UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

AUGUSTO KLINGER

O Modelo de Fusão de Rankings Baseado em Análise de Preferência aplicado a Metabusca

Trabalho de Graduação.

Prof. Dr. José Valdeni de Lima
Orientador

Porto Alegre, novembro de 2009

AGRADECIMENTOS

Agradeço a todos que contribuíram, não só para a conclusão deste trabalho, mas para a conclusão de toda a graduação, especialmente:

- Aos meus pais, Hélio e Marion, que me apoiam durante toda a vida, investindo em mim, me sustentando longe de casa, e que me proporcionaram ambiente, educação e conhecimento de alto nível para eu poder me formar, não só como bacharel em Ciência da Computação, mas como pessoa;
- Ao meu orientador, Prof. Valdeni, que me motivou nas pesquisas e contribuiu com muitas idéias para esse trabalho;
- Ao Elmário, com quem tive a oportunidade de trabalhar como bolsista em sua dissertação de mestrado e proporcionou a minha orientação para realizar esse trabalho, e com quem aprendi muitas coisas;
- Aos meu colegas de curso, Fernando, Rafael e Vinícius, grandes amigos que fiz na UFRGS, com quem conviví durante todo o curso e tornaram o caminho muito mais gratificante;
- Ao DACOMP, pelos momentos de lazer entre uma aula e outra;
- A todos os professores que tive na graduação, não só na UFRGS mas também na UNISC, que me aprovaram em suas disciplinas e contribuíram para meu conhecimento;
- A Paula, que me incentivou a estudar e completar a minha graduação na UFRGS;
- Aos meus grandes amigos de Santa Cruz, Felpi, Kipper, e aqueles que vieram de lá também estudar em Porto Alegre, Marco, Henrique, Marta, Elisa, e em especial ao Everton, com quem mantive mais contato durante esses tempos, e ao Moraes, que também esteve bastante presente ao meu lado. Todos eles sempre proporcionando um clima de amizade e descontração saudavel, que me motivou a seguir em frente.

Conteúdo

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CSS	Cascading Style Sheets
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JEE	Java Platform Enterprise Edition
JVM	Java Virtual Machine
MDPREF	Multidimensional Preference Analysis
SVD	Singular Value Decomposition
URL	Uniform Resource Locator

RESUMO

Um sistema de metabusca permite ao usuário obter resultados de diferentes motores de busca a fim de aumentar a abrangência e qualidade de sua consulta. Este trabalho propõe o uso do método de fusão de rankings baseado em Análise de Preferência (MDPREF) para fundir os rankings obtidos de diversos motores de busca, formando um único ranking resultante com maior relevância que os demais individualmente. Para tal foi feito um estudo e desenvolvido um protótipo, buscando explorar um cenário para a aplicação do MDPREF.

ABSTRACT

A metasearch system enable us to obtain results from different search engines in order to increase the coverage and quality of our query. This paper proposes the use of the ranking fusion method based on Preference Analysis (MDPREF) to melt the rankings obtained from different search engines, generating a result ranking with greater relevance than the others individually. For such a study was done and a prototype was developed, trying to explore a scenario aplication for the MDPREF.

1 INTRODUÇÃO

Desde que a Internet se tornou pública, no início da década de 90, o acúmulo de informações na Web cresceu de forma gigantesca. Com isso começaram a surgir ferramentas específicas de busca para recuperar suas páginas. Tais sistemas de busca foram inicialmente desenvolvidos em universidades como meio de catalogar o conteúdo da rede e com o passar do tempo foram evoluindo [20].

Segundo Cendón [2], existem dois tipos de ferramentas de busca na Web: motores de busca e diretórios. Diretórios organizam os sites em categorias, as quais podem conter subcategorias. Motores de busca permitem a busca por palavras-chave a partir de índices em suas bases de dados.

A busca em diretórios pode ser bastante difícil, pois os mesmos foram introduzidos quando a informação na Web ainda podia ser catalogada de forma não automática. Motores de busca são bem mais abrangentes e são essenciais em tempos em que o volume de informações não permite se chegar a bons resultados apenas por simples navegação.

Os resultados de uma pesquisa em um motor de busca tradicional não são buscados da Web diretamente, são buscados em índices previamente construídos. Programas chamados aranhas[1], que são implementadas por cada buscador, processam o conteúdo das páginas num processo chamado *Web Crawling*, responsável pela manutenção do conteúdo dos índices [17]. A forma de exploração do conteúdo varia de motor para motor.

O contínuo crescimento da Web favorece o surgimento de diversos tipos de ferramentas de busca, tornando o mundo dos serviços de busca complexo e volátil. Devido às características específicas de cada ferramenta, os resultados recuperados através de seu uso podem variar enormemente. Existem motores de busca com maior e menor abrangência, especializados em artigos médicos, determinados idiomas, documentos científicos, blogs, livros, etc.

Para obter uma maior amplitude em sua resposta, os usuários passaram a repetir a mesma consulta em diversos motores de busca, e para atender justamente a esta demanda é que surgiram as ferramentas de metabusca [10]. Um sistema de metabusca permite ao usuário pesquisar simultaneamente em vários motores de busca simples, tornando a pesquisa mais vasta e mais eficiente. Metabuscadores geralmente não possuem nenhum tipo de base de dados, baseando-se puramente nos dados de outros mecanismos de busca [1]. Utilizando-se dos índices de diversos motores de busca, é esperado que se obtenha uma abrangência bem maior do universo de sites da *Web*.

> "Os dois primeiros sistemas de meta-busca apareceram quase simultaneamente, em 1995. *Savvy Search* realizava buscas em até 20 outros buscadores por vez e inclusive permitia acesso a alguns diretórios temáticos. No entanto, simplesmente ignorava as opções avançadas dos vários sistemas de busca. Já o

[1] Do inglês *Spiders*, como são conhecidos os algoritmos de *parsing* usados pelos motores de busca

> *MetaCrawler*, que se tornaria mais popular, enfrentava as diferenças de sintaxe entre as opções avançadas dos sistemas de busca criando sua própria sintaxe e convertendo o *input* do usuário no comando correspondente em cada sistema de busca acessado.” [10]

Com o crescente numero de usuários da Internet, os metabuscadores começaram a concorrer não só com os motores de busca mas com outros metabuscadores, fomentando assim o crescimento e o avanço deste tipo de busca.

Este trabalho visa a utilização do método de fusão de *rankings* baseado no MDPREF (método de fusão baseado na análise de preferência multimensional) para classificar os resultados obtidos de diversos motores de busca. Os bons resultados obtidos no trabalho de Dutra [7] devem refletir em um bom *ranking* resultante na metabusca, mostrando assim um cenário ideal para a aplicação do MDPREF.

1.1 Objetivos do Trabalho

1.1.1 Objetivo Geral

Aplicar o método de fusão de *rankings* baseado no MDPREF em um sistema de metabusca, buscando a validação do método em um cenário real e atual.

1.1.2 Objetivos Específicos

O objetivo geral foi dividido em objetivos mais específicos para que possa ser atingido:

- Descrever os conceitos básicos envolvidos;
- Descrever o funcionamento do método MDPREF de modo geral;
- Descrever sistemas de metabusca atuais;
- Estudar características e operação de motores de busca diversos;
- Desenvolver um sistema que recolha os rankings gerados por motores de busca e realize a fusão com o MDPREF;
- Analisar e comparar os resultados obtidos.

1.2 Organização do Trabalho

- Capitulo 2, **Conceitos Básicos**, apresenta os conceitos de metabusca e fusão de *rankings*, tomando como base para explicar o método MDPREF e seu emprego num sistema de metabusca;
- Capitulo 3, **Trabalhos Relacionados**, exemplifica o cenário de metabusca atual encontrado na Web, mostrando informações e características dos metabuscadores mais populares;
- Capitulo 4, **Implementação**, mostra como foi realizada a construção do sistema de metabusca e proporciona uma visão geral do seu funcionamento;
- Capitulo 5, **Avaliação do Sistema de Metabusca proposto**, faz um comparativo dos *rankings* gerados pelo sistema implementado com alguns *rankings* gerados por alguns sistemas apresentados no Capitulo 3;

- Capitulo 6, **CONCLUSÃO**, apresenta as conclusões sobre o trabalho realizado e indica possíveis trabalhos futuros.

2 CONCEITOS BÁSICOS

2.1 Metabusca

Metabusca significa fazer uma busca sobre outra(s) busca(s). Uma ferramenta de metabusca funciona como um agregador de buscadores, fazendo uso das bases de dados dos motores de busca a que submete as consultas. Quando se realiza uma metabusca na Web, a ferramenta envia o pedido para diversos motores de busca simultaneamente como o Google, Yahoo, Bing, e diversos outros menores, e retorna o resultado deles em uma mesma tela. Alguns sistemas de metabusca ainda possuem recursos a mais, como unificar os resultados e alterar o posicionamento deles.

Estas ferramentas de pesquisa, conforme Blattman, Fachin e Rados [1], são utilizadas quando:

- Procura-se por um tópico obscuro;
- Não se obtém resultados em buscas;
- A pesquisa não é complexa;
- Deseja-se recuperar o maior número de documentos possíveis com uma estrutura de sintaxe, assuntos especiais que possam limitar resultados da busca.

Do ponto de vista do usuário não há diferença entre um site de busca tradicional e um de metabusca. Ambos utilizam a mesma clássica interface, onde os elementos principais são uma caixa de texto, em que será digitado o termo alvo da consulta, e o botão de “buscar”.

Para obter os resultados de motores de busca diversos, segundo Tous e Delgado [17], é explorado o fato de que eles usam navegadores como interface, com requisições HTTP e resultados em páginas HTML, o que deixa a porta aberta para outras aplicações, como sistemas de metabusca, disparerem consultas.

A tarefa de um sistema de metabusca pode ser dividida em dois sub-problemas. Um deles é como realizar a consulta a cada motor de busca. O outro consiste em como recuperar a informação retornada. É preciso estudar individualmente cada buscador afim de obter as respostas.

