjunto de endereços web adicionados em seu código fonte, apontando para a página. Assim é possível começar por um site popular, fazendo a suposição de que por ele ser popular pode possuir endereços web com informações mais significativas e que podem ser solicitados com maior frequência do que uma página web não muito popular (YATES; NETO et al., 1999). Entretanto uma página que não é enviada pelo usuário, pode demorar semanas ou até meses para serem encontradas por um rastreador web, dado o imenso tamanho e crescimento da web a cada dia que se passa.

Há uma técnica em que utiliza dois modos diferentes de implementação recursiva para a travessia de endereços web pelo rastreador web, chamado *"breadth-first"* e o *"depth-first"*. A abordagem *breadth-first* é feita de modo em que o rastreador web analise todos os endereços web dentro da página que ele se encontra e assim por diante, permitindo que

seja realizada uma combinação estruturada dos endereços e das páginas web, uma vez que esses endereços podem ser atribuídos à mesma página. Essa abordagem é vasta, uma vez que são rastreados todos os endereços web presentes em uma página, porém é superficial pois não segue por uma trilha fixa de endereços web da página. Já o *depth-first* segue o primeiro endereço de uma página web e logo em seguida primeiro endereço da segunda página web e assim por diante, até que não seja possível seguir adiante e retornando recursivamente (YATES; NETO et al., 1999). Esse tipo de abordagem faz com que o rastreamento de endereços e páginas web seja profundo, porém é possível observar que possui abrangência restrita e que pode deixar que algum endereço de relevância alta para o usuário passe despercebido. Isso mostra que a ordenação em um motor de busca bem implementado pode fazer muita diferença se rastrear primeiro páginas importantes para o usuário. Ambas soluções *breadth-first* e *depth-first* são satisfatórias, porém há muita dificuldade em sincronizar vários rastreadores web para que não peguem a mesma página ou o mesmo endereço web repetidas vezes.

Existe uma técnica que permite a utilização de um ou mais rastreadores web. Ela consiste basicamente dividir a web usando *country codes*[1] ou *internet names*, atribuindo assim um ou mais rastreadores web para cada fração de parte da web e explorá-la até que se esgotem as possibilidades (YATES; NETO et al., 1999).

Toda página que é encontrada pelo rastreador web é mandado para um index, e, considerando o tamanho da web, a implementação de um index de um motor de busca pode ser complicado, devido à infinidade de possibilidades e à análise crítica da aplicação no determinado contexto. É importante lembrar que quando é feita a consulta em um motor de busca, a página não é indexada durante da consulta, e sim em uma data anterior, um dia ou até meses atrás. Assim, é possível que uma consulta não tenha sucesso, encontrando uma página que já deixou de existir. A porcentagem de endereços web inválidos, inexistentes, nos motores de busca podem variar de 2 a 9% de todas as páginas indexadas (YATES; NETO et al., 1999) (SALTON; MCGILL, 1986).

Apesar de todos essas habilidades que um rastreador web possui, um rastreador pode ter problemas com páginas HTML que utilizem frames, que divide a tela em duas ou mais partes, ou hyperlinks associados a imagens. Além disso, um rastreador web não consegue indexar páginas web geradas dinamicamente ou que sejam protegidas por senha (YATES; NETO et al., 1999).

[1] Country Code é um código normalizado pela ISO 3166-1 para identificar os países para utilização no processamento de dados e comunicação

## 2.4 Indexação

O index é uma estrutura de dados de extrema importância, construída em cima de textos, em um motor de busca. É por meio desta estrutura de dados que é possível realizar uma busca rápida, através de pesquisas pré-prontas, sobre um grande volume de dados (YATES; NETO et al., 1999). Para construir um index, é necessário que sejam implementadas formas para que sejam tratados os textos que são encontrados na web, pois nem toda palavra possui valor significativo para uma consulta. Alguns motores de busca desistem da ideia de realizar operações com esses textos, pois indexando todas as palavras, sem que haja operação textual, de um texto em determinada página web pode tornar a consulta do usuário uma tarefa mais simples e intuitiva (SALTON; MCGILL, 1986). Essa ideia de indexar todas as palavras de um texto sem realizar qualquer operação textual não é das melhores, pois indexar todas as palavras pode fazer com que o index se torne "barulhento", que o index acabe por armazenar palavras demais, assim como palavras que não terão influência nenhuma sobre alguma consulta. As operações textuais são realizadas durante o pré-processamento de documentos. Até certo tempo, os motores de busca se preocupavam mais com a velocidade em que a recuperação de informação era executada, do que em trazer a informação que é realmente necessária, importante e requisitada pelo usuário. O motivo disso é que um motor de busca precisa fazer uma quantidade determinada de consultas por uma quantidade determinada de tempo, evitando que esse número de consultas seja muito baixo ou que a quantidade de tempo seja muito alta. Uma boa técnica para a diminuição de tempo para uma consulta é a utilização de compressão de texto, pois um bom algoritmo de compressão consegue diminuir o texto para 30 a 35% de seu tamanho original, fazendo com que o texto ocupe menos espaço em sua armazenagem (YATES; NETO et al., 1999). Apesar de parecer uma boa alternativa, o tempo que um texto comprimido pode diminuir em uma consulta é compensado pelo tempo que é necessário para descomprimir e o próprio tempo necessário para comprimir esse texto, fazendo assim que um tempo anule o outro, ou seja, o tempo para comprimir, consultar, recuperar o texto e descomprimir será o mesmo que apenas consultar e recuperar o texto, além de tornar o design e a implementação de um motor de busca mais complexo (YATES; NETO et al., 1999).

Há cinco operações textuais para o procedimento de pré-processamento de documento, porém três delas possuem uma abordagem interessante dentro do contexto de motor de busca web, são elas o *stemming*, a seleção de termos indexados e o *Thesauri*.

### 2.4.1 *Stemming*

A operação textual de *stemming* tem como objetivo remover os prefixos e os sufixos de todas as palavras, assim, a recuperação de documentos possuirá variações sintáticas da consulta (YATES; NETO et al., 1999).

*Stem* significa a porção de palavra que resta após remover seus sufixos e prefixos (afixos), como por exemplo cafeteria, se removermos seu afixo resta apenas café. A técnica de *stem* permite que a performance de um motor de busca em sua recuperação de informação seja otimizada, reduzindo o que pode vir variar de uma palavra raiz para uma palavra de uma consulta, pode ser um termo mais intuitivo ou dentro da intenção do usuário, além também de diminuir a quantidade de termos indexados reduzindo o tamanho da estrutura de indexação.

Frakes (FRAKES, 1992) realizou experimentos favoráveis a utilização de *stemming* em motores de busca, porém seus resultados não foram satisfatórios, causando receio de vários motores de busca. Até o ano de 1999, não foi implementada essa técnica de pré-processamento para seus indexes.

Frakes descreve quatro técnicas para o *stemming* que são a remoção de afixo, consulta a tabela, variedade de sucessor e *n-grams* (YATES; NETO et al., 1999). A remoção de afixo é um *stemming* intuitivo, pois qualquer pessoa com conhecimento em determinado idioma pode implementar essa técnica de *stemming* com eficiência. Em várias línguas, no respectivo vocabulário, é mais comum encontrar palavras com sufixos ao invés de prefixos, assim essa técnica de *stemming* dá mais importância a remoção de sufixos. A técnica de consulta a tabela tem como objetivo armazenar a maior quantidade possível de stem em um banco de dados. É considerado uma técnica simples, porém trabalhosa demais, uma vez que para o bom funcionamento dela, seria necessário armazenar os *stem's* de um idioma inteiro, e como teoricamente não há dados relacionados ao *stem* prontos, isso demanda muito tempo e possivelmente um profissional especializado na língua que se deseja utilizar essa técnica. A operação textual de *stemming* de variedade de sucessor tem como objetivo determinar morfemas que podem vir a não ter expressividade em uma consulta, consiste em remover uma parte de uma palavra que, com ou sem ela, os resultados recuperados serão os mesmos. Essa técnica se difere da remoção de afixos pelo fato de haver a possibilidade de considerar afixos em uma busca, desde que possuam valor significativo para a consulta ou estejam dentro da intuição do usuário, enquanto a remoção de afixos tem o objetivo de remover todos os afixos, mas principalmente os sufixo. E por fim, *n-grams* no contexto computacional é uma sequência de $n$ itens para uma dada sequência de palavras e esses itens podem ser qualquer tipo de termo relacionado ao estudo da língua em questão, sendo considerada uma técnica de operação de texto mais relacionado ao agrupamento de termos do que tratamento de *stem*.

Até o ano de 1999 era possível encontrar três ou quatro tipos de algoritmos para a remoção de sufixo, porém o mais popular é o algoritmo de Porter, pela sua simplicidade e elegância (YATES; NETO et al., 1999), além de oferecer resultados satisfatórios de mesma qualidade de algoritmos não tão simples. Esse algoritmo trata a operação com texto através de uma lista de sufixo para realizar o corte de sufixo, caracterizando o *stemming*

de remoção de afixo. Um exemplo deste algoritmo é a conversão das palavras no plural por sua respectiva forma no singular. Na língua inglesa consiste em basicamente remover a letra "s" de uma palavra no plural, para língua portuguesa esse simples tratamento não pode ser executado, uma vez que nem sempre a palavra em sua forma no plural possui apenas a letra "s" como sufixo.

### 2.4.2 Seleção de Termos Indexados

Como citado anteriormente, vários motores de busca tratam o index selecionando todas as palavras de um texto, causando um index poluído. Um alternativa a isso é selecionar apenas palavras que têm a chance de possuir um algum significado na consulta ou na intenção do usuário, limitando assim as palavras que serão selecionadas e utilizadas como termos indexados.

Geralmente são os substantivos que apresentam maior valor em uma consulta do usuário, então uma boa estratégia é selecionar todos os substantivos dentro de um texto, excluindo todas as outras palavras pertencentes à classes gramaticais diferentes. É importante também fazer maior utilização de vários substantivos como se fosse apenas um termo indexado, adotando grupos de substantivos. Um grupo de substantivos são substantivos que estão sintaticamente distantes no texto, em termos de quantidade de palavras entre os substantivos, porém que não exceda uma certa quantidade pré-definida de palavras entre eles. Utilizando essa técnica é possível obter um index mais robusto, mais rico em questão de expressões e quantidades diferentes de possíveis intenções do usuário.

### 2.4.3 *Thesauri*

*Thesaurus* significa tesouro de palavras (*treasury words*) e é uma palavra de origem latina: seu plural é *thesauri* e o singular *thesaurus*. Essa operação de texto tem como objetivo formar um tesouro de palavras dentro de determinado contexto ou assunto, e para cada palavra dentro do tesouro indexado, formar um conjunto de palavras relacionadas, que geralmente são derivadas de um sinônimo (YATES; NETO et al., 1999). Apesar das aparências, um *thesaurus* envolve um trabalho de tratamento e normalização do vocabulário, além de incluir uma estrutura muito mais complexa do que uma simples lista de palavras com seus respectivos sinônimos (YATES; NETO et al., 1999).

Um *thesaurus* muito popular é o elaborado por Peter Roget, que é de natureza geral, isto é, não especifica um certo domínio de conhecimento. O objetivo do *thesaurus* de Roget é associar vários sinônimos que compõem a classe de *thesaurus* (YATES; NETO et al., 1999).

De acordo com Fosket (FOSKETT, 1997), os objetivos de formar um *thesaurus*

são basicamente, dar assistência para os usuários encontrando os termos para formar uma *query* adequada e formar hierarquia classificada das palavras que possibilitam a ampliação ou estreitamento da requisição da consulta atual de acordo com a necessidade do usuário, aumentando a chance de encontrar o que o usuário busca mesmo que sua consulta não seja formulada na melhor forma. A motivação para utilizar o *thesaurus* como uma operação de texto em um motor de busca é utilizar vocabulário controlado para indexar e buscar, pois um vocabulário controlado apresenta vantagens como a normalização de conceitos de index, além de diminuir a poluição dos termos indexados, através da identificação de termos com significado semântico e pôr fim a recuperação de documentos baseada em conceito ao invés de palavras (YATES; NETO et al., 1999).

Até 1999 não se via muita vantagem e clareza na utilização de *thesaurus* dentro do contexto web. Até então não era possível associar nenhuma área de conhecimento com documentos, pois a base de documentos da web é muito vasta, além de sua mudança ser muito rápida e dinâmica. Dentro do contexto web, a hierarquia de classificação de termos indexados se mostrava uma técnica que poderia ser útil e interessante dentro do contexto web, tomando como exemplo as páginas web que possuem seus conteúdos divididos por categorias, geralmente como sites de notícias, podendo associar cada categoria das notícias à um nível hierárquico alto e alguns substantivos ou conjunto de substantivos subordinados à essas categorias. No entanto a maior vantagem que se via na implementação de *thesaurus*, até 1999, era que a maioria dos motores de busca indexavam todas as palavras, sem nenhum critério de tratamento de texto ou de indexação, assim *thesaurus* poderia fornecer um alternativa à diminuição de termos indexados e indexar termos que apenas possuem algum valor para uma consulta do usuário.

Geralmente um termo indexado com a operação de texto *thesaurus* representa uma unidade básica para transmitir ideias, que podem ser palavras individuais ou um grupo de palavras, e geralmente são substantivos, pois como já citado, são os substantivos que possuem maior valor para a intenção do usuário em uma consulta e por ser a parte mais concreta de um discurso (YATES; NETO et al., 1999). Na maioria das vezes, essa unidade básica não consegue transmitir uma ideia através de uma única palavra, por isso geralmente são utilizados um grupo de palavras. Um exemplo disso é o conjunto de palavras “mísseis balísticos”. Se o motor de busca indexar uma das duas palavras, por exemplo, apenas a palavra “mísseis” não vai necessariamente encontrar dados relacionados a “mísseis balísticos”, além disso, é possível perceber que as palavras se encontram no plural, a ideia é que *thesaurus* representa um conjunto, uma categoria de dados, ou de coisas, tratando naturalmente um elemento dentro de um conjunto no plural, entretanto esse tipo de tratamento não é exclusivo, há também termos indexados no singular, porém o plural é preferencial (YATES; NETO et al., 1999). Além do próprio termo, os *thesauri* prezam pela relação entre termo indexado e sua definição, permitindo assim que um termo seja relacionado dentro de uma área de conhecimento com maior precisão.

Dentro da relação de termos relacionados na operação de texto de indexação *thesaurus*, um termo *thesaurus* é geralmente composto por sinônimos ou palavras próximas aos seus sinônimos, além disso, esses termos relacionados podem ser gerados através da quantidade de vezes que eles podem vir a aparecerem juntos em determinado texto. Esses relacionamentos também são estruturados de forma hierárquica e são formados com o objetivo de se relacionar com termos expansores (*boarder* representado por BT), que visam expandir a consulta através de termos que podem ser do interesse ou intenção do usuário ou por termos estreitadores (*narrower* representado por NT), que irá estreitar a consulta do usuário para uma área específica contida em determinada área de conhecimento dentro do index de *thesaurus*. Porém, os termos também podem ser estruturados de forma não hierárquica, quando isso ocorre, são simplesmente chamados de termos relacionados (*related terms* representado por RT). Os relacionamentos de BT e NT são mais fáceis de identificar do que relacionamentos de RT, uma vez que como relacionamentos BT e NT são hierárquicos, BT é associado com uma classe, que pode vir a ser uma área de conhecimento, e seu NT relacionado será associado com as instancias dessa classe (YATES; NETO et al., 1999), já o relacionamento RT tem de pertencer a um contexto específico e às necessidades dos usuários, além de ser difícil de identificar sem conhecimentos de especialistas no contexto determinado (YATES; NETO et al., 1999). No contexto web, o *thesaurus* parece ser muito interessante dentro da situação de que o usuário faz consultas errôneas, que não são exatamente o que estão procurando, porém está dentro de sua intenção. Em uma situação semelhante a está, o tratamento da consulta do usuário expandida com termos relacionados parece uma opção muito promissora, porém essa técnica não funciona de forma adequada, pois as relações que são estabelecidas em um *thesaurus* geralmente não se aplicam no contexto de uma consulta, então uma alternativa a isso seria determinar alguma coisa parecida com relacionamento de *thesaurus* durante o tempo de busca, mas para um motor de busca realizar esse tipo de procedimento durante o tempo de consulta acarretaria em demora excessiva e isso no contexto web não é nada viável.

## 2.5 Ordenação

A ordenação é o componente responsável por pontuar e ordenar os documentos, fazendo com que no ato da recuperação, os documentos sejam recuperados de forma que satisfaça o usuário. É centrada em satisfazer a necessidade do usuário, trazendo a informação que o usuário realmente necessita, através de medições estabelecendo critérios de relevância e pesos para realizar a ordenação. É também considerado o principal objetivo e núcleo na modelagem de um sistema de recuperação da informação (YATES; NETO et al., 1999).

Assim como a consulta, a ordenação não deve fazer acesso direto ao texto, somente

o index deve fazer isso. Até o período de 1995 a 1999 não havia muita informação sobre algum algoritmo específico de ordenação, adicionando a isso a dificuldade de comparar os motores de busca que existiam até o momento, pois esses motores possuíam muitas diferenças e qualidades distintas, além de melhorarem continuamente.

Yuwono e Lee (YUWONO; LEE, 1996) fizeram a proposta de três algoritmos de ordenação chamados *boolean spread*, *vector spread* e mais-citados. Os algoritmos *boolean spread* e *vector spread* são algoritmos normais de ordenação do modelo booleano e do modelo vetorial, respectivamente, porém a implementação de adição das páginas que são apontadas por páginas que estão contidas na resposta de uma consulta ou por páginas que fazem referência para a página na resposta. O algoritmo mais-citado tem como base os termos que estão contidos em uma página que possui algum link para uma das páginas na resposta.

Alguns algoritmos de ordenação mais modernos também utilizam hyperlinks como uma forma de critérios ordenar páginas. Essa é uma diferença notável entre os dados que podem ser recuperados via web e um sistema de recuperação de informação normal, uma vez que o sistema web utiliza-se de páginas que contém hyperlinks podendo assim utilizá-los para a ordenação de uma página, ao contrário do sistema de informação de recuperação normal que não possui páginas e consequentemente, também não possui hyperlinks. A quantidade de hyperlinks que apontam para uma página pode ser um indicativo de que a página possui relevância alta e é importante para determinado contexto, além de poder ser usada para medir a popularidade da página. Juntamente a isso, links que podem ser similares em diferentes páginas podem indicar certo grau de relacionamento entre elas. Três algoritmos muito conhecidos que apresentam bem essas definições e relações de hyperlinks citadas são o *WebQuery*, HITS (*Hypertext Induced Topic Search*), que dependem diretamente da consulta realizada pelo usuário e o último é o *PageRank* que durante muito tempo foi o algoritmo utilizado pelo famoso motor de busca Google (BRIN; PAGE, 1998). Esses algoritmos de ordenação serão explicados mais adiante com maiores detalhes.

# 3 Algoritmos de Ordenação

## 3.1 Introdução

Os algoritmos de ordenação que são selecionados para serem utilizados dentro de um motor de busca depende do contexto em que serão aplicados. Esses algoritmos podem ser divididos em três categorias distintas: os algoritmos baseados em termos, os algoritmos de ordenação estática e os algoritmos de ordenação dinâmica.

Os algoritmos baseados em termos são bastante utilizados em bibliotecas. Por meio da quantidade de termos no título ou corpo do texto, são realizados cálculos, na maioria distribuições de probabilidade ou tratamento de eventos randômicos, para chegar até a pontuação de determinado documento.

Os algoritmos de ordenação estática trabalham com quantidade de links que apontam de uma página para outra, são muito bem conceituados e muito utilizados em contexto web pelos grandes motores de busca.

E por fim, os algoritmos de ordenação dinâmica utilizam princípios muito parecidos com os algoritmos de ordenação estática, porém adicionam o feedback usuário como um parâmetro para o cálculo da pontuação da página. Também são muito utilizados para contexto web.

A seguir serão apresentados alguns dos algoritmos mais utilizados ou conceituados dentro destas três classificações.

## 3.2 Algoritmos Baseados em Termos

O Terrier é uma ferramenta que baseia suas buscas através do paradigma de *Divergence From Randomness* (DFR) que é um dos primeiros modelos de recuperação da informação (TERRIER, 2011). O modelo DFR segue seguinte princípio: quanto maior a divergência da frequência do termo dentro de um documento especifico a partir da sua frequência na coleção, maior a informação carregada pela palavra $T$ no documento $D$. O peso do termo é inversamente proporcional à probabilidade de encontrar o termo dentro do documento $D$ de acordo com o seguinte modelo distinto de aleatoriedade $M$(TERRIER, 2011):

$$peso(T \mid D)\alpha - \log \cdot Probabilidade_M(T \epsilon D \mid Colecao) \tag{3.1}$$

onde $M$ será o modelo de aleatoriedade que realizará o cálculo da probabilidade. Os modelos de DFR são baseados em modelos de aleatoriedade como o modelo de divergência de aproximação binomial ou a distribuição de bose-einstein e a partir desses modelos são

construídos os algoritmos utilizados para realizar a ordenação dos documentos recuperados.

Esses algoritmos são o *Best Match*(BM25), o *Divergence From Independence Model*(DFI), o *Divergence From Randomness free from parameters*(DRFree), o DFR *model based on Hypergeometric distribution and Laplace Normalization*(DLH) e por fim o *Poisson model and Euclidean Norm*(PL2). Como a arquitetura do Terrier utiliza-se de classes abstratas, algumas variáveis são levadas em consideração para a implementação dos algoritmos:

### 3.2.1 BM25

O BM25, também conhecido como Okapi BM25, ordena um conjunto de documentos baseados nos termos da consulta que aparecem em cada um dos documentos indexados sem levar em consideração a relação entre dois termos dentro do documento, por exemplo seu sentido diferenciado caso seja um substantivo composto. O BM25 é um algoritmo que sofreu mudanças desde seu nascimento em 1976 por criado Stephen E. Robertson(ROBERTSON; JONES, 1976). Em sua primeira forma, o BM possui a seguinte formula:

$$\frac{(k_3+1)\cdot q}{(k_3+q)} \cdot \frac{(k_1+1)\cdot f}{(k_1\cdot L+f)} \cdot \log \frac{(r+0,5)\cdot(N-n-R+r+0,5)}{(n-r+0,5)\cdot(R-r+0,5)} \tag{3.2}$$

onde $k_1$ e $k_3$ são constantes, $q$ é a frequência do termo na consulta, $f$ é a frequência do termo no documento, $n$ é a quantidade de documentos indexados pelo termo consultado, $N$ é o total de documentos na coleção, $r$ é a quantidade de documentos relevantes indexados pelo termo consultado, $R$ é a quantidade total de documentos relevantes e $L$ é a normalização do comprimento do documento, o comprimento de determinado documento dividido pelo comprimento médio de documentos na coleção (XAPIAN, 2013). No segundo passo Stephen E. Robertson cria o modelo BM11 utilizando o primeiro BM adicionando o termo que indica a quantidade e termos em uma consulta, chegando a formula:

$$k_2 \cdot n_q \cdot \frac{(1-L)}{(1+L)} \tag{3.3}$$

onde $k_2$ é a constante e $n_2$ é a quantidade de termos na consulta, o comprimento da consulta.

No terceiro passo é criado o BM15 que consiste em substituir $(k_1 + f)$ por $(k_1 \cdot L + f)$ na expressão definida por BM.

E por fim cria-se o BM25, combinando o BM11 e o BM15 com um fator de escala b, fazendo com que BM15 se transforme em BM11 enquanto seu resultado varia de 0 a 1 (XAPIAN, 2013). A formula para o BM25 é:

$$\frac{(k_3+1)\cdot q}{(k_3+q)} \cdot \frac{(k_1+1)\cdot f}{(K+f)} \cdot \log \frac{(r+0,5)\cdot(N-n-R+r+0,5)}{(n-r+0,5)\cdot(R-r+0,5)} \tag{3.4}$$

onde $K$ é $k_1(b \cdot L + (1 - b))$.

No Terrier encontra-se o algoritmo do BM25 da seguinte maneira:

$$\log_2 \left( \frac{N - n + 0,5}{n + 0,5} \right) \cdot \left[ (k_1 + 1) \cdot \frac{f}{K + f} \right] \cdot \left[ (k_1 + 1) \cdot \frac{f}{K + f} \right] \cdot \left[ (k_3 + 1) \cdot \frac{q}{k_3 + 1} \right] \quad (3.5)$$

seguindo as mesmas variáveis definidas pelo BM.

