Uma Proposta para o Uso de Detecção de Versões de Páginas Web para Melhoria do Desempenho do Algoritmo de *PageRank**

Glauber Rodrigues da Silva, Renata de Matos Galante, Deise de Brum Saccol

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{grsilva,renata,deise}@inf.ufrgs.br

Abstract. *In order to improve the document ranking, search engines are based on some characteristics, such as: use of keywords, analysis of page content, analysis of link structure, attribution of popularity scores, and so on. However, new versions of top-ranked pages may not appear in the top of the list, since they are not detected a priori. To solve this problem, this paper considers the use of page version detection for performance improvement of the PageRank, a well-known algorithm in the information retrieval area. The paper also presents some experiments performed with the Nutch engine.*

Resumo. *A fim de melhorar o ranking de documentos, os motores de busca baseiam-se em algumas características, tais como: uso de palavras-chave, análise de conteúdo das páginas, análise de estrutura de links, atribuição de escores de popularidade, entre outros. No entanto, novas versões de páginas bem ranqueadas podem não aparecer no topo da lista, uma vez que não são detectadas a priori. Para solucionar este problema, este artigo propõe o uso de detecção de versões de páginas Web para melhoria do desempenho do PageRank, um conhecido algoritmo na área de recuperação de informação. O artigo também apresenta alguns experimentos realizados no motor de busca Nutch.*

1. Introdução

A natureza distribuída das informações disponíveis na Internet levou à busca constante de maneiras eficientes de executar consultas sobre uma grande coleção de documentos. Os motores de busca para *Web* realizam essa tarefa, porém, devido ao grande tamanho da coleção, o número de documentos relevantes para uma consulta pode facilmente ter milhares de itens. Devido a essa quantidade de informação, os algoritmos utilizados na área de recuperação de informação tradicional não obtiveram resultados satisfatórios, visto que realizavam uma análise sobre o conteúdo das páginas para montar um *ranking*. A estrutura de *links* comum nos documentos HTML fornece uma maneira de estimar quais são as páginas mais "populares", partindo-se da premissa de que quanto maior o número de *links* que apontam para uma certa página (*links* esses vindos também de páginas populares), mais popular essa página é.

Os algoritmos que levam em conta essa estrutura de *links* na montagem do *ranking* dos resultados de uma pesquisa são chamados de algoritmos de análise de *links*. O algoritmo de *PageRank* [Page, Brin, Motwani and Winograd 1998] é o que mais tem destaque nesta família de algoritmos. O termo *PageRank* é comumente atribuído a um

* Este trabalho foi parcialmente suportado pelo projeto Pronex FAPERGS número 0408933.

número que diz o quanto uma página *Web* é relevante na Internet, ou seja, quanto maior esse número, maior será a relevância da página.

No entanto, a velocidade em que a estrutura da Internet se modifica acaba por trazer alguns problemas ao algoritmo de *PageRank*. Novas versões de uma página *Web* com alto grau de relevância não irão ter o mesmo grau de relevância da página original, mesmo se tratando de versões de um mesmo documento. Por exemplo, considerando uma página *Web p,* um fator importante para constituir o *ranking* pelos motores de busca é a quantidade de outras páginas que apontam para *p.* Contudo, novas versões desta página terão uma baixa taxa de *links* que apontam para elas (*links* entrantes), pois as páginas que apontam para *p* não sabem da existência de suas novas versões. Nesse contexto, a detecção de versões pode melhorar o posicionamento no *ranking* de versões baseado nos *links* que apontam para *p.*

Neste contexto, o objetivo deste trabalho é especificar uma proposta de um mecanismo para detecção automática de versões de páginas *Web*, visando melhorar o cálculo do *PageRank*. Pretende-se fazer com que o grau de relevância de uma página seja computado, considerando o escore de suas versões antigas, mantendo assim, o grau de importância que a página obteve ao longo do tempo. Em outras palavras, isto significa que uma nova versão da página ou a alteração de sua localização não causa prejuízo ao seu posicionamento no *ranking* de resultados do motor de busca. Para validar a proposta foram realizados experimentos com o motor de busca de código aberto *Nutch* [Nutch 2007]. A principal contribuição desse trabalho é apresentar uma proposta para o uso de detecção de versões de páginas *Web* para melhoria do desempenho do algoritmo de *PageRank*.