Descobrir as particularidades de cada motor de busca pode ser uma tarefa trabalhosa, e não há garantias de que as interfaces não serão mudadas. Isso torna a manutenção de um sistema de metabusca bastante difícil. O ideal seria verificar automaticamente as mudanças que possam ocorrer em cada mecanismo de busca, afim de deixar o sistema sempre atualizado e funcional.

O cenário perfeito para a aplicação de metabusca seria aquele no qual se tem uma ordem completa de todos os elementos do universo em todos os *rankings*. Porém isso não é possível pois cada motor de busca tem uma cobertura diferente da Web [9]. É improvável que os motores de busca sejam capazes de rankear toda a coleção de páginas da Web, que cresce a uma taxa bastante rápida.

Na sessão de Trabalhos Relacionados são citadas algumas das principais ferramentas de metabusca que estão atualmente em funcionamento.

2.2 Fusão de Rankings

Um *ranking* é um conjunto de elementos ordenados de forma que, para cada dois itens, o primeiro é maior, menor ou igual ao segundo [19]. Usualmente são representados como listas. Alguns autores, como Hsu e Palumbo [11], tratam *rankings* como permutações de elementos.

Fusão de *Rankings* é o problema de computar um *ranking* consensual, dados diversos *rankings* individuais contendo elementos classificados por diferentes juízes [16]. Um juíz é quem determina a ordem dos elementos de um dado universo, portanto um motor de busca é um juíz.

Existem vários métodos para se realizar a fusão de *rankings*. Os métodos utilizam as informações dos rankings individuais, como o ordinal associado a um elemento (*rank*[2]), uma nota dada a cada elemento (*score*[3]) ou o próprio conteúdo, com o objetivo de encontrar o melhor *ranking* resultante.

2.3 MDPREF

MDPREF é um método de mapeamento perceptual, técnica proveniente na área de *Marketing*, onde o estudo das preferências de grupos de consumidores em relação a determinados produtos é de extremo interesse. O mapeamento perceptual permite a análise conjunta dos atributos de um produto, podendo gerar um gráfico com o posicionamento, em um espaço comum, dos produtos e consumidores com suas respectivas preferências. Em suma, métodos de mapeamento perceptual permitem determinar a preferência de um grupo de indivíduos em relação a um conjunto de elementos.

> "MDPREF é baseado no modelo desenvolvido por J. D. Carrol e J. J. Chang em 1973, que faz uso do teorema de decomposição de Eckart-Young (SVD) ou na análise de componente principal (PCA), executado sobre os dados de preferência gerados da avaliação dada pelos consumidores para cada produto."[7]

Graficamente, cada juíz (responsável por um *ranking*) ou grupo de juízes é representado como um vetor, o qual indica a sua direção de preferência, os estímulos são representados como pontos. Cada ponto de estímulo é projetado sobre cada vetor, revelando a sua preferência. No gráfico (Figura 1), as dimensões são as duas primeiras componetes principais. Conforme Dutra [7], a primeira componente representa a informação que é mais saliente na avaliação dos julgadores, e, a segunda componente representa a direção da preferência dos julgadores, sendo estas ortogonais.

---

[2] Cada elemento está associado a uma posição.

[3] Cada elemento está associado a um valor.

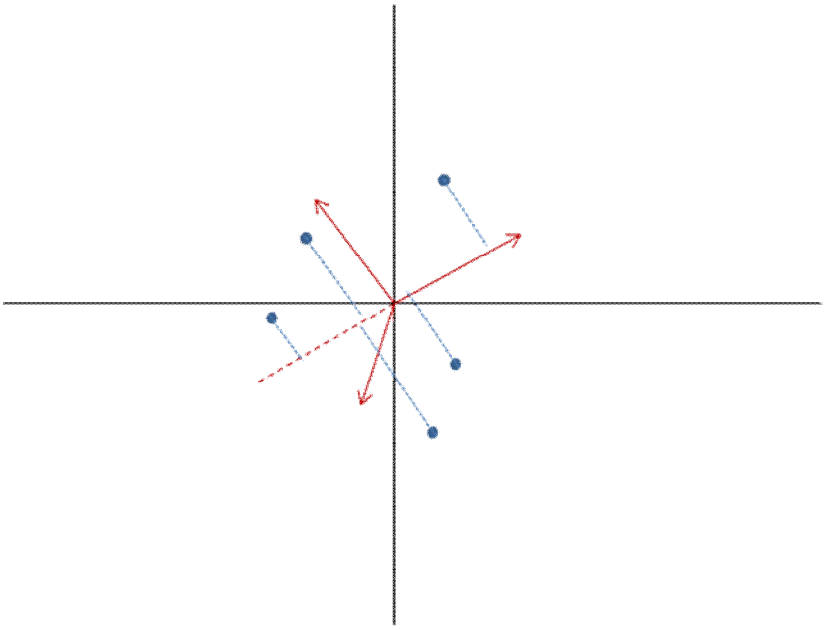


Figura 1. Gráfico de estímulos e vetores juízes

Em termos mais técnicos, o mapa é gerado baseado no modelo de análise de componente principal. Uma matriz de dados de **i** juízes por **j** estímulos é decomposta em duas matrizes menores. A primeira dessas duas matrizes é a matriz de escores da componente principal, de tamanho **i** por **t** (t é o numero de dimensões). A segunda matriz é chamada de matriz da componente principal e seu tamanho é **t** por **j**. Essa matriz representa os **j** estímulos ao longo das **t** dimensões da componente principal [6].

2.4 Fusão de Rankings baseada no MDPREF

O modelo de fusão baseado na Análise de Preferência foi proposto por Dutra [7], em sua dissertação de Mestrado. Os dados de entrada são uma matriz com os *rankings* de cada juíz e uma matriz de pesos. Portanto, cada juíz pode receber um peso diferente, o que irá influir no resultado final da fusão. A matriz dos *rankings* é colocada na forma de um *cluster*[4] de *rankings*, de tamanho **p** por **n,** sendo **p** a dimensão dos objetos avaliados e **n** a dimensão dos juízes.

Iniciando o processo, primeiramente é contruída uma matriz tridimencional D (matriz de *scores* primários) a partir da matriz de *rankings* (*cluster* de *rankings*). Em uma das dimensões de D estão os juízes, e as outras duas representam a comparação entre pares de elementos. Cada par *jk* de elementos é avaliado, para o juíz *i*, da seguinte forma:

- D[i][j][k] = 1, se o elemento *j* foi avaliado melhor que *k*
- D[i][j][k] = -1, se o elemento *j* foi avaliado pior que *k*
- D[i][j][k] = 0, se o elemento *j* foi avaliado igual *k* ou não foi avaliado.

A partir da matriz D e da matriz de pesos é formada uma matriz bidimensional S (matriz de *scores* secundários), que define a diferença entre a preferência dos elementos

[4] Agregado

j sobre *k* para cada juíz. O trecho de código a seguir (Figura 2) exemplifica como é calculada essa diferença e como é considerada a contribuição dos pesos atribuídos aos juízes.

```
S = new double[cluster.n][cluster.p];
for (int i = 0; i < cluster.n; i++) {
    for (int j = 0; j < cluster.p; j++) {
        int aux = 0;
        for (int k = 0; k < cluster.p; k++) {
            aux += D[i][j][k] - D[i][k][j];
        }
        S[i][j] = Math.sqrt(cluster.rankings.get(i).weigth) * aux;
    }
}
```

Figura 2. Calculo da matriz S

Como pode-se ver na Figura 2, S é de tamanho **n** por **p**. Cada elemento seu é preenchido com o acumulo na variável *aux* dos resultados da diferença do valores nas posições *jk* e *kj*. Ou seja, imaginando D como *i* matrizes **p** por **p,** cada linha e coluna de toda matriz *i* é condensada em um elemento de S, que contém o acumulo das diferenças dos valores entre linhas e colunas de mesmo índice (linha 1 com coluna 1, linha 2 com coluna 2, e assim por diante). O valor acumulado na variável *aux* é multiplicado pela raíz quadrada do peso do *ranking* do juíz *i* e colocado em S na posição *ij*, correspondente ao juíz e objeto avaliado respectivamente.

Através da decomposição matricial SVD de S, são obtidas três novas matrizes: U, L e A. L é uma matriz diagonal dos autovalores. U e A são matrizes que contém os autovetores, com suas colunas sendo ortogonais. As três matrizes são ordenadas de acordo com a magnitude dos autovalores e utilizadas para obter as matrizes solução X e Y.

Apenas as duas componentes mais significativas de U, L e A são utilizadas para formar X e Y, procedendo-se da seguinte forma:

- X = UL
- Y = A

O vetor de preferência é calculado com base na matriz X, e normalizado, conforme o trecho de código a seguir (Figura 3). Cada uma das duas componentes do vetor é dada pelo somatório das linhas da matriz X em cada uma das duas colunas.

```
double r[] = new double[2]; //vetor r
vetPref = new double[1][2];
// calcula r e normaliza o vetor de preferência (vetPref)
for (int col = 0; col<2; col++)
    for (int row=0; row<cluster.n; row++)
        r[col] += x[row][col];
double normar = Math.sqrt(r[0]*r[0]+r[1]*r[1]);
vetPref[0][0] = r[0]/normar;
vetPref[0][1] = r[1]/normar;
```

Figura 3. Calculo do vetor de preferência

No código acima (Figura 3), *r* é o vetor ainda não normalizado. A normalização do vetor é feita tirando a raíz quadrada da soma de suas componentes ao quadrado (sétima linha), e dividindo as componentes por esse valor (armazenado na variável *normar*).

Finalmente, a matriz Y é projetada sobre o vetor de preferência, resultando no *ranking* consensual.