### 3.2.2 DFI

No modelo DFI é assumido que os termos mais de um dado documento são as palavras que a frequência diverge da frequência sugerida pelo modelo de independência. A noção de independência estabelece que se a razão da frequência de dois termos distintos são as mesmas sobre os documentos recuperados, eles são independentes dos documentos. O modelo de DFI estabelece uma relação entre os termos e os documentos, correspondente ao esquema TF x IDF onde TF representa a frequência do termos e IDF representa o inverso da frequência de documentos. Nesta relação, o DFI corresponde ao TF. O IDF é um fator que depende da coleção de documentos indexados e que identifica os termos que se concentram em poucos documentos da coleção (DINCER; KOCABASE; KARAOGLAN, 2009). Afirma-se que os melhores termos devem ter elevado nível de frequência de termo mas baixa frequência de coleção, assim levando pra notação do DFI, os melhores termos devem possuir maior pontuação de DFI (DINCER; KOCABASE; KARAOGLAN, 2009). O modelo de peso baseado no TF x IDF relacionado ao DFI é dado por:

$$w_{ij} = DFI_{ij} \cdot IDF_i \quad (3.6)$$

O $DFI_{ij}$ de determinado termo $t_i$ em um documento específico $d_j$ é encontrado através da diferença entre a frequência do determinado termo no documento específico ($x_{ij}$) de acordo com o modelo de independência determinado por(DINCER; KOCABASE; KARAOGLAN, 2009):

$$e_{ij} = x_{ij} \cdot \left( \frac{x_j}{x_{...}} \right) \quad (3.7)$$

onde $x_i$ é igual a frequência total de $t_i$ na coleção de documentos e $x_j$ é o comprimento do documento $d_j$ assim a frequência de termos deve ser distribuída proporcionalmente ao comprimento dos documentos.

Um método de atribuição de peso à um termo deve identificar os documentos que possuem mais afinidade com os termos fornecidos na consulta, para isso a formula do DFI é dada por:

$$DFI_{ij} = \frac{x_{ij} - e_{ij}}{\sqrt{e_{ij}}} \quad (3.8)$$

Na ferramenta Terrier, encontra-se o algoritmo DFI implementado como:

$$q \cdot \log_2 \left( 1 + \frac{f - e_{ij}}{\sqrt{e_{ij}}} \right) \quad (3.9)$$

onde $e_{ij}$ é igual ao produto da frequência do termo na coleção pela divisão do comprimento do documento pela quantidade de tokens na coleção.

### 3.2.3 DFRee

O DFRee é um algoritmo que segue o paradigma do DFR, e tem como objetivo calcular a média do número de bits extra para codificar um token dos termos fornecidos para a consulta de acordo com o DFR. Há duas formas para provar o conceito da probabilidade, o primeiro consistem em considerar apenas um documento da coleção e o segundo consiste em colher uma amostra do documento de acordo com as estatísticas da coleção inteira, assim o DFRee realiza o cálculo da média entre as medidas dos dois documentos, determinado como seu produto interno.

No Terrier a fórmula do DFRee é implementada como:

$$q \cdot norm \cdot \{f \cdot [\log_2 (prior \cdot InvPrior)]\} + (f+1) \cdot \log_2 (post \cdot InvPrior) + 0,5 \cdot \log_2 (post \cdot InvPrior) \quad (3.10)$$

onde *prior* é dada pela razão entre a quantidade de termos dentro do documento, $f$, pelo comprimento do documento, docLength, post é definido como:

$$post = \frac{(f+1)}{docLength + 1} \quad (3.11)$$

$InvPrior$ é a razão entre a quantidade de *tokens* dentro de um documento pela frequência do termo dentro da coleção e *norm* é igual ao produto da frequência de termos dentro de um documento pelo log na base 2 da razão entre *post* pelo *prior*.

### 3.2.4 DLH

O DLH é um algoritmo que faz utilização da distribuição hipergeométrica e da normalização de caminhada randômica de Laplace.

A distribuição hipergeométrica é uma distribuição discreta utilizada para encontrar a probabilidade de retirar $X$ elementos do tipo A em uma determinada sequência de $n$ extrações de uma amostra do tamanho $N$, com $M$ elementos do tipo a e $N$-$M$ elementos do tipo $B$, sem que haja reposição das amostras. Definindo o $X$ como uma variável aleatória, essa variável é responsável por encontrar a quantidade de elementos do tipo $A$. Assim, define-se a distribuição de probabilidade de $X$ como (ACTION, 2011):

$$\mathbb{P}(X = x) = \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}} \quad (3.12)$$

A normalização de caminhada randômica de Laplace é muito utilizada quando se trata de caminhadas por grafos. A normalização de Laplace para um grafo é definida como(SPIELMAN, 2007):

$$N_G = D_G^{\frac{-1}{2}} * L_G * D_G^{\frac{-1}{2}} \quad (3.13)$$

Através desses conceitos, o algoritmo implementado no Terrier é definido como:

$$q \cdot \left[ \frac{f \cdot \log_2\left(\left[f \cdot \frac{ADL}{docLength}\right] \cdot \left[\frac{n}{ct}\right]\right) + [(docLength - f) \cdot \log_2(1-tf)] + [0{,}5 \cdot \log_2(2 \cdot \pi \cdot f \cdot (1-tf))]}{f-k} \right] \quad (3.14)$$

onde $ADL$ é a média do comprimento dos documentos, $docLength$ é o comprimento dos documentos, $n$ é a quantidade de documentos indexados, $ct$ é a frequência do termo na coleção de documentos, $tf$ é a razão entre a quantidade de termos dentro de um documento, $f$, e o comprimento do documento, e $k$ é uma constante de normalização.

### 3.2.5 PL2

O algoritmo PL2 faz utilização de dois modelos matemáticos combinando-os para sua implementação, a distribuição de Poisson e a normalização euclidiana. A distribuição de Poisson calcula a probabilidade de que um evento ocorra em um determinado intervalo de tempo. Na distribuição de Poisson a quantidade de ocorrências de um evento em um intervalo de tempo, independe da quantidade de ocorrências do evento em qualquer outro intervalo distinto. A probabilidade de haver ocorrências simultâneas é muito próxima ao zero, a quantidade média de ocorrências por unidade de tempo é constante ao longo do tempo, as ocorrências são distribuídas de maneira uniforme sobre o intervalo de tempo em questão. A quantidade de ocorrências durante qualquer intervalo de tempo depende somente da duração deste intervalo, quanto maior o intervalo, maior o número de ocorrências (DELAWARE, 2013).

A formula da distribuição de Poisson é definida por:

$$P(x) = \frac{\left(\lambda^2 \cdot e^{-\lambda}\right)}{x!} \quad (3.15)$$

onde $x$ é a quantidade de ocorrências do evento em um determinado intervalo de tempo, $\lambda$ é a taxa de ocorrência do evento $x$ e $e \approx 2{,}718$ é uma constante natural.

A normalização euclidiana (ou comprimento euclidiano) em um vetor$u = (u_1, u_2, ..., un)$ na dimensão $R^n$, é definida por(ANTON; RORRES, 2002):

$$\|u\| = (u * u)^{\frac{1}{2}} = \sqrt{u_1^2 + u_2^2 + ... + u_n^2} \quad (3.16)$$

Através destes dois conceitos matemáticos o Terrier realiza a implementação do PL2 como:

$$NORM \cdot q \cdot \left\{TF \cdot \log_2\left(\frac{1}{fn}\right) + [fn \cdot \log_2(e)] + [0,5 \cdot \log_2(2 \cdot \pi \cdot TF)] + [TF \cdot \log_2(TF) - \log_2(e)]\right\} \quad (3.17)$$

sendo que:

$$TF = f \cdot log_2\left(1 + \frac{[c \cdot ADL]}{docLength}\right) \quad (3.18)$$

onde, $f$ é a frequência dos termos dentro do documento, é uma constante de normalização, $ADL$ é a média do comprimento dos documentos e $docLength$ é o comprimento do documento. $NORM$ é definida pela razão de 1 pelo $TF$ somado a 1 e por fim $fn$ é a razão entre a frequência dos termos na coleção de documentos e o número de documentos.

## 3.3 Algoritmos de Ordenação Estática

Com o avanço tecnológico e com o fácil acesso a informação, o usuário vem se tornando cada vez mais exigente, e apesar do Google ocupar a maior parcela de mercado dentro do contexto de motor de busca, o usuário quer, em qualquer motor de busca que seja, os documentos que tenham relação com a sua consulta realizada, e além disso, querem também que esses documentos que podem ser relevantes sejam recuperados para o topo da página do motor de busca, evitando o esforço desnecessário de ter que percorrer várias páginas até que encontre o que procurava.

O primeiro algoritmo elaborado para a ordenação foi o modelo booleano, porém ele não tinha como objetivo o usuário final, em seguida vieram os modelos vetoriais e os probabilísticos. O modelo vetorial utilizava a consulta realizada pelo usuário e documento como dois vetores distintos e realizava o cálculo através da função cosseno, e assim, computando a similaridade entre esses dois vetores. Quanto maior fosse o valor da função cosseno, maior seria a ordenação do documento para determinada consulta. Já o modelo probabilístico utilizava os termos que eram usados em uma consulta do usuário para definir um peso de acordo com ocorrências em determinados documentos, estabelecendo assim que os documentos com maior peso em ocorrência de termos de uma consulta possuem maior relevância do que documentos com menor peso.

Geralmente os algoritmos de ordenação estáticos utilizam esses modelos para explorar a estrutura do documento e assim elaborar uma ordenação dos documentos (SHARMA; ADHAO; MISHRA, 2013).

### 3.3.1 *WebQuery*

O algoritmo de /WebQuery] (YATES; NETO et al., 1999), um algoritmo estático, tem como objetivo pegar várias páginas web dentro de um conjunto e faz sua ordenação com base em quão conectada uma página está com outra. Em seguida são pegas as páginas que estão relacionadas ao conjunto original para poder julgar melhor a ordenação das páginas.

### 3.3.2 *Page Rank*

O *PageRank* é um algoritmo de ordenação muito usado entre os motores de busca e fez parte dos algoritmos de ordenação do google (BRIN; PAGE, 1998) (SHARMA; ADHAO; MISHRA, 2013). O algoritmo de *PageRank* é um algoritmo estático e dependente da estrutura web, consiste em basicamente levar em consideração que uma página possui vários links apontando para ela, levando também em consideração que o usuário nunca iria voltar para uma página que já foi explorada, e então a ordenação da página é calculada de acordo com a ordenação das páginas anteriores e quantidade de backlinks entrando na página (SHARMA; ADHAO; MISHRA, 2013), isto é, se uma página atual é referenciada por vários links de páginas com ordenação alta e se essa mesma página possui grande quantidade de backlinks, então essa página possuirá uma ordenação o muito alta, podendo inferir que as páginas com ordenação alta são importante e relevante, assim as páginas com backlinks para a página atual também serão consideradas importantes e relevantes (YATES; NETO et al., 1999) (SHARMA; SHARMA, ). É possível observar um exemplo de *PageRank* na Figura 2:

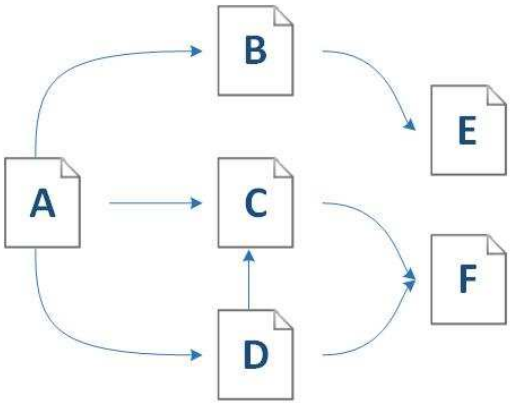


Figura 2 – Page Rank

Na Figura 2 podemos ver as relações entre as páginas e os links que apontam para elas, por exemplo, observamos que a página A fornece vários links, para as páginas B, C e D, as páginas B e C fornecem links apenas para as páginas E e F respectivamente e a página D fornece links para as C e F. Se a página A e B já possuírem alto valor de ordenação, então a página E receberá um valor de ordenação maior ainda.

Baeza (YATES; NETO et al., 1999) leva em consideração que um usuário tem a chance de encontrar um página aleatoriamente com a probabilidade “q” ou segue adiante em um link aleatório dessa mesma página indo parar na próxima página com a probabilidade de “1-q”, lembrando que faz-se a suposição de que o usuário nunca irá voltar para uma página já visitada (YATES; NETO et al., 1999). A probabilidade de se parar em cada página pode ser calculado através de cadeia de Markov (YATES; NETO et al., 1999) e então esse valor é utilizado em uma formula onde “C(a)” é o número de links

que apontam para a página "a" em questão e que todas as páginas entre p1 e pn estão apontando para a página "a". Assim é possível definir o *PageRank*, PR(a) como (YATES; NETO et al., 1999):

$$PR(a) = q + (1 - q) \sum_{i=1}^{n} \frac{PR(p_i)}{C(p_i)} \tag{3.19}$$

onde $q$ deve ser definido pelo sistema, geralmente é atribuído o valor de 0,15. A fórmula de (YATES; NETO et al., 1999) permite que a ordenação de outras páginas seja normalizado pelo número de links na página.

Já (SHARMA; ADHAO; MISHRA, 2013) assim como (SHARMA; SHARMA, ) apresentam uma forma simplifica de cálculo de *PageRank*

$$PR(a) = \sum_{v \epsilon B_a} \frac{PR(v)}{L(v)} \tag{3.20}$$

onde o valor do *PageRank* de uma página "a", $PR(a)$, depende dos valores do PageRank de cada página "v", $PR(v)$, fora do conjunto de páginas que apontam para a página "a", $B_a$, dividido pelo número de links que vem da página "v".

### 3.3.3 *Page Rank* por Peso

(SHARMA; ADHAO; MISHRA, 2013) e (SHARMA; SHARMA, ) fazem menção a um algoritmo de *PageRank* alternativo com o objetivo de melhorar o algoritmo de *PageRank* convencional no que se diz a respeito de ordenar melhor os documentos criado por Wenpu Xing e Ali Ghorbani (XING; GHORBANI, 2004). O algoritmo de *PageRank* por peso utiliza o peso como parâmetro de cálculo da ordenação da página web atribuindo esse novo parâmetro entre os links que apontam para a página, os *incoming links*, *in-links* ou *backlinks* e os links que saem da página em questão, os *outgoing links* ou *out-links*, definindo a popularidade de uma página (SHARMA; SHARMA, ) sem com que haja divisão de valor de ordenação para que sejam atribuídos pesos divididos aos *out-links* de uma página.

O cálculo de um *PageRank* por peso de uma página "a", $WPR(a)$, é definido como (SHARMA; ADHAO; MISHRA, 2013) (XING; GHORBANI, 2004):

$$WPR(a) = q + (1 - q) \sum_{v \epsilon B_a} WPR(v) W_{(v,a)}^{[in]} W_{(v,a)}^{[out]} \tag{3.21}$$

onde $W_{(v,a)}^{[in]}$ é o peso dos *in-links* calculados baseados na quantidade de *in-links* da página "a" e o número de *in-links* para todas as páginas que são referenciadas para a página "v', assim como $W_{(v,a)}^{[out]}$ é o peso dos *out-links* calculados com base na quantidade de *out-links* da página "n" e o número de *out-links* para todas as páginas referenciadas para a página

"m" (SHARMA; ADHAO; MISHRA, 2013), é possível inferir as fórmulas dos pesos como (XING; GHORBANI, 2004):

$$W_{(v,a)}^{[in]} = \frac{I_a}{\sum_{p \epsilon R(v)}} I_p \tag{3.22}$$

$$W_{(v,a)}^{[out]} = \frac{O_a}{\sum_{p \epsilon R(v)}} O_p \tag{3.23}$$

onde $I_a$ e $I_p$ representam o número de *in-link* da página a e da página p, respectivamente e $R(v)$ representa a lista de referência de página da página v assim como $O_a$ e $O_p$ são os números de *out-links* de uma página a e uma página p respectivamente.

### 3.3.4 HITS *(Hypertext Induced Topic Search)*

Segundo Baeza (YATES; NETO et al., 1999) o HITS, um algoritmo de ordenação estática, é uma alternativa melhorada do *WebQuery* proposta por Kleinberg (KLEINBERG, 1999) e trabalha com a ideia de a ordenação depende, além do conteúdo textual, dos links apontam para as páginas na resposta em conjunto com os links que são apontados por essas mesmas páginas na resposta, isto é, ao contrário do *PageRank* que trabalha especificamente e somente com os links que estão apontado para determinada página, somente no sentido dos in-links, esse algoritmo trabalha com ambos os sentidos dos links, leva em consideração os *in-links* e os *out-links* de uma mesma página dentre as páginas que estão como resposta à consulta.

As páginas que possuem links saindo dela se chamam *Hub* e as páginas que recebem links, possuem links apontando para ela, se chamam Responsabilidades (*Authorities*) podendo ser observado essa relação na Figura 3. Uma página não precisa ser apenas um *hub* ou apenas uma responsabilidade, de fato, em teoria, todas ou quase todas as páginas deveriam ser *hubs* e responsabilidades simultaneamente, é possível visualizar esse tipo de relação na Figura 4, mas há muitas páginas que são apenas responsabilidades, o exemplo mais claro disso é uma página que possui apenas uma figura, essa página só possui *in-links* apontando para ela e não possui nenhum *out-link* apontando dela para uma outra página qualquer. Esse tipo de página é bem comum no recurso de busca de imagens do Google.

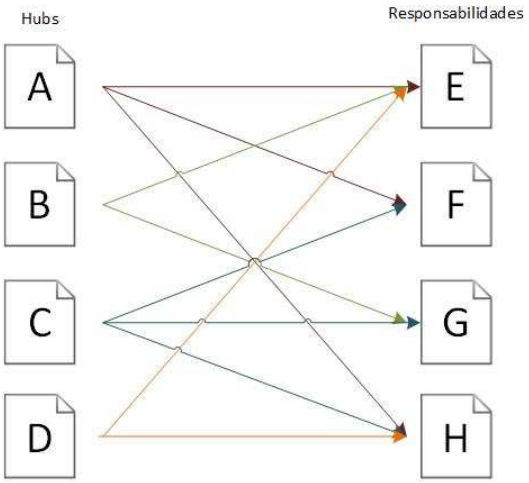


Figura 3 – Hubs e Responsabilidades (Authorities)

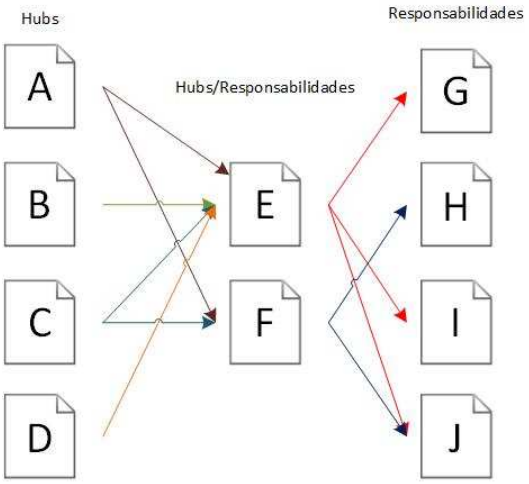


Figura 4 – Hubs e Responsabilidades e Uma Página "Híbrida"

Como HITS é um algoritmo baseado em links, com esses conceitos, considera-se que um bom *hub* é aquele que possui muitos *out-links* para muitas responsabilidades importantes, e que uma boa responsabilidade possui muitos *in-links* de *hubs* populares e importantes (SHARMA; ADHAO; MISHRA, 2013) (SHARMA; SHARMA, ), na prática, o peso da responsabilidade é proporcional a soma dos pesos dos *hubs*, assim como o peso do *hub* é proporcional ao peso das responsabilidades que estão relacionadas (SHARMA; ADHAO; MISHRA, 2013) como representado pela fórmulas a seguir.

$$r^{(p)} = \sum_{q:(q,p)\epsilon E} h^{(q)} \tag{3.24}$$

$$q^{(p)} = \sum_{q:(p,q)\epsilon E} r^{(q)} \tag{3.25}$$

onde o peso de uma responsabilidade, $r^{(p)}$, de determinada página p é dado pelo somatório dos pesos dos hubs, $h^{(q)}$, para todas as páginas q apontando para as páginas p, assim como o peso de um hub é calculado pelo somatório de todos os pesos das responsabilidades para todas as páginas q que são apontadas pela página p (KLEINBERG, 1999). A representação gráfica desse cálculo de peso pode ser representada pela Figura 5.

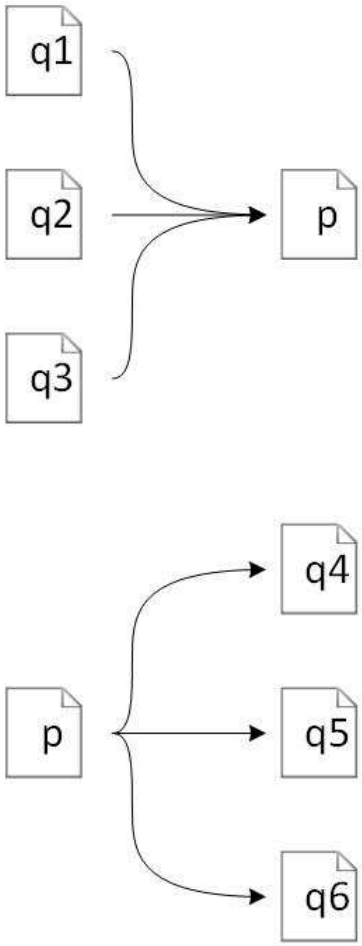


Figura 5 – Cálculo de Peso dos Hubs e Responsabilidades

Esse tipo de relacionamento entre o *hub* e autoridade é chamado de relacionamento mutualmente reforçado (KLEINBERG, 1999).

O HITS analisa a estrutura da web, após a coleta de páginas web, através da construção de um grafo onde os vértices são as páginas web e as arestas são os links que apontam de um *hub* para uma responsabilidade. Esse grafo divide em duas partes todas as páginas recuperadas em determinada consulta, onde de um lado da divisão se encontram os *hubs* e do outros lado encontra-se as responsabilidades (SHARMA; ADHAO; MISHRA, 2013). Basicamente esse processo do HITS é construído em duas etapas, na primeira etapa, o passo de amostragem, um conjunto de páginas que contém conteúdo relevante ou que tem potencial de estar dentro da intenção do usuário de acordo com a consulta do mesmo são coletadas, e na segunda etapa, o passo iterativo, os *hubs* e as responsabilidades são

encontrados e definidos usando o que foi coletado no passo de amostragem formando o grafo (SHARMA; ADHAO; MISHRA, 2013). Assim o HITS busca recuperar apenas uma quantidade limitada de responsabilidades mais relevantes, fornecendo ao usuário o que pode vir a ser o objetivo de sua consulta.

O HITS mesmo com essa abordagem buscando o equilíbrio através da ordenação por meio de *in* e *out-links* possui desvantagens, uma delas é que a página que é considerada popular é atribuída com uma pontuação de ordenação alta, isto é, uma página popular acaba aparecendo como uma das primeiras páginas de uma consulta realizada, mesmo que ela não traga nenhuma informação relevante para a consulta. Outro problema é a fuga do tema do objeto que foi consultado, quando um *hub* possui vários temas, a mesma pontuação de ordenação é dada para todos os *out-links* do *hub*, fazendo também com que as primeiras páginas recuperadas não serão necessariamente relevantes para a consulta do usuário (SHARMA; SHARMA, ).

### 3.3.5 SALSA (*The Stochastic Approach for Link-Structure Analysis*)

O algoritmo de abordagem estocástica[1] para a análise de estruturas de link (SALSA) (LEMPEL; MORAN, 2000) é um algoritmo estático e um algoritmo híbrido baseado nos algoritmos *PageRank* e HITS. Diferente do algoritmo de *PageRank* que trabalha com a técnica de ordenação baseada na consulta independente dos links das páginas, o SALSA é uma técnica que baseada na dependência de links, assim como o algoritmo de HITS (SHARMA; ADHAO; MISHRA, 2013). A abordagem é baseada nas cadeias de Markov e se utiliza de cálculos estatísticos, especificamente as propriedades estocásticas, simulando que o usuário acesse páginas randômicas. Para realizar a caminha randômica, ele utiliza-se do grafo de *hubs* e responsabilidades construído, e caso o algoritmo se localizar em uma página *hub*, então ele randomicamente irá escolher um *out-link* qualquer para caminhas, assim que chegar na página responsabilidade ele escolherá randomicamente um *in-link* qualquer para uma página *hub* e assim por diante (SHARMA; ADHAO; MISHRA, 2013).

