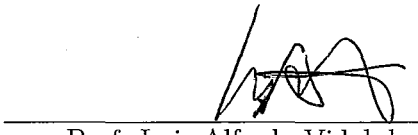
RELP SURFING: UM FRAMEWORK PARA O
RELACIONAMENTO DE PÁGINAS WEB

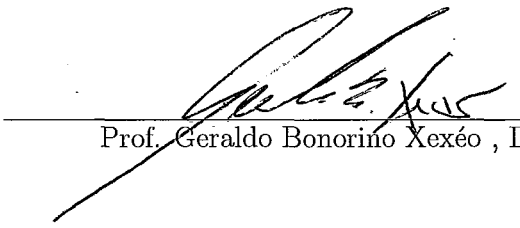
Vinicius Faria Culmant Ramos

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:



Prof. Luis Alfredo Vidal de Carvalho, D. Sc.



Prof. Geraldo Bonorino Xexéo , D. Sc.

Profa. Rosa Maria Esteves Moreira da Costa, D. Sc.

RIO DE JANEIRO, RJ - BRASIL
JUNHO DE 2008

RAMOS, VINICIUS FARIA CULMANT

RelP Surfing: Um Framework para o Relacionamento de Páginas Web [Rio de Janeiro] 2008

XII, 53 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2008)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Web Mining
2. Relacionamento de Páginas Web

I. COPPE/UFRJ II. Título (série)

À minha filha Júlia, por ser
o centro do meu universo.

Agradecimentos

Em primeiro lugar quero agradecer à minha família por ter me apoiado em todos os momentos de minha vida. Todos foram muito importantes nesta longa caminhada. Esta minha família maravilhosa que tem seus alicerces em meu pai e minha mãe. São eles que merecem todos os agradecimentos possíveis, pois sem eles eu não chegaria onde cheguei. Obrigado D. Claudio e D. Sonia. Eles merecem todo e qualquer elogio que eu possa dizer. Agradeço aos meus avós: Eurides, Oswaldo e Lia. Muito obrigado.

Não posso falar em minha família sem falar de meus irmãos. Isabela, Mano e Cássia, obrigado pelo carinho, risadas e apoio. Preciso falar de cada um deles em separado. Agradeço à Cássia pelos dias de risada e carinho que me proporcionou ao lado de seu marido Felipe e suas filhas maravilhosas, Nathália e Duda. Falar da minha irmã caçula é até covardia com todos, porque a caçula da casa é sempre a que ganha os maiores elogios e com ela não seria diferente. Era ela que brigava comigo, que conversava comigo, que brincava comigo etc. Era a Isabela que me fazia sorrir e olhar sempre pra frente em busca de meus objetivos. E o Mano e sua esposa Ana são muito especiais nesta caminhada. A minha cunhada sempre interessada em saber quando eu terminaria o mestrado e o meu irmão sempre interessado em me levar para jogar futebol. Foram inúmeras as vezes que saímos com esta finalidade. Esse era o meu passatempo predileto.

Agradeço à minha filha Júlia, sem dúvida tornou-se o centro do meu universo. Antes de nascer ela já incentivava o seu pai a terminar o mestrado e a acordar cedo para trabalhar em sua dissertação. Agradeço também à Keila, mãe da minha filha. Agradeço pelo carinho e apoio que me foi dado por toda elaboração deste trabalho.

Em muitos momentos esse apoio foi de extrema importância, principalmente quando se tratava de cuidar de nossa filha. Obrigado.

Agradeço à Taísa por seu companheirismo, carinho, apoio e dedicação, principalmente na etapa final desta dissertação. Era ela que preparava o meu café para eu ficar acordado até mais tarde, que me ajudava a revisar os capítulos da dissertação e que dedicava parte de seu tempo para me acompanhar. Todas essas palavras são poucas se comparado ao seu carinho por mim e tudo que posso agradecer a ela.

Obrigado pela força Nini, Rachel e Ayla. Estes são grandes amigos e merecem meus agradecimentos. Estes aqui são meus cumpadres e minha afilhada, respectivamente. Eles foram, em muitos momentos, peças fundamentais para a conclusão desta dissertação.

Se eu não falasse desse grande amigo, co-orientador, incentivador e péssimo jogador de futebol chamado Alexandre Stauffer, esses agradecimentos não estariam completo. Foi ele que meu auxiliou em todas as etapas de minha graduação e mestrado. Esse cara conhecido apenas como Stauffer é responsável por minha dedicação aos estudos. Foi o Stauffer que me ensinou a fazer pesquisa. Foi o Stauffer que me ensinou a escrever um texto acadêmico. Foi o Stauffer que me auxiliou a encontrar grande parte do conhecimento que tenho hoje. Para isso eu tive que ensiná-lo a jogar futebol, mas parece que ele ainda não aprendeu. Obrigado Stauffer pela amizade, emails, correções, esporros e tudo mais que você já fez por mim.

Obrigado Luis Alfredo que, além de orientador, também tornou-se um grande amigo. Uma pessoa dona de um vasto conhecimento. Uma pessoa dona de um enorme senso de humor. Em todos os momentos deste trabalho me incentivou e me apoiou. Obrigado pela enorme contribuição em meu conhecimento que me foi passada.

Agradeço às minhas amigas Miriam e Taís, minha orientadora e co-orientadora nos tempos de iniciação científica e de projeto final de curso. Elas são grandes responsáveis por eu construir essa carreira acadêmica. Foram elas que me incentivaram a fazer pesquisas científicas e até mesmo orientar alunos de graduação.

Dentre todos os meus amigos eu gostaria de falar de alguns em especial. Obrigado Leizer que, apesar de ter esse nome, tornou-se um de meus melhores amigos em pouco tempo. Foi meu amigo nos dias de futebol, nos dias de cerveja e nos dias de estudos também. Amigo para todos os momentos. Quero agradecer também à dupla: Fábio e Jesus. Ficamos amigos em pouco tempo e já fazem parte do meu ciclo de melhores amigos. Essenciais nas noites de cerveja e de papo-furado. Entre tantos outros grandes amigos quero agradecer ao Dill e sua esposa Camila pois tenho grande admiração e respeito por eles. A admiração ao Dill vem não só nos estudos mas também no futebol e na cachaça.

São muitos os meus grandes amigos e não posso dissertar sobre todos neste pequeno espaço, mas gostaria de lembrar de vários amigos de faculdade, de futebol e de cerveja. Obrigado Calisto, Papel, Izaías, Digão, Targino, Carol, Duran, Marquinho Tesouro, Léo Gaúcho, Thiago, Saulo e Marina. Todos vocês foram importantes para a realização deste trabalho. Agradeço também aos funcionários do Programa de Engenharia de Sistemas e Computação: Solange, Sônia, Claudia, Lúcia, Alex, Leandro, Guty e Itamar pelo apoio que me deram durante esta etapa.

Agradeço aos professores Jano Moreira e Roberto Blaschek pela oportunidade em seus projetos que contribuiu para meu crescimento profissional e acadêmico.

Agradeço à professora Rosa Costa e ao professor Geraldo Xexéo por aceitarem o convite de participação na banca deste trabalho.

Agradeço ao CNPq pelo apoio financeiro.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

RELP SURFING: UM FRAMEWORK PARA O RELACIONAMENTO DE PÁGINAS WEB

Vinicius Faria Culmant Ramos
Junho/2008

Orientador: Luis Alfredo Vidal de Carvalho

Programa: Engenharia de Sistemas e Computação

Este trabalho consiste em procurar páginas Web semelhantes usando a estrutura dos links na Web. Mais especificamente, dada uma página Web, a qual chamamos de página atual em alusão ao fato de que tipicamente esta é a página atualmente sendo visitada por um usuário, procuramos na estrutura dos links encontrar outras páginas Web que são semelhantes à atual. Modelamos a Web como um grafo direcionado, ao qual chamamos de grafo Web. Cada nó desse grafo representa uma página Web e as arestas representam os links entre as páginas Web. Introduzimos um algoritmo que realiza uma busca aleatória nesse grafo começando pela página atual. A idéia básica é atribuir pesos às arestas desse grafo de forma a fazer essa busca tender em direção às páginas Web que são mais semelhantes à atual. Nesse trabalho, discutimos três métodos para atribuir pesos as arestas. Nosso algoritmo lembra e, de fato, é inspirado em outros algoritmos; porém ele vai além por explorar mais profundamente o grafo Web, permitindo que a busca aleatória alcance nós relativamente mais distantes da página atual.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

RELP SURFING: A FRAMEWORK FOR
RELATING WEB PAGES

Vinicius Faria Culmant Ramos
June/2008

Advisor: Luis Alfredo Vidal de Carvalho

Department: Systems Engineering and Computer Science

This work consists of searching for similar Web pages using the structure of the links in the Web. More specifically, we are given a Web page, which we call the current Web page in allusion to the fact that it is typically the Web page being currently visited by a user, and search this link structure to find other Web pages that are similar to the current one. We model the structure of the Web as a weighted directed graph, to which we refer as the Web graph. Each node of the Web graph represents a Web page and its edges represent the links between Web pages. We introduce an algorithm that performs a kind of random walk on this graph starting from the node that represents the current Web page. The basic idea is to set up the weights of the edges in such a way to bias the random walk towards the nodes that tend to be more similar to the current Web page. In this work we discuss three methods for attributing weights to the edge. Our algorithm resembles and in fact is inspired in previous works; it extends on those works by exploring the Web graph more thoroughly, allowing the random walk to reach nodes that are further away from the current Web page.

Sumário

Lista de Figuras

Lista de Algoritmos

Lista de Tabelas

Capítulo 1

Introdução

A *World Wide Web* (ou simplesmente Web) é um sistema de relacionamento de hipertexto acessado pela Internet. A sua criação remete-se ao começo da década de 80, quando Berners-Lee, então pesquisador do laboratório CERN na Suíça, notou a dificuldade de físicos em todo o mundo de compartilharem informações sem que tivessem uma máquina específica ou fossem obrigados a instalar programas de apresentação de informação em suas máquinas. Em 1989, Berners-Lee desenvolveu uma proposta para a Web, que foi aperfeiçoada em 1990 com a cooperação de Cailliau [1].

No final de 1997 a Web atingiu a marca de 3 milhões de páginas e mais de 200 milhões de links entre elas [2]. Em meados de 2000, Lawrence e Giles estimaram que a Web possuia aproximadamente 800 milhões de páginas [3] e em 2005 este número cresceu assustadoramente para algo em torno de 11.5 bilhões de páginas [4].

O rápido crescimento da Web deu-se muito à sua fácil utilização, incluindo o fácil desenvolvimento e publicação de uma página na Web. Outro fator importante para esse crescimento é a diversidade de formas com que as informações podem ser publicadas na Web. Por exemplo, pode-se disponibilizar vídeos e imagens de forma simples e eficaz, bastando carregá-las em um servidor Web e fazer a ligação (link) deles com uma página Web; dessa forma eles poderão ser visualizados por praticamente qualquer navegador Web do mercado. Além disso, o acesso desses recursos é também muito simples. Basta o usuário possuir o endereço URL (Unified Resource Linkage Locator) da página que contém o link para este recurso.

Por outro lado, a facilidade de desenvolvimento e publicação de páginas acabou por transformar a Web em uma grande produtora de lixo eletrônico. O fato de não haver validação no conteúdo publicado na Web favorece a proliferação de páginas com erros de sintaxe ou de conteúdo. Portanto, nos dias de hoje, cabe ao usuário certificar-se da corretude da informação que ele encontra na Web.

A Web proporcionou oportunidades nunca dantes imaginadas para os processos de recuperação de informação (*IR* - do homônimo em inglês "*Information Retrieval*"), haja vista que a sua alta expansão tornou a busca e recuperação de informação cada vez mais complexa e essencial.

A *IR* é uma disciplina multidisciplinar cujo principal e seu maior objetivo é a busca por informação, seja em documentos, entre documentos ou em metadados destes documentos. Em se tratando da Web, as principais ferramentas de *IR* presentes no dia a dia dos usuários são as ferramentas de busca, como por exemplo *Yahoo!* [5], *Google* [6] e *Live Search* [7]. Estas ferramentas são responsáveis por recuperar informações de páginas Web a partir de um pequeno conjunto de palavras-chave digitadas pelos usuários.

É um desafio incomensurável para a *IR* responder todas as questões dos usuários de forma rápida e eficaz. Reconhecendo esta dificuldade, nos concentramos em uma pequena porém significativa parte desse desafio, que é o estudo do relacionamento entre páginas Web. Já que os usuários precisam ter acesso rápido a informações de qualidade, a busca automática por páginas Web relacionadas àquelas atualmente sendo visualizada pelo usuário é evidentemente de grande valia. É precisamente este o tema principal deste trabalho.