A visualização gráfica pode ser vista na Figura 4 abaixo. Os pontos são todos os elementos avaliados, contidos em Y. O vetor de preferência é representado como uma linha com um ponto no final. Os algarismos são os rótulos de identificação de cada elemento.

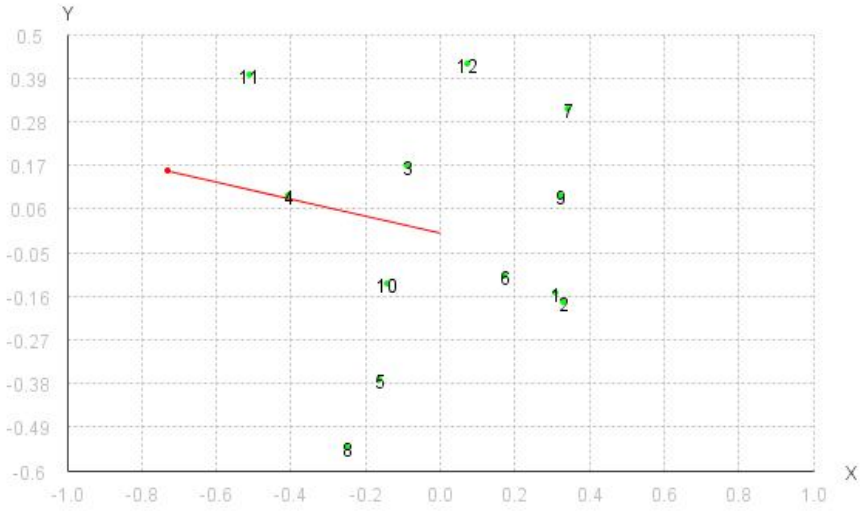


Figura 4. Gráfico resultante de fusão baseada no MDPREF

2.5 Distâncias

A distância entre listas, ou *rankings*, pode ser calculada de diversas formas. É a principal métrica no contexto de *ranks*. Basicamente, essas distâncias podem ser de desordem, medindo o quão desorganizado um *ranking* está em relação a outro, ou espaciais, medindo o numero de movimentos necessários para ir de um vértice a outro em um espaço multidimensional [7].

A Distância de Spearman footrule é do tipo espacial. Ela retorna a soma das diferenças de posição em cada lista para cada elemento, sendo assim, aplicável a somete duas listas. Sendo r_1 e r_2 dois *rankings* contendo os elementos do universo U, defini-se a distância de Spearman footrule por:

$$F(r_1, r_2) = \sum_{i=1}^{|U|} \left| f_{r_1}^{r}(i) - f_{r_2}^{r}(i) \right| \quad \textbf{(1)}$$

Existem diversas variações na literatura para o cálculo de *F*, porém não serão abordadas.

A distância Kendall, por sua vez, é uma medida tanto espacial como de desordem [14]. Dados dois *rankings*, a distância Kendall (espacial) entre eles é o numero de pares discordantes. Já a interpretação de desordem computa o numero de trocas necessárias para tornar um *ranking* r_1 igual a um *ranking* r_2.

Formalmente, a distância Kendall entre dois *rankings* r_1 e r_2, contendo todos os elementos do universo U, é dada por:

$$K(r_1, r_2) = \sum_{\{i,j\} \in U} k_{ij}(r_1, r_2) \tag{2}$$

Sendo que $k_{ij}(r_1, r_2)$ é avaliado recebendo um valor 0 ou 1, da seguinte forma:

- 0, se elemento i foi melhor avaliado que j em r_1 e em r_2 ou se elemento i foi avaliado pior que j em r_1 e em r_2;
- 1, caso contrário.

Outra distância baseada em trocas é a de Cayley, que também conta o numero de trocas de elementos para chegar de uma lista em outra, mas que permite trocas arbitrárias ao invés de somente adjacentes como na distância Kendall, conforme Marden [14]. A distância de Cayley pode ser calculada diretamente fazendo as trocas em um dos *rankings* até chegar ao outro. Por exemplo, dados r_1=[3,1,2,4] e r_2=[1,2,3,4], são necessárias duas trocas arbitrárias de elementos para deixar os dois *rankings* na mesma ordem, conforme a Tabela 1.

Tabela 1. Distância Cayley por trocas diretas

Trocas	Rank
0	**[3,1,2,4] = r_1**
1	**[3,**2,1**,4]**
2	**[**1**,2,**3**,4] = r_2**

A distância de Hamming é outra medida de desordem. Ela calcula o numero de elementos discordantes entre dois *rankings*. Consiste em remover todos os elementos que estiverem em posição errada e então colocá-los um por um nas posições certas.

Como todas essas medidas trabalham sempre com *rankings* completos, ou seja, que possuem todos os elementos do universo, para se calcular a distância entre *rankings* com elementos divergentes é necessária uma operação chamada *completamento de ranking* [8]. Para completar um ranking é feita a adição dos elementos que aparecem no outro *ranking*, preservando a ordem. Ou seja, dado um universo *U* formado pela união dos elementos dos *rankings* r_1 e r_2, o completamento de r_1 em relação a r_2 resulta em um *ranking* completo em relação a *U*, composto pelos elementos de r_1 acrescido dos elementos de r_2 não pertecentes a r_1, ao final deste, preservando a ordem de r_2.

3 TRABALHOS RELACIONADOS

Existe uma diversidade de sistemas de metabusca na Web. Cada um deles apresenta seus próprios critérios para a ordenação do ranking resultante e nem sempre tais critérios são expostos, dado que tais sistemas precisam ter um diferencial, a fim de sobreviver à concorrência.

O Ixquick [13], disponível em 18 linguagens, combina resultados de até 11 motores de busca, permitindo ao usuário desabilitar alguns se desejar. Seu algoritmo de classificação dos resultados é simples: dá uma estrela para um site cada vez que aparece entre os dez primeiros resultados de um motor de busca. Assim, se um resultado apresentar sete estrelas significa que apareceu entre os dez primeiros em sete motores de busca. Como os resultados são escolhidos de maneiras diferentes por cada motor de busca, um site com muitas estrelas foi escolhido segundo diversos critérios, destacando-se assim a sua relevância.

O Clusty [3] é um sistema de metabusca da empresa de *softwares* de busca Vivísimo, fundada em 2000. Ele combina os resultados de diversos motores de busca e gera uma lista ordenada baseada na comparação dos *rankings*. A idéia principal do Clusty é agrupar os resultados em tópicos, um processo chamado de clusterização, fazendo com que os usuários consigam direcionar melhor sua busca.

Outro sistema que faz uso de clusterização é o iBoogie [12]. Os clusters são formados baseados na URL, título e descrição da página, e rotulados conforme o conteúdo dos documentos que o compõe. Um mesmo documento pode aparecer em mais de um cluster. Tudo isso é feito por uma espécie de parser. Os termos também são normalizados pelo sistema, eliminando ambiguidades. Além disso, é possível personalizar a busca escolhendo até sete motores de busca diferentes que o sistema dispõe.

Mais um *site* de metabusca bastante fácil de se encontrar na Web é o Dogpile [5]. Seu sistema combina os resultados dos mais populares motores de busca da atualidade: Google, Ask, Live Search e Yahoo!. Os resultados são filtrados e classificados de acordo com o termo entrado, fazendo uso de um algoritmo que separa os links patrocinados dos não patrocinados. Assim, o *ranking* resultante é uma combinação de links patrocinados e não, de acordo com a consulta do usuário. Por exemplo, se for colocada a palavra “comprar” na busca, aparecerão mais links patrocinados. Se o termo pesquisado for científico, os links normais terão mais espaço.

A filosofia altruísta de ajudar cães abandonados é maior chamariz do Dogpile. Utilizando o sistema o usuário está colaborando com a causa, o que acaba se destacando em relação ao resultado da metabusca em si. O mesmo ocorre com o DoGreatGood [4], que se trata de uma expansão do Dogpile. No DoGreatGood o usuário pode colaborar com obras de caridade em geral, e não somente com aquelas relacionadas a cães. O resultado da busca é idêntico em ambos.

O Metacrawler [15] é mais um sistema de metabusca que combina, assim como os dois citados acima, links patrocinados e não, tentando identificar a intenção da busca realizada. Os links são dispostos por ordem de relevância conforme encontrados nos vários motores de busca.

O Zworks [21] realiza a busca em doze motores, classificando os sites de acordo com o numero de aparições nos *rankings*, dos quais são consideradas as vinte primeiras posições. As posições em que o site figura são somadas e usadas como critério de desempate, quanto menor o valor da soma, melhor a classificação. Os sites ganham colocações também quando pertecem a um mesmo domínio, caso um site apareça em somente um *ranking*, ele pode ganhar posições se estiver em um domínio que apareça nos demais *rankings*. Sites que aparecem em pelo menos cinco motores de busca são destacados e considerados altamente relevantes para o termo pesquisado. O Zworks ainda dispõe de uma *Power Search*, que realiza a busca permitindo mais critérios de filtragem, possibilitando, inclusive, desabilitar alguns buscadores, se desejado.

Um metabuscador bastante interessante é o KartOO. Ele possui uma interface totalmente diferente dos demais, e exibe os resultados de forma gráfica. Ele forma uma espécie de mapa, exibindo os sites e as conexões entre eles.

No Brasil alguns exemplos de metabuscadores são o Tay, ainda em desenvolvimento, e o Exploora, que permite a busca em até 4 mecanismos. O Exploora, diferente dos demais, exibe os resultados de cada buscador separadamente, fazendo subdivisões no browser. Basicamente o resultado é como abrir 4 *sites* de busca em janelas diferentes e submeter a mesma consulta em todos eles.

Existem muitos outros sistemas de metabusca na Web, tais como o Mamma, Pandia, Metasearch.com, OneSeek e FlexFinder. Alguns deles visam atender uma determinada região apenas, ou certo conteúdo, ou um idioma, e outros tentam ser o mais abrangentes possível.