O texto deste trabalho está organizado da seguinte forma: na seção 2 é apresentada a base conceitual para a compreensão do funcionamento dos motores de busca para *Web.* Na seção 3, são apresentados os trabalhos relacionados. Na seção 4, são apresentados os experimentos realizados com o *Nutch* e uma proposta de detecção de versões de páginas *Web* para a melhoria de performance do algoritmo de *ranking* utilizado pelo *Nutch*. Por fim, a seção 5 traz as conclusões do trabalho e trabalhos futuros.

2. Base Conceitual: Arquitetura dos Motores de Busca para *Web*

Nesta seção é apresentada a arquitetura dos motores de busca para *Web,* servindo como base para a implementação no *Nutch*, descrita na seção 5. Com o entendimento de cada componente da arquitetura, é possível definir a proposta de modificação dos componentes visando a melhoria do algoritmo de *PageRank*, para que o mesmo passe a considerar versões na atribuição de escore às páginas.

Os motores de busca para *Web* são a personificação de alguns conceitos da área de recuperação de informações para o ambiente de Internet. A Figura 1 [Langville and Meyer 2006] ilustra os elementos de um motor de busca para *Web*, na qual é possível observar dentro da área demarcada os componentes/processos que pertencem à organização interna do motor de busca que podem ser executados/atualizados a qualquer momento. Já os componentes dependentes de consulta são os únicos que são perceptíveis ao usuário e interagem diretamente com o mesmo.

Os principais elementos de um motor de busca são detalhados a seguir.

- **Crawler** - Web *Crawling* é o processo utilizado pelos motores de busca para vasculhar a Internet com o objetivo de fazer *download* das páginas Web para processamentos posteriores, tais como indexação, análise de *links*, entre outros. O processo de *webcrawler* surgiu com os primeiros motores de busca que eram

agressivos ao ponto de buscar as informações dos sites para indexá-las. O software de *crawling* cria robôs, chamados *spiders*, que são os elementos que executam a tarefa de navegação buscando o maior número de páginas possível. A literatura sobre Web *crawler* é abundante e fornece subsídios ao surgimento de várias implementações, inclusive de código aberto, para os algoritmos propostos.