Como citado, o SALSA também elabora um grafo de *hubs* e responsabilidades, utilizando os vértices desse grafo para calcular a pontuação de ordenação de cada *hub* e responsabilidade como seu auto vetor. Entretanto, no algoritmo SALSA, os pesos dos *hubs* e das responsabilidades não são idênticos, e sim distribuídos igualmente entre eles, isto é, para um *hub*, o peso das responsabilidades que ele aponta é dividido igualmente entre eles, assim ocorre também com as responsabilidades, o peso dos *hubs* que apontam para ela são divididos igualmente entre eles.

Assim o salsa não utiliza a utiliza a estrutura mutualmente reforçada que é utili-

[1] Estocástica é o estudo da aplicação do cálculo de probabilidades a dados estatísticos, estabelecendo a existência de variáveis permanentes e regulares, cuja a ação se complica com a de fatores que ocorrem por acaso

zada no algoritmo de HITS, ao invés disso, no SALSA a pontuação que é dada para um hub ou uma responsabilidade é determinado pelos links locais que estão ligando os dois lados do grafo, e não de acordo com a estrutura completa do grafo (SIGNORINI, 2005).

## 3.4 Algoritmos de Ordenação Dinâmica

Como visto logo acima, os algoritmos de ordenação estáticos não consideram a interação com o usuário e além disso possuem problemas de ambiguidade de consulta e consequentemente a intensão do usuário se defere muito da consulta realizada (SHARMA; ADHAO; MISHRA, 2013). Nos algoritmos de ordenação dinâmica possuem uma relação de perda e ganho entre os resultados fornecidos especificamente para a intenção do usuário e de intenções recuperadas (RAMAN; JOACHIMS; SHIVASWAMY, 2011), ou seja, quanto maiores forem os resultados fornecidos para as intenções do usuário, menor será a variedade de possibilidades de intenções que o usuário poderia vir a ter para a consulta em questão.

Os algoritmos de ordenação o dinâmica buscam uma forma de interagir com usuário para fazer uma suposição em cima das intenções dos usuários dentro das intenções possíveis em determinados contextos, assim os resultados do primeiro processo de recuperação são reordenados e trazidos apenas o resultado refinado desse processo ao usuário (SHARMA; ADHAO; MISHRA, 2013).

O foco principal desses algoritmos é a relevância que os documentos possuem e a diversidade de recuperação de resultados. Os algoritmos de ordenação dinâmica de aprendizado estruturado em dois níveis (*Structured learning of two-lvl dynamic rankings*) (RAMAN; JOACHIMS; SHIVASWAMY, 2011), o algoritmo de ordenação genérica diversificada (*GenDer, A Generic Diversified Ranking Algorithm*) (HE et al., 2012), além de alguns outros algoritmos são soluções interessantes para o objetivo da ordenação dinâmica.

### 3.4.1 Ordenação de Dois Níveis Utilizando Informação Mútua (*Two-Level Ranking Using Mutual Information*)

Através do index, vimos que esse componente é importante para o motor de busca e a ordenação de uma palavra não depende exclusivamente do seu significado, mas sim das suas relações com outras palavras. As relações entre as palavras é chamada de informação mútua e ajuda a entender a relevância de um resultado para a consulta de um usuário (SHARMA; ADHAO; MISHRA, 2013).

Essa informação mútua é calculada utilizando a relação entra a probabilidade que duas palavras podem se juntar e continuar fazendo sentido e tendo importância para a consulta e as probabilidades individuais da palavra ser importante para a consulta. Assim

é possível inferir que de duas palavras estão relacionadas entre elas, a probabilidade dessas palavras terem alguém valor para a consulta realizada pelo usuário é muito maior do que uma palavra apenas. Um exemplo de palavras relacionadas são livros, páginas, autor e editora(SHARMA; ADHAO; MISHRA, 2013).

Como o próprio nome do algoritmo diz, ele utiliza de dois níveis para realizar a ordenação, o nível de recuperação e o nível de reordenação.

#### 3.4.1.1 Primeiro Nível: Recuperação de Documentos

No primeiro nível é necessário recuperar os documentos que são relevantes para a consulta do usuário. Para realizar essa atividade é necessário utilizar um método de ordenação dinâmica chamado recuperação de documentos baseado em vetor, utilizando modelo de vetor no espaço. Através desse método a consulta do usuário é utilizada para recuperar todos os documentos que possam vir ser similar a ela.

O documento de informação mútua do index deverá possuir a identificação do documento, número de termos do documento, termos pareados e frequência de palavras e de pares de palavras

#### 3.4.1.2 Segundo Nível: Reordenando Documentos Recuperados

No segundo passo do algoritmo de ordenação por informação mútua é utilizado para fornecer um resultado otimizado para a consulta realizada pelo usuário. Para realizar essa tarefa um motor de reordenação é utilizado para atribuir pontos de ordenação aos documentos que foram recuperados no primeiro nível. As entradas para o funcionamento deste nível é o documento de informação mútua e a quantidade de termos do documento, ambos construídos e calculados no primeiro nível. Assim a reordenação poderá ser realizada de diferentes formas para calcular o valor de similaridade de todos os documentos que possuem relação com a consulta realizada pelo usuário (SHARMA; ADHAO; MISHRA, 2013).

### 3.4.2 Árvore Dinâmica de Ordenação (*Dynamic Ranking Tree*)

O algoritmo de arvore dinâmica de ordenação tem como objetivo controlar o que o usuário realmente quer como resultado de sua consulta, para isso ela forma uma árvore de dados e em cada nó desta arvore é um documento recuperado com caminhos possíveis para documentos que possuam a chance de fazer parte da intenção do usuário em sua consulta (SHARMA; ADHAO; MISHRA, 2013).

Podemos exemplificar o funcionamento do algoritmo de árvore dinâmica através de uma consulta fictícia da palavra *"brown"*, quando essa palavra é passada como uma consulta para um motor de busca, as diferentes possibilidades de recuperação poderão ser

*Brown University*, livro de *Dan Brown*, *Browns* fashion magazine, *Brown's hotels*, *Jerry Brown*, ou simplesmente a palavra *brown* como significado da cor marrom. Esses documentos recuperados através da consulta possuem apenas ligação com a palavra *brown* e farão parte do primeiro nível da arvore dinâmica, abaixo apenas da consulta, e os documentos possam ter relacionamento com algum destes documentos recuperados poderão vir a fazer parte do segundo nível da arvore dinâmica, e assim por diante (SHARMA; ADHAO; MISHRA, 2013).

Quando a árvore dinâmica é construída, é permitido que o usuário escolha qualquer um de seus nós e faça o caminho de "descida" começando pelo primeiro nó, a consulta realizada por ele, fornecendo os documentos que serão relevantes depois da gravação do primeiro nó que ele escolheu abaixo de sua consulta (SHARMA; ADHAO; MISHRA, 2013).

Na Figura 6 é possível observar uma possível estruturação de uma árvore de ordenação dinâmica.

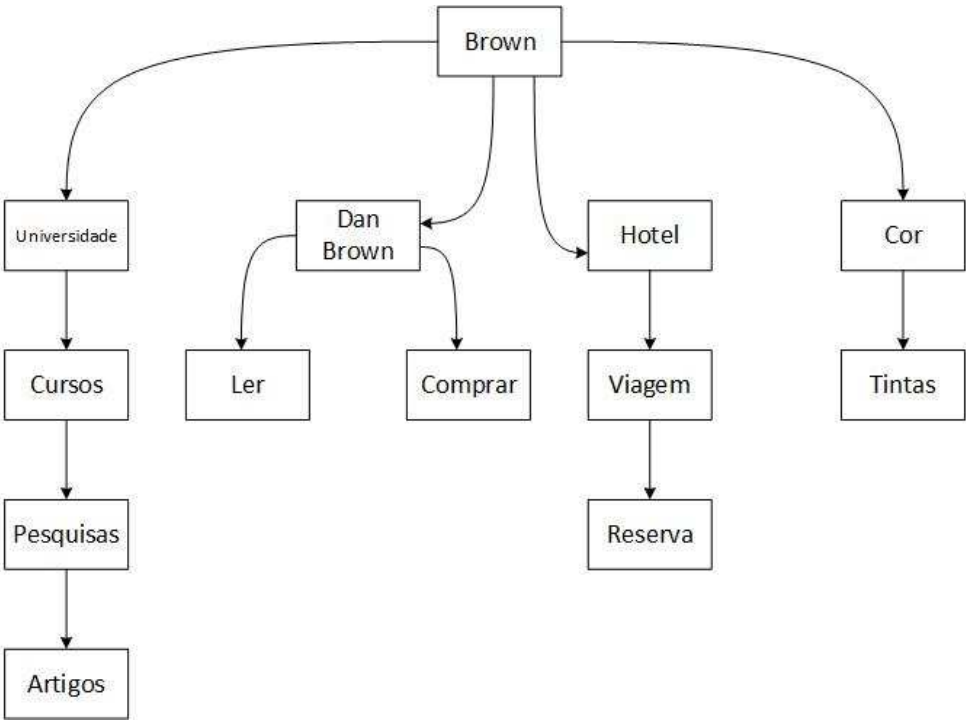


Figura 6 – Árvore Dinâmica de Ordenação

Para fazer a construção da árvore dinâmica é necessário dois algoritmos, o *DynamicMyopic* e o *DynamicLookahead*. O *DynamicMyopic* faz uma abordagem gananciosa, adicionando, depois de cada interação, um documento com o maior valor da variável ganho em utilidade, ordenando assim seus documentos de acordo com o valor de ganho em utilidade. Por outro lado, o algoritmo *DynamicLookahead* utiliza-se do método de *Lookahead* para estimativa, enquanto o algoritmo *Myopic* faz somente uso do valor de ganho em utili-

dade do documento, o algoritmo de *Lookahead* utiliza de valores aproximados de utilidade de duas ou mais sub-árvores do documento em questão. Com isso o documento que será recuperado para determinada consulta do usuário não trará apenas o valor de utilidade do documento, mas sim o valor da sub-árvore toda (SHARMA; ADHAO; MISHRA, 2013).

### 3.4.3 GENDER (*Generic Diversified Ranking Algorithm*)

O algoritmo de ordenação genérica diversificada tem como objetivo fornecer os resultados atendendo a diferentes necessidades possíveis do usuário (SHARMA; ADHAO; MISHRA, 2013) diversificando uma quantidade determinada de documentos.

Essa ordenação é feita usando a função de relevância arbitrária em conjunto com a função de relevância e similaridade (HE et al., 2012) considerando a diversidade o fator principal para poder chegar a uma definição da incerteza e da ambiguidade no sistema de recuperação de informação (SHARMA; ADHAO; MISHRA, 2013), além de ser uma forma interessante de aumentar a cobertura no que diz respeito à intenção do usuário.

A maioria dos algoritmos de diversificação tem como objetivo a extensão da cobertura do tópico, no resultado ou na diversificação dos conjuntos de possíveis resultados. A partir disso o desempenho do algoritmo é medido através das matrizes de relevância, utilidade ou similaridade que possuem relação com o assunto buscado através da consulta e dos documentos recuperados (SHARMA; ADHAO; MISHRA, 2013).

O algoritmo GenDer leva em consideração a relevância para uma consulta em conjunto com a diversificação de resultados como fator principal, entretanto, sempre há uma troca inversamente proporcional entre a relevância e a diversidade, isto é, quando há um ganho em relevância, se perde na diversidade, se o algoritmo focar no fator relevância, ele pode perder alguns documentos que não possuam muito destaque ou não sejam muito populares, mas que tem potencial para satisfazer a consulta ou a intenção do usuário. Por outro lado, se o algoritmo focar na diversidade, os números de documentos relevantes, que realmente fazem parte da intenção do usuário (SHARMA; ADHAO; MISHRA, 2013), podem ficar de fora, pois, popularmente falando, a quantidade é mais importante que a qualidade, o algoritmo tem como objetivo trazer a maior quantidade de informação possível.

Para tratar essa troca inversamente proporcional entre a relevância e a diversidade, o GenDer utiliza de um parâmetro de regularização para manter esse equilíbrio. Além disso, não o GenDer não consegue encontrar o equilíbrio perfeito, fornecendo uma solução otimizada aproximada (HE et al., 2012):

$$arg\,max_{|\tau=k|}\; g\left(\tau\right) = w \sum_{i \epsilon \tau} q_i r_i - \sum_{i,j \epsilon \tau} r_i S_{i,j} r_j \tag{3.26}$$

onde $X$ pode ser considerado um conjunto contendo $n$ documentos candidatos $S$ é uma

matriz de similaridade de tamanho n x n, uma matriz simétrica, $r$ é a função de ordenação, que retorna o valor de relevância para cada documento em X, $\tau$ é o subconjunto de X, possui $k$ elementos, sendo o objetivo principal do algoritmo encontrar o subconjunto $\tau$, $w$ o parâmetro de regularização, para equilibrar a troca inversamente proporcional entre a relevância para a consulta e a diversificação entre o conjunto de documentos recuperados, $g(\tau)$ é a função de qualidade, para calcular a qualidade do documento dentro dos parâmetros de relevância e diversidade e $q$ é o vetor referência n x 1, calculado como:

$$q = Sri_n \tag{3.27}$$

e dá importância para a ordenação do enésimo termo dentro do conjunto de documentos X.

# 4 Desenvolvimento

## 4.1 Introdução

Para o desenvolvimento deste trabalho se viu necessário um motor de busca *open source* para a experimentação. A ferramenta seria utilizada na forma de um *framework*, permitindo que fosse possível a inserção ou a remoção de algoritmos para a experimentação de ordenação. Foram feitas uma séria de pesquisas a respeito de motores de busca *open source* e a ferramenta Terrier (TERRIER, 2011) foi a que mais se adequou ao requisito de inserção e remoção de algoritmos.

Para o levantamento do perfil foram utilizadas as engenharias presentes no *campus* Faculdade do Gama (FGA), com exceção da engenharia aeroespacial, pois a mesma não possui grade curricular definitiva até a realização do trabalho. A grade curricular foi utilizada para o levantamento da quantidade de matérias que cada engenharia possui em determinada área de conhecimento.

Todas as matérias da grade curricular foram alocadas em uma das nove áreas de conhecimentos levantadas e assim, através da parcela curricular que cada área de conhecimento tinha sobre determinada engenharia, foram estabelecidos os perfis.

Para que a realização da experimentação, é necessário que haja um *corpus* indexado ao motor de busca. O Terrier possui seu conjunto de páginas, *logs* de alteração de versão, Javadoc, entre outras páginas, todas no formato HTML disponíveis para a simulação de um *corpus*, mas optou-se por um *corpus* mais aderente aos perfis levantados para o trabalho. Para a formação do *corpus* foram utilizadas 30 páginas do Wikipédia que fazem parte das áreas de conhecimento levantadas e que possam fazer parte de consultas pré-definidas para o experimento.

Na Figura 7 é possível observar a representação gráfica do desenvolvimento do trabalho.

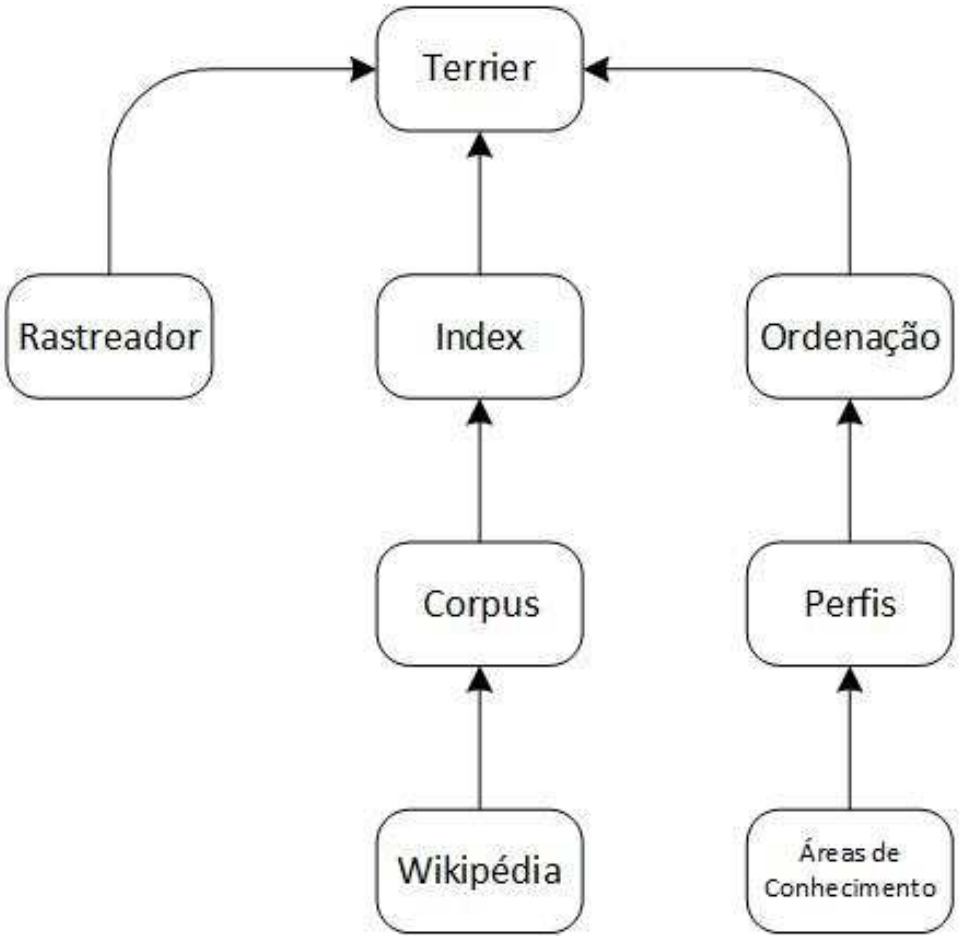


Figura 7 – Desenvolvimento do trabalho

Na Figura 7 é mostrado que o Wikipédia é quem fornece o *corpus* para a indexar a ferramenta e no componente de ordenação foram inseridos os perfis relacionados às áreas de conhecimento levantadas.

## 4.2 Testes

A segunda etapa do trabalho consistiu em testar os algoritmos implementados adaptados de ordenação com os perfis definidos implementados em um conjunto de três classes, isto é, basicamente, tinha o objetivo de analisar a qualidade do algoritmo de ordenação, se o algoritmo ordena os documentos do jeito que o usuário espera.

Para a realização destes testes foi elaborada uma planilha com os pesos para cada perfil e consultas contendo termos que pudessem abranger a maior quantidade de perfis possível, incluindo a busca sem perfil.

Utilizando as métricas de precisão e de *recall*, foi elaborada outra planilha em que relaciona cada busca realizada com cada um dos perfis, levantando os dados das métricas utilizadas

Com estes dados foram elaborados gráficos de precisão x recall (YATES; NETO et al., 1999), que possibilitam a análise das métricas através das curvas construídas podendo, estabelecer qual algoritmo em determinado perfil pode ser mais satisfatório.

Na Figura 8 é possível observar a representação gráfica dos testes realizados:

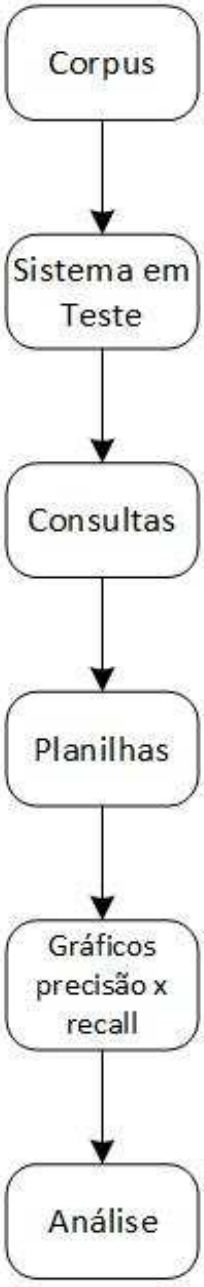


Figura 8 – Fluxo de testes do trabalho

Na Figura 8 é representado o corpus indexado na ferramenta em período de testes para a realização de consultas. Essas consultas geraram planilhas contendo dados a respeito da precisão x *recall* para determinada consulta em determinado perfil e algoritmo gerando gráficos da curva e posteriormente gerando a análise dos dados e dos gráficos.

## 4.3 Ferramenta Terrier

O foco deste trabalho está centralizado no módulo que trata da ordenação dos documentos recuperados. Para suprir a necessidade da utilização dos componentes de *crawling* e indexação sem que houvessem suas respectivas implementações, foi utilizado o motor de busca *open source* chamado Terrier 3.5 (TERRIER, 2011), implementado em Java pela Universidade de Glassgow. O Terrier é uma ferramenta altamente flexível, que implementa o estado da arte da indexação e das funcionalidades de recuperação, além de fornecer uma ótima plataforma para desenvolvimento, avaliação e experimentação a respeito da recuperação de textos (OUNIS et al., 2006). A arquitetura do Terrier é modular, sendo possível identificar características do padrão de projeto *Factory Method*. A documentação da ferramenta recomenda que se utilize as super classes abstratas para, por exemplo, implementar uma classe filha de um novo modelo de ordenação de documentos.

O Terrier possui módulos *desktop* e um módulo para *web*. Durante este trabalho foi utilizado o módulo *desktop*. Um dos motivos para esta escolha foi a fácil implementação da interface do módulo *desktop*, uma vez que foi feita de forma modularizada e compreensível para a manutenção, não possuindo códigos de interface bloqueados e permitindo a inserção de novas funcionalidades como o componente de escolha de perfil e o de escolha de algoritmo. O outro motivo é não haver tempo hábil para compreender o funcionamento da ferramenta como uma plataforma *web*.

## 4.4 *Corpus*

O conjunto de documentos indexados conhecidos que serão recuperados durante a pesquisa chama-se *corpus*. Através de consultas, como por exemplo *"corpus page ranking"* no Google, é possível encontrar vários resultados sugerindo que se utilize o *corpus* do Wikipédia como documentos, utilizando os artigos que existem no Wikipédia para fazerem parte do conjunto de documentos que será indexado e recuperado na busca.

Inicialmente seriam utilizados cerca de cem documentos do Wikipédia, porém a forma com que o trabalho foi modelado e as consultas que seriam posteriormente realizadas, para o levantamento das métricas de desempenho da recuperação de informação, mostram que essa quantidade de documentos onera o controle completo da experimentação da ordenação. Estabeleceu-se então que seriam selecionados três documentos para cada uma das áreas de conhecimento definidas através da definição de perfil e do levantamento de matérias de cada curso. Junto com esses documentos para cada área de conhecimento, foram selecionados também os documentos que definem cada um dos perfis, neste caso as quatro engenharias da UnB campus Gama, que será explicado posteriormente.

A ferramenta Terrier apresenta em sua documentação que seu *encode* é o UTF-8.

Entretanto, caso haja algum comentário ou alguma string com acentos, a sua *build* falha e não é construida devido à diferença de *encode* entre os arquivos de entrada e o script da construção da build. Para contornar este problema, foram retirados todos os acentos de todos os documentos selecionados para indexação, assim como os termos relacionados para cada área de conhecimento.

Para a seleção dos documentos, preocupou-se com a quantidade de termos que pertencem a alguma área de conhecimento e estão contidos dentro do artigo e as possíveis consultas com mais de um termo que poderiam ser realizadas com aquele documento, uma vez que os algoritmos de ordenação do Terrier tomam como base para pontuação dos documentos um cálculo em que leva em consideração a quantidade e a localização dos termos dentro do documento, ao contrário do *page rank* ou HITS que levam em consideração as referencias dos *links* entre as páginas.

