Recuperação de Informação - Máquinas de Busca na Web Processamento de Consultas

Jordan Silva[1]

[1] Instituto de Ciências Exatas
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

jordan@dcc.ufmg.br

1. Objetivo

Este trabalho tem como objetivo a implementação de um sistema de programas para responder eficientemente e eficazmente consultas realizadas sobre o índice gerado a partir dos programas do Trabalho Prático 1.

A implementação deste sistema foi realizada utilizando a linguagem *C++11*; e como entrada de dados, foi utilizada uma coleção de documentos cedida pelo *LATIN* (Laboratório para Tratamento da Informação)[1].

2. Introdução

Nesta série de trabalhos, podemos visualizar as dificuldades e alguns cuidados à serem mitigados na criação de uma máquina de busca. No primeiro trabalho dessa série foi realizada a construção e tratamento dos documentos, dentro de uma coleção limitada; por fim disponibilizando uma forma simples de recuperar esses documentos, através de consultas *booleanas*. O modelo *booleano* de consulta, implementado no primeiro trabalho, não é um bom modelo para ser utilizado no mundo real, devido a dificuldade na manipulação e criação das consultas, como também o aprendizado e utilização pelos usuários. Assim, afim de contornar esses problemas e obter um melhor resultado na recuperação dos documentos, nessa etapa do trabalho foram implementado os modelos: *Espaço Vetorial*, *PageRank*, e *Anchor Text*, tal como uma combinação entre eles.[Baeza-Yates and Ribeiro-Neto 1999]

O trabalho teve como objetivo a melhoria no processamento das consultas através da implementações de modelos de matemáticos. Para a recuperação os documentos e o processamento das consultas de forma eficiente, fez-se necessário realizar alterações e melhorias na máquina de busca gerada pelo primeiro trabalho dessa série. Foram realizadas alterações na geração do *índice invertido*, como na geração de um novo índice para armazenamento de informações de urls referênciadas pelo documento, tal como a geração do ranqueamento da página.

O trabalho será organizado da seguinte maneira. Na próxima seção será tratada o tema de Processador de Consultas, e as técnicas utilizadas como base deste trabalho. A Seção 4, será apresentada a arquitetura do projeto e como este foi construído. A Seção 5, ensinará os procedimentos para a execução do sistema. A Seção 6, será abordada as características da coleção de dados utilizada como também a análise e os resultados obtidos. Ao final será apresentada a conclusão do trabalho.

[1] http://www.latin.dcc.ufmg.br/

3. Processador de Consultas

Os processadores de consultas possuem várias etapas à serem realizadas que serão brevemente descritas abaixo. Esse módulo é a responsável por disponibilizar uma interface para a realização de consultas pelo usuário, tal como o tratamento dessas consultas, avaliação da consulta com os documentos relevantes, e formatação desses resultados para serem retornados ao usuário.

Essa é uma funcionalidade crítica em uma máquina de busca, pois ela ocorre *online*, onde o usuário informa uma consulta e espera recuperar os documentos relevantes para a consulta realizada. Assim, quanto mais etapas e processamentos forem realizados no processador de consultas, mais tempo e processamento ela custará, impactando no tempo de resultado para a pesquisa. Por este motivo, a implementação e os algoritmos a serem utilizados no processador de consulta devem ser cuidadosamente ponderados, deve-se manter um equilíbrio entre a qualidade dos resultados - tempo de espera, para que tenha um menor impacto possível na usabilidade do usuário.

3.1. Tokenização

Assim que um usuário insere uma consulta, o motor de busca deve tokenizar o fluxo de consulta, ou seja, dividi-la em segmentos compreensíveis. Normalmente, um token é definido como uma sequência de caracteres alfa-numérico que ocorre entre o espaço em branco e ou sinais de pontuação. Os caracteres utilizados nesse trabalho para separação dos tokens foram: vírgula, ponto, formatações (e.g., \t, \v, \f, \b), quebras de linhas (e.g., \r, \n), barras (e.g., \, /).

3.2. Análise da Consulta

A análise da consulta ou *parsing* como ele é chamado, deve ser realizado afim de analisar a consulta em termos e operadores da consulta. Assim, desde que a máquina de busca permita com que os usuários utilizem operadores especiais em suas consultas, como os booleanos, adjacentes, ou operadores de proximidade, esses devem ser analisados. Esses operadores podem aparecer sob forma de pontuação reservada (e.g, aspas) ou termos reservados em um formato específico (e.g., AND, OR). No caso de sistemas de linguagem naturais, o processador de consultas irá reconhecer os operadores implicitamente na linguagem utilizada, não importando como esses operadores possam ser expressados (e.g., preposições, conjunções, ordenação).

Nesta etapa, a máquina de busca já pode realizar a buscar pelos termos da consulta no indíce invertido. Este é o ponto no qual a maior parte dos motores de busca disponíveis tentam recuperar os documentos do índice, e também que será realizado nesse trabalho.

3.3. Stop Words e Stemming

Algumas máquinas de busca vão além das etapas anteriores e implementação a remoção de *stop words* e *stemming* dos termos na consulta. Essa etapa pode ser realizada para a melhoria na interpretação da consulta, removendo termos que não possuem contribuição para a consulta (e.g., artigo), como reduzindo termos flexionados ou substituindo por sinônimos (e.g., animais - animal, feliz - alegre).

3.4. Criando a Consulta

Como cada máquina de busca criará uma representação da consulta depende de como o sistema irá realizar a combinação dos termos com os documentos. Se for utilizada uma combinação estatística para a consulta, então esta deve deve coincidir com as representações estatísticas dos documentos no índice. Boas consultas devem conter muitos sinônimos e outros termos, com o intuito de criar uam respresentação completa da consulta. Se o motor de busca realizar combinação booleana, como foi implementado no primeiro trabalho prático, então o sistema deverá criar uma representação contendo um conjunto de termos ligados por **AND**, **OR**, ou **NOT**.

Um sistema de linguagem natural (NLP) irá reconhecer termos individuais, frases e entidades nomeadas. Se ele utilizar qualquer lógica booleana, ele também irá reconhecer os operadores lógicos mencionados anteriormente, e assim, criar uma representação contendo os conjuntos de termos ligados pelos operadores lógicos.