Os sites de busca empregam essencialmente duas estratégias na busca por páginas Web. A primeira delas é associar o conteúdo de cada página às palavras-chave especificadas pelo usuário, ou seja, ao se digitar a palavra "algoritmo", as ferramentas de busca procuram páginas que possuem esta palavra em seu conteúdo e as ordena por um processo de contagem (quanto mais vezes a palavra "algoritmo" aparecer melhor). Uma outra forma de relacionar o que se busca com páginas Web é explorando a estrutura dos links, ou seja, primeiramente se relaciona o que se busca com o conteúdo da página (não precisando lançar mão de contagem) e depois se procura uma ordenação das páginas usando a estrutura de links. Esta estratégia se

baseia na idéia de que a existência de um link de uma página A para uma página B pode ser vista como se A estivesse recomendando a página B.

Sabendo-se das dificuldades encontradas para buscar informações em um ambiente com mais de 11 bilhões de páginas, a pesquisa por algoritmos e heurísticas que auxiliem o usuário nos motiva a achar relacionamentos entre páginas Web. É com esse espírito que estudamos a estrutura dos links na Web de forma a melhor compreendê-la e utilizá-la na busca por esses relacionamentos.

O relacionamento de páginas Web utilizando a estrutura dos links assume pelo menos um dos princípios a seguir:

1. A existência de um link de uma página A para uma página B corresponde a uma recomendação do autor da página A para a página B.

2. A existência de um link entre uma página A e uma página B indica que elas devem tratar do mesmo tema.

Estes dois princípios são essenciais para o desenvolvimento deste trabalho. Exploramos a estrutura dos links para estabelecer relacionamentos entre as páginas Web e a página que o usuário está visitando atualmente, a qual chamamos de página atual. O nosso foco é entender como páginas referenciadas pela página atual se relacionam, visando o estabelecimento de um ranking dessas páginas em termos de grau de relevância para com a página atual.

No Capítulo 2, fazemos uma breve descrição sobre a *IR* e o problema de busca de informações na Web. Apresentamos algumas idéias existentes na literatura para explorar a estrutura dos links na Web visando a ordenação dos resultados de uma busca. Além disso, discutimos com mais detalhes dois algoritmos que utilizam esta estrutura para relacionar páginas Web através de uma URL. No Capítulo 3, introduzimos nosso algoritmo e três variações dele para a busca de páginas relacionadas com a página atual. No Capítulo 4, apresentamos um *framework* e um *Web proxy* utilizados nas implementações para captura dos dados de testes e avaliação do algoritmo proposto. No Capítulo 5, discutimos os resultados apresentados pelo nosso algoritmo. Por fim, no último Capítulo apresentamos as conclusões deste trabalho e também sugestões de temas e principais contribuições para trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

2.1 Introdução

Há decadas que a busca por informações deixou de ser feita em artefatos de papel, como jornais, livros e revistas, e passou a ser feita por artefatos digitais. Não que estes artefatos digitais tenham substituídos os artefatos de papel por completo, mas muitos deixaram de ser fontes principais de pesquisa e de busca por informações. Esta mudança de paradigma cria uma nova área do conhecimento, que é conhecida como "*Information Retrieval*" (*IR*).

A *IR* surgiu assim que documentos digitais passaram a armazenar informações que antes eram datilografadas ou escritas em papel. É uma área multidisciplinar, contando com ciências matemáticas, de computação, bibliotecária, entre outras. Em síntese, as áreas de atuação principais da *IR* consistem na busca por informações em documentos, localização de documentos em base de dados, busca por metadados [1] que descrevem os documentos e ainda busca em bases de dados genéricas abrangendo desde bases de dados relacionais até bases de dados de hipertextos como a *World Wide Web*.

Uma das disciplinas da *IR* é a categorização de textos (*TC* - do homônimo em inglês *Text Categorization*). A *TC* é a disciplina responsável por criar categorias para textos em linguagens naturais de acordo com um conjunto pré-definido de

[1] Conhecida como "dados dos dados", de qualquer ordem em qualquer meio. Um item de metadado descreve uma referência individual ou o conteúdo de um item ou ainda uma coleção de dados incluindo múltiplos itens.

categorias. Nos dias de hoje, a *TC* é largamente utilizada nos mais diversos contextos, desde a indexação de documentos até a filtragem de vocábulos em um texto.

Não se pode falar sobre digitalização e busca por informação sem mencionar a *World Wide Web* (ou simplesmente Web). A Web surgiu na década de 80 e em menos de duas décadas tornou-se o principal meio de difusão de informação. No final da década de 90, a Web se polarizou de forma abrupta, facilitando o acesso e permitindo a difusão maciça de informação. Em 1997 Bharat e Broder estimaram a existência de 200 milhões de páginas Web [2]. Lawrence e Giles, em meados do ano 2000, estimaram o tamanho da Web em um valor um pouco menor que 800 milhões de páginas [3]. Mais recentemente, em 2005, este número cresceu para aproximadamente 11.5 bilhões de páginas Web, segundo estudo de Gulli e Signorini [4].

O crescimento desordenado da Web cria diversas oportunidades para a *IR*, entre as quais podemos destacar a aplicação da *TC* sobre as páginas Web. No começo desta década, as ferramentas de busca utilizavam diversas técnicas de *TC* para ordenar de forma rápida e eficaz as páginas Web. Sebastiani [8] descreve as técnicas utilizadas pela *TC* nesse contexto.

Sobre as ferramentas de busca na Web, a busca por páginas através da análise do conteúdo é bastante eficiente. Porém, ela abre a possibilidade de os criadores de uma página burlarem os métodos de análise, adaptando o conteúdo do site de forma a satisfazer os critérios adotados pelas ferramentas de busca. Por exemplo, um método simples de análise de conteúdo é a contagem do número de vezes que cada palavra aparece em uma página. Portanto, se uma página A contém 10 vezes a palavra W e uma página B contém 5 vezes a mesma palavra, a página A aparecerá ordenada na frente da página B em caso de uma busca pela palavra W. Este é um exemplo clássico para demonstrar como a análise de conteúdo pode causar resultados ruins.

No final dos anos 90, pesquisadores de *IR* desenvolveram técnicas de análise estrutural dos links da Web. Algumas dessas técnicas baseiam-se em técnicas existentes em outras áreas do conhecimento como por exemplo no estudo de redes sociais e de redes de citações científicas.

As redes sociais são representadas como um grafo $G = (V, E)$, onde os nós representam indivíduos e a existência de uma aresta (i, j) significa que o indivíduo i conhece o indivíduo j. Nessas redes, por exemplo, existe o interesse em ordenar os graus de entrada [2] de cada nó [9].

Redes de citações científicas [10] revelam informações sobre a importância e o impacto de um artigo científico no meio acadêmico. Muitos dos métodos desenvolvidos baseiam-se na contagem da quantidade de referências a um determinado artigo [11, 12, 13, 14].

A estrutura de um hipertexto é importantíssima para o funcionamento da Web. É essa estrutura que permite com que usuários se direcionem em sua navegação pela Web. Uma das primeiras análises da estrutura de links dos hipertextos foi feita por Botafogo et al. [15] visando a ordenação de páginas Web. Nesta abordagem as páginas são modeladas como um grafo direcionado, onde os nós representam as páginas e as arestas representam os links entre páginas. Eles definem dois tipos de nós, chamados de *index nodes* e *reference nodes*. Os nós chamados de *index nodes* são aqueles cujo grau de saída é maior do que a média de todos os graus de saída no grafo. Os *reference nodes*, por sua vez, possuem o grau de entrada maior do que a média de todos os graus de entrada no grafo. Uma proposta para a ordenação do resultado de uma busca foi apresentada por Carrière e Kazman [16]. Esta ordenação baseia-se na contagem da quantidade de arestas de um nó (independente de serem arestas de saída ou de entrada).

Entre os algoritmos que analisam a estrutura dos links na Web, destacam-se dois: o algoritmo *HITS* [17], proposto por Kleinberg e descrito mais detalhadamente na Seção 2.2 e o algoritmo *PageRank* [18], proposto por Page et al., que foi implementado juntamente a uma ferramenta de busca que nos dias atuais é a ferramenta de busca mais utilizada no mundo [6]. O *PageRank* é descrito na Seção 2.3.

Outros dois algoritmos merecem nossa atenção, são eles: *Companion* e *Cocitacion*, ambos propostos por Dean e Henzinger [19]. Estes algoritmos são descritos na Seção 2.4 e também fazem uma análise estrutural dos links da Web. Eles têm como objetivo principal o relacionamento de páginas, que é o foco principal desta dissertação.

[2] Quantidade de arestas que apontam para um nó

2.2 Algoritmo *HITS*

O principal recurso das páginas Web é o hiperlink. São os hiperlinks que proporcionam toda navegação pela Web. Até a publicação de Kleinberg [17], a estrutura deste importante recurso havia sido pouco explorada, tanto por ferramentas de busca quanto por pesquisadores de *IR*.

Na Seção 2.1, mencionamos tentativas de utilização da estrutura dos links da Web em processos de contagem de referências. Essa contagem funciona pura e simplesmente para avaliar a importância de algum ator em um mesmo nicho de informação, sendo esse ator um sistema, uma pessoa ou uma organização. Essas referências existiam em locais específicos, como em páginas Web de jornais ou revistas. Entretanto, com a difusão da *World Wide Web*, as referências entre sites deixaram de estar limitadas a pequenos nichos ou grupos.

Em se tratando de referências, as ferramentas de busca na Web procuram ordenar páginas por sua relevância. Um dos métodos utilizados para estimar a relevância de uma página é um processo de contagem de referências, ou seja, dado uma página A qualquer, a estimativa da relevância de A é igual ao número de páginas que a referencia. Esse processo é claramente inspirado nos métodos utilizados para a ordenação de artigos acadêmicos.

Podemos definir um grafo sobre a estrutura dos hiperlinks pela Web. Cada página Web é representada por um nó no grafo e cada aresta representa a referência de uma página a outra. Portanto, esse grafo é direcionado, pois a existência de um hiperlink de uma página para outra não implica na existência de um hiperlink na direção oposta.

A existância da aresta $(X \rightarrow Y)$ no grafo indica que o autor da página X faz referência à página Y. Espera-se que autores de páginas Web referenciem conteúdos que lhes são relevantes, então os autores da página X atribuem um pouco de "autoridade" à página Y.

Este conceito de "autoridade" (*authority*) norteou o trabalho de Kleinberg. Em seu trabalho, as páginas Web e seus links são representados por um grafo direcionado, como definimos acima. Para melhor entendermos a análise estrutural dos hipertextos no trabalho de Kleinberg precisamos definir as páginas chamadas de *authorities* e as chamadas de *hubs*.

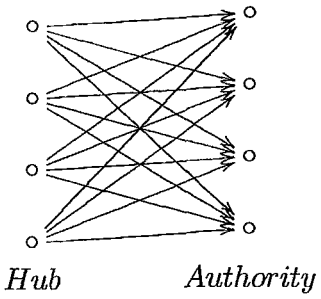


Figura 2.1: Representação de *Hubs* e *Authorities*

Para especificarmos *authority* e *hub*, vamos usar de um exemplo simples: uma página Web p que referencia uma página Web q. Portanto o criador da página p especificou um certo grau de "autoridade" à página q. Com isso, é fácil acharmos *authorities* em potencial. Basta buscarmos por páginas muito referenciadas ou com alto grau de entrada. As páginas Web chamadas de *hubs* são aquelas que têm um grau de saída alto (veja Figura 2.1).

Com esta idéia, Kleinberg desenvolveu um algoritmo chamado *HITS* (Hypertext Induced Topic Selection) que analisa os links de um conjunto de páginas Web e os ordena de acordo com sua relevância. A entrada do algoritmo é um subgrafo da Web com as páginas a serem analisadas e ordenadas[3]. O *HITS* é um algoritmo iterativo. Para cada página p, atribui-se uma variável não-negativa $x^{(p)}$ chamada peso *authority* e uma variável não-negativa $y^{(p)}$ chamada peso *hub*. A seguir, apresentamos o algoritmo proposto por Kleinberg:

[3]O subgrafo utilizado foi obtido utilizando-se ferramentas de buscas populares na época, como por exemplo, o *yahoo!* [5] e o *alta vista* [20].