## 4.5 Perfis

O levantamento de perfil foi uma atividade importante neste trabalho, pois foi através do levantamento de perfis que se estabeleceu os relacionamentos entre o peso e a ordenação de resultado. No decorrer da primeira etapa do trabalho, planejava-se levantar perfis entre sexos e idades, de acordo com pesquisas e horários ou *sites* em que determinado perfil de usuário mantém como "rotina". Porém essa possibilidade foi arquitetada antes mesmo de definir como e o que seria implementado a respeito do motor de busca: nesse ponto havia sido apenas levantado quais os motores de busca ou *frameworks open sources* que poderiam vir a ser utilizados no trabalho. Assim que foi definido o Terrier como a ferramenta de motor de busca para o trabalho e analisando o funcionamento de seus algoritmos de ordenação baseados nos termos contidos nos documentos, surgiu necessidade de implementar perfis que correspondessem a essa estrutura, nascendo assim a ideia de levantamento de perfis das engenharias da UnB no campus da Faculdade do Gama (FGA).

O levantamento deste perfil consiste basicamente em analisar a grade curricular de cada uma das engenharias da FGA, automotiva, eletrônica, de energia e de *software*, e através deste currículo analisar qual a porcentagem de matérias que cada um dos cursos possui em determinada área de conhecimento. Por exemplo, a engenharia de *software* possui pouco mais de 50% de seu currículo concentrado na área de conhecimento de computação, porém apenas 6% em química.

Através deste levantamento de perfis e da análise da porcentagem de cada área de conhecimento, foram estabelecidos pesos proporcionais para os termos em que pertencem à área de conhecimento com maior porcentagem para algumas das engenharias. Se pesquisarmos, por exemplo, pelo termo "computação", o perfil de engenharia de software terá um peso consideravelmente maior do que o perfil de engenharia automotiva.

Posteriormente foram elaborados pesos proporcionais à quantidade. Por exemplo, a engenharia de *software* possui exatos 54% das matérias de sua grade concentradas na área de conhecimento de computação, então foi atribuido um peso de 1,54 aos termos relacionados à computação, aumentando em 54% o valor deste termo, porém esse peso é muito pequeno, não sendo o suficiente para influenciar na consulta de diferentes perfis. Foi então elaborado duas divisões de pesos, a primeira, na Tabela 1, em que é atribuido 1000, 100 e 1 para a área de conhecimento com maior quantidade de matérias em determinada engenharia, a segunda maior área de conhecimento e o restante das áreas de conhecimento para determinada engenharia respectivamente, e a segunda atribuição, na Tabela 2, de 10000, 5000, 1000, 100 e 1 para a maior, segunda maior, terceira maior, quarta maior e o restante das áreas de conhecimento para determinada engenharia. Com esta escala maior de valores, é possivel atribuir pontuação maior para os documentos relacionados com a consulta, resultando em diferentes ordenações de resultados.

| Peso | | | | |
|---|---|---|---|---|
| Pefis | 1.000 | 100 | 10 | 1 |
| Engenharia Automotiva | Engenharia Mecânica | Matemática | Engenharias | Demais Áreas de Conhecimento |
| Engenharia Eletrônica | Engenharia Elétrica | Engenharias | Matemática | |
| Engenharia de Energia | Matemática | Engenharia Mecânica | Engenharias | |
| Engenharia de Software | Computação | Engenharias | Matemática | |

Tabela 1 – Tabela de pesos para os perfis na escala 1000

| Peso | | | | | | |
|---|---|---|---|---|---|---|
| Perfis | 10.000 | 5.000 | 1.000 | 500 | 100 | 1 |
| Engenharia Automotiva | Engenharia Mecanica | Matemática | Engenharias | Física | Engenharia Elétrica | Demais Áreas de Conhecimento |
| Engenharia Eletrôica | Engenharia Elétrica | Engenharias | Matemática | Física | Engenharia Mecânica | |
| Engenharia de Energia | Matemática | Engenharia Mecânica | Engenharias | Física | Química | |
| Engenharia de Software | Computação | Engenharias | Matemática | Engenharia Elétrica | Engenharia Mecânica | |

Tabela 2 – Tabela de pesos para os perfis na escala 10.000

As consultas de apenas um termo não trazem resultados satisfatórios, pois muitas das vezes os algoritmos aumentam apenas a escala das pontuações atribuidas aos documentos. Para uma consulta de um único termo, por exemplo “computação”, os documentos recuperados, em teoria, trazem a mesma ordem para diferentes perfis. Uma consulta computação nunca irá trazer o documento “computação” para engenharia de *software* e o documento “matemática” para a engenharia automotiva, sempre trará aos dois o documento “computação”, com pontuação diferenciada de acordo com o peso do perfil.

## 4.6 Áreas de Conhecimento

As áreas de conhecimento definem a qual conjunto pré-determinado de conhecimentos específicos alguma matéria ou documento pertence, isto é, a matéria "Cálculo 1", comum a todas as engenharias, pertence a área de conhecimento "matemática" definida para o trabalho. As áreas selecionadas são algumas das estabelecidas pela CAPES (CAPES, 2012).

As áreas de conhecimento escolhidas não foram em um nível alto de detalhes. Em outras palavras, a matéria "Álgebra Linear" não foi definida como pertencente a área de conhecimento "álgebra" e sim na "matemática", para que não houvesse uma quantidade excessiva de áreas de conhecimento, acarretando em uma grande fragmentação das áreas com poucas matérias em cada uma delas. Entretanto, houve a necessidade da inclusão de áreas de conhecimento diretamente relacionadas a alguma engenharia, por exemplo, não foi possível encaixar a matéria "Gestão da Produção e Qualidade" em alguma das áreas de conhecimento como "física" ou "quimica" e sim na "Engenharia de Produção".

Após a seleção das áreas de conhecimento que fariam parte do trabalho, foi mapeado então as áreas de conhecimento que todas as matérias do currículo de todas as engenharias, com exceção da engenharia aeroespacial, pertencem. Depois de realizado esse mapeamento, calculou-se a porcentagem de matérias de determinada engenharia que pertencem à uma área de conhecimento específica, analisando assim qual área de conhecimento possui maior peso para determinada engenharia. As Tabelas 3 e 4 mostram a porcentagem de cada área de conhecimento para cada engenharia:

| | Engenharia Automotiva | | Engenharia Eletrônica | |
|---|---|---|---|---|
| Áreas de Conhecimento | Total de Créditos | Porcentagem Dentro do Currículo | Total de Créditos | Porcentagem Dentro do Currículo |
| Matemática | 58 | 23% | 42 | 17% |
| Física | 37 | 15% | 33 | 13% |
| Química | 14 | 6% | 6 | 2% |
| Computação | 6 | 2% | 6 | 2% |
| Engenharia de Produção | 8 | 3% | 4 | 2% |
| Engenharia Sanitária | 6 | 2% | 6 | 2% |
| Engenharia Elétrica | 14 | 6% | 88 | 36% |
| Engenharia Mecânica | 62 | 25% | 14 | 6% |
| Engenharias | 42 | 17% | 48 | 19% |

Tabela 3 – Tabela de áreas de conhecimento para a Engenharia Automotiva e Engenharia Eletrônica

| | Engenharia de Energia | | Engenharia de Software | |
|---|---|---|---|---|
| Áreas de Conhecimento | Total de Créditos | Porcentagem Dentro do Currículo | Total de Créditos | Porcentagem Dentro do Currículo |
| Matemática | 49 | 19% | 34 | 14% |
| Física | 37 | 14% | 6 | 3% |
| Química | 26 | 10% | 6 | 2% |
| Computação | 6 | 2% | 132 | 55% |
| Engenharia de Produção | 10 | 4% | 4 | 2% |
| Engenharia Sanitária | 21 | 8% | 4 | 2% |
| Engenharia Elétrica | 21 | 8% | 6 | 2% |
| Engenharia Mecânica | 45 | 18% | 6 | 2% |
| Engenharias | 42 | 16% | 44 | 18% |

Tabela 4 – Tabela de áreas de conhecimento para a Engenharia de Energia e Engenharia de Software

## 4.7 Precisão e *Recall*

Existem alguns métodos para validar ou avaliar o desempenho de um motor de busca. Geralmente, quando se trata a respeito de desempenho, a variável tempo é colocada em questão. Para avaliar o desempenho de um motor de busca com a variável tempo, faz maior sentido avaliar o módulo de indexação do motor de busca, uma vez que ele é o responsável por tratar da velocidade em que a informação é recuperada. No caso especifico deste trabalho, o que está sendo avaliado o é o desempenho do algoritmo de ordenação, isto é, analisaremos o quão preciso é o algoritmo no que se diz respeito a recuperar o documento esperado para determinada busca (YATES; NETO et al., 1999).

Para entender como está sendo avaliado desempenho do algoritmo é necessário entender as duas variáveis para o levantamento, o *recall* e a precisão(YATES; NETO et al., 1999). O *recall* é a fração dos documentos relevantes recuperados, podendo ser representado por:

$$Recall = \frac{|Ra|}{|R|} \tag{4.1}$$

onde $|Ra|$ é a quantidade de documentos recuperados, que pertencem ao conjunto de documentos julgados relevantes para determinada consulta, e $|R|$ é a quantidade de documentos do conjunto de documentos relevantes. Já a precisão é a fração dos documentos recuperados que pertencem ao conjunto de documentos relevantes pela quantidade de documentos recuperados até então, podendo ser determinado como:

$$Precisão = \frac{|Ra|}{|A|} \tag{4.2}$$

Possuindo os valores das duas variáveis, é necessário a construção do gráfico de precisão x *recall* para a realização de uma análise consistente e adequada.

Levando em consideração o caso mais simples do trabalho, utiliza-se a consulta "química física", que nas atribuições de peso, nenhum dos perfis possui peso para as áreas

de conhecimento física ou química. Na atribuição de peso, são atribuídos as três áreas de conhecimento mais influentes em cada um dos perfis, de acordo com as grades do cursos específicos da FGA. Julgando que na consulta "química física" possuímos o seguinte conjunto R = motor de combustão interna, biorrefinaria, máquina térmica de documentos relevantes para a consulta. E na determinada consulta encontramos o resultado apresentado na Figura 9:

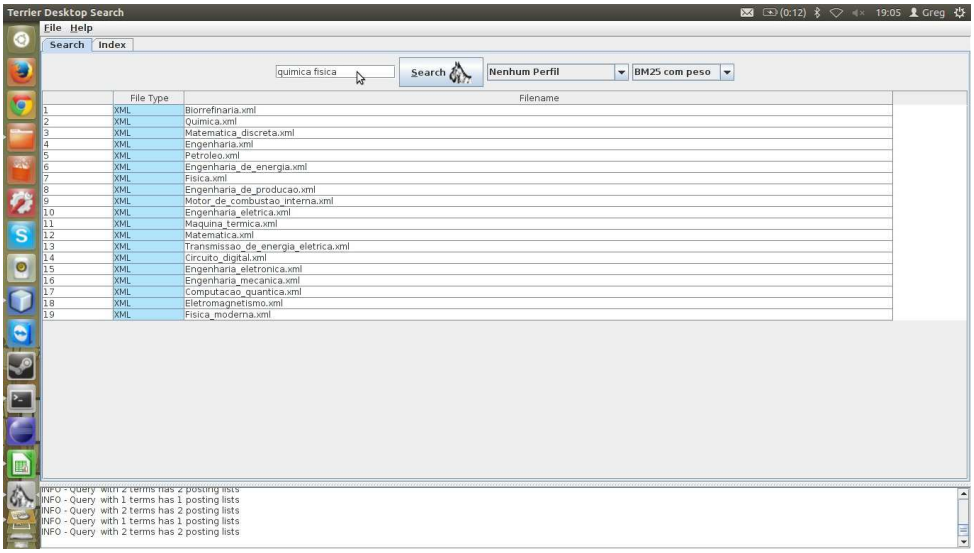


Figura 9 – Consulta "quimica fisica" sem nenhum perfil escolhido

lembrando que pela consulta não possuir peso para nenhum perfil levantado, a escolha do perfil não influencia na consulta. Através do resultado podemos observar que o primeiro resultado, o documento "biorrefinaria" faz parte do conjunto de documentos relevantes, e como ele é o primeiro de três documentos a ser recuperado, determinamos seu *recall* igual a 33,3% com a precisão igual a 100%, já o segundo documento relevante é trazido na nona posição, com o *recall* igual a 66,6% e precisão igual a 22,22% e por fim, o último documento é trazido na posição 11, com o recall de 100% e precisão de 27,27%. Com esses dados podemos construir o gráfico da Figura 10:


Figura 10 – *Recall* x Precisão para a consulta "quimica fisica" sem nenhum perfil escolhido

Através desse gráfico podemos ver que nos *recalls* mais baixos a precisão é satisfatória, no enquanto há um gargalo em que a precisão diminui muito e então retorna a aumentar, podendo deduzir que em uma quantidade maior de documentos dentro de um *corpus*, essa consulta com esse algoritmo tende a aumentar a precisão conforme o *recall* também aumenta.

O gráfico de *recall* x precisão é muito útil para a comparação de algoritmos. Na Figura 11 podemos observar um gráfico comparando os perfis das engenharias da FGA, além da consulta sem perfil para determinado algoritmo:

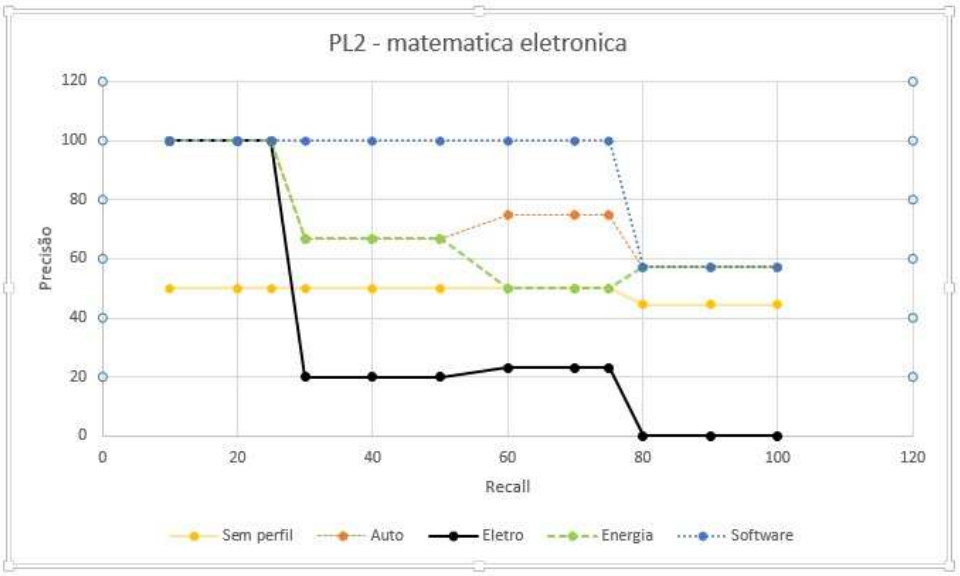


Figura 11 – *Recall* x Precisão para a consulta "matematica eletronica" para todos os perfis

Com a análise do gráfico, é possível julgar que para a consulta "matemática eletrônica" o perfil da engenharia do software traz os resultados mais satisfatórios, isto é, os documentos esperados para as consultas são os primeiros a serem trazidos, e posteriormente se equilibra com os demais perfis. Já o perfil da engenharia eletrônica, a linha em preto no gráfico, a partir do recall de 75% sua precisão cai para 0%, o motivo disso é que o último documento do conjunto relevante a ser recuperado, é recuperado no recall de 75%, neste caso, na consulta "matemática eletrônica" no perfil de engenharia de eletrônica não recupera um dos documentos relevantes do conjunto e o último documento relevante é recuperado com precisão de 23,08% e como já citado o recall igual a 75%.

O modelo padrão de *recall* x precisão, julga um conjunto de 10 documentos relevantes, plotando um gráfico que pode ser construído com maior detalhes. Para a construção destes gráficos, que foram levados em conta 3 e 4 documentos relevantes respectivamente, foi necessário realizar a interpolação dos dados. Considerando o caso do primeiro gráfico, no recall de 33,3% o algoritmo possui precisão de 100%, então para a construção do gráfico levamos em consideração os pontos 10, 20, 30 e 33,3% com precisão de 100%, já no segundo ponto de recall, de 66,6%, a precisão é igual a 22,22%, então levamos em consideração os pontos 40, 50, 60 e 66,6% com precisão igual a 22,22% e por fim, o recall de 100% é igual a precisão de 27,27% então nos pontos 70, 80, 90 e 100 consideramos a precisão igual a 27,27%. A análise de recall x precisão não leva em consideração a interação do usuário, sendo analisado os perfis como se fossem diferentes algoritmos de ordenação. Para uma análise como esta seria necessário a utilização das variáveis de cobertura e novidade, as quais não se aplicam a este contexto, uma vez que o usuário não conhece os documentos que serão recuperados, pois a indexação dos documentos é constante, diferente do contexto aplicado, em que todos os documentos a serem recuperados são previamente definidos.

## 4.8 Análise dos Algoritmos

As análises serão feitas com base nos gráficos construídos com diferentes consultas e perfis, onde a precisão é o eixo Y e o *recall* é o eixo X. As consultas realizadas são todas feitas sem acentuação, uma vez que foram removidos todos os acentos dos documentos e a busca deve ser do termo exato que se encontra no documento.

Para as análises foram feitas três consultas: "matematica computacao" que possui o conjunto de documentos relevantes constituído pelos documentos "matematica discreta", "computacao quantica", "computacao" e "matematica", a consulta "quimica fisica", que possui os documentos relevantes "motor de combustao", "biorrefinaria" e "maquina tarmica" e a última consulta de "matematica eletronica" que possui os documentos relevantes "matematica", "engenharia eletrica", "engenharia eletronica" e "circuito digital".

### 4.8.1 BM25

O algoritmo BM25 é o único algoritmo que não aceitou a implementação de peso e perfis. O BM25 apenas aplica uma escala aos documentos: não afeta a pesquisa de acordo com o perfis escolhido, e sim aumenta a pontuação dos documentos que possuem os termos. Entretanto é um bom algoritmo se levado em conta apenas a sua ordenação sem aplicação do perfil. Como é possível observar nas Figuras 12 e 13, o BM25 em todas as consultas possuiu uma precisão de 100% no primeiro ou nos primeiros *recalls*.

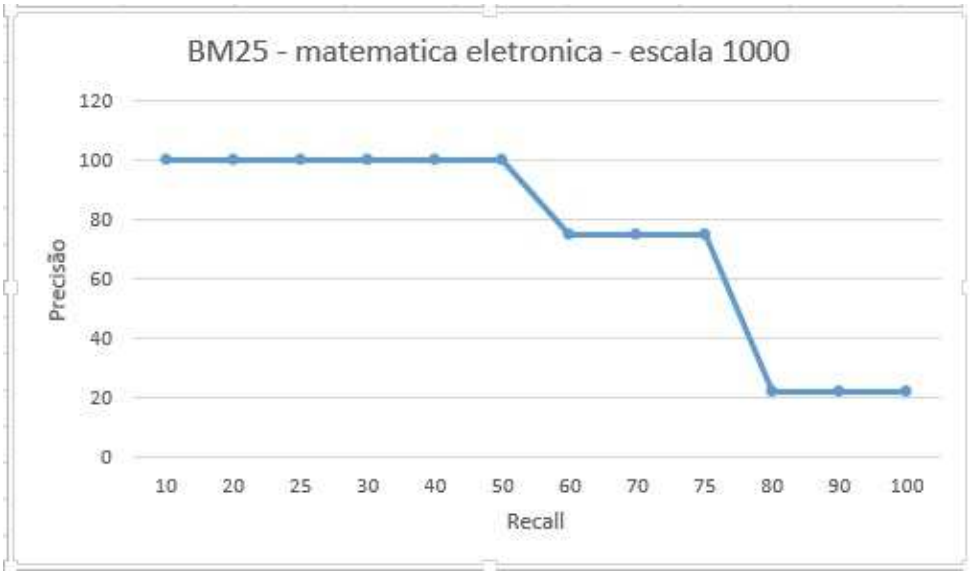


Figura 12 – Recall x Precisão para a consulta “matematica eletronica” para todos os perfis Algoritmo BM25 escala de peso 1000

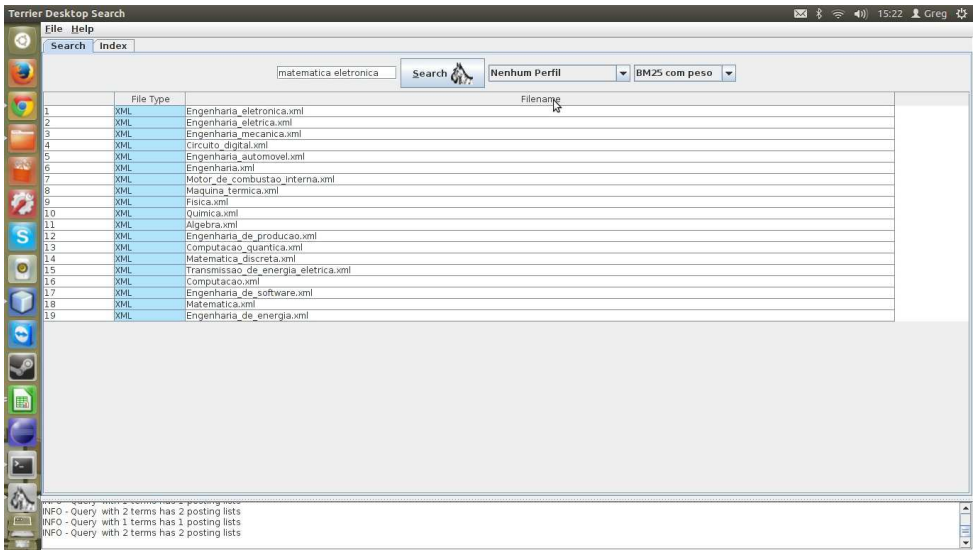


Figura 13 – Consulta “matematica eletronica” para qualquer perfil Algoritmo BM25 escala de peso 1000

### 4.8.2 DFI0

O algoritmo DFI0 é um algoritmo que diferencia apenas quando há ou não há perfil. Ele possui a curva relativamente satisfatória quando não há perfil, tendo 100% de precisão no primeiro *recall*, entretanto traz resultados tanto quanto imprecisos para determinadas consultas, como é possível observar nas duas Figuras 14 e 15.

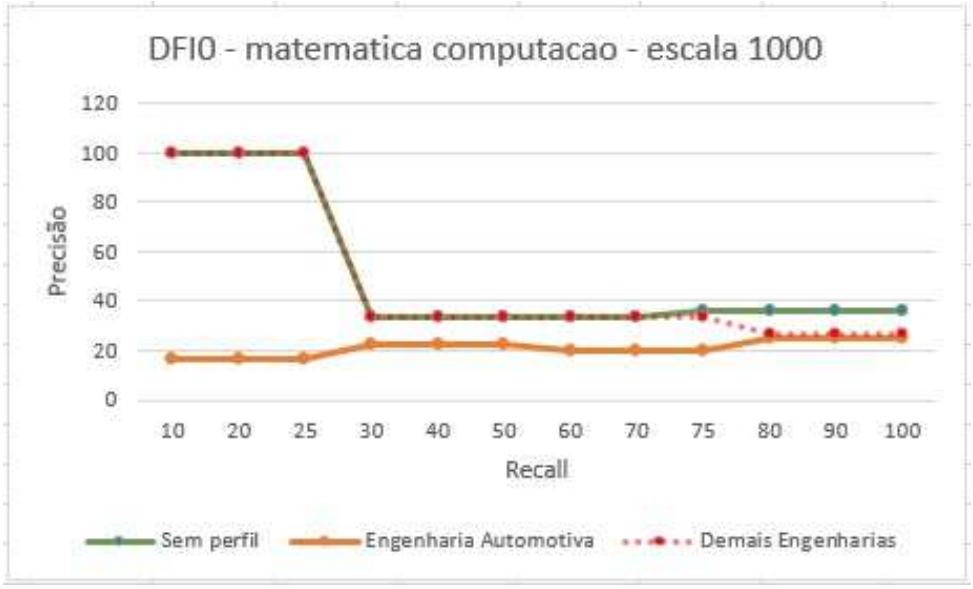


Figura 14 – Consulta "matematica computacao" no Algoritmo DFI0 escala de peso 1000

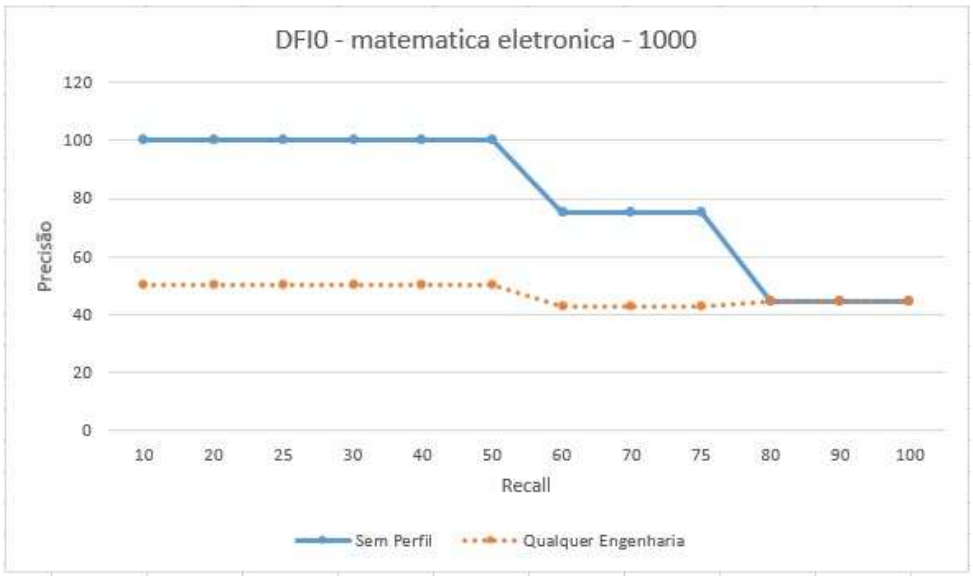


Figura 15 – Consulta "matematica eletronica" no Algoritmo DFI0 escala de peso 1000

No primeiro gráfico, Figura 14, na consulta "matematica computacao", é possível observar que sem perfil e para todos os outros perfis com exceção da engenharia automo-

tiva, possuem os primeiros *recalls* de 100%, enquanto a engenharia automotiva começa com o *recall* de cerca de 20% e aumenta muito pouco chegando perto dos 30%.