4 IMPLEMENTAÇÃO

O protótipo foi implementado em Java, no ambiente de desenvolvimento *NetBeans*, e faz uso de dois pacotes: o pacote *mdpref*, desenvolvido por Augusto Klinger para o trabalho de dissertação de Elmário G. Dutra [7], que provê o mecanismo básico para a fusão baseada na Análise de Preferência, e o pacote *htmlparser*, disponível na Web, que permite acesso e leitura de páginas da Web via suas classes Java.

O projeto do sistema de metabusca está dividido em três grupos: Interface do Usuário, Motores de Busca e Núcleo.

4.1 Interface do Usuário

É onde se localizam as classes que possuem entrada e saída para o usuário do sistema de metabusca. Para fins de testes rápidos possui uma janela para uso local, como se fosse um programa comum. Mas é um *applet*[5] o elemento principal de interação.

Um *applet* foi escolhido como interface pois é de fácil publicação na Web e não demanda um servidor, pois roda diretamente na máquina do cliente. O *applet* desenvolvido pode ser testado através de um navegador qualquer com suporte a Java, com a JVM devidamente instalada.

A Figura 5, a seguir, mostra a interface atual do sistema, que pode ser acessado na Web por qualquer um. A partir dessa figura é possível já observar um *ranking* completo gerado pelo metabuscador.

Basicamente todos os métodos das classes deste pacote servem para visualização dos resultados e interações. Um método *search* é o principal, que representa a ação do clique no botão de busca. Isso faz com que a entrada do usuário digitada na caixa de texto seja enviada para o Núcleo (seção 4.3) e, após o processamento, seja exibido o resultado.

[5] www.inf.ufrgs.br/~aklinger/metasearch/

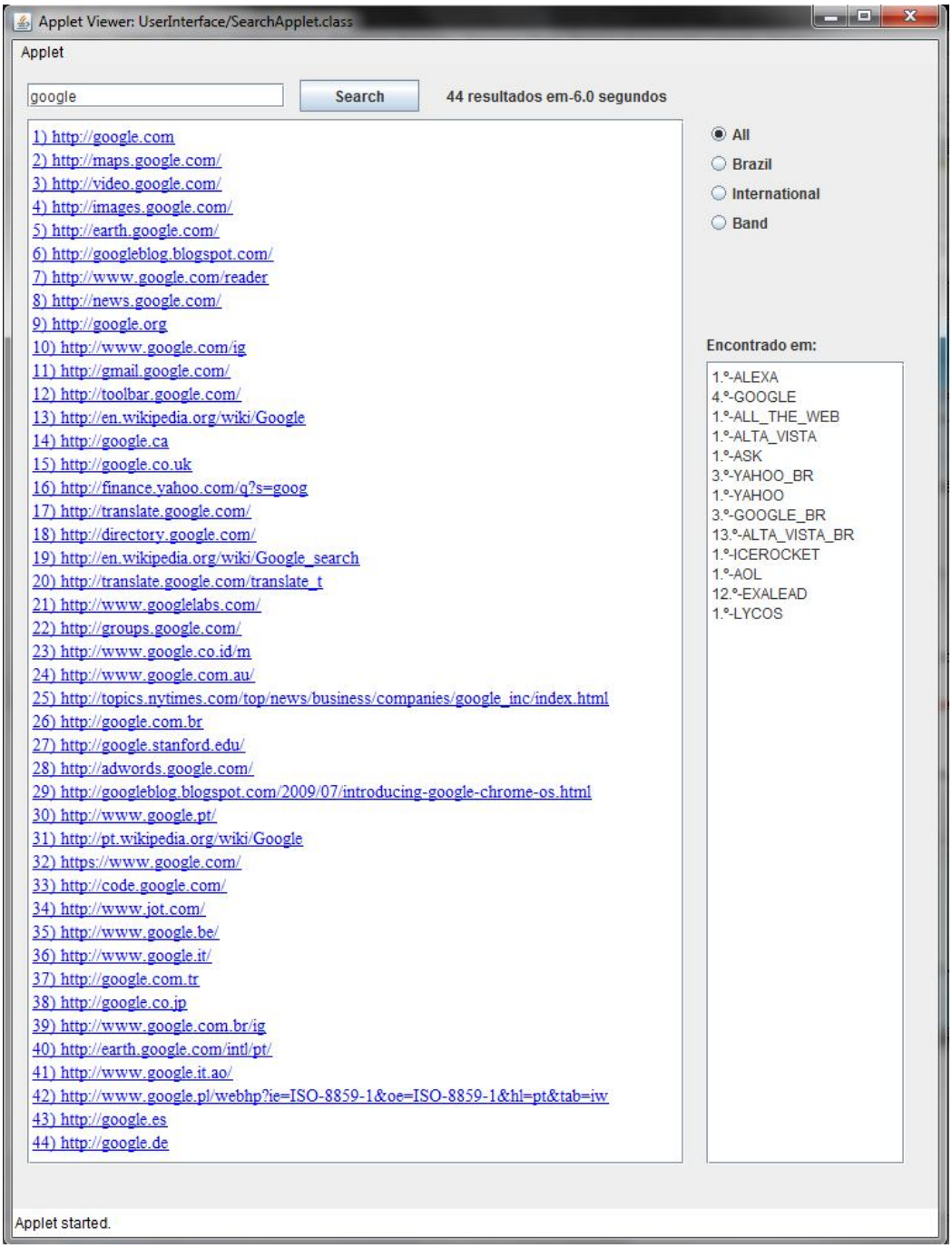


Figura 5. Interface do sistema

Como nota-se na Figura 5, a interface segue os padrões adotados pela maioria dos sistemas de busca, com uma caixa de texto para o usuário digitar o termo da pesquisa e um botão de busca ao lado. Abaixo são exibidos os resultados. Ao lado ainda são exibidos os motores de busca e posição em que foi encontrado o link que estiver selecionado pelo cursor do mouse (no momento em que foi extraída a Figura 5, o mouse estava posicionado no primeiro link do ranking). Os botões de rádio servem para alterar

os pesos de certos motores de busca no momento da fusão, de acordo com a orientação desejada para a pesquisa.

4.2 Motores de Busca

Todas as informações referentes aos motores de busca suportados são armazenadas neste pacote. Para cada motor de busca é armazenada a sua forma de consulta (*query*), seu separador (caractere usado para separar palavras no lugar de um espaço em branco), a classe *CSS* dos resultados de uma consulta, e a *string*[6] para a próxima página. Não é realizada nenhuma ação dentro deste pacote. Todos os motores de busca são descritos por um *enum*, como mostrado abaixo. Pela Figura 6 também pode-se identificar todos os dezessete motores de busca que são utilizados.

```
public enum SearchEngine {
    GOOGLE_BR ("http://www.google.com.br/search?as_q=","+","l","&start=10&sa=N"),
    GOOGLE("http://www.google.com/search?hl=en&q=","+","l","&start=10&sa=N"),
    GOOGLE_BOOKS ("http://books.google.com/books?q=","+","link_aux","&start=10"),
    GOOGLE_BLOGS ("http://blogsearch.google.com/blogsearch?hl=en&ie=UTF-8&q=","+","f1","&start=10"),
    YAHOO_BR ("http://br.search.yahoo.com/search?p=","+","yschttl spt","&b=11"),
    YAHOO("http://search.yahoo.com/search?p=","+","yschttl spt","&b=11"),
    ASK ("http://www.ask.com/web?q=","+","L4","&page=2"),
    CUIL ("http://www.cuil.com/search?q=","+","title_link","#page=2"),
    ALTA_VISTA ("http://www.altavista.com/web/results?q=","+","res","&stq=10"),
    ALTA_VISTA_BR ("http://www.altavista.com.br/web/results?q=","+","res","&stq=10"),
    AOL ("http://aolsearcht12.search.aol.com/aol/search?invocationType=comsearch40&query=","+","find","&page=2"),
    LYCOS ("http://search.lycos.com/?query=","+","large","&page2=1&tab=web"),
    EXALEAD ("http://www.exalead.com/search/results?q=","+","url","&b=10"),
    ALEXA ("http://www.alexa.com/search?q=","+","offsite",null),
    ICEROCKET ("http://www.icerocket.com/search?tab=web&fr=h&q=","+","main_link","&p=2"),
    ENTIREWEB ("http://www.entireweb.com/search0?q=","+","result_title","&of=10"),
    ALL_THE_WEB ("http://www.alltheweb.com/search?cat=web&cs=iso88591&q=","+","res","&o=10");

    private final String query, separator, ccsClass, page2;

    SearchEngine (String query, String separator, String ccsClass, String page2) {
        this.query = query;
        this.separator = separator;
        this.ccsClass = ccsClass;
        this.page2 = page2;
    }
```

Figura 6. Enum descritor dos motores de busca.

O formato de cada consulta, bem como o separador, são importantes para poder passar o termo da busca como argumento para o motor de busca em questão. A consulta montada será usada para obter os resultados, através do pacote *htmlparser*, em métodos do Núcelo (4.3). A Figura 7, a seguir, identifica os elementos de uma consulta base.

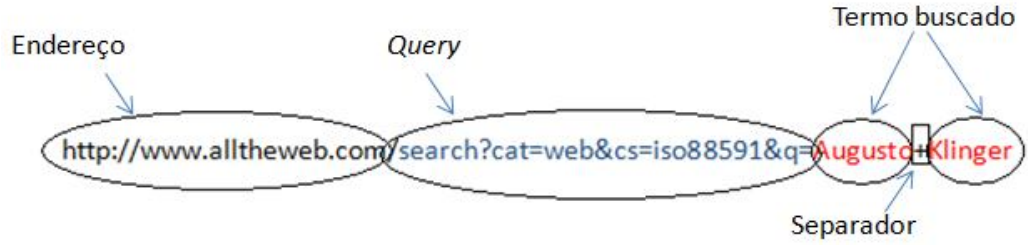


Figura 7. Formato de consulta por *query*.