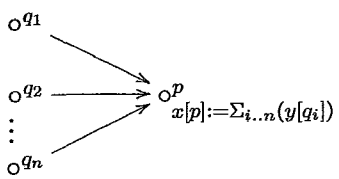


Figura 2.2: Operação Φ

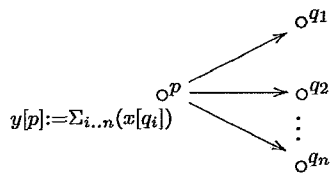


Figura 2.3: Operação Θ

G : Grafo que representa a Web;
k : Número inteiro;
Seja z um vetor (1,1,1,...,1) $\in \Re^n$;
$x_0 \leftarrow z$;
$y_0 \leftarrow z$;
para cada $i = 1, 2, 3, ..., k$ **faça**
  $x'_i \leftarrow \Phi(x_{i-1}, y_{i-1})$;
  $y'_i \leftarrow \Theta(x'_i, y_{i-1})$;
  $x_i \leftarrow$ Normalizar x'_i;
  $y_i \leftarrow$ Normalizar y'_i;
fim
retorna (x_k, y_k);

Algoritmo 1: Algoritmo *HITS*

As operações Φ e Θ representam atualizações dos valores dos pesos $x^{(p)}$ e $y^{(p)}$ (veja figuras 2.2 e 2.3), respectivamente. Elas são definidas da seguinte forma:

$$\Phi \ : \quad x^{(p)} \leftarrow \sum_{(q \rightarrow p) \in E} y^{(q)}$$

$$\Theta \ : \quad y^{(p)} \leftarrow \sum_{(p \rightarrow q) \in E} x^{(q)}$$

A criação deste novo conceito estabeleceu um novo paradigma para as ferramentas de busca e para os pesquisadores de *IR*. A simples idéia de *authorities* como páginas influenciadas ou concentradoras de informações e *hubs* como páginas que influenciam ou são difusoras de informações conseguem alterar a forma com

que as ferramentas de buscas na Web operam e, também, como alguns aplicativos, estabelecem relacionamentos entre páginas. Muitas das pesquisas que despontaram utilizaram dessa idéia. Além disso, este novo conceito norteia os métodos propostos neste trabalho.

Na Seção 2.3 apresentamos um outro algoritmo de ordenação de páginas Web que, apesar de não estar relacionado diretamente com o algoritmo de Kleinberg, possui algumas semelhanças com o *HITS*, principalmente no que tange a ordenação através do grau de entrada e de saída de um nó.

2.3 *PageRank*

Enquanto as ferramentas de busca criavam alternativas para ordenar os resultados de suas buscas utilizando o conteúdo das páginas, pesquisadores priorizavam o entendimento completo da estrutura dos links pela Web. Nas Seções 2.1 e 2.2, discutimos a importância de se utilizar a estrutura dos links para o relacionamento de páginas Web ou a ordenação de resultados de busca.

É neste sentido que Brin et al. [18] descrevem um algoritmo para ordenação de resultados de buscas. Diferentemente da abordagem de Kleinberg [17], na qual a entrada de seu algoritmo era o resultado da busca em algum dos diversos sites de busca (*Alta Vista* [20], *Yahoo!* [5], entre outros), Brin e Page criaram a sua própria base de dados e a sua própria ferramenta de busca, chamada de *Google* [21]. Esta ferramenta viria a se tornar a principal ferramenta de busca dos dias de hoje.

Na época, a base de dados do *Google* estava populada com 75 milhões de URLs e mais de 320 milhões de links. Esses valores são relativamente altos, pois no ano 2000 (2 anos após a publicação do artigo de Brin e Page) Lawrence e Giles [3] estimaram o tamanho da Web em um valor um pouco menor do que 800 milhões de páginas.

O algoritmo de Brin e Page trabalha em cima de todo o grafo Web. O algoritmo baseia-se na contagem dos graus de entrada e de saída de cada um dos nós. Este conceito está fortemente ligado às pesquisas acadêmicas em torno do grau de importância de um artigo acadêmico [22, 23]. A intuição é que artigos muito referenciados ou muito citados tendem a ser mais importantes do que outros artigos menos citados. Assim como artigos que citam muitos outros artigos costumam ser boas referências no meio acadêmico.

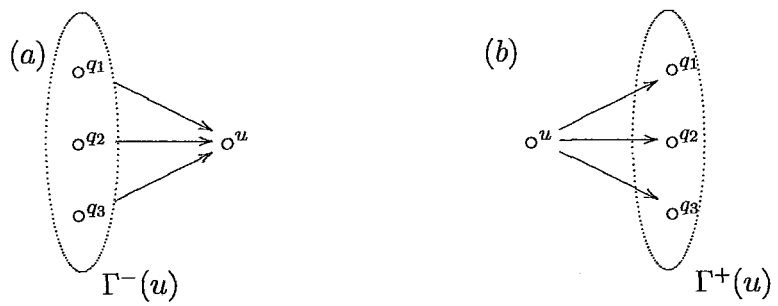


Figura 2.4: (a) $\Gamma^-(u)$ - representa os "pais" de u; (b) $\Gamma^+(u)$ - representa os "filhos" de u

A definição do algoritmo *PageRank* utiliza essa idéia das referências dos artigos acadêmicos. Este algoritmo é recursivo e, em cada iteração, ele atualiza o peso das arestas de um nó do grafo usando, basicamente, os graus de entrada e saída deste nó. O valor armazenado é considerado o grau de importância de cada nó (ou grau de "autoridade" de cada nó sobre os outros) e é este valor que eles chamam de *PageRank*. O grau de "autoridade" de um nó, como os próprios autores citam, faz referência ao conceito de *authority* descrito na Seção 2.2.

Para a definição do *PageRank*, precisamos de alguns conceitos básicos de teoria dos grafos. Seja u uma página Web. Denotamos por $\Gamma^+(u)$ o conjunto de páginas Web que u aponta e $\Gamma^-(u)$ o conjunto de páginas Web que apontam para u (veja Figura 2.4).

Seja A uma matriz quadrada onde as linhas e colunas representam páginas Web. Seja u e v duas páginas Web. Seja $N_u^+ = |\Gamma^+(u)|$ o número de links apontados por u. Definimos o elemento (u, v) da matriz A como $\frac{1}{N_u^+}$ se a aresta $(u \rightarrow v)$ existe e 0 no caso contrário.

Uma maneira de se definir o *PageRank* é vendo A como a matriz de transição de uma cadeia de Markov [24] definida sobre o grafo Web.

A partir disso, podemos definir a forma de se calcular o *PageRank* para cada um dos nós do grafo.

Definição do *PageRank*: Seja $E(u)$ uma distribuição de probabilidade sobre as páginas Web, isto é, $E(u) > 0$ para todo u e $\sum_u E(u) = 1$. Seja $c < 1$ um parâmetro positivo. O *PageRank* consiste na distribuição estacionária da cadeia de Markov

sobre páginas Web com matriz de transição dada por $cA + (1-c)E$. Em palavras, essa cadeia de Markov equivale ao seguinte processo. Dada uma página Web u, com probabilidade c escolhemos um dos links de u e transitamos para a página apontada por u e, com probabilidade $(1-c)$, escolhemos uma página usando a distribuição E.

Com esta definição, podemos apresentar o algoritmo que computa os valores *PageRank* de cada nó. Seja S um vetor sobre as páginas Web. O *PageRank*[4] pode ser computado da seguinte forma:

$R_0 \leftarrow S$;
repita
  $R_{i+1} \leftarrow AR_i$;
  $d \leftarrow \|R_i\|_l - \|R_{i+1}\|_l$;
  $R_{i+1} \leftarrow R_{i+1} + dE$;
  $\delta \leftarrow \|R_{i+1} - R_i\|_l$;
até $\delta > \epsilon$;

Algoritmo 2: Algoritmo Iterativo que calcula o valor do *PageRank* de cada nó.

Por se tratar de um algoritmo recursivo e por ter milhões de páginas Web (nós do grafo) a serem analisadas, o custo para se obter o valor de cada nó é alto a priori. Porém, esse custo é ainda relativamente pequeno se comparado ao custo de se indexar todo o texto de páginas Web.

Um problema óbvio deste algoritmo consiste na sua ineficiência quando o grafo é atualizado frequentemente com inserção e remoção de nós e arestas. Portanto, seria natural pensar que a aplicação desse método é ineficiente para um grafo que tenha a dimensão da Web, pois a cada minuto milhares de modificações são feitas. Esta inviabilidade não é de todo verdade, pois o método proposto atribui valores maiores para o *PageRank* de nós que possuem grau de saída ou o grau de entrada alto. Sabemos que na Web a criação ou remoção de páginas com graus de entrada ou saída alto não acontece com freqüência. Por isso, o valor do *PageRank* de cada um dos nós não precisa ser modificado a cada atualização do grafo, podendo ser atualizado apenas de tempos em tempos.

[4]A convergência do algoritmo apresentado a seguir foge do escopo deste trabalho e a sua prova completa pode ser encontrada em [18].

2.4 *Companion* e *Cocitation*

Alternativamente às Seções 2.2 e 2.3, Dean e Hezinger propõem dois métodos de relacionamento de páginas através do endereço URL [19].

Um relacionamento entre duas páginas Web significa que elas possuem o mesmo tópico, mas não necessariamente o mesmo conteúdo ou a mesma semântica. Para exemplificar, se dermos o endereço *www.cbf.com.br*[5], o método deverá retornar páginas relacionadas ao futebol, como páginas de clubes ou campeonatos de futebol. Esta abordagem requer que o usuário saiba exatamente a página que está procurando, diferentemente do problema de busca na Web elucidado anteriormente neste capítulo.

A principal preocupação dos autores nessa abordagem é fornecer uma resposta boa e rápida ao usuário. Para isso, são utilizadas apenas informações dos links da Web, ou seja, apenas a estrutura de links que a Web fornece. Para se obter essa informação, os autores construíram uma ferramenta que explora a estrutura de links da Web [25]. Apenas os links e a ordem com que esses links aparecem nas páginas são utilizados, e nenhum conteúdo das páginas é analisado.

Os dois algoritmos são chamados de *Companion* e *Cocitation*. O primeiro deles é baseado no algoritmo HITS proposto por Kleinberg (ver Seção 2.2). Em seu artigo, Kleinberg sugere a utilização do algoritmo HITS para o relacionamento de páginas, e apresenta diversas evidências de sua potencialidade. O segundo algoritmo apresenta páginas que são co-citadas pela entrada do algoritmo, ou seja, a URL correspondente. Por exemplo, se uma página C possui link para a página A e para a página B, então A e B são co-citadas por C, como demonstrado na Figura 2.5.

2.4.1 Algoritmo *Companion*

O algoritmo *companion* tem como entrada de dados uma URL u e consiste nos 4 passos a seguir:

1. Gerar o grafo de *vizinhança* de u;

2. Eliminar nós duplicados no grafo;

[5]Página da Confederação Brasileira de Futebol, órgão máximo do futebol brasileiro

Figura 2.5: (a) Páginas A e B co-citadas por C; (b) Grafo não-direcionado de co-citação

3. Computar pesos das arestas baseados em conexões *domínio* a *domínio*;

4. Calcular valor de *hub* e de *authority* de cada um dos nós do grafo e retornar os 10 melhores *authorities*.

Esses passos são explicados mais detalhadamente abaixo.

Passo 1 - Gerando o grafo de vizinhança de um nó

Dado uma URL u construimos um grafo direcionado com os nós próximos ao nó u no grafo Web. O grafo é constituído pelos seguintes nós e arestas:

1. u;

2. máximo de B nós em $\Gamma^{-}(u)$ e, para cada nó v pai de u, escolher até BF nós em $\Gamma^{+}(v)$ diferentes de u (mais detalhes em [19]); e

3. máximo de F nós em $\Gamma^{+}(u)$ e, para cada nó v filho de u escolher FB pais de v diferentes de u (mais detalhes em [19]).

Os valores para B, BF, F e FB utilizados em [19] foram adaptados com os experimentos realizados, não havendo valores máximos e mínimos limitando esses parâmetros. A estimativa dos autores é que para valores altos de B e F e valores baixos de BF e FB o algoritmo é mais eficaz.

No item 2, a escolha de B nós pais de u respeita os seguintes critérios: se u tiver mais do que B pais, escolhe-se B pais aleatoriamente; caso contrário, escolhe-se todos os pais de u. No mesmo item 2, a escolha de BF nós que sejam filhos dos pais de u respeita os critérios: se um pai x de u tiver mais do que BF +1

filhos, escolhe-se os $BF/2$ filhos que são apontados na página x pelos $BF/2$ links imediatamente anteriores ao link de u em x e, de forma semelhante, também os $BF/2$ filhos apontados na página x pelos $BF/2$ links imediatamente posteriores ao link que aponta para u em x. Se a página x tiver menos do que $BF + 1$ links, escolhe-se obviamente todos os links de x diferentes de u.

No item 3, a escolha de F nós filhos de u respeita os seguintes critérios: se u tiver mais do que F filhos, escolhe-se os filhos apontados pelos primeiros F links de u; caso contrário, escolhe-se todos os filhos de u. Para a escolha dos FB nós, segue-se os seguintes critérios: se um filho de u tiver mais do $BF + 1$ pais, escolhe-se os BF pais com maiores graus de entrada; caso contrário, escolhe-se todos os pais dos filhos de u.

Passo 2 - Eliminar nós duplicados

Dois nós são considerados duplicados se possuírem cada um mais do que 10 links e se possuírem ao menos 95% dos links em comum. Nesse caso, os dois nós são combinados em um único nó contendo todos os links de ambos.

Passo 3 - Computar pesos das arestas

Para calcular os valor dos pesos das arestas foi utilizado o algoritmo proposto por Bharat e Henzinger [26]. Para as arestas onde os nós de suas extremidades são de um mesmo domínio Web o peso é 0. Se uma página u for apontada por k páginas de um mesmo domínio Web, porém distinto do domínio de u, então para cada uma dessas arestas de entrada de u atribui-se o peso *authority* $\frac{1}{k}$. Se uma página u possui link para ℓ páginas de um mesmo domínio Web, porém distinto do domínio u, então para cada uma dessas arestas de saída de u atribui-se o peso *hub* $\frac{1}{\ell}$.