Já na consulta "matematica eletronica", o segundo gráfico, possui precisão de 100% até metade dos *recalls*, porém traz resultados ruins para a consulta com qualquer perfil, tendo me média sua precisão cerca de 50% durante quase todo o *recall*.

### 4.8.3 PL2

O PL2 é um algoritmo que apresentou bons resultados quando realizava-se consulta com perfil determinado, trazendo, na consulta "matematica computacao", até o *recall* 75% apresenta a precisão igual a 100% e mantendo precisão de 80% até o último *recall*, e mantendo a precisão nas consultas sem perfil entre 40 e 50%, como observa-se na Figura 16.

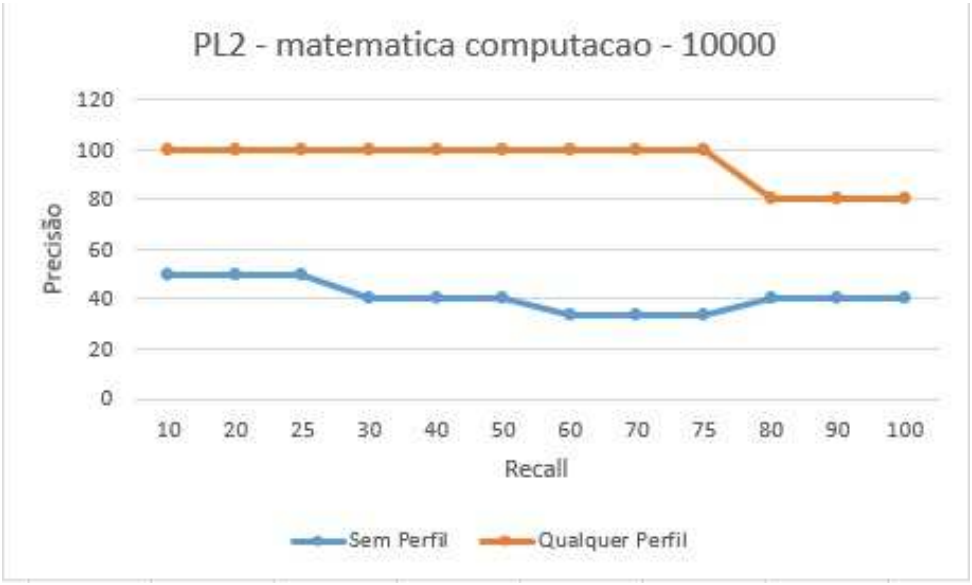


Figura 16 – Consulta "matematica computacao" no Algoritmo PL2 escala de peso 10000

Já para a consulta "fisica quimica", como se observa na Figura 17, quando aumentada a escala do peso para dez mil, há diferença apenas entre o perfil da engenharia de energia e os demais perfis, sendo que nos primeiros *recalls* apresenta uma precisão menor para engenharia de energia do que os demais perfis, porém nos últimos *recalls* possui precisão maior para a engenharia automotiva. Quando possui a escala de mil para os pesos, não há alteração na precisão entre perfis para esta consulta.

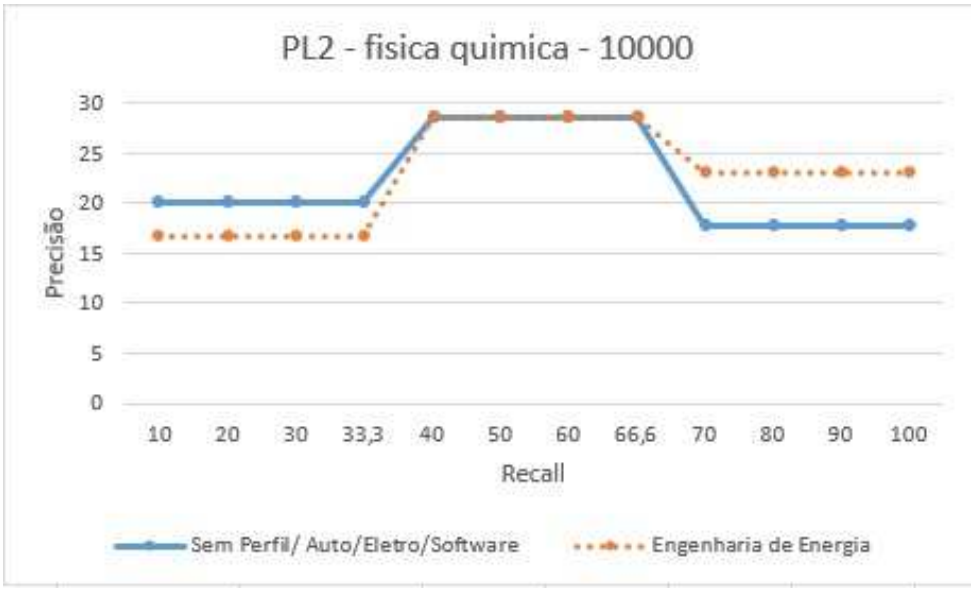


Figura 17 – Consulta “fisica quimica” no Algoritmo PL2 escala de peso 10000

O PL2 é um algoritmo que, para as duas consultas acima, não apresentou diferença significativa entre os resultados recuperados para os diferentes perfis, porém é o único algoritmo que apresentou as cinco diferentes curvas de precisão x *recall*. Apresentou resultado diferente para cada um dos perfis para a consulta “matematica eletronica” somente quando o peso para o perfil está definido na escala de mil. Como observa-se na Figura 18, o algoritmo PL2 é o algoritmo usado com maior alcance entre os perfis.

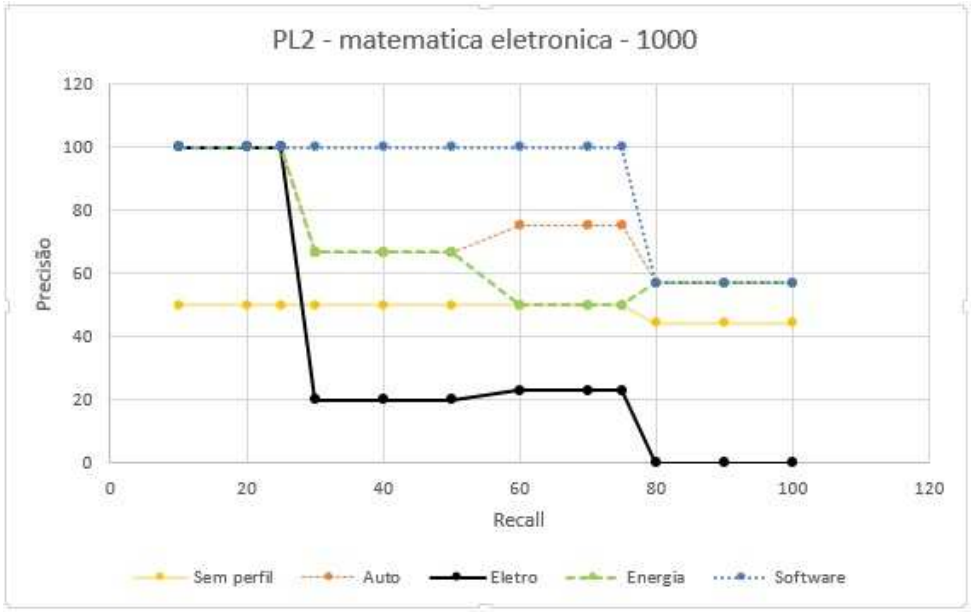


Figura 18 – Consulta “matematica eletronica” no Algoritmo PL2 escala de peso 1000

### 4.8.4 DLH

Foi dispendioso fazer com que o algoritmo DLH aceitasse a implementação de perfil. No caso da consulta "matematica computacao", tanto quando os pesos para os perfis na escala de dez mil como para a escala de mil são associados, o DLH não apresenta ordenação variada para os diferentes perfis, acarretando na diferença de apenas três conjuntos de precisão x *recall*, como é possível observar na Figura 19:

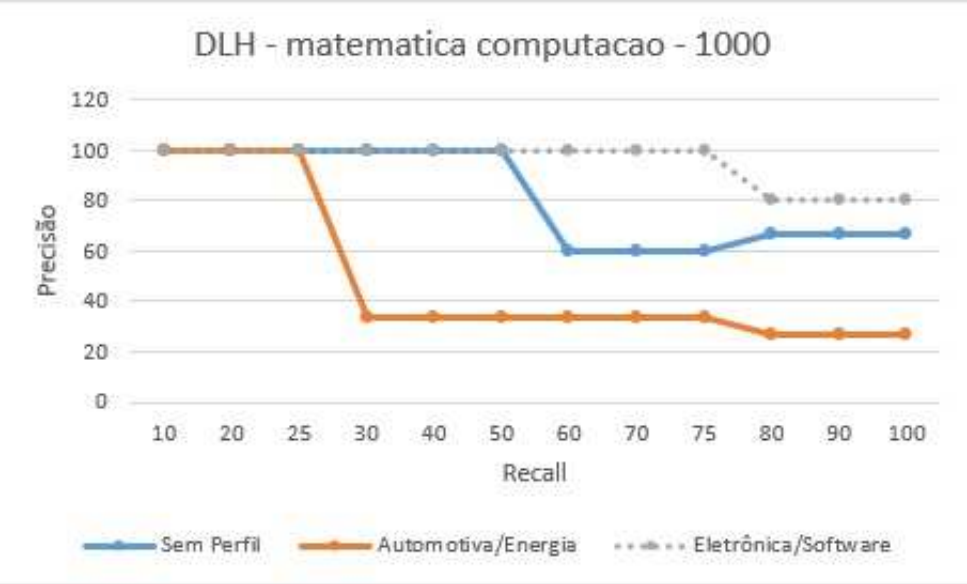


Figura 19 – Consulta "matematica computacao" no Algoritmo DLH escala de peso 1000

é possível que nos perfis da engenharia automotiva e da engenharia de energia a precisão atinge baixos níveis a partir dos 30% de recall. Já na consulta realizada sem perfil o algoritmos traz bons resultados, tendo sua precisão diminuída apenas no recall de 50%. Na consulta realizada no perfil de engenharia de software ou da engenharia eletrônica, o algoritmo traz ótimos resultados, mantendo sua precisão de 100% até o recall de 75% e posteriormente diminuindo sua precisão apenas para 80%.

Mesmo apresentando o gráfico restrito para a consulta acima, o DLH pode ser um algoritmo com potencial de alcance alto como é possível observar na Figura 20:

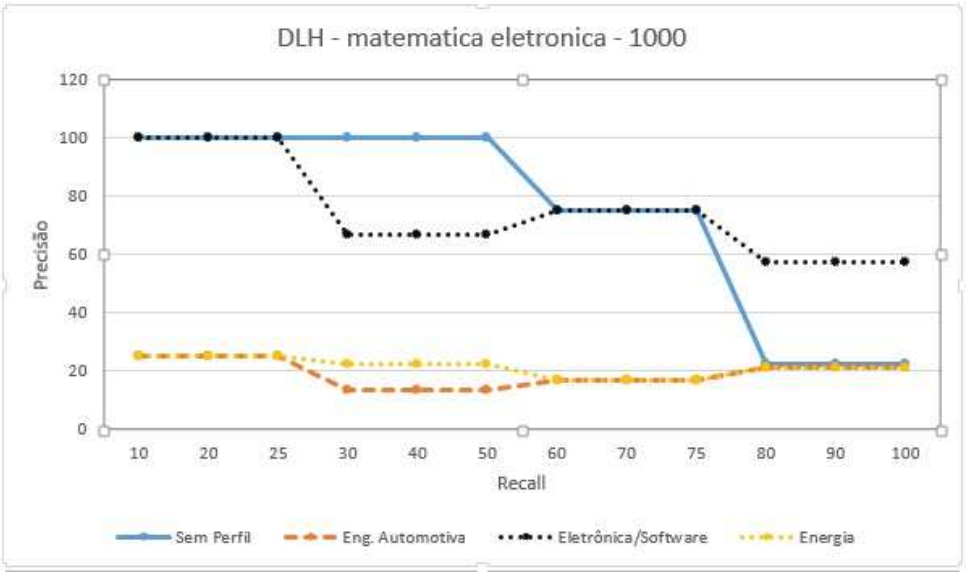


Figura 20 – Consulta “matematica eletronica” no Algoritmo DLH escala de peso 1000

porém há possibilidade de identificar que alto alcance não implica na boa precisão. Nos perfis de engenharia automotiva e engenharia de energia, os níveis de precisão não são satisfatórios, sendo restritos aos valores entre 16 e 25%. O nível de precisão da consulta realizada sem perfil mostra resultados ligeiramente satisfatórios, mostrando possuir precisão de 100% até 50% do *recall*, e posteriormente, nos *recall's* finais, apresenta o mesmo índice que o perfil de engenharia automotiva e engenharia de energia. A consulta para os perfis de engenharia de software e de engenharia eletrônica no DLH são os que mostram a precisão mais satisfatória, pois mesmo obtendo precisão de 100% até o *recall* de 25% apenas, manteve sua precisão sempre acima dos 50%.

### 4.8.5 DFRee

No algoritmo DFRee, para a consulta “matematica computacao” na escala de mil para os pesos, a curva precisão x *recall* foi exatamente igual para todos os perfis, como é possível observar na Figura 21, pois o algoritmo recuperou os documentos relevantes na mesma ordem para todos os perfis, trocando de ordem apenas os documentos não relevantes. Isso aconteceu pois a escala de mil atribui peso apenas para as três áreas de conhecimento de maior significância para determinada engenharia.

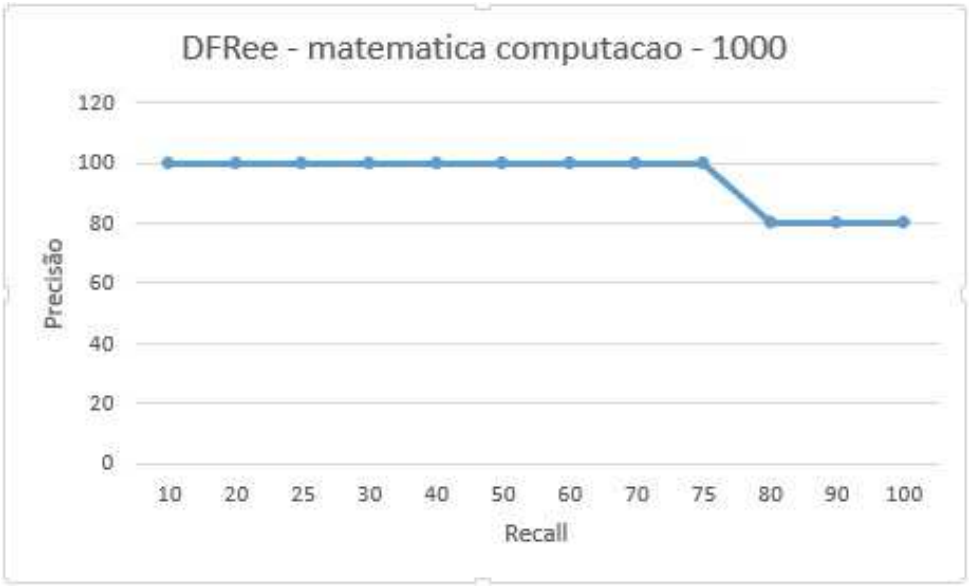


Figura 21 – Consulta “matematica computacao” no Algoritmo DFRee escala de peso 1000

Para a mesma consulta, com o peso na escala de dez mil, houveram três diferentes curvas para a precisão x *recall* como observa-se na Figura 22. O perfil de engenharia automotiva manteve baixos índices de precisão, porém sempre aumentando conforme o *recall* também aumenta, sendo possível inferir que se a quantidade de documentos relevantes fosse maior, a sua precisão chegaria a níveis satisfatórios para altos *recall's*, o que pouco usual. Para a consulta sem perfil e para os perfis de engenharia de software e engenharia eletrônica, para ambos, o DFRee trouxe ótimos resultados, entretanto é possível afirmar que a consulta sem perfil obteve melhores resultados, uma vez que no *recall* de 50% a consulta sem perfil manteve sua precisão de 100% enquanto os perfis de software e eletrônica obtiveram precisão de 75%.

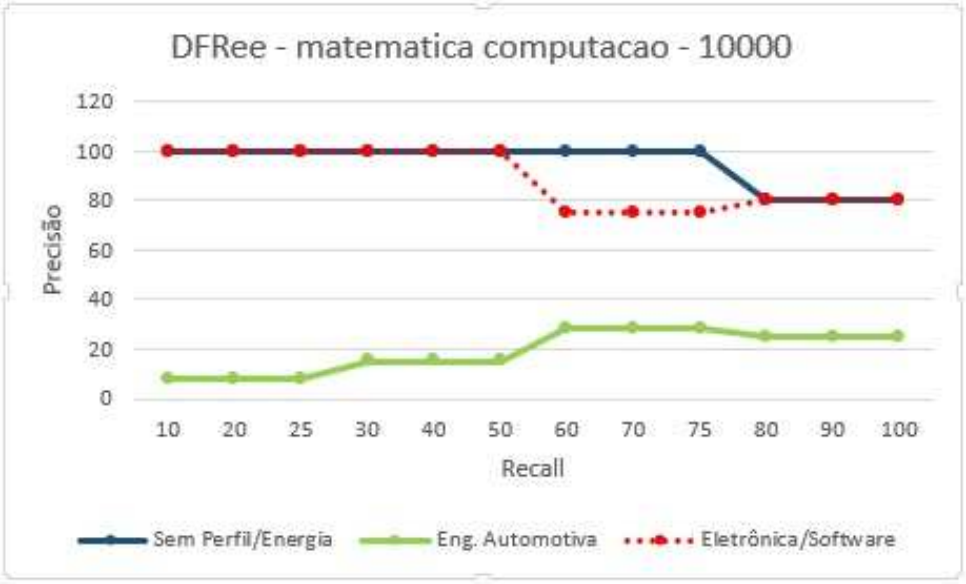


Figura 22 – Consulta "matematica computacao" no Algoritmo DFRee escala de peso 10000

Assim como o DLH, o DFRee é um algoritmo com potencial para alcances altos, como se observa na Figura 23. Na consulta "matematica eletronica" tanto na escala de mil quanto para a escala de dez mil, o algoritmo apresenta cinco curvas variadas para a precisão x *recall*, trazendo os resultados para suas diferentes consultas realizada em cada perfil. Na Figura 23 é possível perceber que a precisão se mantém a mesma para todos os perfis até o *recall* de 50%, posteriormente, a precisão de todos os perfis diminui variando de cerca de 20% para a engenharia de software e engenharia eletrônica, até cerca de 60% para a consulta realizada sem perfil.

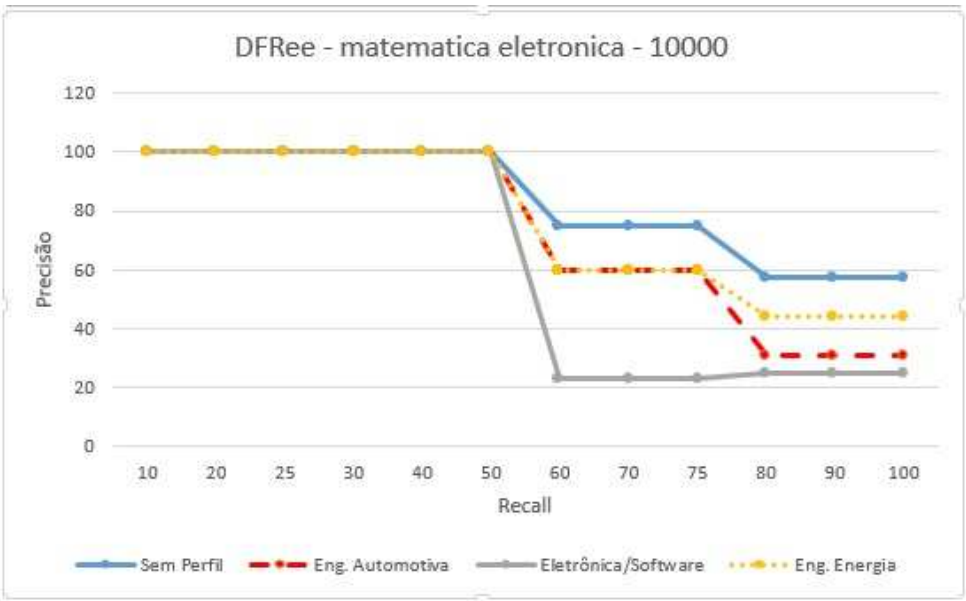


Figura 23 – Consulta "matematica eletronica" no Algoritmo DFRee escala de peso 10000

## 4.9 Resultados

A avaliação dos algoritmos com três consultas diferentes gerou quinze tabelas para cada consulta e em cada uma das escalas de peso, mil e dez mil, totalizando trinta diferentes tabelas comparando a precisão e *recall* de cada perfil para os diferentes algoritmos escolhidos para o experimento. Para cada tabela gerada, foi gerado também um gráfico correspondente para a análise visual da curva de precisão x *recall*. Em cada uma das tabelas é feito o cálculo da média simples entre as precisões encontradas. Para eleger os melhores algoritmos, não basta ter as maiores médias, mas também oferecer variedade nos resultados, representado por várias curvas dentro do gráfico respectivo à tabela ou de acordo com as imagens da consulta realizada na ferramenta. O conjunto variedade e maiores precisões indicam quais são os melhores algoritmos. Para definir qual é o melhor algoritmo geral, foi levado em consideração a quantidade de vezes que o algoritmo foi considerado melhor entre as consultas da mesma escala. Através destas definições, se estabeleceu o melhor algoritmo para a consulta e o melhor algoritmo geral.

### 4.9.1 Melhores Algoritmos para as Consultas

Na consulta "matematica computacao" para o peso de escala mil, à primeira vista, o algoritmo DFRee pode passar a impressão de fugir dos padrões estabelecidos para a eleição do melhor algoritmo. Entretanto o algoritmo recupera documentos na ordenação diferente mas os documentos relevantes para a consulta são recuperados na mesma ordem para os diferentes perfis, acarretando em uma curva única no gráfico para cada um dos perfis. O algoritmo DFRee traz a surpreendente média de 95% de precisão para a consulta em questão, podendo, sem dúvidas, ser considerado o melhor algoritmo para esta consulta.

Já para a escala de dez mil, o algoritmo com maior média de precisão é o PL2 entretanto, o PL2 nesta consulta não atende o segundo requisito, a variedade de resultados entre os perfis, mantendo iguais a ordenação dos documentos para todos os perfis, exceto quando não há perfil selecionado. Desta forma, o melhor algoritmo é também o DFRee, pois recupera três conjuntos distintos de resultados, a maior quantidade para esta consulta, e mesmo possuindo uma das médias de precisão muito baixas, cerca de 19%, suas outras médias são altíssimas, de 88,75 e 95%.