[6] Conjunto de caracteres

A classe CSS serve para identificar os elementos do *ranking* retornado pelo motor de busca. O pacote *htmlparser* irá ler e obter os links que estiverem com esta classe dentro da página retornada pela consulta, como veremos mais adiante. O quarto campo do *enum*, Figura 6, denominado *page2*, será utilizado para acessar a segunda página de resultados, quando houver.

Os motores de busca utilizados foram selecionados a partir de uma lista [18]. Para serem incluídos no pacote precisaram atender a alguns critérios:

I. Retornar resultados coerentes a partir de certos termos;
II. Expôr a forma da consulta;
III. Ter uma classe CSS definida para os elementos do *ranking* gerado.

Para o primeiro critério foram testados termos genéricos como "*search engine*", "*metasearch*", "*beatles*", tais que pudessem certamente retornar resultados dada a quantidade de sites contendo tais palavras. O segundo critério foi testado pela submissão de consultas através das próprias interfaces dos buscadores, observando o endereço de retorno como mostrado na Figura 8.

Figura 8. Observação do critério *II* de exposição da consulta

O terceiro critério, referente ao CSS, foi testado também via observações, mas com o auxílio de uma rotina. Foi implementada uma função que lê uma URL de consulta e, para cada link encontrado, imprime o seu endereço e a correspondente classe CSS. Assim, de posse dessa lista, é possível comparar os sites dela com os sites apresentados no *browser* afim de identificar uma classe CSS unica para os links do *ranking* resultante. Caso não haja uma classe unica para os elementos do *ranking*, o motor de busca é descartado.

Os motores de busca que atenderam aos critérios tiveram seus dados armazenados no *enum* mostrado na Figura 6.

4.3 Núcleo

É onde está a parte principal do projeto. As classes do núcleo disparam as consultas aos motores de busca, lêem os resultados e preparam a fusão. Nesta sessão serão descritas as quatro classes fundamentais do sistema.

4.3.1 Submit

Classe que recebe o termo a ser buscado como parâmetro e passa para todos os motores de busca. Cada motor de busca é invocado dentro de uma *thread* diferente, que é implementada pela classe SubmitRunner (4.3.2). Após todos os motores de busca terem devolvido seus resultados, eles são armazenados em uma lista de *rankings*, que será usada pelo clusterizador (4.3.4).

```
public Submit(String term){
    rankings = new Vector<ArrayList<String>>();
    motor = new ArrayList<String>();
    for (SearchEngine searchEngine : SearchEngine.values()){
        SubmitRunner sRunner = new SubmitRunner(searchEngine, term, rankings, motor);
        Thread t = new Thread(sRunner);
        incCounter();
        t.start();
    }
    while (getCounter() > 0) {
        try {
            Thread.sleep(900);
//          System.out.println("Esperando as threads");
        } catch (Exception e) {
            System.out.println("Erro: " + e.getMessage());
        }
    }
```

Figura 9. Funcionamento principal da classe Submit

Como é mostrado no trecho de código acima (Figura 9), um laço cria uma instância de SubmitRunner para cada motor de busca e a passa por parâmetro para a classe *Thread* do Java. Um contador é usado para controlar o numero de *threads* disparadas, assim no laço seguinte as *threads* esperam umas pelas outras até todas terem finalizado e o contador estar em zero. O decremento do contador é feito dentro de cada instância.

Também é importante notar que é criado um vetor (*Vector*[7]) de *rankings*. Cada *ranking* é representado como uma lista (*ArrayList*[8]) de *Strings*. É nessa estrutura, denominada **rankings**, que serão armazenados todos os resultados retornados pelos motores de busca. A estrutura **motor** será utilizada para identificar de onde veio cada resultado.

4.3.2 SubmitRunner

Classe que extende a classe *Thread* do Java. Basicamente, ao disparar uma instância de SubmitRunner, é instanciado um objeto *Searcher* (4.3.3) e invocado o método *goSearch*, que será descrito a seguir em 4.3.3. O *ranking* resultante é devolvido pelo *Searcher* e adicionado a uma lista parcial dos *rankings* já obtidos.

---

[7] Classe Java que permite acesso simultâneo a seus elementos

[8] Classe Java que provê todos os métodos para gerenciar uma lista como sendo um único objeto

```
public void run() {
    Searcher searcher = new Searcher();
    searcher.goSearch(engine, term);
    ArrayList<String> resultRank = searcher.getRanking();
    if (resultRank.size() > 0) {
        resultRank = searcher.getRanking();
        rankings.add(resultRank);
        motor.add(engine.toString());
    }
    Submit.decCounter();
}
```

Figura 10. Método run() de SubmitRunner

O método *run* apresentado acima na Figura 10, principal de uma *Thread*, faz a consulta em um motor de busca (**engine**), verifica se retornou resultados e então os armazena na estrutura de **rankings.** O nome do motor de busca é armazenado na estrutura **motor**, tornando assim possível a identificação do buscador utilizado que retornou dado ranking, já que ambos estarão na n-ésima coluna em suas respectivas estruturas.

Antes de sair de *run*, o contador de *threads* é decrementado na classe Submit.

4.3.3 Searcher

Aqui se encontra o método *goSearch,* que recebe dois parâmetros de entrada: o motor de busca e o termo alvo da busca. Primeiramente é montada a consulta, conforme mostrada na Figura 7, de acordo com o motor de busca recebido, os espaços em branco do termo alvo, quando presentes, são substituído pelo caractere separador, e o termo alvo é então concatenado a *query* e ao endereço, formando uma URL.

A URL formada é passada como parâmetro para o objeto Parser do pacote *htmlparser,* que extrai todos os links da página. Em cada link é filtrado o atributo *class* e são adicionados ao *ranking* somente os links que possuem a classe CSS de resultado da consulta, identificada previamente. Assim são evitados os demais links de patrocínios, propagandas, navigacionais e outras coisas que não são pertinentes ao *ranking*.

São considerados somente os vinte primeiros resultados de cada motor de busca, sendo que o processo de extração dos links da segunda página é semelhante, a diferença está na formação da URL, onde é concatenado também o *string* que caracteriza a segunda página.

4.3.4 Clusterizer

Classe responsável por fazer a preparação dos resultados para a fusão. Os diversos *rankings* são aqui dispostos em uma representação matricial única. Clusterizer recebe o termo alvo da busca e repassa para sua instância de Submit (4.3.1), que irá devolver os *rankings* de todos os motores de busca que tiverem retornado. Então é feita uma lista com todos os sites encontrados, considerando os vinte primeiros resultados e eliminando links repetidos.

São considerados links repetidos todos aqueles que levam a mesma página. Por exemplo: www.site.com/index, www.site.com/index.htm e www.site.com. Os três endereços resultam no mesmo local, logo é armazenada somente uma entrada na lista para este site e qualquer uma das variações que possa aparecer dele será computada para

essa entrada. Porém, se os links forem muito diferentes e mesmo assim levarem ao mesmo documento, haverá duplicação.

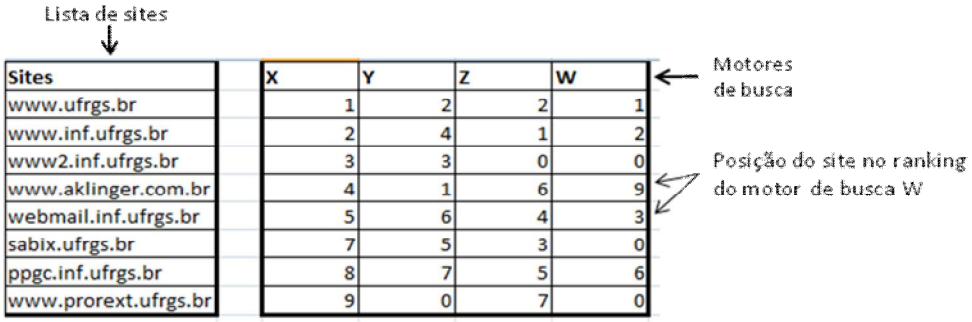
Afim de encontrar casos de multiplos endereços resultando na mesma página, ao efetuar as comparações entre endereços são removidos algumas partes listadas a seguir:

- "http://" e "www.", nos endereços que apresentarem no início;
- "/", "/index.htm", "/index.html" e "/index", nos endereços que apresentarem no final.

Para melhorar a eficiência e qualidade do *ranking* final, alguns sites ainda são removidos da lista, isso explica o fato de ter, na Figura 11, posições faltando entre intervalos. Os critérios se baseiam em:

- Tamanho da URL muito grande. Indica uma provável concatenação de URLs feita por algum motor de busca ou ainda um erro de geração dinâmica do link.
- Site encontrado em apenas um *ranking*. A classificação desse site não ficaria boa, uma vez que levaria zeros de todos os demais *rankings* no momento da fusão. Portanto é considerado desprezível para o resultado final, visando-se os melhores.

A partir da lista de sites é feita a matriz que será usada para a fusão. Cada site da lista corresponde a uma linha da matriz e cada motor de busca é representado por uma coluna. Em cada elemento da matriz vai a posição em que o site representado por aquela linha obteve no motor de busca correspondente a sua coluna. Quando um site não aparece no *ranking* de um motor de busca a célula correspondente recebe o valor zero. A Figura 11 ilustra o parágrafo.



Sites	X	Y	Z	W
www.ufrgs.br	1	2	2	1
www.inf.ufrgs.br	2	4	1	2
www2.inf.ufrgs.br	3	3	0	0
www.aklinger.com.br	4	1	6	9
webmail.inf.ufrgs.br	5	6	4	3
sabix.ufrgs.br	7	5	3	0
ppgc.inf.ufrgs.br	8	7	5	6
www.prorext.ufrgs.br	9	0	7	0

Figura 11. Ilustração da composição da matriz para a fusão

A matriz montada agora está pronta para ser passada para o pacote do MDPREF. Após realizada a fusão o pacote *mdpref*[9] irá devolver um *ranking* resultante. Detalhes de como é feita a fusão se encontram no trabalho de Elmário G. Dutra [7].