3.5. Expansão de Consulta

Como os usuários costumam informar somente uma sequência de termos para uma consulta, é altamente provável que as informações que eles precisam possam ter sido expressadas no documento utilizando sinônimos, ao invés de os termos da consulta exatas. Portanto, os sistemas mais sofisticados podem expandir a consulta em todos os possíveis termos e sinônimos e talvez até mesmo alterando os termos da consulta, adicionando novos termos (e.g., vertebrados - mamíferos, aves, répteis).

3.6. Ponderação dos Termos na Consulta

O passo final no processamento de consultas envolve a computação dos pesos para os termos na consulta. Algumas vezes o usuário pode controlar essa etapa, indicando qual o peso de cada termo ou simplesmente informando qual termo ou conceito é mais importante na consulta, e que este deve aparecer em cada documento recuperado para assegurar a relevância deste.

Permitir a realização dessa ponderação de pesos pelo usuário é pouco comum, e é muito difícil determinar a importância relativa dos termos nas consultas. Assim, é fortemente recomendado que esse peso seja computado ao invés de informado pelo usuário, por diversas razões. Primeiramente, eles não sabem o que mais existe no banco de dados, e esses termos serão ponderados para serem comparados com o banco de dados como um todo. A maioria dos usuários procuram informações sobre um assunto desconhecido, assim, eles podem não saber a terminologia correta a ser utilizada.

Algumas máquinas de busca implementam essa computação dos pesos para os termos da consulta, mas utilizam de forma implícita, tratando o primeiro termo como maior importância, ou utilizando outras formas estatísticas para a determinação de cada peso. Após essa ponderação, as máquinas de busca utilizam dessa informação para prover a lista de documentos para o usuário.

Utilizamos nesse trabalho a ponderação $tf \times idf$, onde tf é a frequência do termo na consulta (Equação 1), e o idf é representado pelo inverso da frequência do termo nos documentos da coleção (Equação 2). [Baeza-Yates and Ribeiro-Neto 1999]

$$tf_{i,q} = \begin{Bmatrix} 1 + \log f_{i,q} & \text{if } f_{i,q} > 0 \\ 0 & \text{senão} \end{Bmatrix} \quad (1)$$

$$if d_i = \log \frac{N}{n_i} \quad (2)$$

Após esta etapa final, a consulta que foi expandida, e agora com a ponderação dos termos, eserá utilizada para retornar os documentos relevantes dentro do índice invertido.

4. Arquitetura do Projeto

Este trabalho foi desenvolvido como continuação do primeiro trabalho prático, com o intuito de possibilitar à recuperação dos documentos de uma forma mais eficiente que a implementada anteriormente, através da utilização de métodos de classificação dos documentos. Também foi disponibilizado serviços RESTful, e implementado uma interface web, para a realização de consultas de uma forma mais amigável para os usuários.

Nesta seção abordaremos a implementação do servidor web e os métodos de classificação utilizados.

4.1. Servidor Web

O desenvolvimento do servidor web foi realizado utilizando a biblioteca Simple-Web-Server [eidheim 2015], servindo como auxílio para disponibilização dos serviços RESTful, e facilitando assim a utilização e consumo de qualquer interface através da API de Busca. O servidor foi configurado para funcionar na **porta 8080**, utilizando **4 threads**.

As buscas são realizadas através do serviço **POST \search** onde deverá ser realizada uma requisição JSON,

```
curl -d '{ "query": "ana maria braga", "vms": 1, "at": 1, "pr": 1, "page": 0}'
 localhost:8080/search
```

onde o campo *query* deverá ser informado a consulta. As propriedades *vms*, *at* e *pr*, deverão ser definidos valores booleanos (i.e., 0-falso, 1-verdadeiro) para a escolha dos modelos a ser utilizado na busca, *Modelo Vetorial*, *Anchor Text*, *PageRank* respectivamente. E por fim, na propriedade *page* deverá ser informada o número da página de resultados a ser retornado. Este método retornará o resultado da busca com uma paginação a cada 10 resultados retornados.


```
{
  "page": 0,
  "totalPages": 3993,
  "result": [
    {
      "id": "17242",
      "url": "http://anamariabraga.org/",
      "title": "",
      "description": "",
      "keywords": "",
      "author": "",
      "pageRank": "4.32727e-05"
    },
    {
      "id": "18677",
      "url": "http://desciclo.pedia.ws/wiki/Ana_Maria_Braga",
      "title": "",
```

```
      "description": "",
      "keywords": "",
      "author": "",
      "pageRank": "4.32727e-05"
    }
  ]
}
```


A página Web foi desenvolvida em **html**, utilizando **jQuery**[2] e o estilo **boostrap**[3] para agilizar o desenvolvimento. Foram utilizadas outras bibliotecas externas como a **jquery.endless-scroll** [4] para auxiliar na paginação dos documentos e carregamento automático deles, assim que seja visualizado o último item da página, será carregado os documentos da próxima página. A página web (Figura 1) poderá ser acessada pelo endereço `http://localhost:8080/index.html`, assim que o servidor web for inicializado.

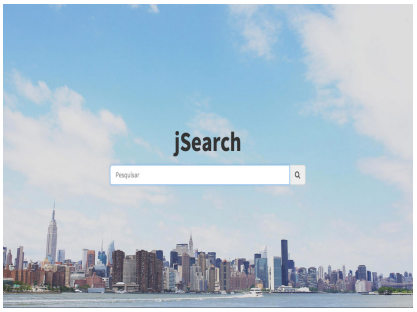

Figura 1. Página Web

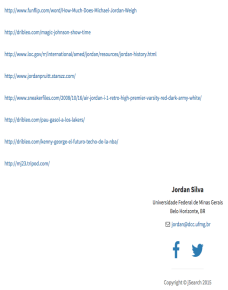

Figura 2. Realizando uma busca

4.2. Classificação

Os modelos utilizados para realizar a classificação e ranqueamento (i.e., ordenação dos documentos) será descrita neste tópico, onde os modelos à serem abordados serão o **Espaço Vetorial**, **PageRank**, **Anchor Text**, e as combinações entre eles.