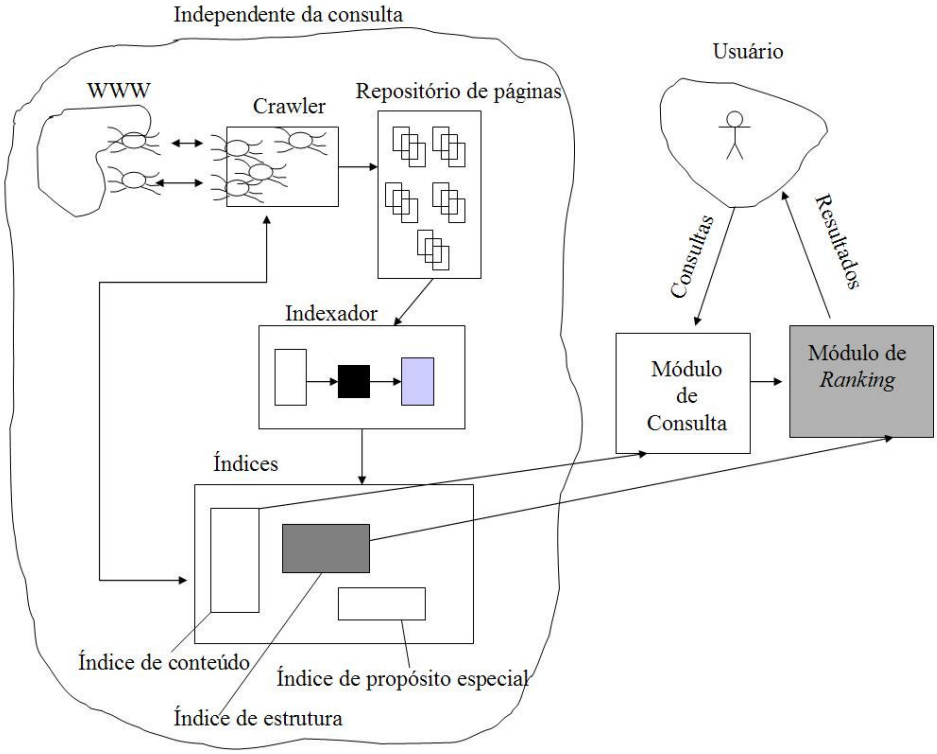

Figura 1. Elementos de um motor de busca para *Web*

- **Repositório de Páginas** - Assim que os *spiders* fazem *download* das novas páginas, todo o seu conteúdo é armazenado no repositório de páginas. As páginas permanecem ao repositório até que sejam consumidas pelo indexador. Atualmente, alguns motores de busca não descartam o conteúdo das páginas, mesmo depois de indexadas, propiciando aos seus usuários um mecanismo de cache de visualização, caso as páginas originais não estejam mais disponíveis.
- **Indexador** - O indexador examina cada nova página no repositório de páginas e extrai as principais informações da mesma, visando criar uma descrição compacta para armazenar em vários índices.
- **Índices** - Os índices contêm informações importantes sobre cada página e podem ter diferentes funções e formas de armazenamento. O índice de conteúdo contém informações textuais, tais como: título, conteúdo e palavras-chave. Essas informações são armazenadas de forma comprimida. O índice de estrutura armazena as informações que dizem respeito ao grafo montado pelas páginas (vértices) e os *links* entre elas (arestas). Podem existir ainda índices com propósitos especiais, como: índice de imagens, índice de documentos binários, entre outros.
- **Módulo de Consulta** - O módulo de consulta converte a consulta entrada pelo usuário em linguagem natural em uma consulta que o motor de busca pode compreender, e consulta os vários índices para respondê-la. Usualmente é

consultado o índice de conteúdo e esse por sua vez consulta um arquivo invertido para saber quais páginas são relevantes para a consulta feita pelo usuário. Essa coleção obtida pelo módulo de consulta é repassada ao módulo de *ranking*.

- **Módulo de Ranking** - O módulo de *ranking* recebe a lista de páginas relevantes do módulo de consultas e as ordena levando em conta algum critério. A lista ordenada traz nas primeiras posições os documentos que possivelmente satisfazem melhor o desejo do usuário. Em um motor de busca para Web, o mecanismo de *ranking* acaba se tornando um dos componentes mais importantes, pois os resultados de uma busca freqüentemente têm milhões de itens. Os motores de busca combinam dois valores para montar um escore único para classificação, são eles o escore de conteúdo e o escore de popularidade [Langville and Meyer 2006]. Muitas regras são utilizadas para atribuir o escore de conteúdo de uma página. Por exemplo: se algum argumento da consulta aparece no título ou na descrição da página, essa terá mais pontos do que se o argumento da consulta aparece somente no corpo da página. Já o escore de popularidade pode ser atribuído, por exemplo, analisando-se a estrutura formada pelos *links* das páginas da *Web*.

3. Trabalhos Relacionados

Os primeiros algoritmos para ordenação dos resultados das consultas submetidas a um motor de busca eram dependentes da consulta e realizavam uma análise sobre o conteúdo das páginas para montar um *ranking*. Comumente era verificado se as palavras-chave da busca estavam no título da página, ou estavam nos metadados de descrição da mesma e quantas ocorrências existiam das palavras-chave no conteúdo da página. De posse desses valores, era atribuído um escore a cada página que satisfazia os critérios da consulta e se montava a ordenação com base nos escores obtidos.

Com o crescimento substancial do número de documentos na *Web*, o processamento do total do escore em tempo de execução da consulta acabou se tornando inviável, podendo tornar o tempo de espera do usuário muito maior do que o esperado. Além disso, a ordenação baseada somente no conteúdo das páginas não colocava no topo do *ranking* as páginas esperadas pelo usuário.

Em 1997, surgem os primeiros algoritmos que consideravam na computação do escore das páginas a "popularidade" das mesmas. Esse escore era computado através de um modelo de análise de *links*. Esse modelo faz uma análise do grafo, cujos vértices são as páginas e os *links* são as arestas, considerando que um *link* da página A para B é um "voto de popularidade" para a página B. Quanto maior for o número de *links* (vindo de páginas também "populares") que apontam para uma certa página, maiores as chances de ela ter um escore alto. Dentre esses algoritmos o que mais teve destaque foi o *PageRank*.