Um esquema geral do funcionamento do sistema de metabusca proposto é apresentado na Figura 12, resumindo este capítulo.

[9] *mdpref* é o nome do pacote que implementa o método MDPREF. Sempre que o nome for minúsculo é uma referência ao pacote Java.

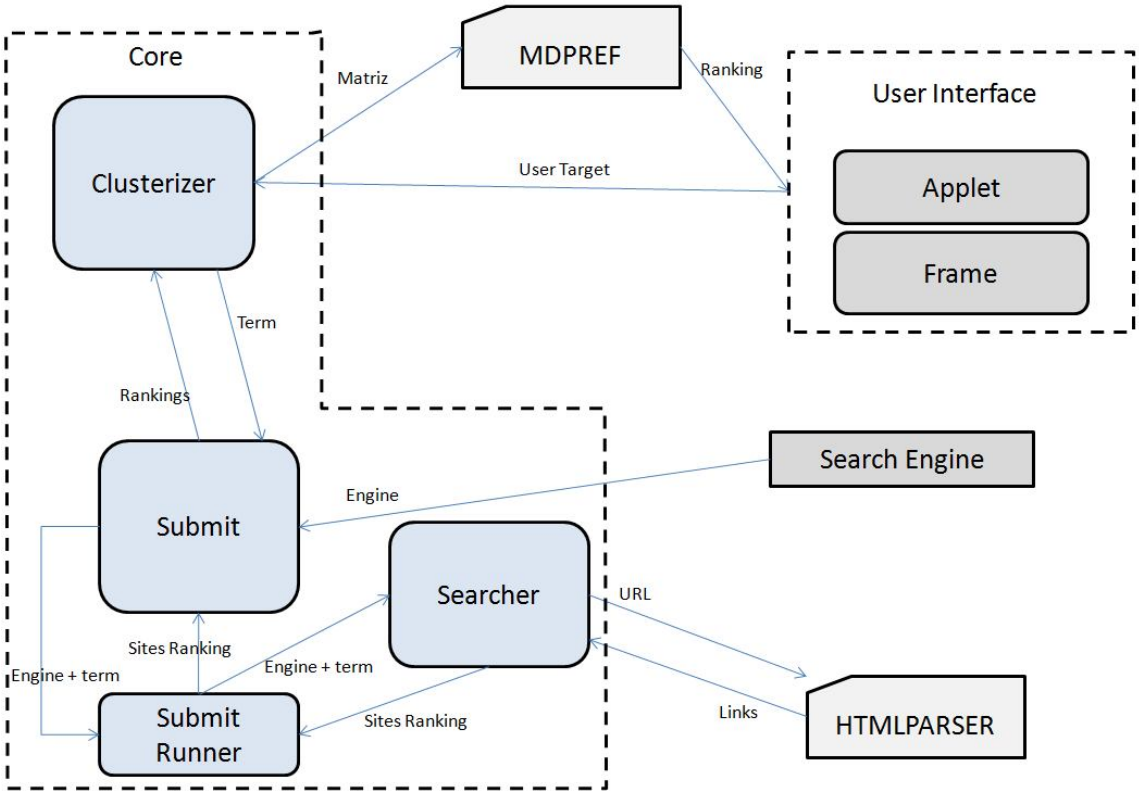


Figura 12. Esquema do projeto do sistema

4.4 Tecnologias

A contrução do sistema foi feita com o uso das tecnologias apresentadas na Tabela 2.

Tabela 2. Tecnologias envolvidas na implementação do sistema

Categoria	**Tecnologia**	**URL**
Ambiente de Desenvolvimento	NetBeans 6.5, NetBeans 6.7	http://www.netbeans.org/
Linguagem de Programação	Java	http://java.sun.com
Biblioteca MDPREF	mdpref package	http://www.inf.ufrgs.br/~aklinger
Biblioteca Matricial	JAMA	http://math.nist.gov/javanumerics/jama/
Biblioteca de *Parsing*	HTML Parser	http://htmlparser.sourceforge.net/

5 Avaliação do Sistema de Metabusca proposto

Para avaliar o sistema proposto foram feitas comparações com outros metabuscadores, todos eles apresentados na sessão de trabalhos relacionados (3). Foram utilizados para comparação somente aqueles que dispunham da informação de quais motores de busca utilizam. As medidas utilizadas foram algumas das distâncias apresentadas na seção 2.5.

Com o objetivo de realizar uma comparação mais justa, os motores de busca do sistema aqui proposto, que não fazem parte da metabusca dos outros a que foi comparado, foram zerados. Assim, pode-se comparar somente o resultado da fusão, sendo que o universo de onde foram extraídos os dados são os mesmos.

A comparação foi feita com o Ixquick, iBoogie e Dogpile. Cada um deles utilizando os seguintes motores de busca:

- Ixquick: Ask, AllTheWeb, AltaVista, EntireWeb, Yahoo;
- iBoogie: Ask, AllTheWeb, AltaVista, Bing, Yahoo;
- Dogpile: Ask, Bing, Google, Yahoo.

Foram considerados somente os dez primeiros resultados dos *rankings*. Os termos utilizados foram os seguintes:

- Termo comum: metasearch
- Termo composto (dois termos disintos designando um alvo): informatica ufrgs
- Termo comum composto (com aspas): “pink floyd”

5.1 Testes

Os termos foram buscados em cada um dos três metabuscadores citados acima, além do sistema baseado no MDPREF proposto neste trabalho. Nas sub-sessões abaixo podemos ver os *rankings* gerados por cada metabuscador, separados por termo, em comparação aos *rankings* gerados pelo MDPREF, de acordo com os motores de busca usados pelos metabuscadores de terceiros.

5.1.1 Ixquick vs. MDPREF

Utilizando cinco motores de busca: Ask, AllTheWeb, AltaVista, EntireWeb e Yahoo, foram obtidos os rankings apresentados nas Tabelas 3, 4 e 5. Apesar do Ixquick possibilitar o uso de mais motores de busca além destes, a pesquisa foi restrita a eles por serem comuns entre os dois metabuscadores.

Tabela 3. Rankings do Ixquick e MDPREF para termo: metaserach.

	Ixquick	**MDPREF**
1	www.metasearch.com	www.mamma.com
2	www.ixquick.com	www.metasearch.com
3	www.tay.com.br	www.ixquick.com
4	www.ohloh.net/tags/metasearch	www.dogpile.com
5	www.arch.com.br/produtos/detector/...	www.lib.berkeley.edu/TeachingLib/...
6	www.mamma.com	en.wikipedia.org/wiki/Metasearch_...
7	www.dogpile.com	www.metacrawler.com
8	www.exl.com.br/metalib_metasearch	www.allmetasearch.com
9	en.wikipedia.org/wiki/Metasearch_...	www.dogpile.com/dogpile/ws/...
10	pesquisa.b-on.pt/ V/ STFC8FL...	www.answers.com/topic/metasearch...

Pela Tabela 3 pode-se ver que o resultado da classificação para a busca do termo *metasearch* foi divergente, sendo que apenas cinco sites figuram nos dois *rankings*, e em posições não coincidentes. Em casos como esse, utiliza-se a operação de *completamento de rankings*, e então pode-se calcular a distância entre eles. Sendo assim, cada *ranking* fica com quinze elementos, sendo que o numero de pares discordantes (distância de Hamming) é exatamente quinze.

Tabela 4. Rankings do Ixquick e MDPREF para os termos: informatica ufrgs

	Ixquick	**MDPREF**
1	www.inf.ufrgs.br	www.inf.ufrgs.br/en/index.php?...
2	penta.ufrgs.br/edu/home_edu.htm	citeseer.ist.psu.edu/cis?q=Fl%E1vio+...
3	penta2.ufrgs.br/edu/home_edu.htm	www.inf.ufrgs.br/
4	www.niee.ufrgs.br	opera.inrialpes.fr/CEMT/Workshop/C...
5	www.baguete.com.br/noticias...	wikimapia.org/144650/pt/Instituto...
6	pt.wikipedia.org/wiki/Instituto_de...	www-gppd.inf.ufrgs.br/~pilla/home.html
7	www.pgie.ufrgs.br	msdn90.e-academy.com/elms...
8	www.famed.ufrgs.br	wcga05.lncc.br/text/7482.pdf
9	www.sbc.org.br/index.php?language...	sites.google.com/site/jvflimahome/
10	www.seer.ufrgs.br/index.php/rita/...	www.aaai.org/Papers/AAAI/2007...

Buscando os termos *informatica ufrgs*, os resultados foram ainda mais contrastantes entre Ixquick e MDPREF. Somente um site figura nos dois *rankings*, justamente o do Instituto de Informática da UFRGS, como mostra a Tabela 4. A Distância de Hamming entre os *rankings* é dezenove.

Tabela 5. Rankings do Ixquick e MDPREF para o termo: “pink floyd”.

	Ixquick	**MDPREF**
1	www.cifras.com.br/pink-floyd	www.pinkfloyd.co.uk
2	www.lastfm.com.br/music/Pink+Floyd	en.wikipedia.org/wiki/Pink_Floyd
3	www.youtube.com/watch?v=A1wamMLHmhU	www.pinkfloyd.com
4	lyricskeeper.com.br/pt/pink-floyd/hey-you.html	www.pinkfloyd-co.com
5	musica.terra.com.br/ interna/ 0...	www.rhapsody.com/pink-floyd
6	www.flickr.com/search/?q=PinkFloyd	www.pinkfloyd.net
7	pt.wikipedia.org/wiki/Pink_Floyd	www.last.fm/music/Pink+Floyd
8	www.whiplash.net/materias/curiosidades/064483	www.pinkfloydonline.com
9	www.myspace.com/aknatha	www.pinkfloydz.com
10	www.pinkfloyd.com	www.pinkfloyd.co.uk/echoes

Finalmente, no último teste com o sistema Ixquick, com o termo “*pink floyd*” (Tabela 5), foram obtidos apenas dois sites iguais nos *rankings*, o que resulta numa Distância de Hamming igual a dezoito, mostrando que a classificação usada pelo Ixquick é bastante diferente da gerada pelo método MDPREF.