4.2.1. Espaço Vetorial

O modelo vetorial proposto inicialmente por [Salton et al. 1975], reconhece que a utilização do modelo Booleano é muito limitado e propõe um arcabouço onde o casamento parcial entre uma consulta e um documento é possível. Isso é realizado pela a atribuição de pesos não binários aos termos do índice, tanto nas consultas, quanto nos documentos. Desta maneira, com os pesos agora atribuídos, será realizada o cálculo da similaridade entre cada documento armazenado no sistema e a consulta informada. A ordenação desses documentos será realizada de forma decrescente de acordo com o valor da similaridade calculada anteriormente, assim, o modelo vetorial leva em consideração documentos que casaram os termos da consulta de forma parcial. Como resultado principal, a ordenação dos documentos retornados é muito mais preciso que o conjunto de documentos retornado pelo modelo Booleano. [Baeza-Yates and Ribeiro-Neto 1999]

[2]http://jquery.com/
[3]http://getbootstrap.com/
[4]https://github.com/fredwu/jquery-endless-scroll

Esse modelo é representado através de um vetor t-dimensional onde cada valor desse vetor será dado pelo peso do termo na coleção. Os pesos $w_{i,j}$ assoaciados à um par termo-documento (k_i, d_j) serão positivos e valores reais. É assumido que todos os termos indexados são independentes e representados como vetores unitários de um espaço *t-dimensional*, no qual t é a quantidade de termos indexados no documento. As representações do documento d_j e da consulta q são vetores *t-dimensionais* dados por

$$\begin{aligned} \overrightarrow{d_j} &= (w_{1,j}, w_{2,j}, \cdots, w_{t,j}) \\ \overrightarrow{q} &= (w_{1,q}, w_{2,q}, \cdots, w_{t,q}) \end{aligned} \tag{3}$$

onde $w_{i,q}$ é o peso associado para cada termo da consulta (k_i, q), onde $w_{i,q} \geq 0$.

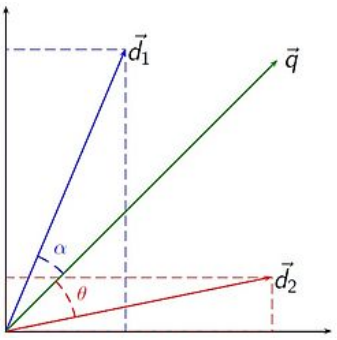

Figura 3. Representação do modelo vetorial

Assim os documentos d_j e a consulta q serão representadas como *vetores t-dimensionais*, como mostrado na Figura 4.2.1. O modelo vetorial avalia a similaridade dos documentos d_j com a consulta q como a correlação entre os vetores $\overrightarrow{d_j}$ e $\overrightarrow{q}$. Essa correlação pode ser quantificada, por exemplo, pela fórmula do cosseno (Equação 4) entre os dois vetores. Isto é,

$$sim(d_j, q) = \frac{\overrightarrow{d_j} \cdot \overrightarrow{q}}{\left|\overrightarrow{d_j}\right| \times |\overrightarrow{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}} \tag{4}$$

onde $\left|\overrightarrow{d_j}\right|$ e $|\overrightarrow{q}|$ são as normalizações do vetor dos documentos e da consulta (i.e., afim de penalizar documentos muito grandes), e $\overrightarrow{d_j} \cdot \overrightarrow{q}$ é o produto interno dos dois vetores. Podemos observar que fator de normalização $|\overrightarrow{q}|$ não afeta o ranqueamento do resultado, devido a ser o mesmo valor para todos os documentos, assim, este será removido da fórmula. Assim, o valor de normalização dos documentos será dado como $\left|\overrightarrow{d_j}\right|$, que representa o tamanho do documento (i.e., quantidade de termos do documento). Os pesos utilizados no modelo vetorial e utilizado neste trabalho é o TF-IDF, assim adotaremos a seguinte fórmula para o cálculo de cada peso,

$$\begin{aligned} w_{i,q} &= (1 + \log f_{i,q}) \times \log \frac{N}{n_i} \\ w_{i,j} &= (1 + \log f_{i,j}) \times \log \frac{N}{n_i} \end{aligned} \tag{5}$$

onde $f_{i,q}$ é a frequência do termo k_i no texto da consulta q, assim como $f_{i,j}$ será a frequência do termo dentro do documento d_j.

4.2.2. PageRank

O algoritmo PageRank foi desenvolvido pelos fundadores do Google[5], onde tem como objetivo medir a importância de uma página, avaliando a quantidade de referências (i.e., hiperligações) existem para essa página. Esta técnica segue como base que quanto mais referências existem para uma determinada página, mais importante essa página é, ou no nosso caso, relevante. O PageRank representa a probabilidade de um determinado usuário, navegando aleatoriamente pela hiperligações das páginas, alcançar uma determinada página. [Page et al. 1999]

O cálculo realizado pelo PageRank é considerado escalável, e pode ser executado em um tempo consideravelmente pequeno mesmo que a quantidade de páginas seja aumentada consideravelmente. Esse cálculo funciona de uma forma iterativa, necessitando várias iterações até que as diferenças dos valores determinados para as páginas alcancem um valor mínimo pré-determinado, que será o valor de convergência para o PageRank. A fórmula é determinada por,

$$PR^i(u) = \frac{1-d}{N} + d \sum_{v \in B_u} \frac{PR^i(v)}{L(v)} \tag{6}$$

onde o valor de uma página u, depende dos valores de PageRank de cada página v contida no conjunto B_u (i.e., cojunto de todas as páginas que referênciam u), dividido pelo número de referências $L(v)$ existentes em v. O parâmetro d funciona como um fator de amortecimento, suavizando o algoritmo e criando uma probabilidade de uma determinada página, que não possui nenhum link externo, poder navegar para qualquer página aleatoriamente dado o peso d. É normalmente atribuído o valor **0,85** para o valor de amortecimento, e este que utilizamos neste trabalho.

4.2.3. Anchor Text

A classificação utilizando este modelo é baseada na criação de um índice invertido tal como foi realizado no primeiro trabalho. O índice gerado pelo modelo *Anchor Text*, como o próprio nome diz, ele é focado na coleta e extração de informações de referênciamento entre páginas, extraindo assim os textos âncora das hiperligações da página. Este novo índice gerado pelo nosso sistema realizou tratamentos e validações nesses novos termos e assim, esses termos seriam referênciados para o documento, tal qual foi realizado no primeiro trabalho prático, diferenciando somente o arquivo para um novo índice. O modelo *Anchor Text* por fim realizará uma busca e classificação dos documentos somente por este novo índice gerado, utilizando a mesma forma descrita no modelo de Espaço Vetorial(4.2.1) alterando somente o índice à ser utilizado.