PageRank por definição é um número que mede a relevância de cada página *Web* na Internet. Isto significa que quanto maior o número, maior será a relevância da página. Diferente do HITS [Keinberg 1999], o algoritmo de *PageRank* é independente de consulta. Dessa forma, é possível ordenar todas, ou quase todas, as páginas da Internet pelo seu grau de relevância, ou ainda, comparar com um conjunto de páginas *Web*.

O algoritmo *PageRank* tem como principal objetivo simular o comportamento de um usuário ao navegar na Internet. Em outras palavras, o número do *PageRank* de uma página diz respeito a quão fácil/difícil seria encontrar esta página partindo de uma URL randômica e, seguindo os *links*, se tentasse chegar a página desejada. Quanto maior o valor

resultante *PageRank*, mais fácil seria encontrar a página procurada [Brin and Page 1998]. Para entender como o *PageRank* funciona, é preciso pensar que um *link* da página A para a página C é um voto que a página A dá para a página C. Esse "voto" tem um peso baseado na relevância que a página A tem.

Para um melhor entendimento da atribuição de escores pelo algoritmo de *PageRank* é preciso considerar o grafo formado pelas páginas *Web* (vértices) e os *links* entre as mesmas (arestas). Esse grafo é comumente chamado de *WebGraph* e um exemplo dele é apresentado na Figura 2.

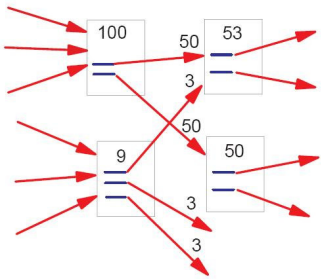

Figura 2. Calculando o *PageRank*

Na Figura 2 foi adicionado o "peso" de cada voto/*link* e é observado que o *PageRank* de cada página é a soma dos pesos atribuídos aos links "entrantes" (Exemplo 50 + 3 = 53). Além disso, é constatado que o *PageRank* das páginas é dividido pelo número de links "saintes" no momento de "computar o voto" a uma página referenciada (Exemplo 9/3 = 3) [Page, Brin, Motwani and Winograd 1998].

Após a publicação do artigo *The PageRank Citation Ranking: Bringing Order to the Web* [Page, Brin, Motwani and Winograd 1998] surgiram vários outros trabalhos propondo técnicas para melhorar a lógica e o desempenho do algoritmo de *PageRank*, comumente chamados de variantes do algoritmo original, pois a idéia central de *ranking* baseado na análise dos *links* é sempre mantida. Um exemplo foi a proposta de variante do algoritmo de *PageRank* que leva em conta os atributos das páginas *Web* [Baeza-yates and Davis 2004]. Em outro trabalho foi proposto um conjunto de algoritmos baseados em *links* (*link-based*), os quais propagam a importância das páginas através dos *links* [Baeza-yates, Boldi and Castillo 2006].

Um outro algoritmo de análise de *links* bastante citado na literatura é o HITS, o qual não atribui um único escore às páginas e sim dois valores, um para *links* que saem da página e outro para *links* apontam para a página (*hub e authority*) e é processado em tempo de consulta e não em tempo de indexação [Keinberg 1999]. Também existe uma proposta de uma combinação dos algoritmos de *PageRank* e HITS para a criação de um *framework* unificado para análise de *links* [Ding, He, Husbands, Zha and Simon 2002].

Outra linha de pesquisa envolve a proposta de variantes do algoritmo de *PageRank* os quais levam em conta a evolução temporal do *webgraph* [Berberich, Bedathur and Weikum 2006]. Existem também trabalhos realizados de análise temporal de wikigrafos, nos quais, dentre várias outras características, são analisados a correlação entre o *PageRank* e o grau de entrada, ao longo de um período de tempo [Buriol, Castillo, Donato, Leonardi and Millozzi 2006].