Passo 4 - Calcular valores para *hubs* e *authorities*

Neste passo, calcula-se os valores para *hubs* e *authorities* através do algoritmo *imp* de Bharat e Henzinger [26]. Este algoritmo é uma extensão do algoritmo HITS com pesos nas arestas. O cálculo dos valores de *hubs* e *authorities* é feito da seguinte forma:

Inicializar os elementos do vetor *hub* H com valor 1.0;
Inicializar os elementos do vetor *authority* A com valor 1.0;
enquanto *vetores* H *e* A *não convergirem* **faça**
  para todo *nó* n *do grafo de vizinhança* N **faça**
    $A[n] \leftarrow \sum_{(n' \rightarrow n) \in arestas(N)} H[n'] \times peso\_authority(n' \rightarrow n)$
  fim
  para todo n *em* N **faça**
    $H[n] \leftarrow \sum_{(n \rightarrow n') \in arestas(N)} A[n'] \times peso\_hub(n \rightarrow n')$
  fim
  Normalizar vetores A e H;
fim

Algoritmo 3: *Companion*: Cálculo dos valores de *hubs* e *authorities*

O retorno do algoritmo *Companion* são os 10 nós com os melhores valores de *authorities* calculados. De fato, este algoritmo é muito parecido com o algoritmo *HITS*. Uma das principais diferenças é a escolha de alguns nós considerando-se a ordem em que os links aparecem nas páginas. Na Seção 2.4.2 apresentamos o algoritmo *Cocitation* dos mesmos autores.

2.4.2 Algoritmo *Cocitation*

Dois nós são co-citados se tiverem um nó pai em comum. Chamamos esses nós de nós "irmãos" (ver Figura 2.5). O grau de co-citação de dois nós é a quantidade de nós pais que eles possuem em comum. Este algoritmo é bem simples e trabalha em cima desta idéia. O algoritmo é descrito como:

1. Selecionar os nós v_i que são filhos dos nós pais de u ($\Gamma^+(\Gamma^-(u))$);
2. Para cada nó v_i calcular o grau de co-citação entre ele e o nó u;
3. Retornar os 10 nós com maiores graus de co-citação em relação a u.

De fato, encontrar nós co-citados, em muitos casos, é bastante difícil. Por isso, foi implementada uma heurística na qual quando não há número suficiente de nós co-citados, retira-se uma parte do endereço URL usado. Por exemplo, se u for a URL *http://www.algoritmo.com/x/y/z* e não existir um número suficiente de nós co-citados então o algoritmo é reiniciado utilizando-se o novo endereço

http://www.algoritmo.com/x/y. A heurística é aplicada recursivamente até que se ache uma quantidade suficiente de nós co-citados ou que se chegue ao domínio da URL, que neste exemplo é *http://www.algoritmo.com*.

Capítulo 3

Algoritmos Propostos

Em diversos pontos deste trabalho enfatizamos a importância da estrutura dos links na Web para o relacionamento de páginas. Apresentamos diversos trabalhos relacionados para demonstrar este fato. No Capítulo 2 vimos definições e algoritmos importantes para o relacionamento de páginas Web. Neste capítulo, abordamos fatores que nos levaram a desenvolver novos algoritmos e mesclar alguns dos algoritmos existentes para serem utilizados no relacionamento de páginas.

Enquanto os algotimos *HITS* e *Companion* enfatizam o relacionamento entre *hubs* e *authorities*, o algoritmo *Cocitation* foca apenas no relacionamente entre páginas pela estrutura dos links. O *PageRank*, por sua vez, utiliza links com pesos normalizados e navegação na Web (*web surfing*) através de modelos de busca aleatória (*random walk models*). Todos eles demonstraram-se muito eficazes em seus respectivos testes.

Os algoritmos *HITS* e *Companion* possuem alguns problemas dos quais citamos os dois abaixo:

- Se a maioria das páginas do grafo de vizinhança analisado for de um tema diferente da página requerida, então os melhores *hubs* e *authorities* serão consequentemente de um tema diferente. Uma possível solução seria adicionar informações do texto da página visitada [26];

- Ambos consideram apenas parte dos links da Web, portanto a adição de algumas arestas a alguns nós pode modificar o resultado da ordenação dos *hubs* e *authorities* [27].

A solução dos dois pontos citados acima gera outros problemas. Por exemplo, a adição de informações sobre o texto da página implica no relacionamento de textos, que é por si só um problema bastante desafiador. Além disso, se aumentarmos o tamanho do grafo a ser analisado, aumentamos também o tempo de processamento do algoritmo. Encontrar o balanceamento ótimo entre eficiência versus eficácia é também um problema bastante complexo. Portanto levando em consideração a análise desses pontos na validação da eficiência desses algoritmos.

O algoritmo *Cocitation* possui um problema mais fundamental, pois a afirmação de que duas páginas são relacionadas se elas são co-citadas por uma mesma página não é sempre verdade. Por exemplo, links para *softwares* requeridos para visualização de uma página, na maioria das vezes, não se relacionam com os outros links da mesma página. Uma solução seria a utilização do texto das páginas a serem relacionadas, o que nos remete mais uma vez ao problema de relacionamento de texto.

A eficiência do algoritmo *PageRank* está relacionada ao seu pré-processamento. Para recordarmos, a Web é representada por um grafo direcionado onde as páginas e os links são representados pelos nós e arestas do grafo, respectivamente. Além disso, para cada aresta do grafo associa-se um peso. Este peso é normalizado e o seu cálculo é feito utilizando-se um modelo de busca aleatória explicado na Seção 2.3. Este cálculo e sua respectiva normalização são responsáveis pelo pré-processamento das informações do grafo, tornando inviável a utilização deste algoritmo em computadores de pequeno porte.

Temos como objetivo criar algoritmos que mesclem características dos algoritmos discutidos no Capítulo 2. Não temos como objetivo reinventar algoritmos para desqualificar os algoritmos já existentes nem tampouco corrigir os problemas encontrados em cada um deles. Queremos unificá-los em uma ferramenta que indique páginas relacionadas às visitadas pelo usuário.

Todos os algoritmos listados no Capítulo 2 são de certa forma relacionados, pois todos procuram de certa forma ordenar as páginas pelos seus graus de entrada, incluindo o algoritmo *PageRank* [28]. Por isso, todos os nossos algoritmos priorizam nós com alto grau de entrada.

O modelo de busca aleatória referenciado por Brin e Page representa a navegação pela Web de um usuário qualquer. Ou seja, um usuário inicializa a navegação em

uma página qualquer, transita para uma outra página apontada pela página inicial e segue visitando as páginas usando a estrutura de links da Web. A qualquer momento o usuário pode também decidir visitar uma página que não é apontada pela página atual. Este modelo é representado pelo algoritmo *PageRank* e pode ser visto como a distribuição estacionária da cadeia de Markov definida pelo conjunto de nós e arestas que representa a Web.

Cada transição da cadeia de Markov é representada pelas arestas do grafo, e a probabilidade de transitar por uma certa aresta é especificada, no algoritmo *PageRank*, pelo peso dessa aresta no grafo. Então, a probabilidade de um usuário navegar de uma página A para uma página B é definida pelo peso da aresta $(A \rightarrow B)$.

A representação da Web como um grafo direcionado é altamente explorada na literatura [17, 18, 21, 19, 26, 28, 29, 30]. Da mesma forma, utilizamos um grafo direcionado para representar a estrutura de links da Web e chamamos esse grafo de grafo Web. O grafo Web que utilizamos em nossos testes foi armazenado e capturado utilizando um *Web proxy*. Este armazenamento e a captura dos dados é detalhada no Capítulo 4.

Em nossos algoritmos estão presentes os conceitos de grau de entrada de um nó e modelo de busca aleatória. O modelo de busca aleatória define o método de iteração de nossos algoritmos, pois cada iteração simboliza uma transição na cadeia de Markov do grafo Web.

Em linhas gerais nosso algoritmo consiste em pegar o nó do grafo Web que representa a página atual e, a partir dele, executar uma busca aleatória por um certo número pré-determinado de passos. A página visitada ao final da busca é a saída do algoritmo.

Como vimos anteriormente, dado um nó u com um filho v, a transição de u para v depende do peso da aresta $(u \rightarrow v)$. Estudamos três métodos para a computação do peso desta aresta:

Alg01. Grau de entrada do nó v (ver Figura 3.1);

Alg02. Grau de saída dos pais do nó v (ver Figura 3.2);

Alg03. Grau de entrada dos pais do nó v (ver Figura 3.3).

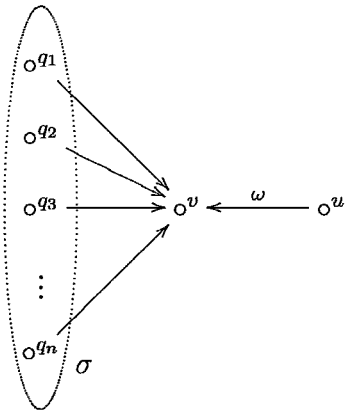


Figura 3.1: Peso da aresta $u \rightarrow v : \omega =$ número de nós do conjunto σ

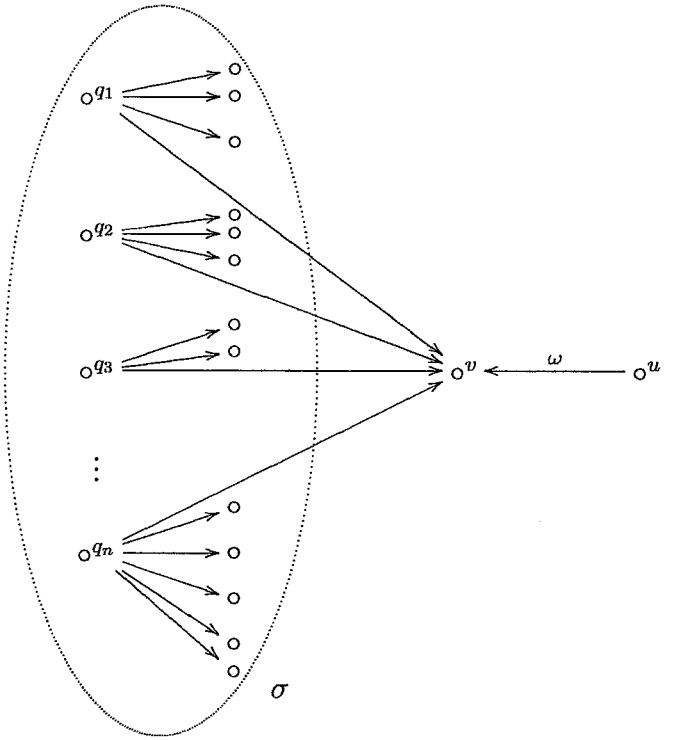


Figura 3.2: Peso da aresta $u \rightarrow v : \omega =$ grau de entrada dos nós do conjunto σ

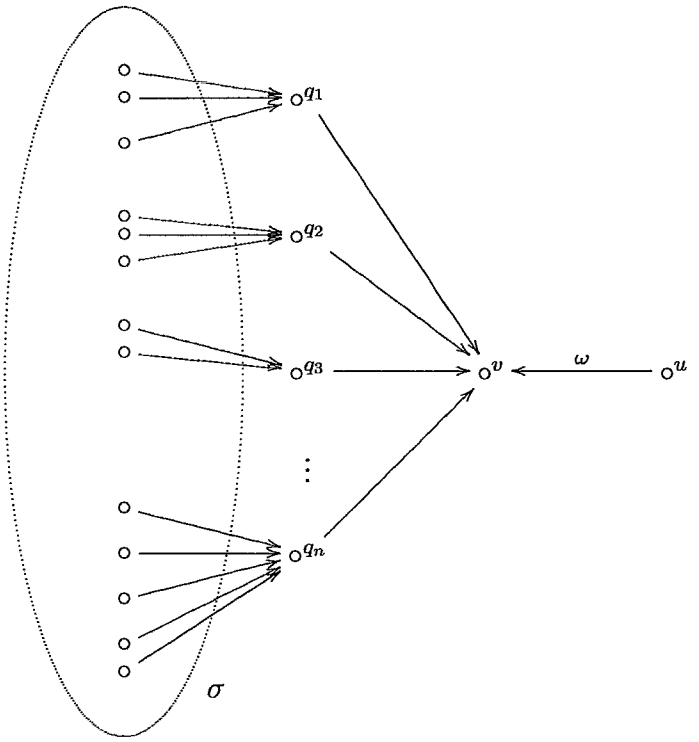


Figura 3.3: Peso da aresta $u \rightarrow v : \omega =$ grau de saída dos nós do conjunto σ

O método Alg01 prioriza os nós considerados bons *authorities*, atribuindo maiores pesos aos nós mais referenciados. A discussão de que este método é bom baseia-se na idéia mencionada no Capítulo 2 de que páginas muito referenciadas tendem a possuir maior qualidade na apresentação da informação e que quando uma página v é referenciada por uma página u, o autor da página u está de certo modo dando "autoridade" à página v.