[5] `http://www.google.com/`

4.2.4. Combinação entre os Modelos

Os modelos propostos acima podem e devem funcionar em conjunto pela máquina de busca. A partir da combinação entre os modelos, através de pesos previamente especificados, esses modelos se complementam de forma que melhoram significantemente a precisão dos resultados da pesquisa. A combinação entre os modelos se dá por,

$$score(q, d_j) = \alpha \times scoreV(q, d_j) + \gamma \times PR(d_j) + \beta \times scoreA(q, d_j) \qquad (7)$$

onde os respectivos pesos α, γ e β serão constantes e definidos durante a inicialização do da máquina de busca. Essas constantes têm como função equilibrar os pesos de cada modelo ou priorizar os modelos mais relevantes, garantindo assim uma melhor classificação dos documentos para retorno ao usuário. *Os valores dos pesos devem variar de 0 à 1, sendo que a soma final de todo pesos deverá ser 1.*

5. Executando o Sistema

A execução do sistema deverá ser realizada via linha de comando, utilizando o **C++11**. Neste trabalho fez-se necessário a divisão do arquivo binário em 3 arquivos, onde está o sistema de indexação, o sistema de busca, e o servidor web.

./indexer <directory> <index> - Este sistema é responsável pela realização da indexação dos documentos. Os parâmetros *directory* e *index* são obrigatórios, onde respectivamente representam o diretório onde o índice se encontra, e o nome do arquivo de índice. O resultado desta indexação será gerado e armazenado em uma pasta chamada *output* dentro da pasta onde o programa se encontra.

./searcher [-vsm|-at|-pr <query>] - Este sistema é responsável pela realização da consulta ao índice gerado. Para a execução desse sistema, é necessário que seja realizada a indexação à partir do primeiro sistema, e que os respectivos arquivos gerados estejam no diretório *output*, onde foram gerados pelo sistema de indexação. Os parâmetros *-vsm*, *-at*, *-pr*, são opcionais, sendo que se nenhum desses forem informados, será assumido automaticamente que o método de classificação dos documentos a ser utilizado pelo processador de consultas será o modelo vetorial espacial. Os termos da consulta serão informados pelo parâmetro *query*, onde não se faz necessário nenhum caracter especial para concatenação da consulta (e.g., ./searcher -vsm ana maria braga).

-vsm Classificação utilizando o Modelo espaço vetorial.
-at Classificação utilizando o Anchor Text.
-pr Classificação por PageRank.

./server - Este sistema é responsável pela inicialização do servidor web, onde será disponibilizado uma interface amigável para a realização das consultas. Para que esse sistema funcione perfeitamente, é necessário realizar a indexação dos documentos utilizando o sistema de indexação disponível nesse pacote. Os arquivos de índices gerados pelo sistema deverão permanecer na pasta *output*, juntamente com os três sistemas (i.e., indexer, searcher, server). Este servidor web poderá ser utilizado com qualquer sistema web através de requisições RESTful descritas nesse documento, retornando assim os documentos relevantes para a consulta.

6. Análise e Resultados

Nesta seção será analisado a implementação realizada, a coleção indexada, como também os resultados obtidos nesse trabalho. Os testes e a implementação foram realizadas utilizando uma coleção cedida pelo Laboratório para Tratamento da Informação da Universidade Federal de Minas Gerais. A coleção utilizada possui: 945.642 documentos; contendo o tamanho de 4,7GB (Tabela 1). Os experimentos foram realizados utilizando um **MacBook Pro (13-inch, Mid 2012); 2,5 GHz Intel Core i5; 16 GB 1600 MHz DDR3; OSX Yosemite 10.10.3 (14D136)**.

A indexação dos documentos fez-se necessária mais uma vez nessa etapa do trabalho, foi necessário revisar a indexação devido à alguns problemas encontrados, como também acrescentar a criação de um novo índice. O índice do *Anchor Text* foi gerado separado como pode-se notar na pasta *output*, onde ele foi criado com o nome de *ianchor.index*, juntamente com o seu vocabulário *anchor_vocabulary.terms*. Na geração deste índice foi realizado um tratamento na coleta das hiperligações, tratando o caminho do endereço de ligação, e também na coleta do texto âncora referente ao link. Essas informações foram utilizadas tanto na recuperação da informação pelo *Anchor Text* quanto pelo modelo do *PageRank*, onde é necessário realizar o cálculo utilizando as referências das páginas (i.e., hiperligações).

Descrição	Valor
Quantidade de termos indexados	5.463.169
Quantidade de termos - Anchor Text	2.363.214
Tamanho do índice - Termos	3,8GB
Tamanho do índice - Anchor Text	3,8GB
Tamanho do vocabulário - Termos	114,9MB
Tamanho do vocabulário - Anchor Text	114,9MB

Tabela 1. Informações da coleção utilizada

Como no primeiro trabalho, neste também acompanhamos o comportamento do vocabulário e o seu crescimento durante a indexação dos documentos. A Figura 4 exibe o crescimento dos vocabulários a cada 10 mil documentos indexados.

Após a realização do processo de geração do índice, conseguimos então calcular o primeiro modelo de classificação o **PageRank**. O cálculo desse modelo será realizado para todos os documentos conforme descrito na Seção (4.2.2). O PageRank nos testes utilizando uma taxa de convergência de *0.00001*, foi necessário realizar *55 iterações* até que convergisse (Figura 5). A taxa de convergência foi medida pela maior diferença do novo PageRank calculado, assim, o documento que obtivesse a maior diferença de PageRank na iteração com o valor da iteração passada irá servir como suporte para esse cálculo.

Como podemos notar na Tabela 2, ao final dos cálculos, é gerada um classificação de todos os documentos da coleção e seu ranqueamento, de acordo com os valores computados por esse modelo.