Apesar de existirem várias pospostas para variantes do algoritmo de *PageRank*, um problema em aberto é o de considerar versões antigas das páginas *Web* para melhorar a

relevâncias das versões atuais. Por exemplo, o *ranking* de um motor de busca envolve a freqüência e a localização das palavras chave em uma página *p*. O motor de busca verifica se as palavras-chave aparecem em *p* e qual a localização delas na página (título, primeiros parágrafos, entre outros) para atribuir um escore de relevância às mesmas, podendo considerar outras técnicas de recuperação de informação. Além disso, um fator importante para constituir o ranking pelos motores de busca é a quantidade de outras páginas que apontam para *p*. Contudo, novas versões de *p* terão uma baixa taxa de *links* que apontam para elas (*links* entrantes), pois as páginas que apontam para *p* não sabem da existência de novas versões de *p*. Nesse contexto, a detecção de versões poderia melhorar o posicionamento no *ranking* de versões de *p* baseado nos *links* que apontam para *p*.

4. O *ranking* do *Nutch* e uma Proposta para sua Melhoria

Esta seção relata os experimentos realizados com o algoritmo de *ranking* utilizado pelo motor de busca *Nutch* [Nutch 2007]. Ao fim da seção é apresentada uma proposta de como o algoritmo funciona quando adaptado para considerar versões de páginas *Web*.

4.1. Nutch

Nutch [Nutch 2007] é um projeto que implementa um motor de busca para Web, totalmente em código aberto. O *Nutch* tem sido largamente utilizado por organizações que queiram implementar dentro do seu ambiente um mecanismo rápido e eficiente para indexação e busca de documentos de diversos formatos dentro de sua Intranet. Por ser de código aberto, permite um alto grau de personalização, tanto na sua indexação, como na interface do usuário.

O *Nutch* tem sido utilizado largamente pela comunidade de código aberto quando o assunto é motor de busca. Por ter o seu código aberto, ser de fácil instalação e utilização (pode ser instalado até mesmo em um computador com baixa capacidade de processamento) e ser baseado num modelo de *plugins*, os desenvolvedores tem facilidade em realizar alterações para adequá-lo às necessidades de cada projeto.

4.2. Criação de um Site Fictício

Para realização dos experimentos foi criado um site com oito páginas que fazem parte de um suposto site de uma instituição acadêmica. Uma das páginas é o índice enquanto as demais são páginas pessoais dos colaboradores da instituição. Todas as páginas pessoais têm o mesmo conteúdo: o nome do colaborador, a frase “Banco de Dados” e *links* para outras páginas pessoais (o número de links pode variar de zero a sete). A página de índice tem *links* para todas as outras páginas. A Figura 3 apresenta a página *page5.html*.

```
<html>
<head>
</head>
<body>

<h3>Augusto Perini</h3>
<li>Banco de Dados</li>

<li><a href="page2.html">Renata Galante</a></li>
<li><a href="page4.html">Viviane Moreira</a></li>
<li><a href="page7.html">Carlos Heuser</a></li>

</body>
```

Figura 3. Página pessoal de Augusto Perini: page5.html

Os *links* entre as páginas pessoais foram cuidadosamente colocados para que algumas páginas pessoais tivessem mais *links* apontando para elas. O objetivo final é criar páginas com maior e menor relevância e que tivessem no seu conteúdo a frase "Banco de Dados". A Figura 4 mostra o grafo que representa as páginas pessoais do site e os *links* entre as mesmas.

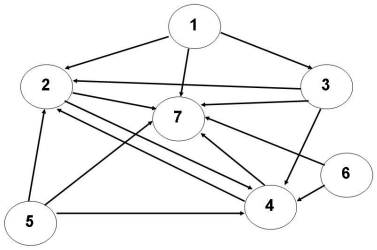

Figura 4. Grafo das páginas pessoais do site fictício

A páginas foram publicadas em um servidor *Web* para propiciar a navegação. Em seguida, foram executados os passos de *crawling* e busca, conforme descritos a seguir.

4.3. *Crawling*

Com o site fictício publicado, a próxima tarefa foi executar a *crawling* do site para que o *Nutch* pudesse indexá-lo e examinar as suas estruturas. O *crawler* no *Nutch* é um processo que roda independente dos demais. Através de um *script* é possível indicar uma lista de URLs iniciais para o início do processo, domínios, a profundidade e amplitude em que o *crawler* irá seguir os *links* e ainda o diretório onde serão salvos os índices.