O método Alg02 se parece muito com o método Alg01, pois também valoriza os nós considerados bons *authorities*. Entretanto, este método acrescenta mais um parâmetro de confiança, que é a certeza de que o nó também é apontado por bons *hubs*. Ou seja, dado um nó v, os pais de v devem ser bons *hubs* (grau de saída alto) e o nó v precisa ser um bom *authority*. Esta idéia foi discutida nas Seções 2.2 e 2.4.

O método Alg03 inverte a idéia do método Alg02 em relação ao pai do nó v. Ao invés de valorizar os nós cujos pais têm grau de saída alto, ele valoriza os nós cujos pais possuem graus de entrada alto, ou seja, os pais de um nó v qualquer devem ser bons *authorities*. Para ilustrar a idéia por trás desse método, considere páginas de um mesmo domínio, sendo uma delas de conteúdo mais restrito. A página com conteúdo mais restrito pode ter poucas referências externas, porém seus pais podem vir a ter muito mais referências, por possuírem um conteúdo mais amplo. Por exemplo, conside o site da CBF [1], um site sobre *futebol* que, por se tratar do órgao máximo de futebol do Brasil, é referenciado por diversos outros sites no mundo. Considere também um dos tópicos deste site sobre arbitragem (*http://www.cbf.com.br/sitearbitros*). Se um link A aponta para o site de arbitragem da CBF, por seu pai (site da CBF) ser muito referenciado, o peso do link A será alto, diferentemente dos métodos anteriores.

Inspirado no método *Companion* discutido na Seção 2.4.1, adicionamos ao algoritmo uma heurística para priorizar links entre páginas de diferentes domínios Web. Em cada passo da busca aleatória, se a página possuir mais de 10 links para páginas em seu próprio domínio, então com probabilidade $0,9$ consideramos apenas os links para páginas de domínios Web distintos da página atual, forçando a busca a transitar para um outro domínio Web. Com probabilidade $0,1$ consideramos todos os links da página atual.

[1] Confederação Brasileira de Futebol, órgão máximo do futebol brasileiro - *http://www.cbf.com.br*.

Após a definição dos métodos e do modelo de busca aleatória, apresentamos o nosso algoritmo iterativo para o relacionamento de páginas Web. A entrada do algoritmo é uma URL e a quantidade de passos a serem realizados sobre a cadeia de Markov:

Entrada: URL, numPassos;

Saída: $URL_{resposta}$;

1 $u \leftarrow URL$;
2 $resposta \leftarrow u$;
3 $\rho_{total} \leftarrow 1$;
4 **enquanto** $numPassos <> 0$ **faça**
5 | *HeuristicaDeCorte(u)*;
6 | **para cada** *filho v de u* **faça**
7 | | $\omega \leftarrow$ peso de $(u \rightarrow v)$ de acordo com o método escolhido (Alg01, Alg02 ou Alg03);
8 | | $\rho_{total} \leftarrow \rho_{total} + \omega$;
9 | | $\rho \leftarrow (\omega / \rho_{total})$;
10 | | $\beta \leftarrow chooseEdge(\rho)$;
11 | | **se** $\beta = verdadeiro$ **então**
12 | | | $URL_{resposta} \leftarrow v$;
13 | | **fim**
14 | **fim**
15 | $numPassos \leftarrow numPassos - 1$;
16 | $u \leftarrow URL_{resposta}$;
17 **fim**

Algoritmo 4: Algoritmo Iterativo RS

No passo 5 do Algoritmo 4 chama-se a função *HeurísticaDeCorte(u)* que recebe como parâmetro uma URL. Esta função implementa a heurística que discutimos anteriormente para priorizar os links para domínios Web diferentes do domínio de u. Isso ajuda a evitar que o algoritmo fique preso dentro de um mesmo domínio caso encontre páginas com *botões voltar*, *links de menus*, *links para voltar ao topo da página* etc.

No passo 10 do Algoritmo 4 chama-se a função *chooseEdge(ρ)* que recebe como parâmetro um valor ρ entre 0 e 1. Este valor ρ representa a probabilidade de

transição da página u para a página v. Nesta função pega-se um número aleatório entre 0 e 1 e, caso ele seja inferior ao valor ρ, retorna-se verdadeiro, indicando uma possível transição para a página v. Caso contrário, retorna-se falso.

Os dados de entrada do algoritmo são uma URL e a quantidade de passos a serem executados pela cadeia de Markov. A quantidade de passos é importante para o nosso algoritmo pois é este dado que diferencia o nosso algoritmo de todos os outros apresentados. Um dos nossos objetivos é discutir a quantidade de passos máxima que podemos caminhar sobre a cadeia de Markov e, ainda assim, garantirmos o relacionamento entre páginas.

Enquanto que um valor muito pequeno para o número de passos faz a busca aleatória ficar muito restrita a poucas páginas, um valor excessivamente grande para o número de passos é igualmente ruim. Note que a cadeia de Markov definida por esse processo é ergódica e, portanto, converge para uma distribuição estacionária única que independe do nó onde a busca é iniciada. Portanto, se o número de passos for muito elevado, o algoritmo retornará uma página aproximadamente de acordo com a distribuição estacionária dessa cadeia de Markov e, portanto, a saída do algoritmo será de certa forma independente da URL fornecida de entrada.

O Capítulo 4 discute o processo involvendo a implementação desses algoritmos, incluindo a fase de captura dos dados para testes.

Capítulo 4

Implementações

4.1 Introdução

A fim de avaliar a eficácia dos algoritmos apresentados no Capítulo 3, precisamos obter uma pequena porém significativa amostra da estrutura dos links da Web. Para isso implementamos algumas ferramentas, sendo a principal das quais um *Web proxy*[1] que captura os links da página atual sendo visitada.

Este capítulo apresenta as ferramentas utilizadas na implementação dos algoritmos vistos no Capítulo 3. Na Seção 4.2 descrevemos uma arquitetura para servidores Web que é a base das implementações descritas neste capítulo. Na Seção 4.3 descrevemos a implementação de um *Web proxy*, o qual foi utilizado na captura dos dados de testes deste trabalho. A captura dos dados de testes e a obtenção dos resultados foram implementados em momentos distintos, por isso dedicamos uma seção para cada uma dessas etapas. Dessa forma na Seção 4.4 apresentamos a classe que captura as informações da Web (ou seja, os dados de testes) e na Seção 4.5 discutimos a implementação da classe de obtenção dos resultados a partir dos dados de teste.

[1] *Web proxy* é um tipo de servidor que atua como intermediário, recebendo requisições clientes (usualmente provenientes de protocolo HTTP) e as repassando ao servidor de destino.

4.2 *Brazil Framework*

O *Brazil Framework* [31] é uma arquitetura para servidores Web mantido pela SUN [32] que visa conectar pessoas a informações, sistemas ou outros dispositivos computacionais de forma segura e fácil. A arquitetura foi desenvolvida com a linguagem de programação Java e, por ser uma linguagem orientada a objetos, possui inúmeras facilidades de reuso de código.

O *Brazil Framework* é constituido principalmente de dois objetos denomindados Servidor e Requisição e uma interface chamada *Handler*. Cada um destes objetos possui características importantes para o *framework*, sendo a principal delas o recebimento da requisição do usuário e o envio da resposta gerada por essa requisição. O protocolo para envio e recebimento das requisições dos usuários é o HTTP, o qual é protocolo mais utilizado na Web [33]. O *handler* é uma interface que tem por objetivo prover um mecanismo onde funcionalidades podem ser adicionadas à arquitetura do *Brazil Framework*.

O projeto *Brazil* foi desenvolvido para servir como um portal ou agregador de conteúdo. Com esse objetivo, ele foi desenvolvido a fim de servir de intermediário entre os servidores de conteúdo e os usuários, possibilitando a personalização de páginas Web e conteúdos diversos e, além disso, sendo flexível o suficiente para se ajustar a praticamente todo tipo de material atualmente disponível na Web.

A implementação da arquitetura do *Brazil Framework* baseia-se na flexibilidade e portabilidade. A portabilidade se deve basicamente ao fato dela ter sido implementada na linguagem Java, que é independente de plataforma e, portanto, totalmente portável. A flexibilidade da arquitetura deve-se ao fato de que os desenvolvedores podem utilizá-la de três formas distintas: *servidor Web*, *micro-servidor* ou *meta-servidor*.

Cada uma dessas formas possui a sua própria função. Os servidores Web, por exemplo, são responsáveis por receber requisições dos navegadores e enviar como resposta a página Web solicitada. Os meta-servidores, também comumente chamados de *Web proxies*, servem como interface entre os navegadores e servidores Web. Sua função se restringe basicamente a receber a requisição dos navegadores e repassá-la ao servidor Web apropriado. Os *micro-servidores*, em contrapartida, são servidores que possuem pequenos materiais mas que não são suportados por

navegadores. Por esse motivo estes servidores precisam trabalhar em conjunto com os *meta-servidores.*

O emprego desta arquitetura como um *meta-servidor* é um dos pilares deste trabalho. Essencialmente, esta abordagem permite que o servidor utilize conteúdos de milhares de servidores Web existentes. Esta implementação nos permite buscar informações de outros servidores Web e apresentá-las num formato visual padronizado e personalizado para o site. Resumidamente, a tecnologia responde a uma requisição URL feita por um navegador Web, recupera o conteúdo em pequenas partes (que podem estar localizados em vários servidores diferentes), analisa e junta todo o conteúdo em uma só página, faz as transformações visuais necessárias e envia o resultado para o usuário.

As inúmeras funcionalidades e a enorme flexibilidade do *Brazil Framework* fizeram com que ele se tornasse uma ferramenta essencial para este trabalho. A captura de informações de páginas Web e a análise do conteúdo destas páginas são dois passos importantes para o armazenamento e a avaliação da estrutura dos links da Web. Como discutido nos Capítulos 1 e 2, a exploração da estrutura dos links da Web atinge diretamente o foco principal desta dissertação.

Na Seção 4.3 discutimos a implementação de um *meta-servidor* ou *Web proxy* implementado sobre a arquitetura do *Brazil Framework.* Na Seção 4.4 apresentamos a implementação de uma classe sobre essa arquitetura. Esta classe é a responsável pela captura da estrutura de links das páginas Web.

4.3 *Pro-Active Webfilter*

Pro-active Webfilter (*PAW*) é um *Web proxy* de código aberto e que pode ser modificado sob a licensa do Apache [34]. É um projeto desenvolvido com tecnologia Java baseado no *Brazil Framework.* Haja vista que ambos são desenvolvidos com tecnologia Java, o servidor *PAW* possui a mesma portabilidade do *Brazil Framework.*

O *PAW* é uma ferramenta de fácil utilização, tanto para usuários quanto para desenvolvedores, e pode ser dividido em dois componentes básicos:

- Servidor *PAW* - responsável pela filtragem das requisições HTTP e resposta ao usuário;

- GUI *PAW* - ferramenta gráfica para administrar o servidor *PAW*;

Por se tratar de um *Web proxy*, ele permite a manipulação de requisições HTTP e de respostas no formato HTML. Essas manipulações são feitas através de *handlers* e de *filters*. Sucintamente, os *handlers* são responsáveis pelas requisições HTTP, ao passo que os *filters* se encarregam do envio dos dados no formato HTML para o usuário.

A interface gráfica é de fácil utilização, o que propicia a manutenção do servidor e dos *filters* e *handlers* de maneira rápida e eficiente. Isto advem do fato de que estes dois componentes possuem arquivos de configuração escritos na linguagem XML.

Na próxima seção é apresentada a implementação de um *filter*, que é utilizado para recuperar a estrutura de links da Web através da análise do documento HTML gerado para ser enviado ao usuário.

4.4 Captura dos Dados

O crescimento desordenado da World Wide Web ainda é instrumento de pesquisa em todo o mundo. Hoje em dia, a utilização da Web por milhões de usuários dificulta a coleta de dados estruturais como um todo. Milhares de pessoas publicam informações a todo momento. Publicar informações significa atualizar sites ou criar novos endereços URLs. Essas atividades são efetuadas pelos próprios usuários e a validação dos links existentes na página não é feita por qualquer ferramenta ou compilador.

As páginas Web, em sua maioria, são criadas utilizando-se a linguagem HTML (Hypertext Markup Language), a qual não recebe qualquer validação de seus links. Quando falamos de validação de um link, nos referimos à existência da URL apontada por este link. Isto deve-se ao fato de que a linguagem HTML é uma linguagem limitada à marcação de texto e não possui um compilador[2] ou um interpretador.[3] Portanto, criar ferramentas que varrem o código em busca

[2]Programas que leêm o código fonte de uma linguagem de programação e o transforma em linguagem de máquina para ser executado pelo processador. Após compilados, os programas podem ser executados inúmeras vezes.