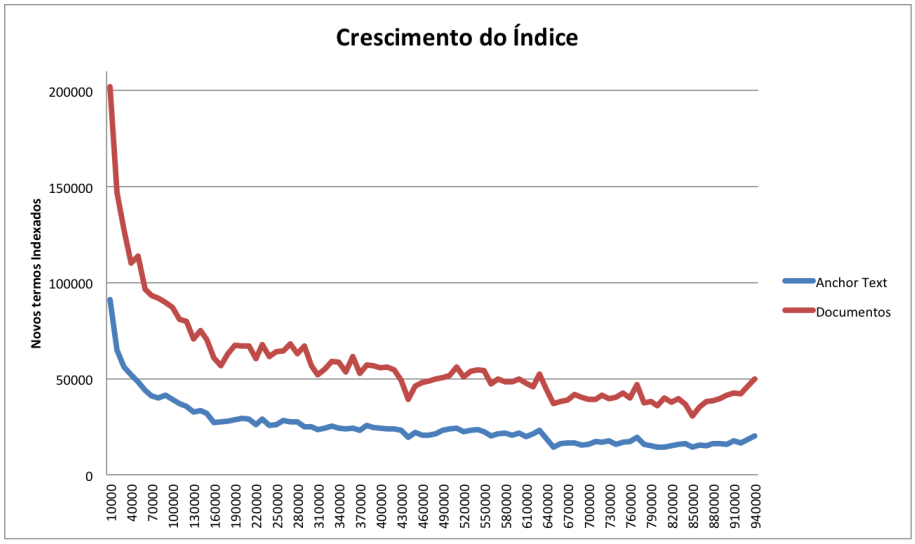

Figura 4. Crescimento do Índice

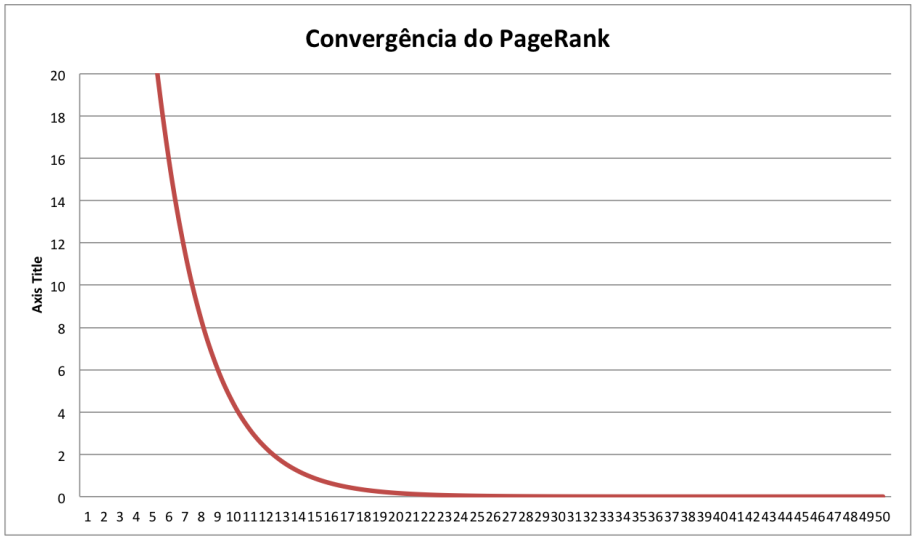

Figura 5. Convergência do PageRank

Posição	Descrição
1	http://www.blogger.com/
2	http://wordpress.com/
3	http://www.blogger.com/features
4	http://blogsofnote.blogspot.com/
5	http://www.thefreesite.com/
943852	http://acrossit.wordpress.com/tag/saas/
943853	http://blocs.gracianet.cat/category/4/5/page/15
943854	http://www.globeofblogs.com/?x=title&letter=f&page=101

Tabela 2. Top documentos mais e menos relevantes pelo PageRank

Assim que o cálculo do **PageRank** é finalizado o sistema já está pronto para ser utilizado e as consultas serem realizadas. Como avaliação, utilizamos **34 consultas**, onde possuímos um gabarito contendo as páginas mais relevantes para cada consulta, conse-

guindo assim mensurar a qualidade da máquina de busca e das classificações. Essas consultas foram identificadas em duas categorias *Navegacionais e Informacionais* (Tabela 3).

Tabela 3. Consultas

Tabela 4. Navegacionais

Baixaki	Oi
Caixa Economica Federal	Orkut
Casa e Video	Panico
Claro	Receita Federal
Detran	Record
Globo	Terra
Gmail	Uol
Google	Vivo
Hotmail	Yahoo
IG	Youtube
Mercado Livre	
Msn	

Tabela 5. Informacionais

Ana Maria Braga
Concursos
Esporte
Frases de Amor
Funk
Jogos de Meninas
Jogo Online
Mario
Naruto
Poquer
Previsao do Tempo
Rio de Janeiro

Neste trabalho realizamos os experimentos utilizando os três modelos propostos na sessão de Classificação (Sessão 4.2), como também a combinação entre eles. Desta forma, conseguimos avaliar a precisão da máquina de busca e a revocação através das métricas de Precisão $\times$ Revocação. Avaliamos primeiramente os três modelos básicos **Espaço Vetorail**, **PageRank** e **Anchor Text** para todas as **34 consultas** e avaliamos a média da precisão e revocação (Figura 6). Podemos notar que o modelo de classificação do **Anchor Text** se sai um pouco melhor inicialmente que o modelo **Espaço Vetorial**, e logo na interpolação de 10% o modelo espaço vetorial já se sai melhor que o **Anchor Text** puro. Uma curiosidade que conseguimos notar é como o modelo do **PageRank** se saiu muito ruim na precisão em comparação com os outros modelos. Acreditamos que possa ser alguma amostragem ou comportamento da base utilizada nos experimentos, assim, desfavorecendo o modelo do PageRank.

Revocação	Precisão	Run ID
0.0	0.8834	vms4-pr3-at9*
0.1	0.6757	vms4-pr3-at9*
0.2	0.509	vms5-pr1-at9*
0.3	0.2985	vms5-pr2-at10
0.4	0.2481	vms7-pr4-at4
0.5	0.1764	vms9-pr8-at2
0.6	0.1407	vms9-pr8-at1
0.7	0.1184	vms2-pr3-at9*
0.8	0.0566	vms10-pr1-at9*
0.9	0.0246	vms10-pr1-at9*
1.0	0.0049	vms10-pr1-at9*

Tabela 6. Precisão x Revocação - Melhores runs em cada revocação

Após esse experimento, realizamos um teste das combinações do modelos e seus respectivos pesos. Foi realizada uma validação massiva de testes afim de encontrar a melhor combinação de pesos para a nossa amostragem de teste, assim, foram realizadas **34 mil consultas**, sendo que testamos para cada combinação de modelo a variação de