No caso desse experimento, foi configurado o *crawler* do *Nutch* para iniciar o processo pela página de índice do site criado anteriormente. O processo mostrou-se bem rápido para esse pequeno conjunto de páginas.

4.4. Busca

Com os índices povoados, o próximo passo foi instalar a interface do motor de busca. Para isso foi necessária a instalação do servidor *Web* Apache Tomcat. A única configuração necessária para o *Nutch* realizar as buscas sobre as páginas criadas na seção 4.2 foi indicar a localização dos índices criados na seção 4.3. Com as tarefas preliminares realizadas, o *Nutch* fica apto a receber consultas.

Além do funcionamento do motor de busca como um todo, o experimento tinha o intuito de verificar se o algoritmo de *ranking* do Nutch funciona de forma adequada. O *Nutch* utiliza uma variável do algoritmo OPIC para atribuir escores às páginas [Krugler 2006]. A Figura 5 mostra os resultados para a pesquisa procurando pela frase "Banco de Dados".

Analisando o experimento foi verificado que algoritmo de *ranking* utilizado pelo *Nutch* é eficiente, visto que todas as páginas tinham estruturas muito parecidas, tinham a frase "Banco de Dados" na mesma localização. A ordem dos resultados ficou totalmente de acordo com a popularidade das páginas apresentadas na Figura 4: página 7, 2, 4, 3, 1, 5 e 6.

4.5. Detecção de Versões de Páginas *Web*

A seção 4.4 apresentou uma análise positiva do algoritmo de *ranking* utilizado pelo *Nutch*. O *Nutch* apresentou um uma ordenação satisfatória, mas não a ideal. Observando a Figura

5 nota-se que a página *page6.html* ficou em último lugar na classificação. Analisando a Figura 4 tem-se a explicação desse mau posicionamento: a *page6.html* não tem nenhum *link* apontando para ela. Chega-se a conclusão que a ordenação está correta.