[3]Programas que leêm o código fonte de uma linguagem de programação e o transforma em linguagem de máquina, sendo que esta transformação acontece imediatamente antes que cada

de padrões (chamados de *parser*) é um trabalho árduo e exaustivo. Nesta seção apresentamos um *parser* de documentos HTML que buscam endereços URL dentro destes documentos.

Os diversos formatos de apresentação de links admitidos pela linguagem HTML corroboram a idéia de difícil criação de um *parser* para essa linguagem. Este não é o principal problema para se criar um parser e talvez seja o menor deles. Os grandes problemas são realmente os abundantes erros de digitação e de sintaxe existentes nas páginas Web. A Figura 4.1 ilustra um erro de sintaxe frequentemente encontrado nas páginas Web e largamente aceito pelos navegadores Web modernos. Na Figura 4.2 mostramos a maneira correta com que esse exemplo deve ser escrito de acordo com as definições HTML.

```
<html>
        <head>
                <title>Página de Erro de Sintaxe HTML</title>
        </head>
        <body>
                <font size='+2'>Erros de Sintaxe em arquivo HTML</font><br>
                <a href='http://www.cos.ufrj.br/~vinicius'>Página Oficial de Vinicius Ramos
                <a href=http://www.cos.ufrj.br >PESC/COPPE</a>
                <br>
                Final da página.
        </body>
</html>
```

Figura 4.1: Exemplos de erros de sintaxe em um arquivo HTML

Em documentos HTML, as marcações do texto são feitas por *tags* que devem aparecer entre os caracteres < e >. Em sua maioria, as marcações são delimitadas por um *tag* de abertura e um *tag* de fechamento, como por exemplo e , respectivamente (para maiores detalhes sobre a especificação HTML, veja [35]).

Na Figura 4.1 está faltando o fechamento da tag <a>, o que caracteriza um erro de sintaxe. Porém isso não inviabiliza a visualização do documento pelo navegador, o qual é capaz de mostrar salva uma ligeira diferença de formato em relação a

linha do programa seja executada. Assim, a cada execução do programa, uma nova execução do interpretador se faz necessária.

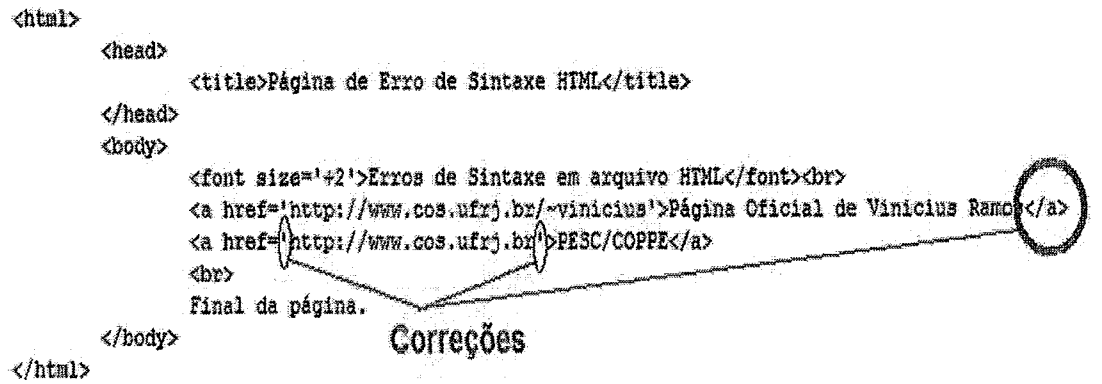


Figura 4.2: Correções de sintaxe do HTML da Figura 4.1

versão com a *tag* de fechamento . O problema de visualização da página Web é mínimo pois, apesar de a página não ser apresentada exatamente como o seu criador gostaria que fosse, ela continua sendo válida e o link também continua sendo válido. A grande dificuldade é na criação de um *parser* HTML para interpretar páginas com esses erros de sintaxe.

Diferente de linguagens de programação que precisam ser compiladas ou interpretadas, a linguagem HTML é apenas para marcação de textos. Por isso, os erros de sintaxe passam despercebidos em HTML, pois a linguagem não possui qualquer verificação anterior à sua publicação. Ou seja, a linguagem HMTL não é interpretada nem compilada, o que facilita a existência de erros de sintaxe conforme os descritos acima.

Como vimos, erros de digitação e erros de sintaxe podem ocorrer em documentos HTML. Podemos prever alguns erros de sintaxe, porém isso não é possível para os erros de digitação. Portanto, ao recolhermos os dados de testes não validamos os endereços URL obtidos. A validação, ou melhor, a verificação da existência do endereço URL só é feita numa etapa posterior, a ser discutida na Seção 4.5.

O *parser* HTML navega pelas diversas *tags* do documento. Na verdade, o *parser* não busca por *tags* HTML especificamente, mas sim por qualquer *tag*, ou seja, qualquer string que esteja entre < e >. Sendo assim, ao encontrarmos as *tags* <a> e , buscamos por seus atributos "href" ou "onclick" e os valores que

eles possuem são precisamente os links apontados pela página Web que está sendo visitada.

Para solucionarmos os problemas apresentados anteriormente (erros de digitação e de sintaxe) utilizamos uma heurística quando o *parser* busca os atributos "href" e "onclick" dentro das tags <a> e . Caso o valor do atributo não esteja delimitado por aspas, por exemplo $href = "www.cbf.com.br"$, o *parser* recupera o valor do atributo apenas até o primeiro espaço vazio. Caso o *parser* não encontre uma *tag* de fechamento , o programa recupera o texto do link até a primeira *tag* após a *tag* <a>. Estas duas heurísticas mostraram-se muito eficientes na solução da maioria dos problemas mencionados.

Como esta classe foi implementada sobre o servidor *PAW*, a sua administração é consequentemente de fácil utilização. Logo, para desabilitá-la bastam-se alguns *clicks* sobre a GUI *PAW*. Além disso, esta GUI permite ordenar os filtros que são aplicados as páginas Web visitadas. A ordenação de filtros permite a adição de novas funcionalidades e aplicação de precedência entre estes filtros, assim filtros podem tratar as informações já alteradas por um dos que antecederam ele. Por exemplo, a implementação da classe para obtenção dos resultados apresentados na Seção 4.5 também foi implementada como um *filter* do servidor *PAW* e é chamada por ele após a recuperação das informações pelo *parser*.

O armazenamento das informações é feito em uma base de dados MySQL [36] por ser de distribuição gratuita e por atender a todas as nossas necessidades quanto à modelagem dos dados. Esta base de dados é distribuída sob a licensa GPL [37]. Outro fator importante na escolha MySQL foi a sua portabilidade, já que ele é disponibilizado em versões para vários sistemas operacionais, proporcionando maior flexibilidade e portabilidade quando utilizado em conjunto com a linguagem de programação Java.

4.5 Obtenção dos Resultados

Nesta seção apresentamos duas classes Java que têm o mesmo objetivo: obter os resultados do relacionamento de páginas utilizando-se os algoritmos descritos no Capítulo 3. Estas classes não são complementares, mas sim idênticas no que diz respeito à sua finalidade. A diferença entre elas é que uma pode ser utilizada *offline*,

enquanto que a outra só pode ser utilizada *online* e, portanto, depende da execução do servidor *PAW*. Nesta seção, iremos nos referir a essas classes como *classe online* e *classe offline*.

A criação das *classes online* e *offline* deve-se ao fato de diminuir o tempo de processamento gerado pelos algoritmos descritos no Capítulo 3. Para termos um melhor desempenho na captura dos dados de teste e na obtenção dos resultados, criamos uma classe para ser utilizada em uma aplicação *desktop*.

Ambas as classes são implementadas com a tecnologia Java e transformam o resultado dos algoritmos em registros de uma tabela da base de dados MySQL. A grande diferença é que na *classe online* o resultado, além de ser armazenado na base de dados MySQL, também é indicado para o usuário como um link na página requisitada.

A etapa de obtenção dos resultados é a mais importante para a validação dos algoritmos propostos e a sua implementação deve utilizar as tecnologias existentes para diminuir o tempo de execução do algoritmo. No Capítulo 5 apresentamos os resultados obtidos.

Capítulo 5

Resultados

Nesse capítulo apresentamos alguns resultados iniciais sobre a eficácia do algoritmo proposto no Capítulo 3. Lembre que esse algoritmo recebe uma URL como entrada, chamada de página atual, e fornece uma outra URL de resultado que, de acordo com os critérios intuitivos descritos em profundidade no Capítulo 3, tende a ser semelhante à página atual. Em linhas gerais, esse algoritmo pega a página atual e realiza uma busca aleatória pelo grafo Web a partir dessa página.

Um parâmetro crucial para a performance do algoritmo consiste no número de passos que o algoritmo performa durante essa busca. Nosso objetivo principal nesse capítulo é mostrar os desafios que a busca por páginas relacionadas oferece e apontar o impacto que o número de passos possui na qualidade dos resultados.

A fim de avaliar o algoritmo proposto selecionamos três endereços Web de conteúdo bem distintos: `java.sun.com`, que é uma página destinada aos desenvolvedores da linguagem de programação Java, `brasil.gov.br`, página oficial do Governo Federal Brasileiro e `cos.ufrj.br`, página do Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ. Para cada uma dessas páginas, executamos nosso algoritmo por várias rodadas, sendo ao todo 1167 rodadas para `java.sun.com`, 833 para `brasil.gov.br` e 1000 para `cos.ufrj.br`.

Utilizamos as ferramentas discutidas no Capítulo 4 para capturar a parte do grafo Web que se localiza na vizinhança dessas três páginas. Para isso, utilizamos o programa Wget em conjunto com o *Web proxy* PAW. O Wget foi utilizado para descobrir os nós que se localizam na vizinhança das páginas `java.sun.com`, `brasil.gov.br` e `cos.ufrj.br`. O PAW, por sua vez, foi utilizado para fazer o

parser detalhado da página e a montagem do subgrafo do grafo Web que foi utilizado nos testes realizados nesse capítulo.

Durante a coleta dos dados notamos que alguns servidores Web detectam o programa Wget e ignoram as requisições provenientes desse programa. Isso dificulta a obtenção do grafo de vizinhança. Dentre os nossos testes, os servidores da `coppe.ufrj.br` e do `cnpq.br` são os únicos que implementam essa política.

A fim de observar o impacto do número de passos na performance do algoritmo, optamos por realizar nossos testes utilizando quatro valores para o número de passos: 1, 3, 5 e 20. Além disso, para cada rodada, testamos o algoritmo usando cada um dos três métodos discutidos no Capítulo 3. Portanto, em suma, para cada uma das três páginas Web, o algoritmo foi rodado diversas vezes, em cada uma das quais utilizamos os três métodos e quatro distintos valores para o número de passos. Dividimos a apresentação desse capítulo em três partes, cada uma delas sendo reservada para a discussão dos resultados de um desses métodos.

5.1 Método Alg01

Nesse método, a cada passo do algoritmo selecionamos um link com probabilidade proporcional ao grau de entrada da página a qual esse link aponta. As tabelas 5.1, 5.2 e 5.3 apresentam os resultados para esse método para as páginas `java.sun.com`, `brasil.gov.br` e `cos.ufrj.br`, respectivamente. A fim de deixar a apresentação mais clara, eliminamos as páginas que foram sugeridas menos de 10 vezes para cada valor do número de passos.

Os resultados para a página `java.sun.com` (Tabela 5.1) mostram claramente dois grupos distintos de páginas Web. O primeiro grupo são aquelas páginas que possuem alto grau de saída no grafo Web, os chamados *hubs*. Quando a busca aleatória atinge um *hub*, raramente a busca retorna para o mesmo *hub* num passo subsequente. Esse comportamento pode ser claramente identificado, por exemplo, nas linhas 1, 2 e 13 da Tabela 5.1. A característica principal desse grupo é que o número de vezes que a página é visitada pelo algoritmo tende a diminuir quando o número de passos é aumentado. Por exemplo, o primeiro passo da busca aleatória foi mais de 49% das vezes para a página da Sun (linha 1). Porém, ao continuar a busca, a página da Sun não é visitada tão frequentemente. A página *sun.com/training* (linha 13) é

outro bom exemplo. Ela não pode ser alcançada pela página java.sun.com em um só passo. Após 3 passos ela é alcançada 49 vezes, porém com 20 passos ela nunca é alcançada.

URL: `java.sun.com`		No de Passos			
#	**Sugestão**	**1**	**3**	**5**	**20**
1	`http://www.sun.com`	573	73	19	0
2	`http://developers.sun.com`	464	137	39	0
3	`http://www.sun.com/software/products/mysql/getit.jsp`	79	13	3	0
4	`http://blogs.sun.com`	9	14	15	23
5	`http://sunsolve.sun.com`	0	290	362	412
6	`https://portal.sun.com/portal/dt`	0	228	250	286
7	`http://shop.sun.com/cart`	0	200	250	238
8	`http://blogs.sun.com/bigadmin`	0	27	47	46
9	`http://www.sun.com/contact`	0	2	10	12
10	`https://suned.sun.com/APPS/US/SESWEB`	0	6	11	12
11	`http://www.sun.com/products`	0	5	3	10
12	`http://www.sun.com/communities`	0	3	10	4
13	`http://www.sun.com/training`	0	49	29	0

Tabela 5.1: Páginas sugeridas pelo método Alg01 para a URL java.sun.com.