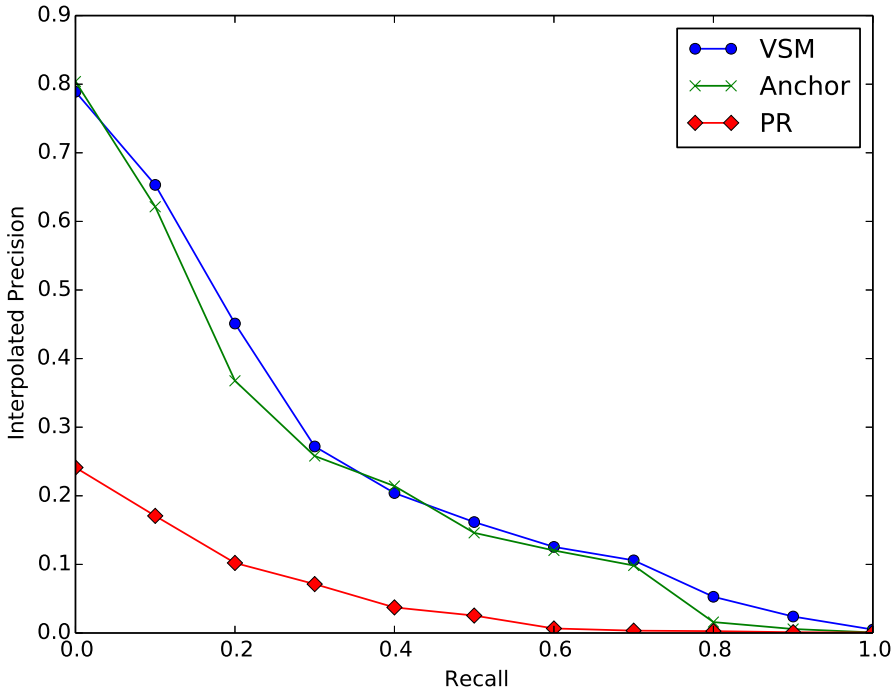

Figura 6. Precision x Recall - Todas as Consultas

cada peso entre 0.1 - 1.0, assim, como são 3 pesos distindos, a quantidade desse teste se deu por $10^3 \times N_q$, sendo que são 10 variações para o peso de cada modelo, multiplicado pela quantidade de consultas à serem realizadas. Os resultados podem ser visualizados nas Tabela 6, onde selecionamos os melhores resultados de precisão e revocação para cada run executada. Os valores contendo um asterisco em frente significa que outras runs obtiveram o mesmo resultado de precisão na respectiva taxa de revocação. Os valores escolhidos foram padronizado de tal forma que notamos que a precisão geralmente é melhor com os modelos de **Espaço Vetorial** e **Anchor Text** maiores, e o **PageRank** menor, com exceção nos pontos de revocação 0.5 e 0.6, onde os pesos do **PageRank** são maiores do que os pesos do **Anchor Text**.

Com esse experimento realizado e avaliado de acordo com o gabarito, conseguimos encontrar a melhor combinação de pesos para o modelo linear dentro da variação testada (Figura 7), assim, percebemos que elas variam muito pouco entre elas, tendo uma vantagem para as runs *vms5-pr-2-at10 e vms7-pr4-at4*. Desta forma, como os dois modelos tem valores muito parecidos para a precisão e revocação, optamos pelo *vms-7-pr4-at4* onde os pesos são mais balanceados. Para a combinação entre os diversos modelos executamos também uma normalização dos seus valores de similaridades, considerando o maior valor como 1, e normalizamos a partir deste. Dessa forma, além da combinação de pesos, os valores dos modelos também sofre uma normalização de similaridade variando de 0.0-1.0.

Realizamos um outro experimento onde constatamos que para essa coleção, a normalização pelo tamanho do documento proposto pela fórmula do Cosseno (Equação 4) e pelo Modelo Espaço Vetorial (Sessão 4.2.1) diminui a precisão e revocação. Temos essa visualização na Figura 8 onde os modelos **N-VSM, N-AT** são os normalizados pelo

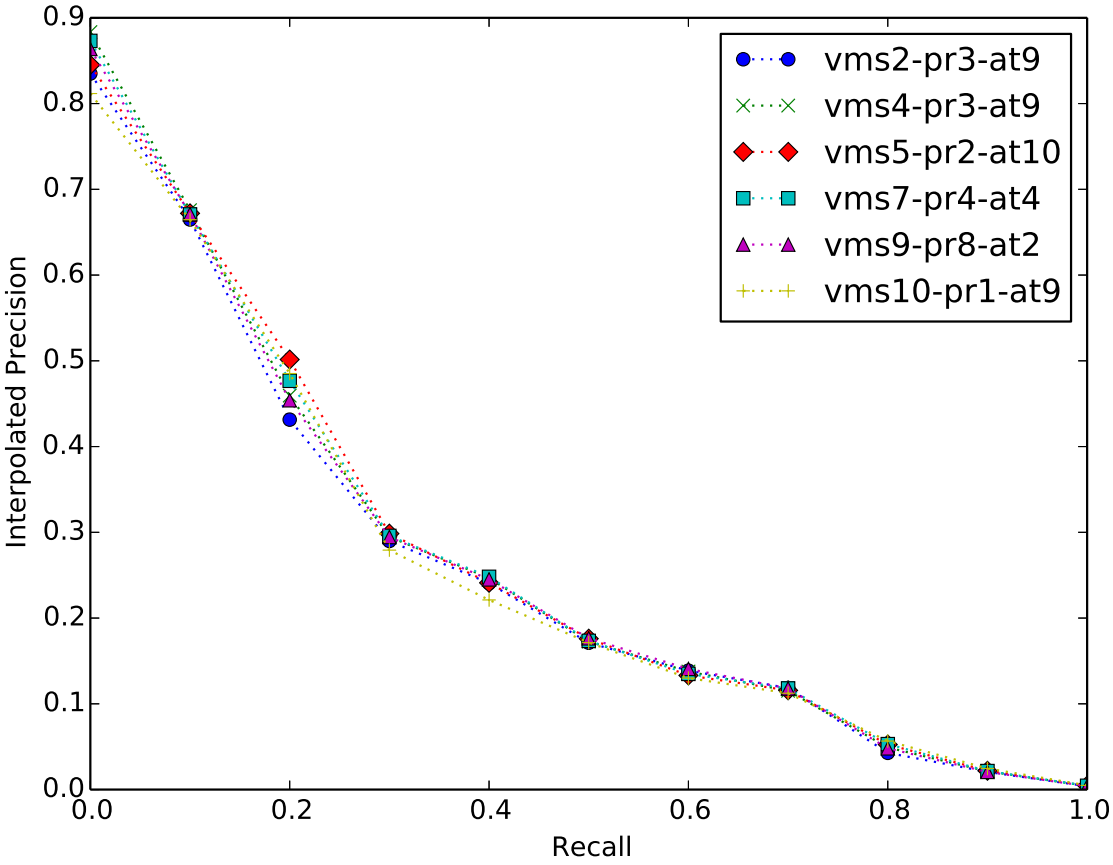

Figura 7. Combinação dos Modelos - Variação dos Pesos

tamanho do documento e os outros dois não. Assim, a normalização pelo tamanho dos documentos foi desconsiderada em todos os experimentos e removida nos modelos de classificação.