5.1.2 iBoogie vs. MDPREF

O iBoogie possibilita a busca em sete motores de busca diferentes, porém as buscas foram restritas a cinco deles, que são comuns ente ele e o metabuscador baseado no MDPREF. Os buscadores utilizados foram: Ask, AllTheWeb, AltaVista, Bing e Yahoo.

Os *rankings* do Bing foram inseridos diretamente no código, uma vez que o buscador não apresenta a característica geral dos motores de busca suportados pelo sistema implementado, já que não possui uma classe CSS específica para os resultados.

Tabela 6. Rankings do iBoogie e MDPREF para o termo: metasearch

	iBoogie	**MDPREF**
1	metasearch.com	www.mamma.com
2	en.wikipedia.org/wiki/Metasearch_engine	metasearch.com
3	www.mamma.com	en.wikipedia.org/wiki/Metasearch_...
4	www.lib.berkeley.edu/TeachingLib...	www.ixquick.com
5	www.dogpile.com	www.dogpile.com
6	www.ixquick.com	www.lib.berkeley.edu/TeachingLib...
7	ixquick.com	www.metacrawler.com
8	www.metacrawler.com	www.answers.com/topic/metasearch...
9	www.dmoz.org/Computers/Internet/...	www.dmoz.org/Computers/Internet/...
10	www.dogpile.com/dogpil...ch/_...	www.2trom.com

No primeiro teste com o metabuscador iBoogie, Tabela 6, nove sites aparecem nos dois *rankings*, sendo que dois deles estão inclusive na mesma posição. Pode-se perceber que o iBoogie não elimina ambiguidades, pois as posições seis e sete de seu *ranking* resultante referenciam o mesmo site. A distância de Hamming entre os *rankings* é igual a nove.

Tabela 7. Rankings do iBoogie e MDPREF para os termos: informatica ufrgs

	iBoogie	**MDPREF**
1	www.inf.ufrgs.br	www.inf.ufrgs.br/en/index.php?option=..
2	www.inf.ufrgs.br/en/ index.php?option=...	citeseer.ist.psu.edu/cis?q=Fl%E1vio+...
3	inf.ufrgs.br/en	opera.inrialpes.fr/CEMT/Workshop/Ce...
4	inf.ufrgs.br/~erika	www.inf.ufrgs.br
5	www.linkedin.com/group...273...	msdn90.e-academy.com/ e...ont /Home...
6	citeseer.ist.psu.edu/cis?q=Fl%E1vio+...	wikimapia.org/144650/pt/Instituto...
7	scratchpad.wikia.com/wiki/Informatica...	www-gppd.inf.ufrgs.br/~pilla/home.html
8	opera.inrialpes.fr/CEMT/Workshop/Ce...	wcga05.lncc.br/text/7482.pdf
9	en.wikipedia.org/wiki/UFRGS	sites.google.com/site/jvflimahome
10	msdn90.e-academy.com/e...ont/Home...	www.aaai.org/Papers/AAAI/2007/A...

Na Tabela 7 pode-se ver novamente, para o termo buscado, dois *rankings* completamente diferentes, mas com cinco elementos em comum. Distância de Hamming entre eles é de quinze.

Tabela 8. Rankings do iBoogie e MDPREF para o termo: “pink floyd”.

	iBoogie	**MDPREF**
1	www.pinkfloyd.com	www.pinkfloyd.com
2	www.pinkfloyd.co.uk	en.wikipedia.org/wiki/Pink_Floyd
3	en.wikipedia.org/wiki/Pink_Floyd	www.pinkfloyd-co.com
4	www.pinkfloyd.net	www.pinkfloyd.net
5	www.pinkfloyd-co.com	www.pinkfloyd.co.uk
6	www.last.fm/music/Pink+Floyd	www.myspace.com/pinkfloyd
7	www.pinkfloydonline.com	www.rhapsody.com/pink-floyd
8	www.vh1.com/artists/az/pink_floyd/bio	www.pinkfloyd.co.uk/echoes/
9	www.pinkfloyd.net/?story=1191831521	www.pinkfloydonline.com
10	rdre1.yahoo.com/click?...	en.wikipedia.org/wiki/Pink%20Floyd?...

No teste com o termo “*pink floyd*” (Tabela 8) foram obtidos seis sites figurando nos dois *rankings*, e dois deles concordando em posição, o que resulta em uma distância de Hamming igual a doze. Interessante notar a concordância na primeira posição.

Pode-se ver claramente, pelos dados apresentados, que a classificação do iBoogie é bastante diferente da classificação feita pelo MDPREF.

5.1.3 Dogpile vs. MDPREF

O Dogpile utiliza quatro motores de busca: Ask, Bing, Google e Yahoo. O sistema implementado foi restrito a eles e os resultados comparativos são mostrados nas Tabelas 9, 10 e 11. Novamente os *rankings* do buscador Bing foram coletados no navegador e inseridos diretamente no código, a respeito do que foi feito com o iBoogie.

Tabela 9. Rankings do Dogpile e MDPREF para o termo: metasearch

	Dogpile	**MDPREF**
1	www.metasearch.com	www.mamma.com
2	www.mamma.com	en.wikipedia.org/wiki/Metasearch_...
3	www.metacrawler.com	www.metasearch.com
4	en.wikipedia.org/wiki/Metasearch_...	www.ixquick.com
5	www.lib.berkeley.edu/TeachingLib...	www.lib.berkeley.edu/TeachingLib/...
6	www.ixquick.com	www.dogpile.com
7	www.dmoz.org/Computers/Internet/...	www.metacrawler.com
8	www.searchengineshowdown.com/multi	www.pandia.com/metasearch/index.html
9	searchenginewatch.com/2156241	metasearch.langenberg.com
10	ixquick.com/uk	searchenginewatch.com/2156241

Com o termo *metasearch* (Tabela 9) foram encontrados sete sites em comum entre os *rankings* gerados pelos dois metabuscadores. Como há somente uma posição concordante, a Distância de Hamming entre os *rankings* é doze.

Tabela 10. Rankings do Dogpile e MDPREF para os termos: informatica ufrgs.

	Dogpile	**MDPREF**
1	www.inf.ufrgs.br	wikimapia.org/144650/pt/Instituto...
2	scratchpad.wikia.com/wiki/Informatica...	www.inf.ufrgs.br
3	www.inf.ufrgs.br/en/index.php?option=...	www.librarything.com/venue/21483/...
4	www.linkedin.com/groupInvitation?...	ppgc.inf.ufrgs.br/index.php?option=...
5	www.librarything.com/venue/21483/...	www.baguete.com.br/noticiasDetalh...
6	inf.ufrgs.br/~erika	www.dimap.ufrn.br/pipermail/logica...
7	wikimapia.org/144650/pt/Instituto...	scratchpad.wikia.com/wiki/Informatica...
8	www.curumin.uwaterloo.ca/~gvgbaran/...	ppgc.inf.ufrgs.br
9	en.wikipedia.org/wiki/UFRGS	www.comentarium.com.br/site.jsp?...
10	www.ufrgs.br/psicoeduc/informatica	leitura.com/descricao.php?id=460678...

No teste com os termos *informatica ufrgs*, visto na Tabela 10, quatro sistes apareceram em ambos os *rankings*, cuja Distância de Hamming é igual a dezesseis.

Tabela 11. Rankings do Dogpile e MDPREF para o termo: “pink floyd”.

	Dogpile	**MDPREF**
1	www.pinkfloyd.co.uk	www.pinkfloyd.com
2	www.pinkfloyd.com	en.wikipedia.org/wiki/Pink_Floyd
3	en.wikipedia.org/wiki/Pink_Floyd	www.pinkfloyd.net
4	www.pinkfloyd.net	www.pinkfloyd-co.com
5	www.pinkfloyd-co.com	www.youtube.com/watch?v=M_bvT-...
6	www.youtube.com/watch?v=M_bvT-...	www.pinkfloydonline.com
7	www.last.fm/music/Pink+Floyd	pt.wikipedia.org/wiki/Pink_Floyd
8	www.pinkfloydonline.com	letras.terra.com.br/pink-floyd
9	www.vh1.com/artists/az/pink_floyd/bio	www.last.fm/music/Pink+Floyd
10	www.pinkfloyd.net/?story=1191831521	www.youtube.com/watch?v=tkJNyQf...

Por fim, na Tabela 11, não houve concordância segundo a classificação de nenhum site para o termo "*pink floyd*", mas sete sites figuram nos dois *rankings*, resultando numa Distância de Hamming igual a treze.

Mais uma vez, observa-se *rankings* bastante divergentes entre os dois sistemas de metabusca.

5.2 Avaliação de Resultados

Visivelmente o modelo de fusão de *rankings* baseado no MDPREF gera uma classificação diferente da gerada pelos sistemas a que foi comparado. A Tabela 12, abaixo, mostra o valor das Distâncias de Hamming do *ranking* gerado pelo metabuscador que faz uso do MDPREF aos *rankings* gerados pelos metabuscadores de terceiros testados.

Tabela 12. Comparativo de Distâncias de Hamming.

	metasearch	informatica ufrgs	"pink floyd"
Ixquick	**15**	**19**	**18**
iBoogie	**9**	**15**	**12**
Dogpile	**12**	**16**	**13**

O iBoogie retornou *rankings* mais próximos ao do MDPREF, mas ainda assim expressivamente divergentes, e também este sistema retornou a maior quantidade de sites que também constam nos *rankings* do MDPREF.

Quando usadas mais de duas palavras na consulta, buscando obter um resultado mais específico, a exemplo dos testes com os termos *informatica ufrgs*, esperando-se obter sites relacionados ao setor de informática da UFRGS, os resultados tendem a divergir mais ainda, sendo que no Ixquick a distância entre os *rankings* quase atingiu o valor máximo, vinte.