Um outro grupo completamente oposto que pode ser observado na Tabela 5.1 é o de páginas que bloqueiam a busca. Veja por exemplo, a página **`sunsolve.sun.com`** (linha 5). Quanto maior o número de passos executados pela busca aleatória, mais vezes essa página foi visitada. Isso ocorre principalmente pois essa página possui poucos links para outros domínios e, portanto, várias vezes a busca alcançou essa página e permaneceu nela até o final da busca. Comportamento semelhante pode ser observado para as linhas 6, 7 e 8. É importante ressaltar que, de uma forma geral, apesar de o número de visitas a essas páginas aumentar com o número de passos, esse aumento tende a ser cada vez menor. Isso ocorre principalmente por que, quando o número de passos é muito grande, a busca aleatória se aproxima de

sua distribuição estacionária e, portanto, o número de visitas a uma determinada página tende a se estabilizar.

Em relação ao conteúdo, podemos observar que a grande maioria das sugestões foram para páginas do site da Sun. Isso não é surpreendente e, de fato, é realmente muito difícil para uma busca aleatória deixar a página da Sun, dado o tamanho imenso de seu site Web. Além disso, a página da Sun possui páginas que são consideradas pelo nosso algoritmo como de domínios distintos, como é o caso das páginas `java.sun.com`, `sun.com`, `developers.sun.com` e `blogs.sun.com`. Essas páginas estão todas interligadas com o site da Sun e, portanto, a busca aleatória acaba ficando presa dentro dessa imensa rede de páginas mesmo utilizando-se a heurística para procurar outros domínios Web.

A Tabela 5.2 mostra os resultados obtidos para a URL `brasil.gov.br`, que são bem mais homogêneos que os obtidos para `java.sun.com`. Um fato um tanto curioso é que a página mais sugerida pelo algoritmo foi `combatadengue.com.br`, o que mostra a importância desse tema na situação atual do Brasil. Aparentemente a principal característica dos resultados mostrados na Tabela 5.2 é a diversidade do conteúdo das páginas Web sugeridas. Por exemplo, além de uma página sobre a Dengue, podemos encontrar páginas sobre a previdência social, medicamentos, saúde pública, concursos públicos, receita federal, polícia federal e até olimpíada de lingua portuguesa. Isso evidencia que a vizinhança do grafo Web em torno da página `brasil.gov.br` tende a ser até certo ponto homogênea com relação aos graus de entrada.

Os últimos resultados para o método Alg01 são apresentados na Tabela 5.3 para a URL `cos.ufrj.br`. Conforme esperado, a maioria das sugestões ficou concentrada nas páginas da UFRJ, COPPE e CNPq (linhas 1, 2 e 3). Podemos observar também a existência de *hubs* como a FAPERJ e CAPES (linhas 4 e 5). A princípio, poderíamos esperar resultados semelhantes para os sites da CAPES e do CNPq, porém há um ponto crucial que precisa ser destacado. No site da CAPES, com execeção de um único link para `brasil.gov.br`, todos os demais links apontam para uma página no domínio da CAPES, inclusive os link para o MEC, CNE, INEP e o próprio CNPq. Eles são apontados para uma página dentro do domínio `capes.gov.br` que faz o redirecionamento ao site dessas instituições. Portanto, quando a busca aleatória atinge o site da CAPES, com alta probabilidade

URL: brasil.gov.br		No de Passos			
#	Sugestão	1	3	5	20
1	http://www.combatadengue.com.br	90	82	75	102
2	http://bvsms.saude.gov.br/bvs/pacsaude	84	79	67	78
3	http://www010.dataprev.gov.br/cws/contexto/consit02/index.html	70	87	81	70
4	http://www.brasil.gov.br	46	0	0	0
5	http://www.previdencia.gov.br/pg_secundarias/beneficios_05.asp	43	40	44	41
6	http://www.anvisa.gov.br/medicamentos/bulas/index.htm	42	36	47	45
7	http://portal.saude.gov.br/portal/saude/cidadao/area.cfm	41	36	39	44
8	http://www.previdencia.gov.br/pg_secundarias/beneficios_03.asp	41	42	41	40
9	http://agenciact.mct.gov.br/index.php/content/view/47239.html	41	36	41	29
10	http://www.servidor.gov.br/concursos/index.htm	40	40	44	43
11	http://www.receita.fazenda.gov.br/Aplicacoes/ATCTA/CPF/ConsultaPublica.asp	40	44	34	35
12	http://portal.mec.gov.br/index.php	39	44	38	52
13	http://olimpiadadelinguaportuguesa.mec.gov.br	39	42	63	38
14	http://www010.dataprev.gov.br/cws/contexto/consit/consit1.html	38	39	41	42
15	http://www.previdencia.gov.br/pg_secundarias/beneficios_06.asp	34	34	40	40
16	https://sinarm.dpf.gov.br/sinarm/sinarm	32	42	45	49
17	http://www.previdencia.gov.br/pg_secundarias/beneficios_02.asp	30	48	41	35
18	http://www.brasil.gov.br/pac	7	10	9	10

Tabela 5.2: Páginas sugeridas pelo método Alg01 para a URL brasil.gov.br.

ela visita o site `brasil.gov.br` que é um site altamente difusor. Por outro lado, o site do CNPq possui alguns links para domínios externos ao `cnpq.br` que, por sua vez, possuem links retornando ao próprio CNPq. Isso faz com que a busca aleatória se mantenha próxima ao site do CNPq.

URL: `cos.ufrj.br`		No de Passos			
#	**Sugestão**	**1**	**3**	**5**	**20**
1	`http://www.ufrj.br`	352	321	356	370
2	`http://www.coppe.ufrj.br`	225	244	208	223
3	`http://www.cnpq.br`	164	167	178	148
4	`http://www.faperj.br`	144	6	0	0
5	`http://www.capes.gov.br`	107	0	0	0
6	`http://www.rederio.br`	0	105	101	121
7	`https://infaperj.faperj.br/efap/tasker.dll/Login`	0	30	31	28
8	`http://bvsms.saude.gov.br/bvs/pacsaude`	0	21	14	16
9	`http://www.combatadengue.com.br`	0	16	12	12
10	`http://www.anvisa.gov.br/medicamentos/bulas/index.htm`	0	7	2	11
11	`http://www.receita.fazenda.gov.br/Aplicacoes/ATCTA/CPF/ConsultaPublica.asp`	0	2	10	4

Tabela 5.3: Páginas sugeridas pelo método Alg01 para a URL cos.ufrj.br.

Um outro fenômeno importante a ser observado nesses resultados é que o site da Rede Rio (linha 6 da Tabela 5.3) é considerado uma página relacionada ao site `cos.ufrj.br`, porém não há nenhum link direto da COS para a Rede Rio. Isso mostra a importância de se considerar uma busca com mais passos.

Uma outra característica importante a ser observada para `cos.ufrj.br` é que algumas das sugestões fornecidas pelo algoritmo não têm, a princípio, relação com o tema principal do site `cos.ufrj.br`, que é educação. Por exemplo, sites relacionados a saúde pública, Dengue, medicamentos e até receita federal (linhas 8 à 11) foram alcançados basicamente através do site `brasil.gov.br` que, por sua vez, pode ser

alcançado, por exemplo, a partir do site da CAPES. Isso evidência a importância de se considerar também informação de contexto a fim de evitar que a busca divirja da rota correta.

5.2 Método Alg02

Recorde que nesse método, em cada passo da busca aleatória, uma página é selecionada com probabilidade proporcional a soma dos graus de saída dos seus pais. Portanto a gente espera que páginas cujos pais são bons *hubs* apareçam em evidência nos resultados. As tabelas 5.4, 5.5 e 5.6 mostram os resultados obtidos por esse método para as páginas `java.sun.com`, `brasil.gov.br` e `cos.ufrj.br`, respectivamente. Como feito para o método Alg01, somente mostramos as páginas que foram sugeridas pelo menos 10 vezes para algum valor do número de passos.

A Tabela 5.4 mostra os resultados obtidos pelo método Alg02 para a página `java.sun.com`. Os resultados foram muito semelhantes aos obtidos pelo método Alg01, mostrando que a busca aleatória na vizinhança da página `java.sun.com` não se altera se buscarmos nós com alto grau de entrada ou nós cujos pais possuem alto grau de saída. Isso indica que no site da Sun os nós que possuem grau de entrada alto também possuem pais com graus de saída alto. Isto é, existem algumas páginas com altíssimo grau de saída que possuem um conjunto comum de páginas filhas. Por isso, a gente pode identificar os mesmos dois grupos: páginas difusoras e páginas que bloqueiam a busca aleatória.

A Tabela 5.5 mostra os resultados para a url `brasil.gov.br`. Para essa página podemos identificar diferenças muito significativas. Primeiramente, note que os resultados ficaram concentrados em um número muito menor de páginas. Isso mostra que esse método restringiu muito mais a busca aleatória. Além disso, pode-se perceber claramente que a página da Dataprev (linha 1) foi catapultada para o primeiro lugar atingindo mais de 50% dos resultados para todos os números de passos. Isso mostra que apesar dessa página não possuir um número de pais muito grande, ela é apontada por páginas que possuem muitos links. É interessante observar também que páginas como a da PAC (linha 5), que apenas atingia posições modestas na tabela para o método Alg01, foi visitada mais vezes no método Alg02.

URL: java.sun.com		No de Passos			
#	**Sugestão**	**1**	**3**	**5**	**20**
1	http://www.sun.com	545	65	17	0
2	http://developers.sun.com	507	147	36	0
3	http://www.sun.com/software/products/mysql/getit.jsp	21	0	0	0
4	http://blogs.sun.com	16	20	31	31
5	https://glassfish.dev.java.net	7	8	7	11
6	http://www.sun.com/software/opensource/java	6	6	6	13
7	http://sunsolve.sun.com	0	287	382	402
8	http://shop.sun.com/cart	0	168	185	204
9	https://portal.sun.com/portal/dt	0	168	205	195
10	http://blogs.sun.com/bigadmin	0	53	62	63
11	https://suned.sun.com/APPS/US/SESWEB	0	9	5	13
12	http://blogs.sun.com/jonathan	0	1	9	12
13	http://www.sun.com/aboutsun	0	2	8	11
14	http://www.sun.com/sales	0	5	5	10
15	http://developers.sun.com/events/techdays/index.jsp	0	6	11	8
16	http://www.netbeans.org	0	3	10	4
17	http://www.sun.com/training	0	47	10	0

Tabela 5.4: Páginas sugeridas pelo método Alg02 para a URL java.sun.com.

Por outro lado, páginas que foram sugeridas muito mais vezes para Alg01 acabaram simplesmente desaparecendo dos resultados para Alg02.

URL: brasil.gov.br		No de Passos			
#	Sugestão	1	3	5	20
1	http://www010.dataprev.gov.br/cws/contexto/consit02/index.html	423	445	433	422
2	http://bvsms.saude.gov.br/bvs/pacsaude	173	168	157	166
3	http://www.combatadengue.com.br	164	168	190	181
4	http://www.brasil.gov.br/noticias/em_questao	34	28	17	36
5	http://www.brasil.gov.br/pac	28	24	36	28
6	http://www.brasil.gov.br	11	0	0	0

Tabela 5.5: Páginas sugeridas pelo método Alg02 para a URL brasil.gov.br.

A Tabela 5.6 apresenta resultados para o método Alg02 para a URL cos.ufrj.br. Os resultados são bem semelhantes aos obtidos para o método Alg01. Podemos, porém, observar que a página da UFRJ (linha 1) teve uma redução significativa no número de visitas. Por outro lado, o fenômeno identificado nas 7 primeiras linhas dessa tabela se repete. As páginas da UFRJ, COPPE e CNPq (linhas 1 à 3) foram bem visitadas para praticamente todos os números de passos. As páginas da FAPERJ e CAPES (linhas 4 e 5) foram visitadas apenas para 1 passo e a página da Rede Rio (linha 6) foi visitada apenas após 3 passos. Ainda em relação ao site da Rede Rio, é importante ressaltar que houve um aumento considerado no número de visitas em relação ao método Alg01. Por fim, podemos notar que o site da Dataprev (linha 7), que não aparecia nos resultados para o método Alg01, aparece relativamente bem posicionado para o método Alg02. Isso, obviamente, se deve ao fato de a busca aleatória atingir o site brasil.gov.br a partir do site da COS e, desse ponto, visitar com alta probabilidade o site da Dataprev.

URL: cos.ufrj.br		No de Passos			
#	Sugestão	1	3	5	20
1	http://www.ufrj.br	230	228	238	248
2	http://www.coppe.ufrj.br	206	222	188	189
3	http://www.cnpq.br	199	221	195	200
4	http://www.faperj.br	198	0	0	0
5	http://www.capes.gov.br	167	0	0	0
6	http://www.rederio.br	0	168	189	197
7	http://www010.dataprev.gov.br/cws/contexto/consit02/index.html	0	94	99	77
8	http://bvsms.saude.gov.br/bvs/pacsaude	0	30	32	42
9	http://www.combatadengue.com.br	0	26	47	35

Tabela 5.6: Páginas sugeridas pelo método Alg02 para a URL cos.ufrj.br.