Após todos os testes os modelos utilizados nesse trabalho resultaram em uma curva de precisão e revocação apresentada na Figura 9, onde vemos que os métodos que se sobressaíram foram as combinações do modelo **Espaço Vetorial**, **Anchor Text**, e tendo um resultado similar quando combinado os três linearmente. Os parâmetros utilizados na combinação e resultados obtidos são apresentados na Tabela 7.

Descrição	Valor
Consultas Realizadas	34
Qtd. Documentos Relevantes	597
Qtd. Documents Relevantes Retornados	510
α - Peso do Modelo Espaço Vetorial	0.48
β - Peso do Modelo Anchor Text	0.26
γ - Peso do Modelo PageRank	0.26

Tabela 7. Informações dos Testes

Por fim, analisamos individualmente as consultas e os modelos propostos, onde percebemos a melhoria dos modelos em determinadas consultas. Na Figura 10 é apresentada essas diferenças para o modelo **Espaço Vetorial** combinado com o **Anchor Text**

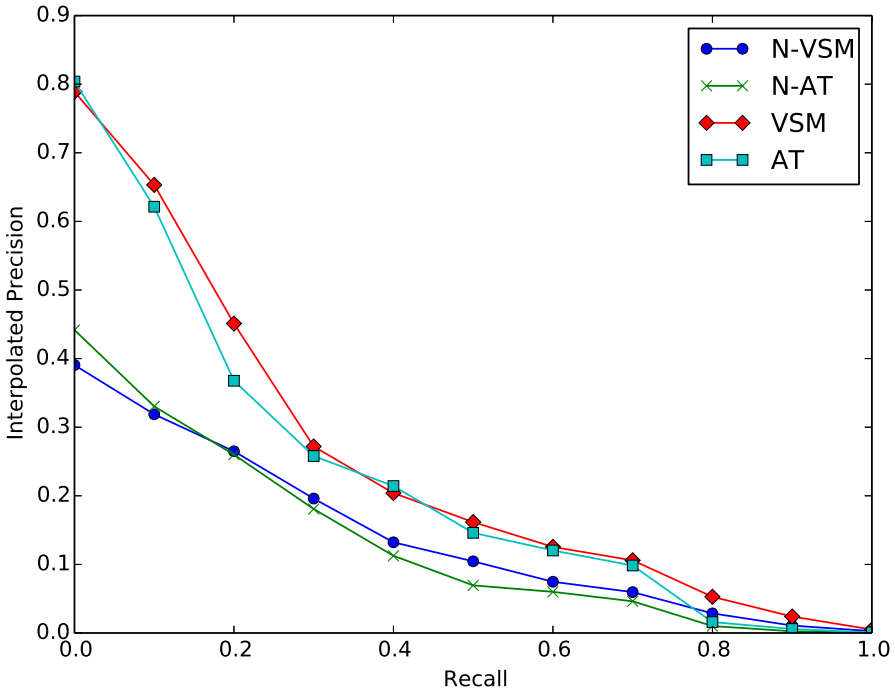

Figura 8. Precisão x Revocação - Normalizada vs Sem Normalização

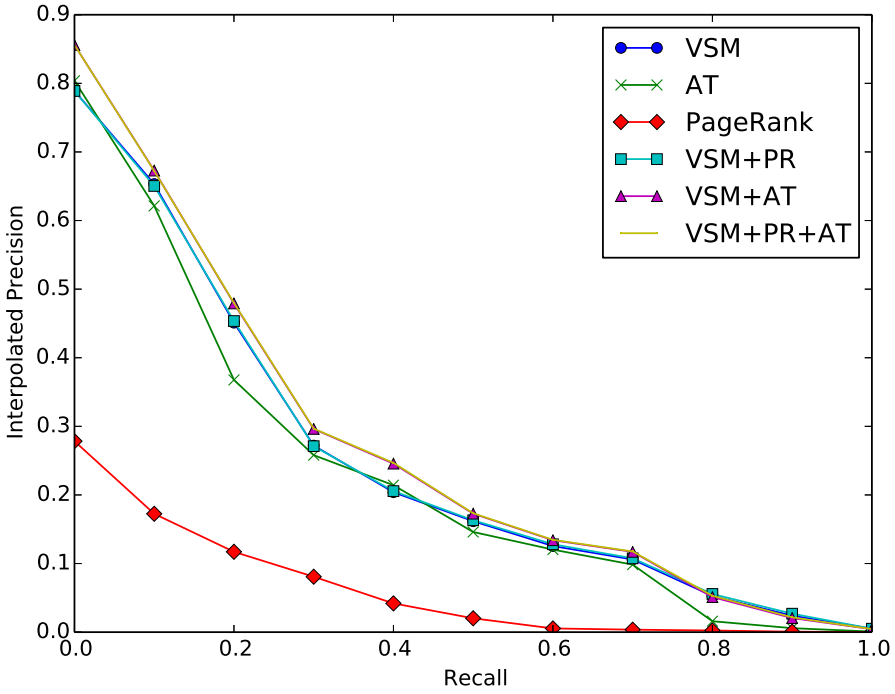

Figura 9. Precisão x Revocação - Combinações dos Modelos

e o **PageRank**, onde podemos visualizar um balanceamento no ranking das consultas, sendo que as consultas com uma avaliação melhor na combinação com o PageRank foram algumas que são mais referênciados, como: *Q2-Baixaki, Q8-Esporte, Q13-Google, Q30-Terra, Q32-Vivo, Q34-Youtube*. Já na Figura 11, podemos notar um grande desba-

lanceamento, onde o PageRank consegue ser melhor apenas nas consultas navegacionais: Q2-Baixaki, Q15-IG, Q30-Terra e Q33-Yahoo; e empatando na consulta Q4-Casa e Vídeo.