Não se pode dizer qual é o melhor *ranking* para dado termo pois é algo que depende muito do critério adotado por cada sistema metabusca. Os sites mais bem posicionados podem ser os mais acessados, os mais refenciados ou os que mais aparecem entre os *rankings.* A comparação feita aqui somente mostra a diferenciação e abrangência do metabuscador usando MDPREF.

Uma comparação entre os resultados obtidos pelo metabuscador proposto, obedecendo as restrições, e com todos os motores de busca abilitados e com pesos iguais, pode ser feita com as tabelas dos *rankings* no Anexo A. Assim, pode-se ter uma idéia de como uma maior abrangência influência no resultado final.

6 Conclusão

Identificar o quanto um motor de busca é melhor que o outro não é uma tarefa simples, pois depende de avaliações subjetivas. De acordo com o termo pesquisado, o *ranking* apresentado pode ser melhor ou pior em um sistema de busca x. A metabusca tenta minimizar o problema aumentando a abrangência, uma vez que leva em conta não só o sistema de classificação dos resultados próprio, mas inclui todos os métodos adotados pelos motores de busca individualmente.

Quanto mais motores de busca, maior a diversidade de critérios e conteúdo. O sistema aqui apresentado, que agrega dezessete buscadores, possui uma grande abrangência, visto que nenhum dos metabuscadores pesquisados usufrui de tantas fontes, quando foi possivel a identificação de tal dado. O resultado é que obtém-se os sites mais conhecidos da Internet, dando a idéia do nível de publicidade no mundo de cada site presente no *ranking* resultante da fusão. Ou seja, o que é apresentado é a visão dos motores de busca com relação ao termo alvo.

De acordo com o MDPREF, nem sempre o site que aparece em mais *rankings* ocupa uma posição melhor no *ranking* final. Através de testes percebe-se que, por exemplo, um site que aparece em seis motores de busca mas tira três primeiros lugares fica melhor posicionado que um site que possa ter figurado entre todos os motores de busca em posições de pouco destaque.

Durante as comparações com outros sistemas de metabusca, ficou clara a diferença dos critérios levados em conta, por diferentes metabuscadores, no momento de fundir os *rankings* gerados pelos motores de busca. Ao final, o MDPREF mostrou-se um bom método para o cenário de metabusca, já que produz *rankings* diferenciados daqueles gerados pelos sistemas de metabusca atuais encontrados na Web.

6.1 Trabalhos Futuros

A pesquisa realizada proporcionou a abertura para alguns trabalhos futuros:

- Estender o numero de motores de busca empregados, ampliando os critérios de inclusão;
- Transpôr o sistema para a plataforma JEE[10] e otimizar o tempo de resposta;
- Estudar a fundo as características de cada motor de busca afim de distribuir os pesos entre eles de acordo com a pesquisa desejada;
- Restringir as buscas a domínios específicos, onde especialistas podem apontar um *ranking* ideal, para poder medir a qualidade do *ranking* resultante;
- Buscar os resultados mais improváveis para o termo buscado, realizando a fusão com *rankings* invertidos, visando mostrar locais inimaginados vinculados ao termo desejado;
- Eliminar resultados repetidos, evitando a multiplicidade de links distintos que levam ao mesmo documento;

[10] Plataforma Java para Web

- Levar em conta a similaridade de termos na busca, isto é, por exemplo, saber que para o termo “Jose Valdeni de Lima”, as ocorrências de “J.V. de Lima” também são válidas;
- Implementar o suporte a operadores booleanos;
- Implementar o suporte a acentos;
- Estudar um novo *ranking*, levando em conta os sites que aparecem acima da vigésima posição e também aqueles que aparecem somente uma vez no cluster;
- Responder a pergunta: na metabusca, é melhor aparecer em muitos motores de busca ou aparecer no topo de poucos?

7 Bibliografia

[1] BLATTMANN, U.; FACHIN, G. R. B.; RADOS, G. J; V. **Recuperar a Informação Eletrônica pela Internet.** Revista da ACB**,** Florianópolis, v.4, no. 4, p. 9-27, 1999. Disponível em: <http://www.ced.ufsc.br/~ursula/papers/buscanet.html>. Acesso em: 2 abr. 2009.

[2] CENDÓN, B. V. Ferramentas de busca na *Web*. **Ciência da Informação**, Brasília, v.30, no.1, 2001. Disponível em: <http://www.scielo.br>. Acesso em: 2 abr. 2009.

[3] **CLUSTY**. *About* do site. Disponível em: <http://www.clusty.com>. Acesso em: 5 abr. 2009.

[4] **DOGREATGOOD**. *About* do site. Disponível em: <http://www.dogreatgood.com>. Acesso em: 6 abr. 2009.

[5] **DOGPILE**. *About* do site. Disponível em: <http://www.dogpile.com>. Acesso em: 5 abr. 2009.

[6] DUNN-RANKIN, Peter et al. **Scaling Methods.** 2ª ed. Lawrence Erlbaum, 2004. 221p. Cap 13: Mapping Individual Preference.

[7] DUTRA, E. G. J. **Um Modelo de Fusão de *Rankings* Baseado em Análise de Preferência.** 2008. 74 f. Dissertação (Mestrado) – Ciência da Computação, UFRGS, Porto Alegre. 2008.

[8] DUTRA, E. G. J.; LIMA, J. V. Supplement of partial ranks to the data fusion. In: BRAZILIAN SYMPOSIUM ON MULTIMEDIA AND THE WEB, WEBMEDIA, 12., 2006, Natal, Rio Grande do Norte. **Proceedings…** New York: ACM, 2006. p. 148-154.

[9] DWORK, C et. al. **Rank Aggregation Methods for the Web**. WWW10, May 1-5, 2001.

[10] FRAGOSO, S. Quem Procura Acha? O Impacto dos Buscadores dobre o Modelo Distributivo da World Wide Web. **Revista de Economía Política de las Tecnologías de la Información y Comunicación.** Vol. 9, 2007. Disponível em: <http://www.eptic.com.br/arquivos/Revistas>. Acesso em: 5 abr. 2009.

[11] HSU, D. F.; PALUMBO, A. **A Study of Data Fusion in Cayley Graphs G(Sn,Pn)**. In Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks, 2004.

[12] **IBOOGIE.** *About* do site. Disponível em: <http://www.iboogie.com>. Acesso em: 5 ago. 2009.

[13] **IXQUICK**. *About* do site. Disponível em: <http://www.ixquick.com>. Acesso em: 5 abr. 2009.

[14] MARDEN, J. I. **Analyzing and Modeling Rank Data**. London: Chapman & Hall, 1995.

[15] **METACRAWLER**. *About* do site. Disponível em: <http://www.metacrawler.com>. Acesso em: 5 ago. 2009.

[16] RENDA, M. E.; STRACCIA, U. **Metasearch:** Rank vs. Score Based Rank List Fusion Methods (without Training Data). Pisa: Instituto di Elaborazione della Informazione, 2002.

[17] TOUS, R.; DELGADO, J. Advanced Meta-Search of News In The Web. VWF Berlim, 2002.

[18] WIKIPEDIA. **List of Search Engines**. Disponível em: <http://en.wikipedia.org/>. Acesso em: 1 abr. 2009.

[19] WIKIPEDIA. **Ranking.** Disponível em: <http://en.wikipedia.org/>. Acesso em: 10 Jul. 2009.
[20] WIKIPEDIA. **Web Search Engine.** Disponível em: <http://en.wikipedia.org/>. Acesso em: 2 abr. 2009.
[21] **ZWROKS**. *About* do site. Disponível em: <http://www.zworks.com>. Acesso em: 10 out. 2009.

ANEXO A
RANKINGS GERADOS PELO SISTEMA FINAL

Nas Tabelas 13, 14 e 15 são apresentados os *rankings* gerados pelo sistema de metabusca implementado contando com todos os dezessete motores de busca, de acordo com os termos do capítulo 5.

Tabela 13. *Ranking* final pelo MDPREF para termo: metasearch

1	www.mamma.com
2	en.wikipedia.org/wiki/Metasearch_engine
3	www.metasearch.com
4	www.dogpile.com
5	www.lib.berkeley.edu/TeachingLib/Guides/Internet/MetaSearch.html
6	www.ixquick.com
7	www.metacrawler.com/
8	www.pandia.com/metasearch/index.html
9	metasearch.langenberg.com
10	searchenginewatch.com/2156241

Tabela 14. *Ranking* final pelo MDPREF para termo: informatica ufrgs

1	www.inf.ufrgs.br
2	wikimapia.org/144650/pt/Instituto-de-Inform%C3%A1tica-UFRGS
3	ppgc.inf.ufrgs.br
4	scratchpad.wikia.com/wiki/Informatica_UFRGS
5	ppgc.inf.ufrgs.br/index.php?option=com_content&view=category...
6	www.inf.ufrgs.br/en/index.php?option=com_content&view=articl...
7	citeseer.ist.psu.edu/cis?q=Fl%E1vio+Rech+Wagner
8	opera.inrialpes.fr/CEMT/Workshop/Cemt.ppt
9	msdn90.e-academy.com/elms/Storefront/Home.aspx?campus=ufga_info
10	www.baguete.com.br/noticiasDetalhes.php?id=3511330

Tabela 15. *Ranking* final pelo MDPREF para termo: “pink floyd

1	www.pinkfloyd.com
2	www.pinkfloyd.net
3	en.wikipedia.org/wiki/Pink_Floyd
4	www.pinkfloyd-co.com
5	www.pinkfloydonline.com
6	www.last.fm/music/Pink+Floyd
7	www.pinkfloyd.co.uk
8	www.rhapsody.com/pink-floyd
9	www.myspace.com/pinkfloyd
10	www.pinkfloyd.co.uk/echoes