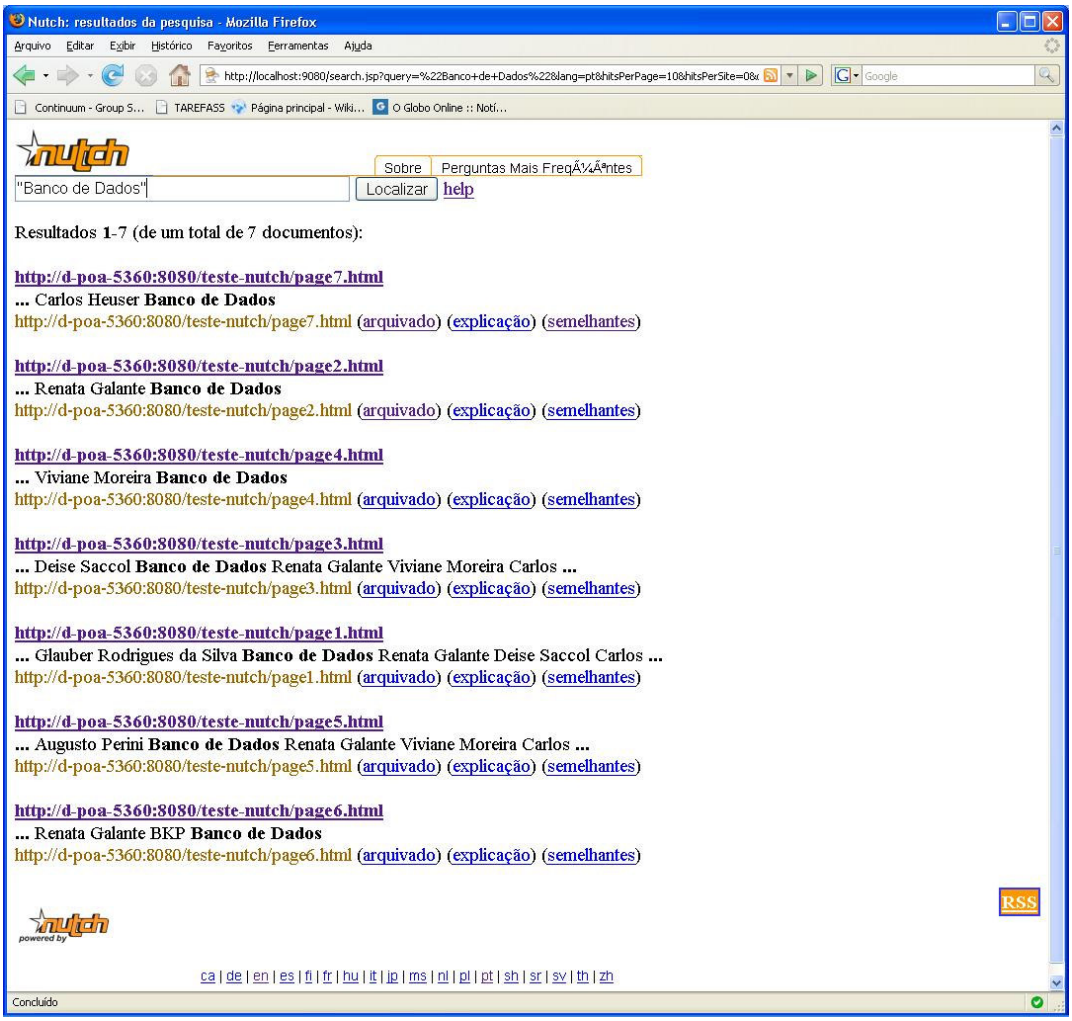

Figura 5. *Nutch*: Pesquisa pela frase "Banco de Dados"

Os algoritmos existentes atualmente na literatura não tratam a detecção de versões para melhorar o cálculo do escore das páginas *Web*, resultando por sua vez em um *ranking* que não considera o escore de versões antigas de páginas para ordenação dos resultados. O problema de detecção de versões para melhorar o cálculo de *PageRank* ainda é um problema em aberto na comunidade científica.

O problema de detecção de versões documentos é um problema bastante explorado na literatura. As soluções propostas para a detecção de documentos XML propõem que os documentos sejam processados como árvores, de modo que informações sobre as diferenças entre duas ou mais versões de um mesmo documento sejam obtidas a partir da comparação das estruturas dessas árvores.

Convém ressaltar o fato de que as técnicas de detecção de réplicas de documentos XML serem facilmente portáveis para detecção de documentos HTML. A fácil transformação de um documento HTML em um documento XHTML endossa essa idéia, já que um documento XHTML nada mais é que um documento XML.

Contudo, apesar da fácil adaptação das técnicas, tem que se ter em mente que um documento HTML é bem mais focado no conteúdo do que na estrutura e essa característica deve ser levada em conta ao aplicar as técnicas até agora apresentadas para documentos XML em documentos HTML.

Voltando a ordenação apresentada na Figura 5, se for considerado que a *page6.html* é uma versão da *page2.html*, seguindo a idéia da proposta, *a page6.html* deveria estar melhor posicionada. A *page6.html* foi inserida propositadamente no site fictício e ela contém o mesmo conteúdo da *page2.html*, com exceção de uma inserção das letras "BKP". A idéia é que o algoritmo proposto seja capaz de detectar (na fase de *crawling*) que a *page6.html* é uma versão da *page2.html* e atribuir um melhor escore à *page2.html*, com base nas suas versões anteriores, para que a mesma possa ser melhor classificada no *ranking* apresentado.

Para a detecção de versões de páginas *Web*, uma forma comum e bastante intuitiva que pode ser utilizada baseia-se em analisar a similaridade entre os documentos. A maioria das técnicas existentes utiliza limiares para definir o ponto de corte apropriado para os valores de similaridade. No entanto, definir o valor do limiar adequado é uma tarefa bastante difícil por várias razões: (i) o valor do limiar não é o mesmo quando se considera diferentes funções de similaridade e (ii) não é semanticamente significativo para o usuário. A fim de solucionar este problema, um trabalho em andamento [Saccol, Edelweiss, Galante and Zaniolo2007] propõe um mecanismo de versões de documentos XML baseado em classificadores *Naïve Bayesian*. Desta forma, a abordagem proposta considera o problema da detecção de versões como um problema de classificação.

O mecanismo de detecção de versões objetiva verificar se dois documentos referem-se a versões de uma mesma página Web. A detecção realiza duas atividades principais: análise de similaridade e classificação de documentos.