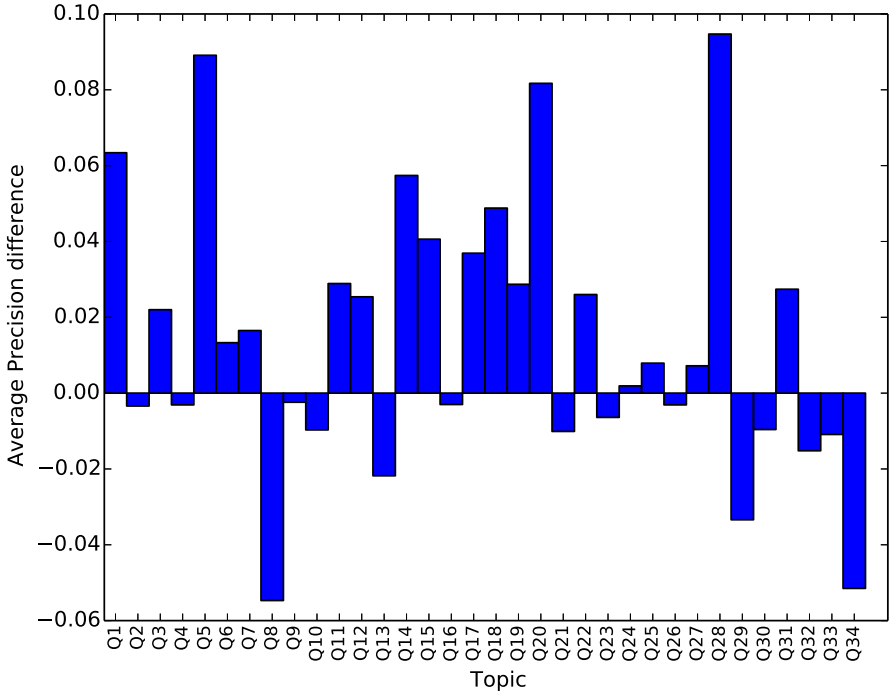

Figura 10. R-Precision - VR+AT vs VR+PR

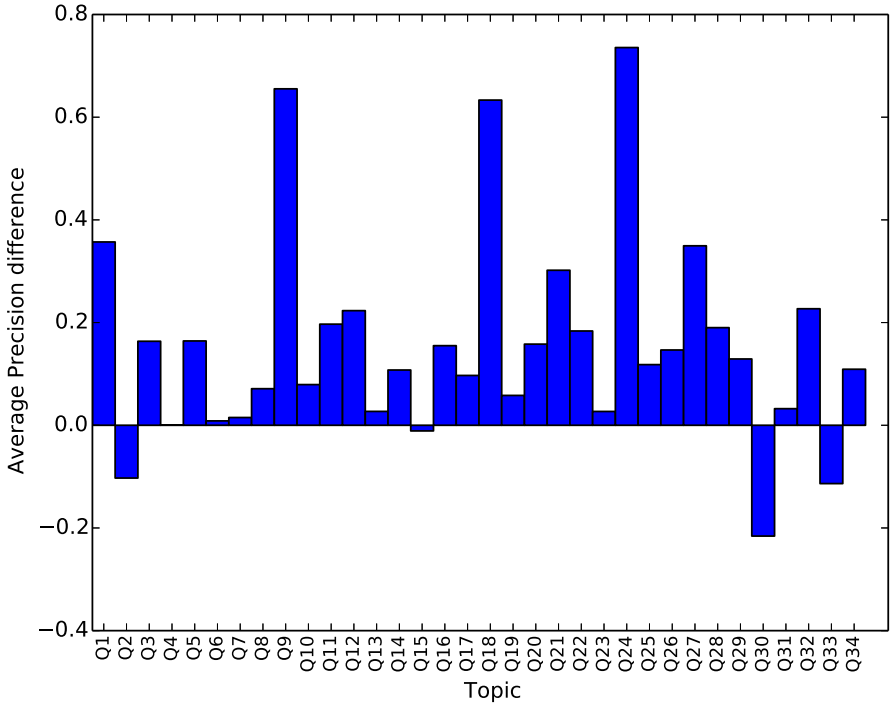

Figura 11. R-Precision - Anchor vs PageRank

7. Conclusão

Neste trabalho notamos como somente adicionando novos modelos de classificações conseguimos recuperar os documentos de forma mais eficiente e com uma precisão muito melhor do que o Modelo Booleano sozinho. Percebemos também como que difícil a criação de uma máquina de busca de uma maneira geral, e os cuidados que devem ser tomados e lembrados à todo tempo, como a utilização de memória, a melhoria de performance contínua sempre para as recuperações dos documentos, como também para a indexação.

Como também no primeiro trabalho, notamos existem diversas formas de implementações em cada algoritmo e que a menor diferença de um tratamento de índice, ou implementação de um PageRank pode alterar consideravelmente os resultados finais.

Conseguimos trabalhar com a realização de testes nesse trabalho e também perceber a importância dele na validação dos modelos de classificação e para comparação com qualquer outro trabalho. Nesta etapa executamos uma massiva gama de testes em cima dos modelos para tentar identificar a melhor combinação entre eles, e notamos que essa etapa exige um esforço muito grande para validação e de processamento para ser executada.

Como qualquer outro trabalho, existem diversas melhorias que podem ser realizadas, desde o tratamento e extração da descrição do conteúdo de uma página, para exibição nos resultados da busca, implementação de novos modelos de classificação e melhorias nos tratamentos dos termos e consultas (e.g., stemmização, expansão da consulta), como também melhorias no buscador web.

Acredito que os objetivos propostos nessa série de trabalhos foram atingidas, criando assim uma máquina de busca de uma forma eficiente, identificar os problemas e notar as dificuldades da construção de uma máquina no mundo real, como as diversas formas e modelos matemáticos para otimizar o processamento de consultas. Os trabalhos proporcionaram um grande conhecimento tanto na linguagem de programação C++, como também no funcionamento de uma máquina de busca de documentos web, enfrentando vários problemas e dificultades na implementação deste.

Referências

Baeza-Yates, R. A. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

DEV.W3.ORG (2015). Character Entity Reference Chart. `http://dev.w3.org/html5/html-author/charref`. [Online; accessed 03-May-2015].

eidheim (2015). Simple-Web-Server. `https://github.com/eidheim/Simple-Web-Server`. [Online; accessed 10-June-2015].

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The pagerank citation ranking: bringing order to the web.

Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Singhal, A. (2001). Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43.

Witten, I. H., Moffat, A., and Bell, T. C. (1999). *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann.

Ziviani, N. et al. (2004). *Projeto de Algoritmos: com Implementações em Pascal e C*, volume 2. Thomson.