A consulta "fisica quimica" torna difícil a decisão do algoritmo, uma vez que para a escala do peso mil, o único algoritmo que apresenta diferença de ordenação nos resultados é o DFRee, mas são resultados com baixos índices de precisão e apenas duas variações de resultados. Se desconsiderar os critérios estabelecidos, o melhor algoritmo a ser considerado para esta consulta, o BM25 pode ser eleito, entretanto o BM25 é um algoritmo que não aceitou a implementação de perfil. Sendo assim o melhor algoritmo neste caso é o DFI0, pois apresenta a maior média de precisão entre os algoritmos.

No peso de dez mil para a consulta acima, o cenário é exatamente igual, sendo eleito também DFI0. Para a consulta "matematica eletronica" na escala de peso igual a mil, mesmo que o algoritmo DFI0 apresente as maiores médias, apresenta apenas duas variações de curvas nos gráficos, trazendo apenas dois conjuntos distintos de ordenação. O algoritmo PL2 foi o que apresentou maior variedade de curvas no gráfico, trazendo os cinco possíveis resultados para cada um dos perfis, porém possui duas médias relativamente insatisfatórias, abaixo dos 50%. O algoritmo DFRee se mostrou o melhor algoritmo nesta consulta, apresentando quatro curvas distintas no gráfico e todas com uma média de precisão maior que os 60%.

Na escala de peso igual a dez mil, o cenário é bem parecido. O algoritmo DFRee apresenta quatro curvas distintas dentro do gráfico, com as médias de precisão todas maiores que 60%, enquanto o segundo lugar, o PL2, apresenta três curvas diferentes e a média mínima de precisão próxima aos 50%.

### 4.9.2 Melhor Algoritmo Geral

É bem evidente que o algoritmo DFRee é melhor algoritmo dentre os selecionados, para as consultas realizadas e os para os perfis definidos, sendo eleito o melhor algoritmo em quatro das seis consultas com os diferentes pesos. O segundo melhor algoritmo é o DFI0, pelos seus resultados na consulta "fisica química" nas diferentes escalas de peso.

Essa decisão não quer dizer necessariamente que os outros algoritmos utilizados são ruins, muito pelo contrário, o exemplo bem evidente disso é o BM25, que se mostrou um ótimo algoritmo, mas por não aceitar a implementação de perfis, não atinge um dos objetivos principais do trabalho.

# 5 Conclusão

Através dos dados de precisão x *recall* foi possível concluir que dos cinco algoritmos selecionados, apenas o BM25 rejeitou completamente a implementação de perfil. Não importa qual seja a consulta, o BM25 não irá ordenar os resultados de acordo com os perfis. O algoritmo DFI0 aderiu parcialmente à implementação de perfil, seu gráfico de precisão apresenta sempre dois conjuntos distintos de resultados mostrando que dentro dos perfis implementados, ele diferencia apenas dois conjuntos. Os demais algoritmos aceitaram relativamente bem a implementação de perfil, possuindo de três a cinco curvas de precisão, os quatro perfis mais a consulta sem perfil.

De acordo com as duas categorias estabelecidas de melhor algoritmo, a categoria melhor algoritmo para a consulta e a categoria melhor algoritmo geral, houveram somente dois algoritmos distintos que foram eleitos os melhores. Entre os melhores algoritmos para a consulta, o DFI0, das seis consultas, foi eleito em duas o melhor algoritmo enquanto o DFRee foi eleito para o restante. Consequentemente o DFRee foi o algoritmo que melhor satisfez ao requisitos de aderência à implementação de perfil e ao requisito de maior média de precisão relacionada à variedade dos resultados, para as consultas determinadas na experimentação.

Os algoritmos relacionados com busca *web*, os estáticos e os dinâmicos, não permitiram ser utilizados pela ferramenta utilizada no trabalho, pois os algoritmos eram baseados em *links* para realizar a pontuação, e a ferramenta utiliza algoritmos baseados em termos.

Para os trabalhos futuros serão realizadas as implementações de mais algoritmos baseados em termos com o perfil agregado, construir um *corpus* mais robusto, para análise mais detalhada de precisão x *recall* para os algoritmos deste trabalho e do trabalho futuro e por fim, analisar a viabilidade de integração dos algoritmos baseados em termos com os algoritmos estáticos, como o *page rank* e o HITS.

# Referências

ACTION, P. *Distribuição hipergeométrica*. 2011. [Online; accessed 19-Novembro-2013]. Disponível em: <http://www.portalaction.com.br/content/54-distribui%C3%A7%C3%A3o-hipergeom%C3%A9trica>. Citado na página 40.

ANTON, H.; RORRES, C. *Álgebra linear com aplicações*. [S.l.]: Grupo A, 2002. Citado na página 41.

BRIN, S.; PAGE, L. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, Elsevier, v. 30, n. 1, p. 107–117, 1998. Citado 2 vezes nas páginas 35 e 43.

CAPES. *Áreas de Conhecimento*. 2012. [Online; accessed 24-Setembro-2013]. Disponível em: <http://http://www.capes.gov.br/images/stories/download/avaliacao/TabelaAreasConhecimento_072012.pdf>. Citado na página 61.

DELAWARE, U. of. *Poisson Distribution Description*. 2013. Citado na página 41.

DINCER, B. T.; KOCABASE, I.; KARAOGLAN, B. *Irra at trec 2009: Index term weighting based on divergence from independence model*. [S.l.], 2009. Citado na página 39.

FOSKETT, D. J. Thesaurus. In: MORGAN KAUFMANN PUBLISHERS INC. *Readings in information retrieval*. [S.l.], 1997. p. 111–134. Citado na página 32.

FRAKES, W. B. *Information retrieval: Data structures & algorithms*. [S.l.]: Pearson Education India, 1992. Citado na página 31.

HE, J. et al. Gender: A generic diversified ranking algorithm. In: *Advances in Neural Information Processing Systems 25*. [S.l.: s.n.], 2012. p. 1151–1159. Citado 2 vezes nas páginas 49 e 52.

KLEINBERG, J. M. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, ACM, v. 46, n. 5, p. 604–632, 1999. Citado 2 vezes nas páginas 45 e 47.

LEMPEL, R.; MORAN, S. The stochastic approach for link-structure analysis (salsa) and the tkc effect. *Computer Networks*, Elsevier, v. 33, n. 1, p. 387–401, 2000. Citado na página 48.

MARKET, N. *Search engines market share*. 2009. [Online; accessed 14-Maio-2013]. Disponível em: <http://marketshare.hitslink.com>. Citado na página 28.

OUNIS, I. et al. Terrier: A High Performance and Scalable Information Retrieval Platform. In: *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*. [S.l.: s.n.], 2006. Citado na página 58.

RAMAN, K.; JOACHIMS, T.; SHIVASWAMY, P. Structured learning of two-level dynamic rankings. In: ACM. *Proceedings of the 20th ACM international conference on Information and knowledge management*. [S.l.], 2011. p. 291–296. Citado na página 49.

ROBERTSON, S. E.; JONES, K. S. Relevance weighting of search terms. *Journal of the American Society for Information science*, Wiley Online Library, v. 27, n. 3, p. 129–146, 1976. Citado na página 38.

SALTON, G.; MCGILL, M. J. Introduction to modern information retrieval. McGraw-Hill, Inc., 1986. Citado 4 vezes nas páginas 25, 26, 29 e 30.

SHARMA, A.; ADHAO, N.; MISHRA, A. A survey: Static and dynamic ranking. *International Journal of Computer Applications*, Foundation of Computer Science, 244 5 th Avenue,# 1526, New York, NY 10001, USA India, v. 70, n. 14, 2013. Citado 11 vezes nas páginas 42, 43, 44, 45, 46, 47, 48, 49, 50, 51 e 52.

SHARMA, D. K.; SHARMA, A. K. A comparative analysis of web page ranking algorithms. Citado 5 vezes nas páginas 27, 43, 44, 46 e 48.

SIGNORINI, A. A survey of ranking algorithms. *Department of Computer Science University of Iowa*, n. 11-C, p. 36–39, 2005. Citado na página 49.

SPIELMAN, D. A. Spectral graph theory and its applications. In: IEEE. *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*. [S.l.], 2007. p. 29–38. Citado na página 40.

TERRIER. *DFR Description*. 2011. [Online; accessed 19-Novembro-2013]. Disponível em: <http://terrier.org/docs/v3.5/dfr_description.html#randomnessmodel>. Citado 3 vezes nas páginas 37, 55 e 58.

TOP, R. to the. *Search Engine Market Share*. 2013. Disponível em: <http://risetothetop.techwyse.com>. Citado na página 28.

XAPIAN. *BM25 Description*. 2013. Citado na página 38.

XING, W.; GHORBANI, A. Weighted pagerank algorithm. In: IEEE. *Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on*. [S.l.], 2004. p. 305–314. Citado 2 vezes nas páginas 44 e 45.

YATES, R. B.; NETO, B. R. et al. *Modern information retrieval*. [S.l.]: ACM press New York, 1999. Citado 17 vezes nas páginas 23, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 42, 43, 44, 45, 57 e 62.

YUWONO, B.; LEE, D. L. Search and ranking algorithms for locating resources on the world wide web. In: IEEE. *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*. [S.l.], 1996. p. 164–171. Citado na página 35.

# Apêndices

# APÊNDICE A – Primeiro Apêndice

Aqui estão todas as consultas realizadas e os resultados recuperados pelos algoritmos, incluindo a imagem já colocada no texto.

## A.1 Escala 1000 para os pesos

### A.1.1 Consulta matemática computação

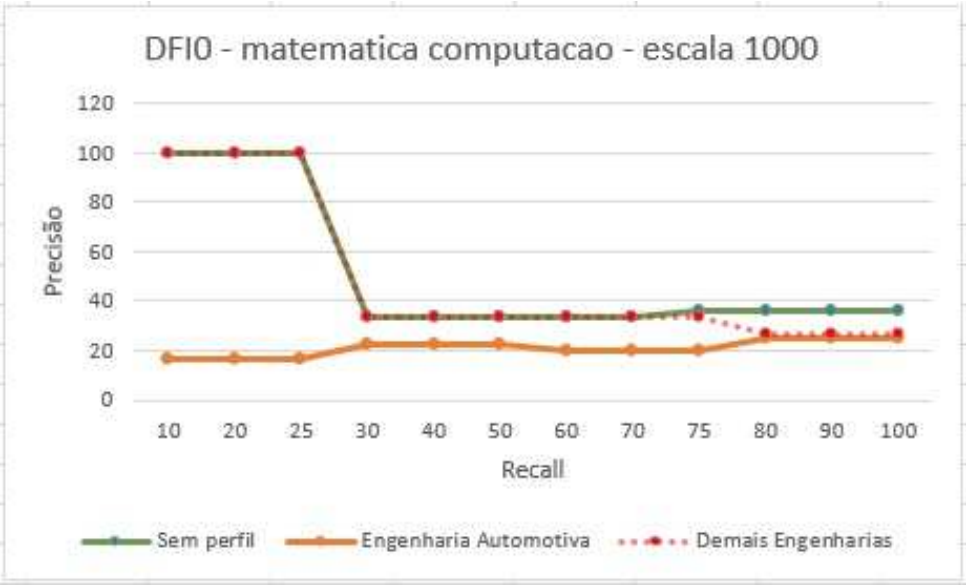


Figura 24 – Consulta “matematica computacao” no Algoritmo DFI0 escala de peso 1000

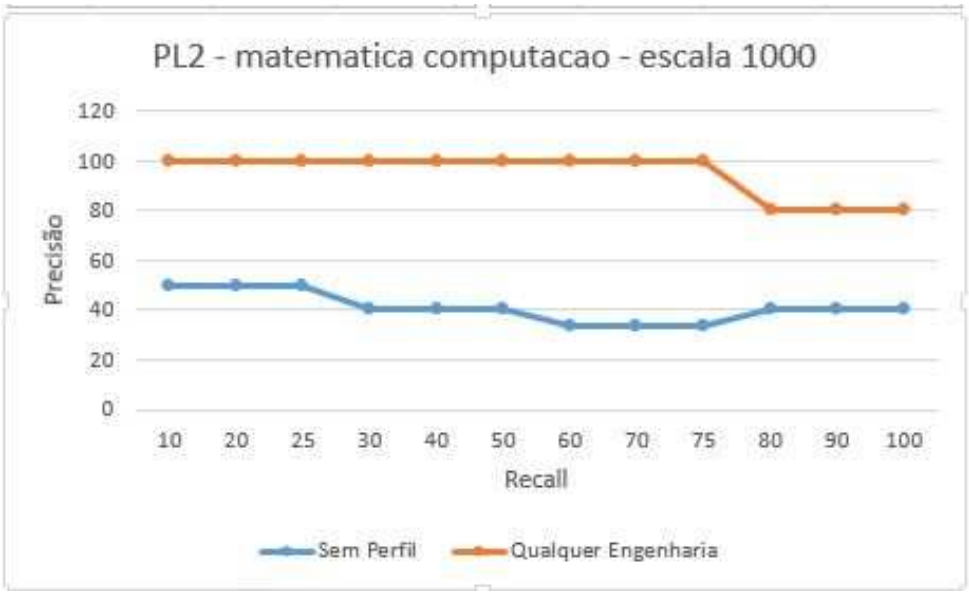


Figura 25 – Consulta “matematica computacao” no Algoritmo PL2 escala de peso 1000


Figura 26 – Consulta “matematica computacao” no Algoritmo BM25 escala de peso 1000

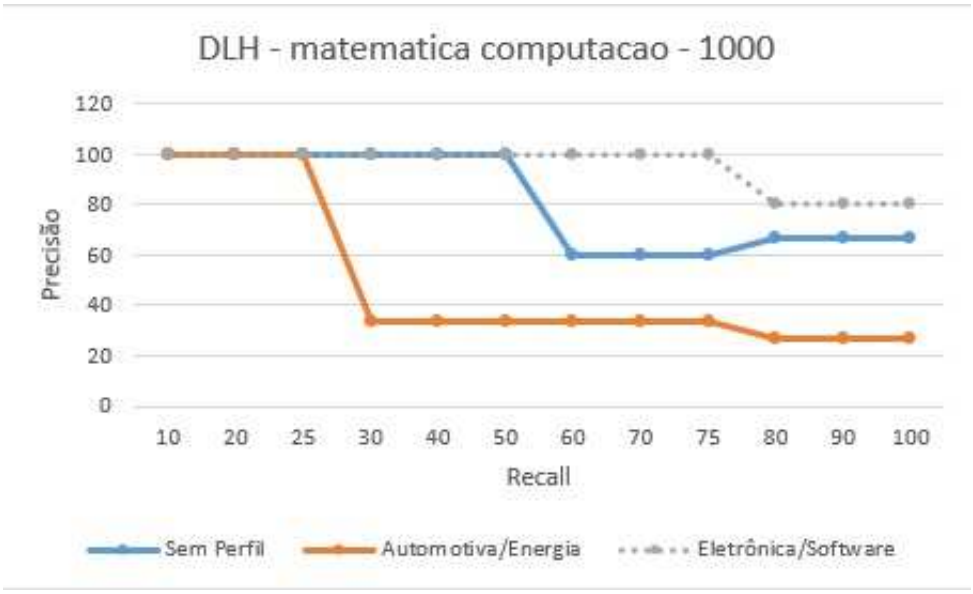


Figura 27 – Consulta "matematica computacao" no Algoritmo DLH escala de peso 1000

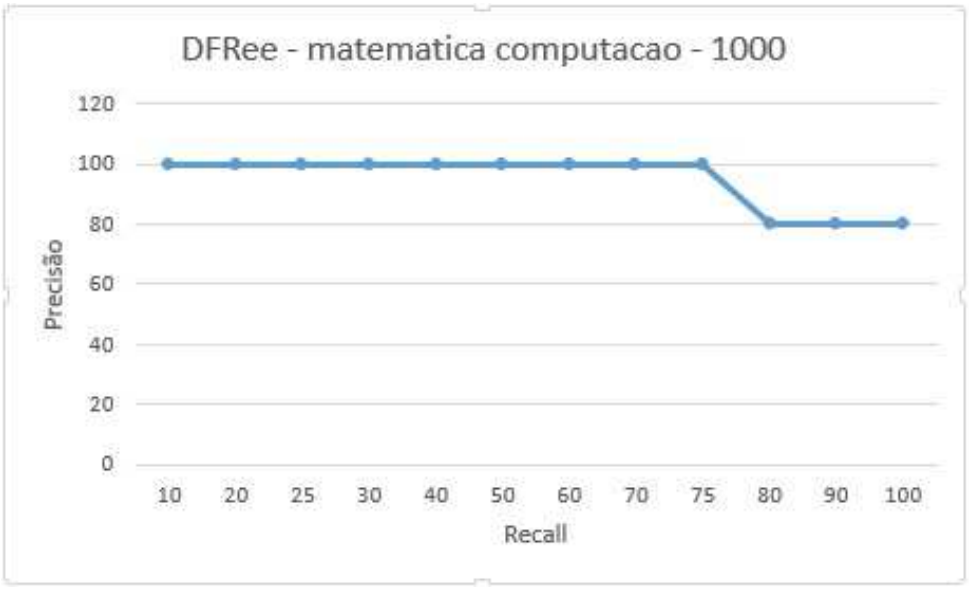


Figura 28 – Consulta "matematica computacao" no Algoritmo DFRee escala de peso 1000

### A.1.2 Consulta física química


Figura 29 – Consulta "fisica quimica" no Algoritmo DFI0 escala de peso 1000


Figura 30 – Consulta "fisica quimica" no Algoritmo PL2 escala de peso 1000


Figura 31 – Consulta "fisica quimica" no Algoritmo BM25 escala de peso 1000


Figura 32 – Consulta "fisica quimica" no Algoritmo DLH escala de peso 1000

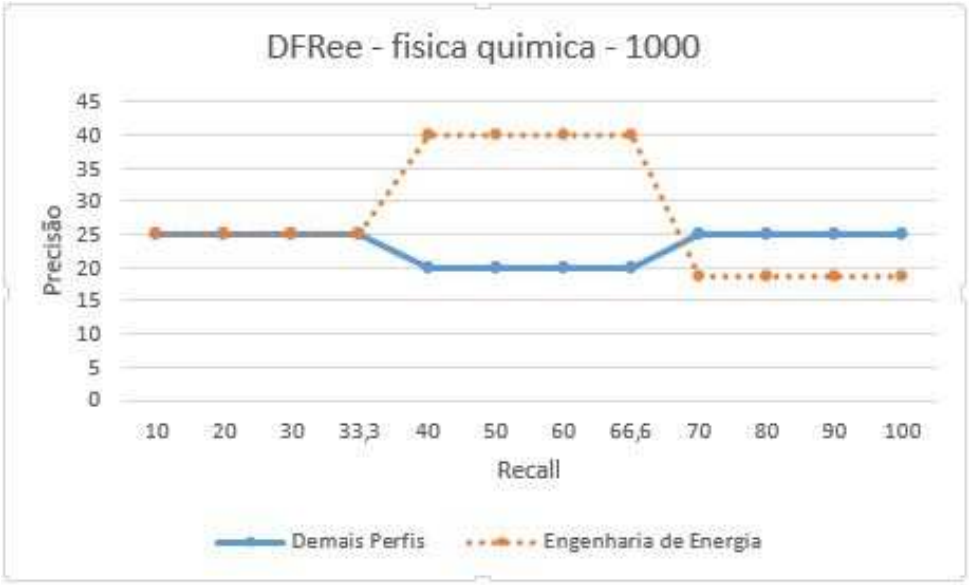


Figura 33 – Consulta "fisica quimica" no Algoritmo DFRee escala de peso 1000

### A.1.3 Consulta matemática eletrônica

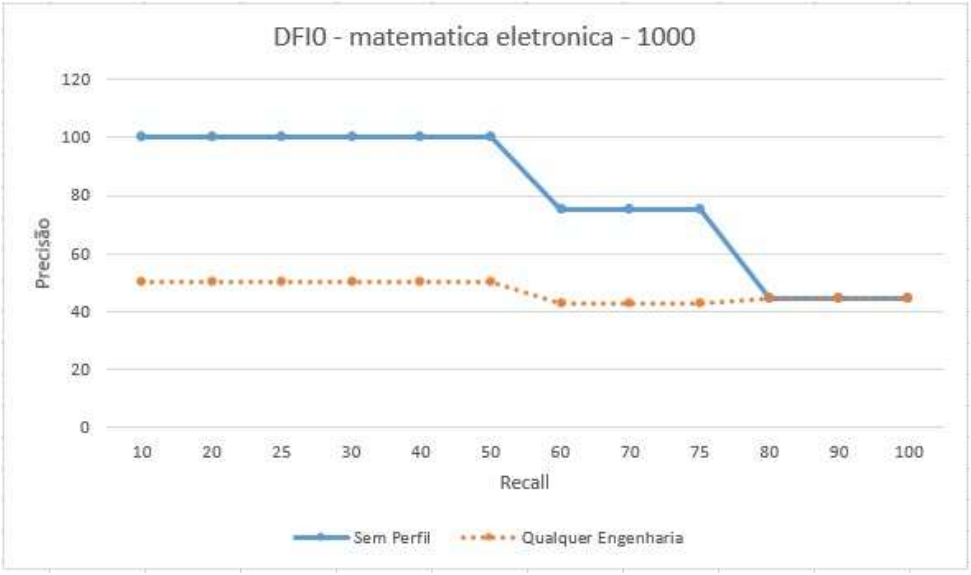


Figura 34 – Consulta "matematica eletronica" no Algoritmo DFI0 escala de peso 1000

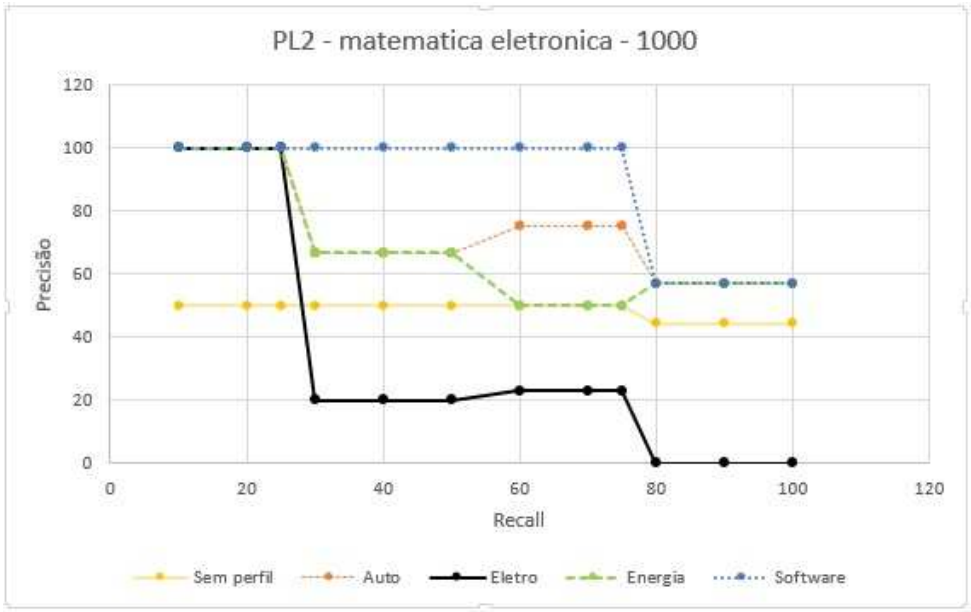


Figura 35 – Consulta "matematica eletronica" no Algoritmo PL2 escala de peso 1000