- **Análise de Similaridade:** esta tarefa é responsável por gerar um valor de similaridade para cada par de documentos. São levados em consideração a similaridade de estrutura e conteúdo entre as páginas. A função de similaridade *sim* entre dois documentos *f1* e *f2* é definida como $sim(f1,f2)=(w_1*F_1 + w_2*F_2 + w_3*F_3 + ... + w_n*F_n)$, onde w_n é um fator que representa a importância de uma certa característica F_n. O detalhamento das funções propostas pode ser encontrado em [Saccol, Edelweiss, Galante and Zaniolo 2007].
- **Classificação:** esta tarefa é responsável por decidir se dois documentos *f1* e *f2* são versões de uma mesma página. A fase de classificação recebe um conjunto de arquivos como treinamento (rotulados com uma classe – versão ou não versão), e produz um modelo (classificador). O classificador atribui o rótulo da classe adequada aos novos pares de documentos.

No artigo mencionado [Saccol, Edelweiss, Galante and Zaniolo2007], vários resultados experimentais são apresentados, os quais mostram que a abordagem produz excelentes resultados em termos de revocação e precisão.

5. Conclusões

Este trabalho apresentou uma proposta sobre o uso de detecção de versões de páginas *Web* para melhoria do desempenho do algoritmo de *PageRank*, abordando as diversas questões associadas a este tema principal. Como resultado desse trabalho, pretende-se definir formalmente uma extensão no algoritmo original de *PageRank* para que o mesmo passe a

considerar versões para a atribuição de escores as páginas *Web*. A fim de atingir tal objetivo, o mecanismo de detecção de versões proposto em [Saccol, Edelweiss, Galante and Zaniolo2007] será incorporado ao *Nutch*. A extensão do algoritmo de *PageRank* proposta deverá ser validada através de experimentos comparativos com o algoritmo original.

Referências

Baeza-yates, R. and Davis, E. (2004) Web page ranking using link attributes. In Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters. Pags: 328 – 329. ISBN: 1-58113-912-8, ACM Press.

Baeza-yates, R., Boldi, P. and Castillo, C. (2006) Generalizing PageRank: damping functions for link-based ranking algorithms. In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. Pags: 308 – 315. ISBN: 1-59593-369-7, ACM Press.

Berberich, K., Bedathur, S. and Weikum, G. (2006) Rank synopses for efficient time travel on the web graph. In proceedings of the 15th ACM international conference on Information and knowledge management. Pags: 864 – 865, ISBN: 1-59593-433-2 ACM Press.

Brin, S. and Page, L. (1998) The anatomy of a large-scale hypertextual Web search engine. 7th WWW Conference, Brisbane, Australia.

Buriol, L.S., Castillo, C., Donato, D., Leonardi, S. and Millozzi, S. (2006) Temporal Analysis of the Wikigraph. In Web Intelligence, IEEE/WIC/ACM International Conference Pags 45 – 51.

Ding, C., He, X., Husbands, P., Zha, H. and Simon, H. D. (2002) PageRank, HITS and a unified framework for link analysis. In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. Pags: 353-354. ISBN: 1-58113-561-0, ACM Press.

Keinberg, J. M. (1999) Authoritative sources in a hyperlinked environment. In Journal of the ACM (JACM) Volume 46. Pags: 604 – 632. ISSN: 0004-5411, ACM Press.

Krugler, K. (2007). Fixing the OPIC algorithm in Nutch. Disponível em http://wiki.apache.org/nutch/FixingOpicScoring . Acessado em out. 2007.

Langville, A. N. and Meyer, C. D. (2006) Google's PageRank and Beyond: The Science of Search Engine Rankings, Princeton University Press, Princeton, NJ.

Nutch (2007). Disponível em http://lucene.apache.org/nutch/. Acessado em dezembro. 2007.

Page, L., Brin, S., Motwani, R. and Winograd T. (1998) The PageRank citation ranking: Bringing order to the Web. Tech. report, Stanford University.

Saccol, D. B.;, Edelweiss, N.;, Galante, R. M.; and Zaniolo, C.. (2007) *XML Version Detection*. In: ACM Symposium on Document Engineering, 2007, Winnipeg, Canada.