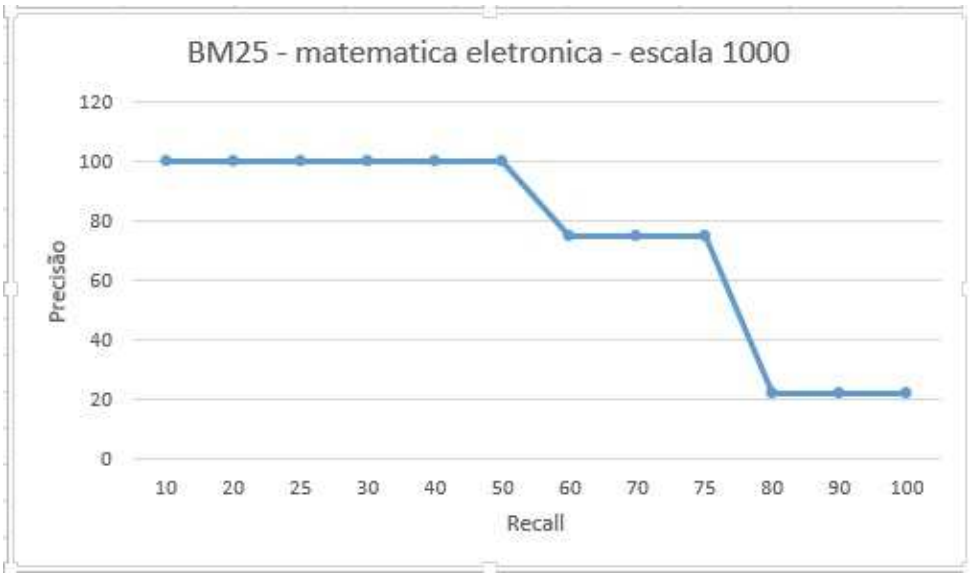


Figura 36 – Consulta "matematica eletronica" no Algoritmo BM25 escala de peso 1000

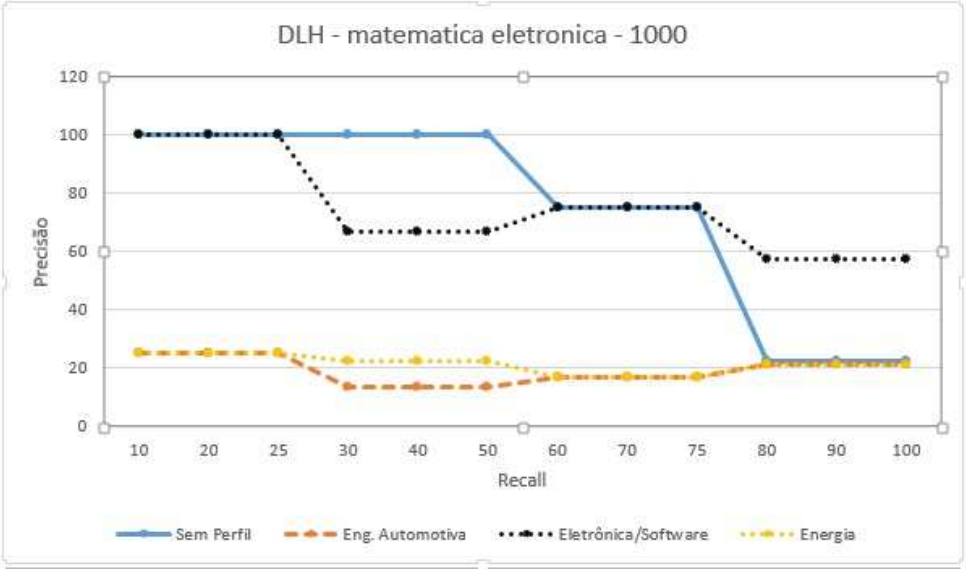


Figura 37 – Consulta “matematica eletronica” no Algoritmo DLH escala de peso 1000

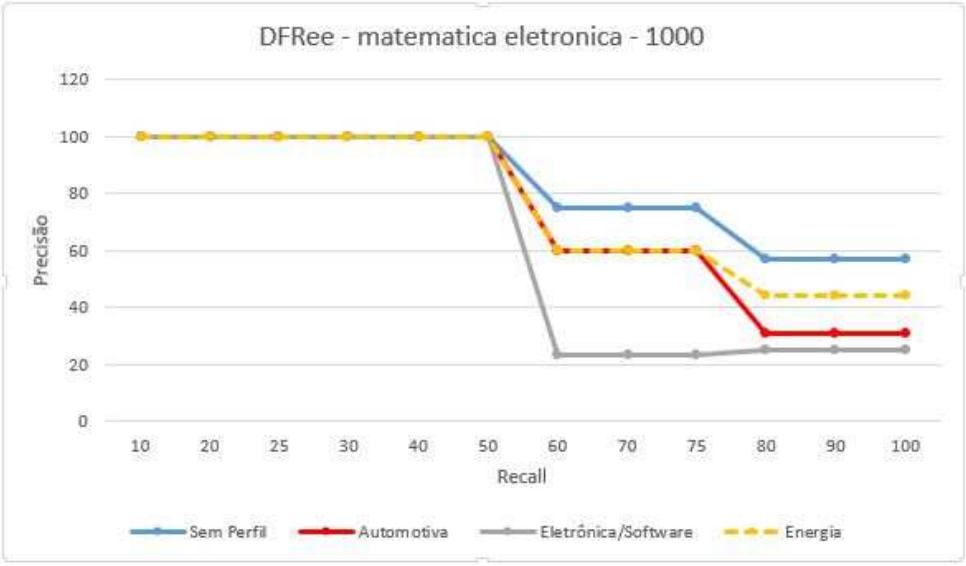


Figura 38 – Consulta “matematica eletronica” no Algoritmo DFRee escala de peso 1000

## A.2 Escala 10000 para os pesos

### A.2.1 Consulta matemática computação

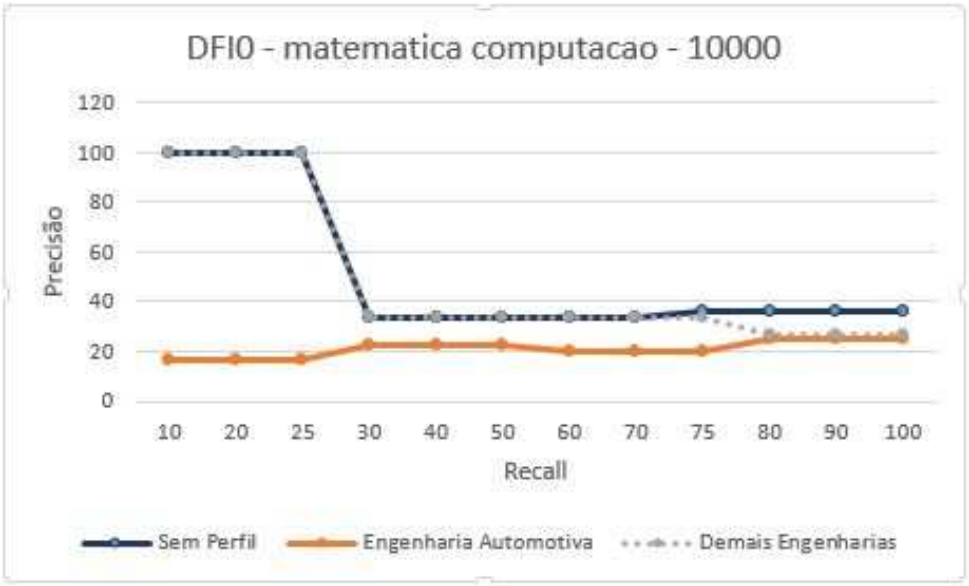


Figura 39 – Consulta "matematica computacao" no Algoritmo DFI0 escala de peso 10000

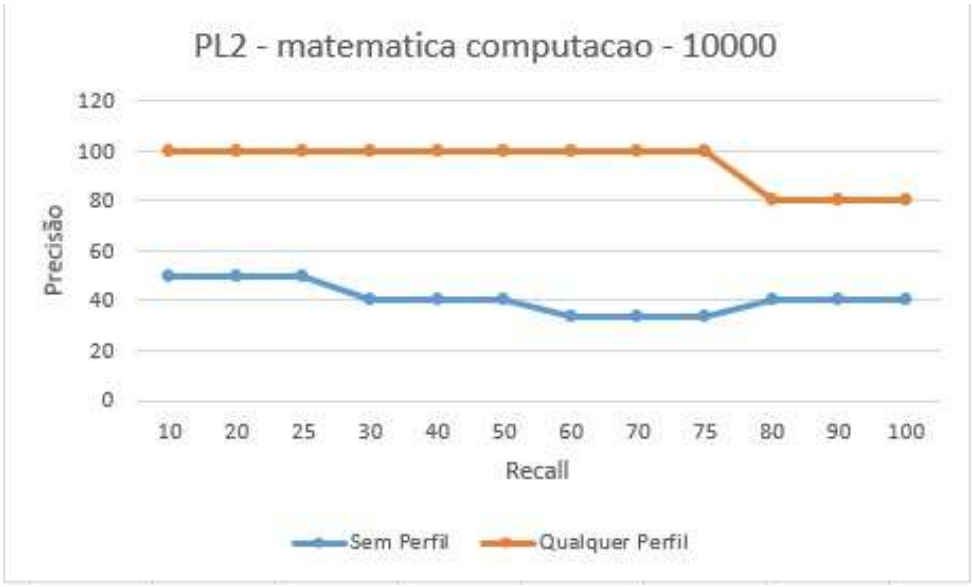


Figura 40 – Consulta "matematica computacao" no Algoritmo PL2 escala de peso 10000

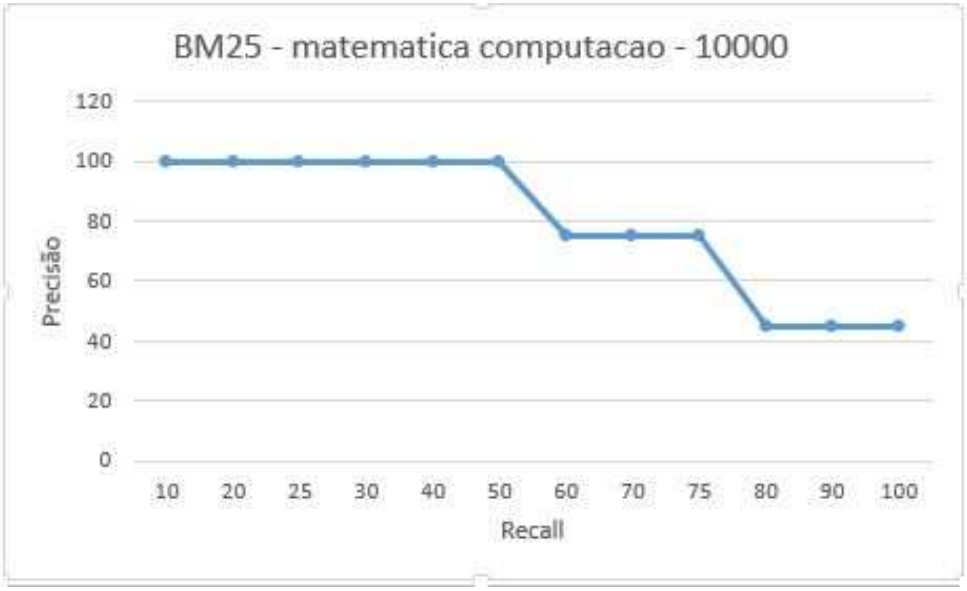


Figura 41 – Consulta “matematica computacao” no Algoritmo BM25 escala de peso 10000

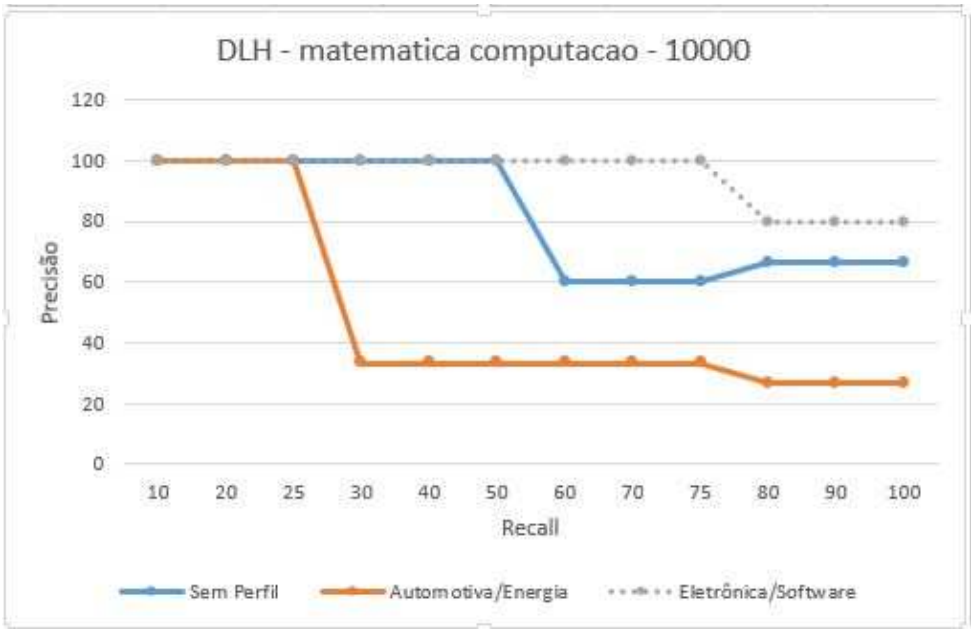


Figura 42 – Consulta “matematica computacao” no Algoritmo DLH escala de peso 10000

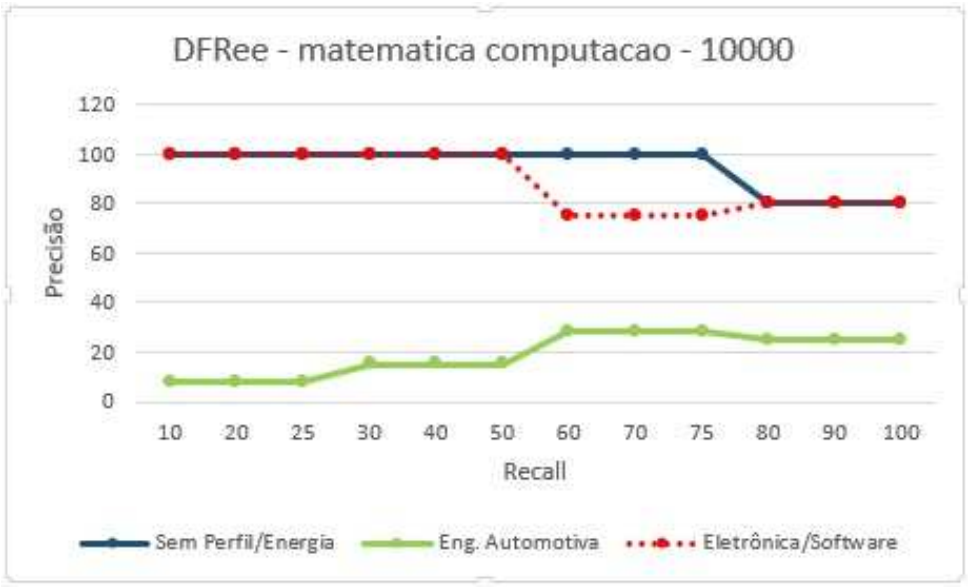


Figura 43 – Consulta "matematica computacao" no Algoritmo DFRee escala de peso 10000

### A.2.2 Consulta física química


Figura 44 – Consulta "fisica quimica" no Algoritmo DFI0 escala de peso 10000

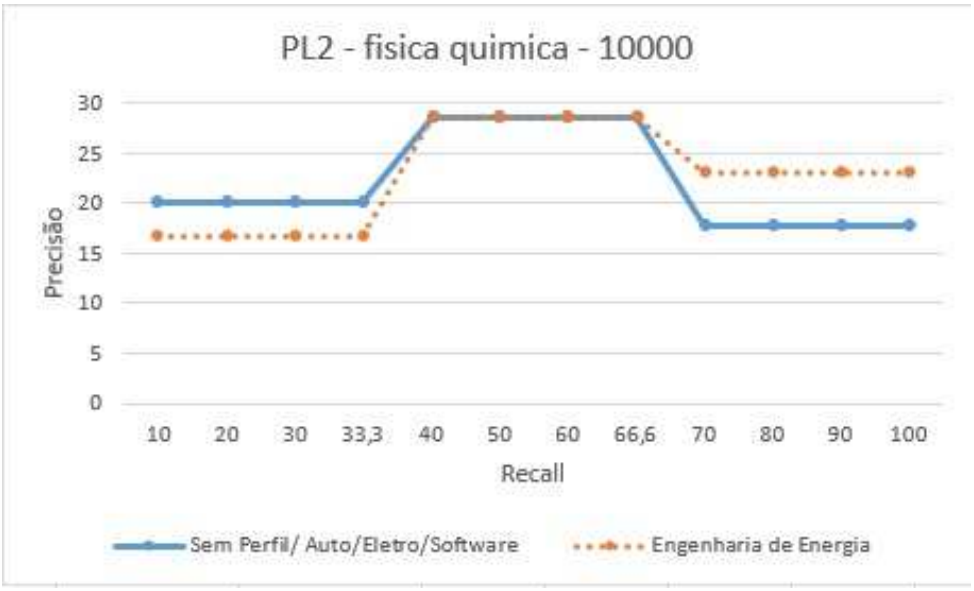


Figura 45 – Consulta "fisica quimica" no Algoritmo PL2 escala de peso 10000


Figura 46 – Consulta "fisica quimica" no Algoritmo BM25 escala de peso 10000

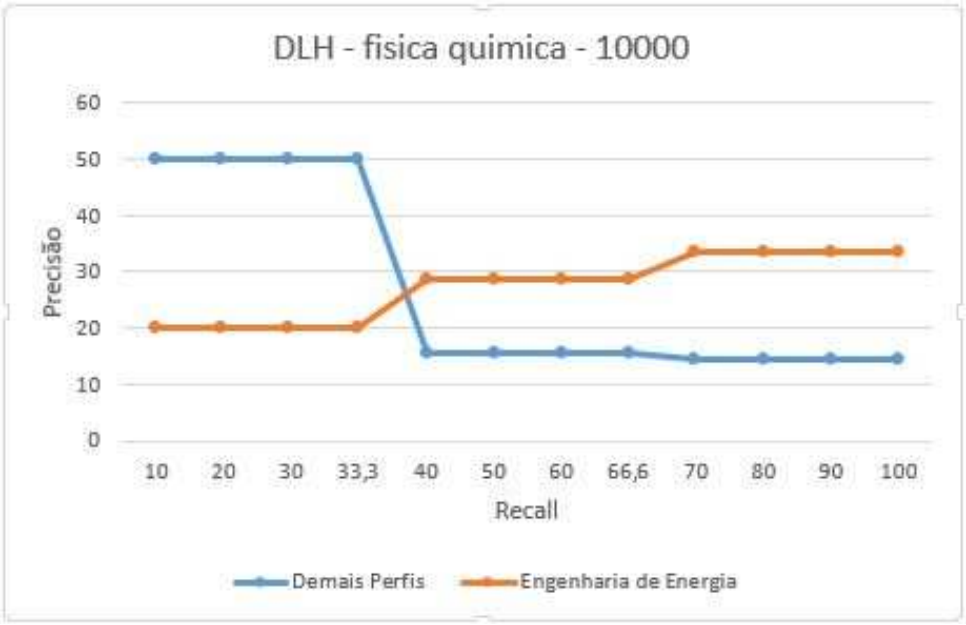


Figura 47 – Consulta "fisica quimica" no Algoritmo DLH escala de peso 10000

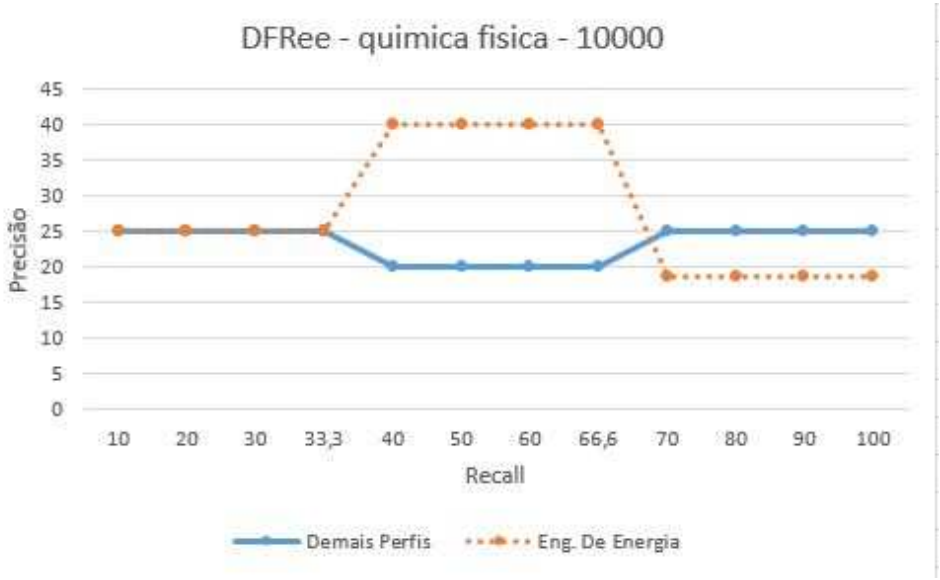


Figura 48 – Consulta "fisica quimica" no Algoritmo DFRee escala de peso 10000

### A.2.3 Consulta matemática eletrônica

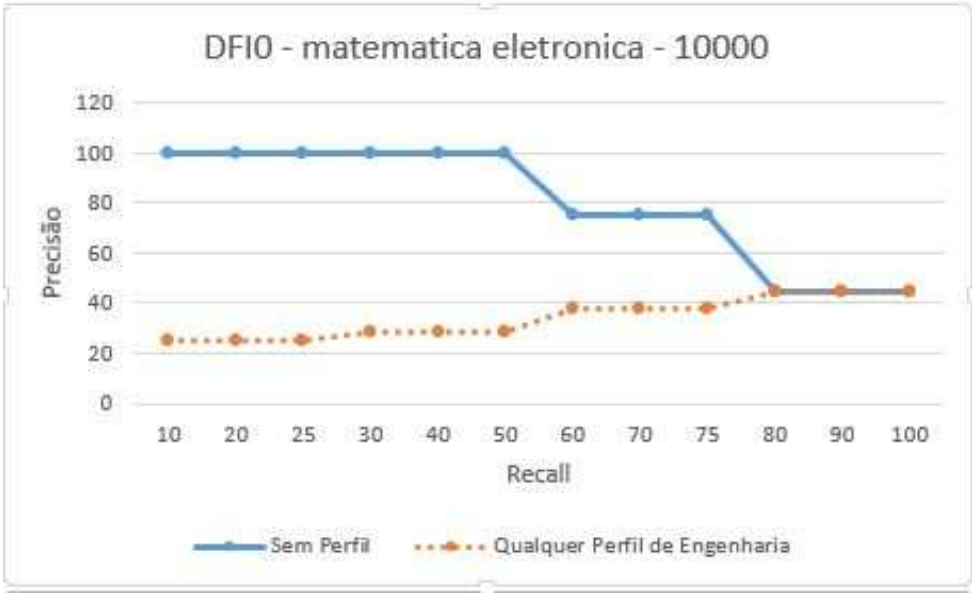


Figura 49 – Consulta "matematica eletronica" no Algoritmo DFI0 escala de peso 10000

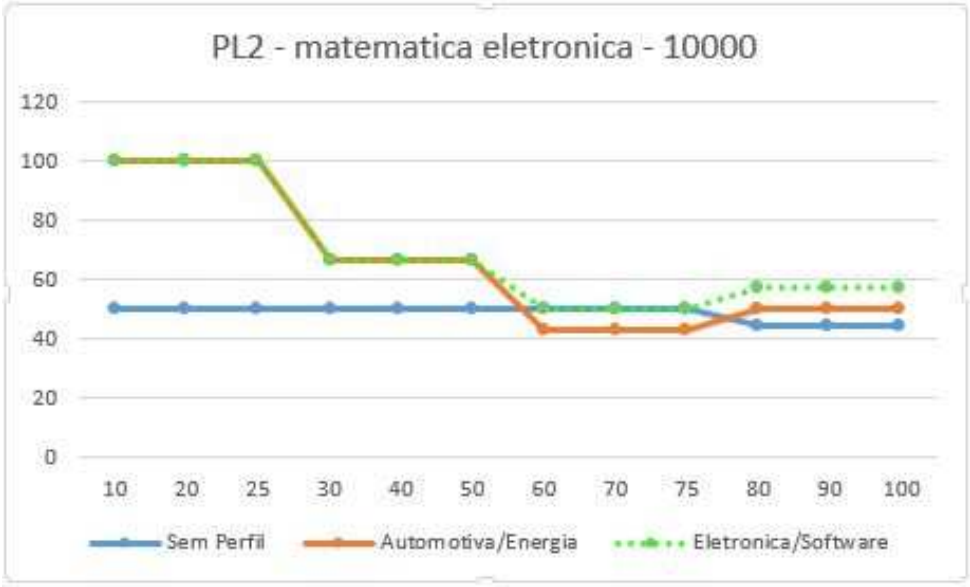


Figura 50 – Consulta "matematica eletronica" no Algoritmo PL2 escala de peso 10000

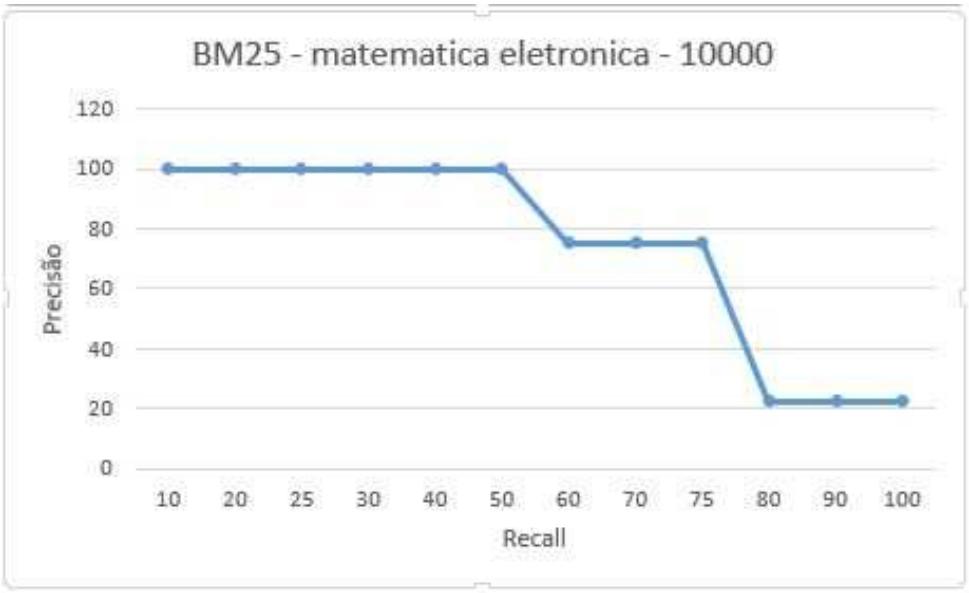


Figura 51 – Consulta “matematica eletronica” no Algoritmo BM25 escala de peso 10000

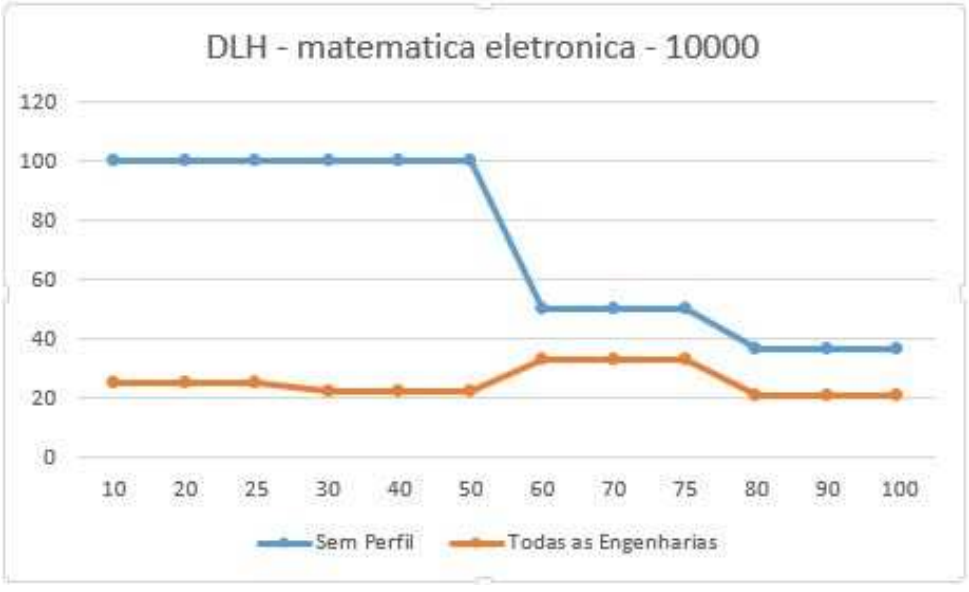


Figura 52 – Consulta “matematica eletronica” no Algoritmo DLH escala de peso 10000

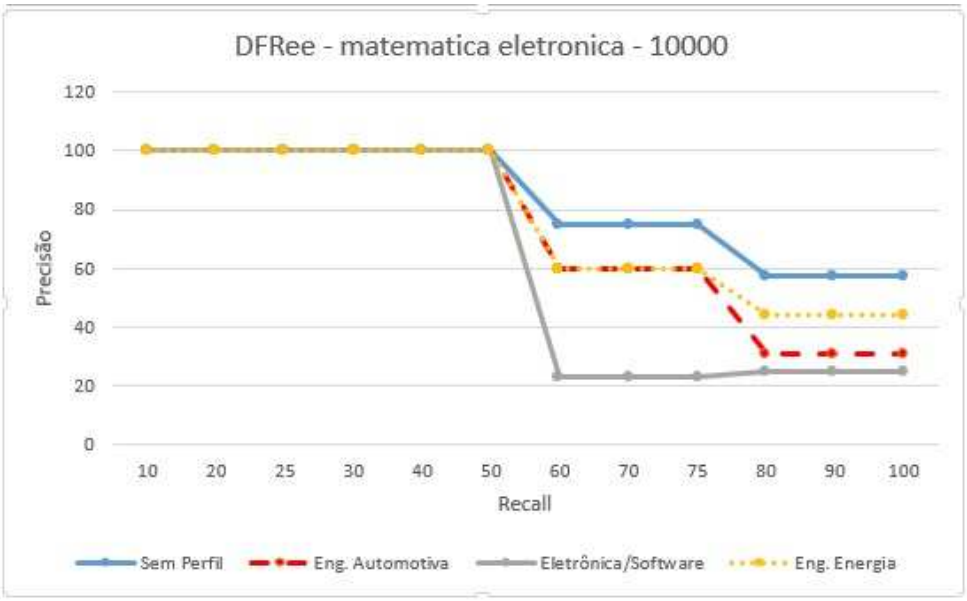


Figura 53 – Consulta “matematica eletronica” no Algoritmo DFRee escala de peso 10000

# APÊNDICE B – Segundo Apêndice

Aqui estão todas as tabelas de precisão x *recall* gerados pelas consultas realizadas, incluindo as tabelas já colocadas no texto. Nas consultas de escala dez mil, só serão inseridas as tabelas que diferenciam entre uma escala e outra.

## B.1 Escala 1000 para os Pesos

### B.1.1 Consulta matemática computação

| DFI0 | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 100 | 16,67 | 100 | Precisão igual a Eng. Eletrônica | |
| 20 | 100 | 16,67 | 100 | | |
| 25 | 100 | 16,67 | 100 | | |
| 30 | 33,33 | 22,22 | 33,33 | | |
| 40 | 33,33 | 22,22 | 33,33 | | |
| 50 | 33,33 | 22,22 | 33,33 | | |
| 60 | 33,33 | 20 | 33,33 | | |
| 70 | 33,33 | 20 | 33,33 | | |
| 75 | 36,36 | 20 | 33,33 | | |
| 80 | 36,36 | 25 | 26,27 | | |
| 90 | 36,36 | 25 | 26,27 | | |
| 100 | 36,36 | 25 | 26,27 | | |
| MÉDIA | 51,01 | 20,97 | 48,33 | | |

| PL2 | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 50 | 100 | Precisão igual para todas as Engenharias | | |
| 20 | 50 | 100 | | | |
| 25 | 50 | 100 | | | |
| 30 | 40 | 100 | | | |
| 40 | 40 | 100 | | | |
| 50 | 40 | 100 | | | |
| 60 | 33,33 | 100 | | | |
| 70 | 33,33 | 100 | | | |
| 75 | 36,36 | 100 | | | |
| 80 | 40 | 80 | | | |
| 90 | 40 | 80 | | | |
| 100 | 40 | 80 | | | |
| MÉDIA | 40,83 | 95 | | | |

| BM25 | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 100 | Precisão igual para todos os perfis | | | |
| 20 | 100 | | | | |
| 25 | 100 | | | | |
| 30 | 100 | | | | |
| 40 | 100 | | | | |
| 50 | 100 | | | | |
| 60 | 75 | | | | |
| 70 | 75 | | | | |
| 75 | 75 | | | | |
| 80 | 44,44 | | | | |
| 90 | 44,44 | | | | |
| 100 | 44,44 | | | | |
| MÉDIA | 79,86 | | | | |

| DLH | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 100 | 100 | 100 | Precisão igual a Eng. Automotiva | Precisão igual a Eng. Eletrônica |
| 20 | 100 | 100 | 100 | | |
| 25 | 100 | 100 | 100 | | |
| 30 | 100 | 33,33 | 100 | | |
| 40 | 100 | 33,33 | 100 | | |
| 50 | 100 | 33,33 | 100 | | |
| 60 | 60 | 33,33 | 100 | | |
| 70 | 60 | 33,33 | 100 | | |
| 75 | 60 | 33,33 | 100 | | |
| 80 | 66,67 | 26,27 | 80 | | |
| 90 | 66,67 | 26,27 | 80 | | |
| 100 | 66,67 | 26,27 | 80 | | |
| MÉDIA | 81,67 | 48,33 | 95 | | |

| DFRee | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 100 | Precisão igual para todos os perfis | | | |
| 20 | 100 | | | | |
| 25 | 100 | | | | |
| 30 | 100 | | | | |
| 40 | 100 | | | | |
| 50 | 100 | | | | |
| 60 | 100 | | | | |
| 70 | 100 | | | | |
| 75 | 100 | | | | |
| 80 | 80 | | | | |
| 90 | 80 | | | | |
| 100 | 80 | | | | |
| MÉDIA | 95 | | | | |

### B.1.2 Consulta física química

| DFI0 | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 20 | Precisão igual para todos os perfis | | | |
| 20 | 20 | | | | |
| 20 | 20 | | | | |
| 33,33 | 20 | | | | |
| 40 | 28,57 | | | | |
| 50 | 28,57 | | | | |
| 60 | 28,57 | | | | |
| 66,66 | 28,57 | | | | |
| 70 | 28,57 | | | | |
| 80 | 17,65 | | | | |
| 90 | 17,65 | | | | |
| 100 | 17,65 | | | | |
| MÉDIA | 22,07 | | | | |

| PL2 | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 20 | Precisão igual para todos os perfis | | | |
| 20 | 20 | | | | |
| 20 | 20 | | | | |
| 33,33 | 20 | | | | |
| 40 | 28,57 | | | | |
| 50 | 28,57 | | | | |
| 60 | 28,57 | | | | |
| 66,66 | 28,57 | | | | |
| 70 | 28,57 | | | | |
| 80 | 17,65 | | | | |
| 90 | 17,65 | | | | |
| 100 | 17,65 | | | | |
| MÉDIA | 22,07 | | | | |

| BM25 | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 100 | Precisão igual para todos os perfis | | | |
| 20 | 100 | | | | |
| 30 | 100 | | | | |
| 33,33 | 100 | | | | |
| 40 | 22,22 | | | | |
| 50 | 22,22 | | | | |
| 60 | 22,22 | | | | |
| 66,66 | 22,22 | | | | |
| 70 | 27,27 | | | | |
| 80 | 27,27 | | | | |
| 90 | 27,27 | | | | |
| 100 | 27,27 | | | | |
| MÉDIA | 49,83 | | | | |

| DLH | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 50 | Precisão igual para todos os perfis | | | |
| 20 | 50 | | | | |
| 30 | 50 | | | | |
| 33,33 | 50 | | | | |
| 40 | 15,38 | | | | |
| 50 | 15,38 | | | | |
| 60 | 15,38 | | | | |
| 66,66 | 15,38 | | | | |
| 70 | 14,29 | | | | |
| 80 | 14,29 | | | | |
| 90 | 14,29 | | | | |
| 100 | 14,29 | | | | |
| MÉDIA | 26,57 | | | | |

| DFree | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 25 | 25 | Precisão igual para todoas as engenharias | | |
| 20 | 25 | 25 | | | |
| 30 | 25 | 25 | | | |
| 33,33 | 25 | 25 | | | |
| 40 | 20 | 40 | | | |
| 50 | 20 | 40 | | | |
| 60 | 20 | 40 | | | |
| 66,66 | 20 | 40 | | | |
| 70 | 25 | 18,75 | | | |
| 80 | 25 | 18,75 | | | |
| 90 | 25 | 18,75 | | | |
| 100 | 25 | 18,75 | | | |
| MÉDIA | 23,33 | 27,91 | | | |

### B.1.3 Consulta matemática eletrônica

| DFI0 | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 100 | 50 | Precisão igual a Eng. de Energia | | |
| 20 | 100 | 50 | | | |
| 25 | 100 | 50 | | | |
| 30 | 100 | 50 | | | |
| 40 | 100 | 50 | | | |
| 50 | 100 | 50 | | | |
| 60 | 75 | 42,86 | | | |
| 70 | 75 | 42,86 | | | |
| 75 | 75 | 42,86 | | | |
| 80 | 44,4 | 44,44 | | | |
| 90 | 44,44 | 44,44 | | | |
| 100 | 44,44 | 44,44 | | | |
| MÉDIA | 79,86 | 46,83 | | | |

| PL2 | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 50 | 100 | 100 | 100 | 100 |
| 20 | 50 | 100 | 100 | 100 | 100 |
| 25 | 50 | 100 | 100 | 100 | 100 |
| 30 | 50 | 66,67 | 20 | 66,67 | 100 |
| 40 | 50 | 66,67 | 20 | 66,67 | 100 |
| 50 | 50 | 66,67 | 20 | 66,67 | 100 |
| 60 | 50 | 75 | 23,08 | 50 | 100 |
| 70 | 50 | 75 | 23,08 | 50 | 100 |
| 75 | 50 | 75 | 23,08 | 50 | 100 |
| 80 | 44,4 | 57,14 | 0 | 57,14 | 57,14 |
| 90 | 44,44 | 57,14 | 0 | 57,14 | 57,14 |
| 100 | 44,44 | 57,14 | 0 | 57,14 | 57,14 |
| MÉDIA | 48,61 | 74,70 | 35,77 | 68,42 | 89,29 |

| BM25 | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 100 | Precisão igual para todos os perfis | | | |
| 20 | 100 | | | | |
| 35 | 100 | | | | |
| 30 | 100 | | | | |
| 40 | 100 | | | | |
| 50 | 100 | | | | |
| 60 | 75 | | | | |
| 70 | 75 | | | | |
| 75 | 75 | | | | |
| 80 | 22,22 | | | | |
| 90 | 22,22 | | | | |
| 100 | 22,22 | | | | |
| MÉDIA | 74,30 | | | | |

| DLH | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 100 | 25 | 100 | 25 | Precisão igual a Eng. Eletrônica |
| 20 | 100 | 25 | 100 | 25 | |
| 25 | 100 | 25 | 100 | 25 | |
| 30 | 100 | 13,33 | 33,33 | 22,22 | |
| 40 | 100 | 13,33 | 33,33 | 22,22 | |
| 50 | 100 | 13,33 | 33,33 | 22,22 | |
| 60 | 75 | 16,67 | 33,33 | 16,67 | |
| 70 | 75 | 16,67 | 33,33 | 16,67 | |
| 75 | 75 | 16,67 | 33,33 | 16,67 | |
| 80 | 22,22 | 21,05 | 26,27 | 21,05 | |
| 90 | 22,22 | 21,05 | 26,27 | 21,05 | |
| 100 | 22,22 | 21,05 | 26,27 | 21,05 | |
| MÉDIA | 74,30 | 19,01 | 74,70 | 21,24 | |

| DLH | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 100 | 100 | 100 | 100 | Precisão igual a Eng. Eletrônica |
| 20 | 100 | 100 | 100 | 100 | |
| 25 | 100 | 100 | 100 | 100 | |
| 30 | 100 | 100 | 100 | 100 | |
| 40 | 100 | 100 | 100 | 100 | |
| 50 | 100 | 100 | 100 | 100 | |
| 60 | 75 | 60 | 23,07 | 23,08 | |
| 70 | 75 | 60 | 23,07 | 23,08 | |
| 75 | 75 | 60 | 23,07 | 23,08 | |
| 80 | 57,14 | 30,77 | 25 | 25 | |
| 90 | 57,14 | 30,77 | 25 | 25 | |
| 100 | 57,14 | 30,77 | 25 | 25 | |
| MÉDIA | 83,04 | 72,69 | 62,02 | 76,11 | |

## B.2 Escala 10000 para os pesos

### B.2.1 Consulta matemática computação

| DFree | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 100 | 8,33 | 100 | Precisão igual a Sem Perfil | Precisão igual a Eng. Eletrônica |
| 20 | 100 | 8,33 | 100 | | |
| 25 | 100 | 8,33 | 100 | | |
| 30 | 100 | 15,39 | 100 | | |
| 40 | 100 | 15,39 | 100 | | |
| 50 | 100 | 15,39 | 100 | | |
| 60 | 100 | 28,57 | 75 | | |
| 70 | 100 | 28,57 | 75 | | |
| 75 | 100 | 28,57 | 75 | | |
| 80 | 80 | 25 | 80 | | |
| 90 | 80 | 25 | 80 | | |
| 100 | 80 | 25 | 80 | | |
| MÉDIA | 95 | 19,32 | 88,75 | | |

### B.2.2 Consulta física química

| DFI0 | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 100 | Precisão igual para todos os perfis | | | |
| 20 | 100 | | | | |
| 20 | 100 | | | | |
| 33,33 | 100 | | | | |
| 40 | 18,18 | | | | |
| 50 | 18,18 | | | | |
| 60 | 18,18 | | | | |
| 66,66 | 18,18 | | | | |
| 70 | 25 | | | | |
| 80 | 25 | | | | |
| 90 | 25 | | | | |
| 100 | 25 | | | | |
| MÉDIA | 47,72 | | | | |

| DFree | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 20 | 16,67 | Precisão igual para todoas as engenharias | | |
| 20 | 20 | 16,67 | | | |
| 30 | 20 | 16,67 | | | |
| 33,33 | 20 | 16,67 | | | |
| 40 | 28,57 | 28,57 | | | |
| 50 | 28,57 | 28,57 | | | |
| 60 | 28,57 | 28,57 | | | |
| 66,66 | 28,57 | 28,57 | | | |
| 70 | 17,65 | 16,08 | | | |
| 80 | 17,65 | 16,08 | | | |
| 90 | 17,65 | 16,08 | | | |
| 100 | 17,65 | 16,08 | | | |
| MÉDIA | 22,07 | 22,77 | | | |

| DLH | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 50 | 20 | Precisão igual para todos os perfis | | |
| 20 | 50 | 20 | | | |
| 30 | 50 | 20 | | | |
| 33,33 | 50 | 20 | | | |
| 40 | 15,38 | 28,57 | | | |
| 50 | 15,38 | 28,57 | | | |
| 60 | 15,38 | 28,57 | | | |
| 66,66 | 15,38 | 28,57 | | | |
| 70 | 14,29 | 33,33 | | | |
| 80 | 14,29 | 33,33 | | | |
| 90 | 14,29 | 33,33 | | | |
| 100 | 14,29 | 33,33 | | | |
| MÉDIA | 26,57 | 27,30 | | | |

### B.2.3 Consulta matemática eletrônica

| DFI0 | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 100 | 25 | Precisão igual a Eng. de Energia | | |
| 20 | 100 | 25 | | | |
| 25 | 100 | 25 | | | |
| 30 | 100 | 28,57 | | | |
| 40 | 100 | 28,57 | | | |
| 50 | 100 | 28,57 | | | |
| 60 | 75 | 37,5 | | | |
| 70 | 75 | 37,5 | | | |
| 75 | 75 | 37,5 | | | |
| 80 | 44,4 | 44,44 | | | |
| 90 | 44,44 | 44,44 | | | |
| 100 | 44,44 | 44,44 | | | |
| MÉDIA | 79,86 | 33,88 | | | |

| PL2 | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 50 | 100 | 100 | Precisão igual a Eng. Automotiva | Precisão igual a Eng. Eletrônica |
| 20 | 50 | 100 | 100 | | |
| 25 | 50 | 100 | 100 | | |
| 30 | 50 | 66,67 | 66,67 | | |
| 40 | 50 | 66,67 | 66,67 | | |
| 50 | 50 | 66,67 | 66,67 | | |
| 60 | 50 | 42,86 | 50 | | |
| 70 | 50 | 42,86 | 50 | | |
| 75 | 50 | 42,86 | 50 | | |
| 80 | 44,44 | 50 | 57,14 | | |
| 90 | 44,44 | 50 | 57,14 | | |
| 100 | 44,44 | 50 | 57,14 | | |
| MÉDIA | 48,61 | 64,88 | 68,45 | | |

| PL2 | | | | | |
|---|---|---|---|---|---|
| Sem Perfil | | Eng. Automotiva | Eng. Eletrônica | Eng. de Energia | Eng. de Software |
| Recall | Precisão | Precisão | Precisão | Precisão | Precisão |
| 10 | 100 | 25 | Precisão igual para todas as Engenharias | | |
| 20 | 100 | 25 | | | |
| 25 | 100 | 25 | | | |
| 30 | 100 | 22,22 | | | |
| 40 | 100 | 22,22 | | | |
| 50 | 100 | 22,22 | | | |
| 60 | 50 | 33,33 | | | |
| 70 | 50 | 33,33 | | | |
| 75 | 50 | 33,33 | | | |
| 80 | 36,36 | 21,05 | | | |
| 90 | 36,36 | 21,05 | | | |
| 100 | 36,36 | 21,05 | | | |
| MÉDIA | 71,59 | 25,40 | | | |

# APÊNDICE C – Terceiro Apêndice

Aqui estão todos os gráficos realizadas através das tabelas de precisão x *recall*, incluindo as imagens já colocadas no texto. As consultas geraram 150 figuras, então colocadas apenas 10 figuras como exemplo das consultas realizadas

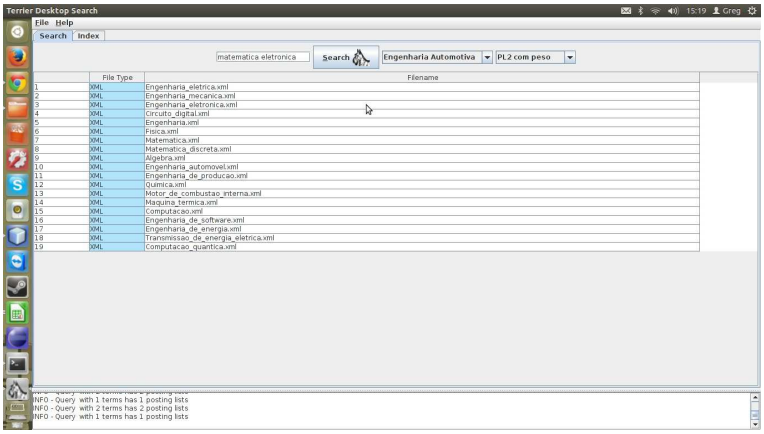


Figura 54 – Consulta “matematica eletronica” no Algoritmo PL2 no perfil Eng. Automotiva

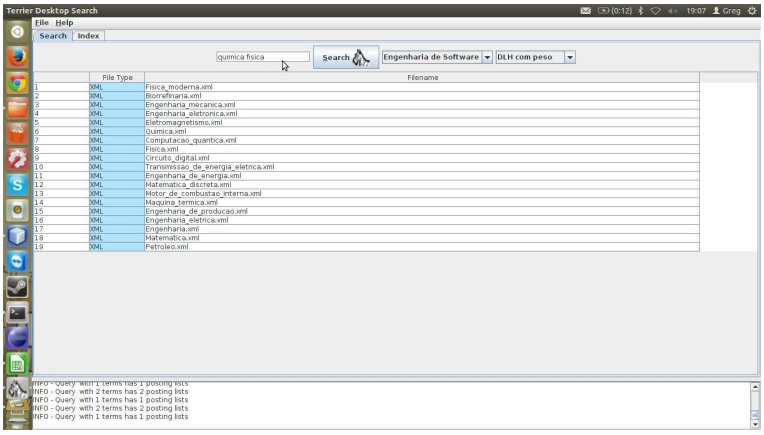


Figura 55 – Consulta “quimica fisica” no Algoritmo DLH no perfil Eng. de Software