5.3 Método Alg03

Nesse método, em cada passo da busca aleatória, uma página é selecionada com probabilidade proporcional a soma dos graus de entrada dos seus pais. As tabelas 5.7, 5.8 e 5.9 mostram os resultados obtidos por esse método para as páginas `java.sun.com`, `brasil.gov.br` e `cos.ufrj.br`, respectivamente. Devido ao grande número de páginas sugeridas para a URL `java.sun.com`, apresentamos nesse caso somente as páginas que foram visitadas pelo menos 18 vezes por um dos valores usados para o número de passos. Para `brasil.gov.br` e `cos.ufrj.br` mostramos todas as páginas que foram visitadas pelo menos 10 vezes para algum dos valores usados para o número de passos da busca.

A Tabela 5.7 mostra os resultados obtidos para `java.sun.com`. De uma maneira geral, podemos observar uma redução significativa no número de visitas de cada página. Nos métodos anteriores, as páginas `sun.com` e `developers.sun.com` (linhas 1 e 2) dominavam completamente a busca aleatória após o primeiro passo. Para o método Alg03, embora essas páginas ainda liderem os resultados para um único passo, o número de visitas caiu consideravelmente. Essa quebra do monopólio

que essas duas páginas exerciam no primeiro passo da busca causou uma maior diversidade nas sugestões fornecidas pelo algoritmo. Porém, isso ainda não permitiu com que páginas fora da estrutura do site da SUN fossem atingidas pela busca. A fim de remediar essa situação, dois casos precisam ser considerados. Primeiro, é preciso verificar se realmente existem links do site da SUN para páginas em outros domínios (fora da estrutura do site da SUN). Uma breve navegação pelo site da SUN nos mostra que esses links são de fato muito reduzidos e, portanto, todos os métodos que se baseiam em uma busca aleatória no grafo Web tendem a ficar "presos" nas páginas dentro do domínio `sun.com`. Uma segunda alternativa seria artificialmente forçar a busca aleatória a procurar essas páginas de outros domínios. Uma forma de fazer isso sem mudar drasticamente o algoritmo proposto é considerar páginas como `developers.sun.com`, `blogs.sun` etc como pertencendo ao mesmo domínio de `sun.com`.

A Tabela 5.8 mostra os resultados para a página `brasil.gov.br` utilizando o método Alg03. Note que os resultados são realmente bem semelhantes aos obtidos com o método Alg02, porém com um detalhe muito importante. O número de visitas é um pouco mais homogêneo para o método Alg03. Note, por exemplo, que o site da Dataprev (linha 1) recebe quase 150 visitas a menos que para o método Alg02. Por outro lado, as páginas das linhas 2 e 3 foram visitadas mais vezes. Isso mostra realmente que o site da Dataprev se encaixa particularmente bem nos critérios para o Alg02.

Por fim, a Tabela 5.9 mostra os resultados obtidos com o método Alg03 para a página `cos.ufrj.br`. Nesse caso podemos identificar uma grande diferença para os resultados anteriores. Primeiramente, podemos notar que as páginas da UFRJ e da COPPE (linhas 1 e 2) foram acessadas muito mais frequentemente. Por outro lado, a página da Rede Rio, que era muitas vezes acessada pelos dois métodos anteriores, nem sequer aparece nos resultados para esse método. Isso nos remete a questionar se o site da Rede Rio é realmente relevante para uma pesquisa iniciada no site da COS. Além disso, talvez o resultado mais surpreendente que pode ser observado na Tabela 5.9 consiste na página da FAPERJ (linha 4). Note que, para os demais métodos, o site da FAPERJ só possuía resultado positivo quando a busca se limitava a um número bem pequeno de passos. Porém, no método Alg03, o site da FAPERJ foi visitado em todos os valores para o número de passos. Isso mostra

URL: java.sun.com		No de Passos			
#	Sugestão	1	3	5	20
1	http://www.sun.com	311	39	17	0
2	http://developers.sun.com	279	96	24	0
3	http://developers.sun.com/events/communityone	44	47	62	46
4	http://www.sun.com/software/products/mysql/getit.jsp	42	6	1	0
5	http://blogs.sun.com	36	60	78	65
6	http://www.sun.com/training/catalog/java/index.xml	23	0	0	0
7	http://www.sun.com/training/certification/java/index.xml	22	0	0	0
8	http://java.net	21	11	18	19
9	http://developers.sun.com/global/newsletters.html	20	8	17	12
10	http://developers.sun.com/global/content_feeds.html	20	6	5	12
11	http://developers.sun.com/global/privacy.html	13	13	16	20
12	http://developers.sun.com/global/berkeley_license.html	12	15	11	18
13	http://developers.sun.com/global/termsofuse.html	12	20	14	12
14	https://openjdk.dev.java.net	9	15	14	18
15	http://sunsolve.sun.com	0	188	207	243
16	https://portal.sun.com/portal/dt	0	108	107	120
17	http://shop.sun.com/cart	0	99	111	114
18	http://blogs.sun.com/bigadmin	0	25	39	41
19	http://www.sun.com/training	0	39	8	0

Tabela 5.7: Páginas sugeridas pelo método Alg03 para a URL java.sun.com.

URL: brasil.gov.br		No de Passos			
#	Sugestão	1	3	5	20
1	http://www010.dataprev.gov.br/cws/contexto/consit02/index.html	275	314	298	287
2	http://www.combatadengue.com.br	226	232	221	241
3	http://bvsms.saude.gov.br/bvs/pacsaude	206	201	218	228
4	http://www.brasil.gov.br	60	0	0	0
5	http://www.brasil.gov.br/noticias/em_questao	34	48	47	37
6	http://www.brasil.gov.br/pac	32	38	49	40

Tabela 5.8: Páginas sugeridas pelo método Alg03 para a URL brasil.gov.br.

que apesar de existirem caminhos para a página da FAPERJ, os outros métodos acabam negligenciando-a devido a forma como atribuem pesos às arestas.

URL: cos.ufrj.br		No de Passos			
#	**Sugestão**	**1**	**3**	**5**	**20**
1	http://www.ufrj.br	464	440	455	413
2	http://www.coppe.ufrj.br	316	356	342	357
3	http://www.cnpq.br	93	95	93	102
4	http://www.faperj.br	82	77	66	79
5	http://www.capes.gov.br	39	0	0	0
6	http://www.combatadengue.com.br	0	6	11	17
7	http://bvsms.saude.gov.br/bvs/pacsaude	0	10	12	16
8	http://www010.dataprev.gov.br/cws/contexto/consit02/index.html	0	11	16	15

Tabela 5.9: Páginas sugeridas pelo método Alg03 para a URL cos.ufrj.br.

Capítulo 6

Conclusão

Este trabalho busca estabelecer relacionamentos entre páginas Web explorando a estrutura de links da Web a fim de auxiliar os usuários a encontrarem informação que desejam. As ferramentas e algoritmos propostas até hoje para este tema avaliam a estrutura de links apenas até um ou dois níveis além da página sendo atualmente visitada pelo usuário. Esse trabalho procura estudar como a utilização de mais níveis dessa estrutura pode auxiliar na busca por páginas relacionadas. Nesse sentido propomos um algoritmo bem genérico para navegar entre os links entre páginas Web e discutimos três métodos que estabelecem os critérios sobre que links são escolhidos durante essa navegação.

Em linhas gerais, o algoritmo proposto visa explorar a estrutura dos links através de uma busca aleatória pelo grafo Web. A cada iteração do algoritmo, escolhemos um dos links da página atual e transitamos para a página apontada por esse link. Discutimos três métodos para atribuir a probabilidade de se escolher cada um dos links. Esses métodos de análise usam o grau de entrada e de saída de cada nó do grafo, e essencialmente atribuem maior probabilidade àqueles nós cujo grau de entrada é alto.

Em se tratando de páginas Web, diversas aplicações sobre os algoritmos propostos emergem. Uma dessas aplicações é a sua utilização em um curso a distância, onde poderíamos avaliar a interface proposta. Em nossa proposta, os conceitos mais relevantes para o curso devem aparecer como bons resultados. Uma outra aplicação bem simples de se imaginar é transformar o *framework* em um *plugin* para navegadores Web. Seu funcionamento seria bem simples, este *plugin* captura

a página atual do usuário, aplica os algoritmos e retorna como sugestão páginas relacionadas à página atual.

Para avaliarmos com mais consistência o algoritmo proposto e realizarmos testes mais detalhados de sua performance, precisamos de uma base de dados com um maior número de páginas. Apesar de atualmente termos em nossa base aproximadamente 200 mil páginas (de quase 10 mil domínios diferentes) e mais de 3 milhões de links, esta quantidade é ainda pequena se comparada ao número de páginas Web existentes.

A utilização de uma base de dados mais ampla cria a necessidade de se otimizar a execução do algoritmo que propomos. Uma possível direção para agilizar a sua execução seria utilizar apenas as páginas mais próximas da URL visitada. Para isso, pode-se fazer o pré-carregamento de parte do grafo Web, diminuindo o tempo de processamento de cada iteração do algoritmo proposto.

Visando melhorar a performance do algoritmo, podemos não somente utilizar os graus de entrada e saída dos nós, mas também podemos usar o texto dos links como uma forma de inferir se aquele link aponta para uma página relevante. Dessa forma, no momento que o usuário clicar em um link, o texto do link é armazenado e utilizado juntamente com os graus de entrada e de saída dos nós para estabelecer a probabilidade de cada link. Isso visaria solucionar o problema de que a busca aleatória divergiu em alguns momentos para páginas de conteúdo não relacionado com a página atual, como ocorreu da página `cos.ufrj.br` para a página `combatadengue.com.br`.

Por fim, vimos que páginas Web de um mesmo domínio Web tendem a possuir conteúdo semelhante. A fim de alcançar uma maior diversidade na saída do algoritmo, devemos favorecer links de domínios Web diferentes. Acreditamos que um mecanismo mais elaborado que o implementado nesse trabalho pode vir a melhorar substancialmente os resultados do nosso algoritmo.

Referências Bibliográficas

[1] BERNERS-LEE, T., E CAILLIAU, R. Worldwideweb: Proposal for a hypertext project. Relatório técnico, CERN, 1990.

[2] BHARAT, K., E BRODER, A. A technique for measuring the relative size and overlap of public web search engines. *Comput. Netw. ISDN Syst. 30*, 1-7 (1998), 379–388.

[3] LAWRENCE, S., E GILES, C. L. Accessibility of information on the web. *Intelligence 11*, 1 (2000), 32–39.

[4] GULLI, A., E SIGNORINI, A. The indexable web is more than 11.5 billion pages. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web* (New York, NY, USA, 2005), ACM, pp. 902–903.

[5] *Yahoo!* `http://www.yahoo.com`.

[6] Google. http://www.google.com.

[7] Live search. http://www.live.com.

[8] SEBASTIANI, F. Machine learning in automated text categorization. *ACM Comput. Surv. 34*, 1 (2002), 1–47.

[9] KATZ, L. A new status index derived from sociometric analysis. *Psychometrika 18*, 1 (March 1953), 39–43.

[10] EGGHE, L., E ROUSSEAU, R. *Introduction to informetrics.* Elsevier, Amsterdam, 1990.

[11] GARFIELD, E. Citation analysis as a tool in journal evaluation. science. *Science 178* (1972), 471.

[12] EGGHE, L. Mathematical relations between impact factors and average number of citations. *Inf. Process. Manage. 24*, 5 (1988), 567–576.

[13] PINSKI, G., E NARIN, F. Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics. *Inf. Process. Manage. 12*, 5 (1976), 297–312.

[14] GELLER, N. L. On the citation influence methodology of pinski and narin. *Inf. Process. Manage. 14*, 2 (1978), 93–95.

[15] BOTAFOGO, R. A., RIVLIN, E., E SHNEIDERMAN, B. Structural analysis of hypertexts: Identifying hierarchies and useful metrics. *ACM Trans. Inf. Syst. 10*, 2 (1992), 142–180.

[16] CARRIÈRE, S. J., E KAZMAN, R. Webquery: Searching and visualizing the web through connectivity. *Computer Networks 29*, 8-13 (1997), 1257–1267.

[17] KLEINBERG, J. M. Authoritative sources in a hyperlinked environment. *J. ACM 46*, 5 (1999), 604–632.

[18] LAWRENCE, P., SERGEY, B., MOTWANI, R., E WINOGRAD, T. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[19] DEAN, J., E HENZINGER, M. R. Finding related pages in the world wide web. *Comput. Netw. 31*, 11-16 (1999), 1467–1479.

[20] Alta vista. `http://www.altavista.com`.

[21] BRIN, S., E PAGE, L. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems 30*, 1–7 (1998), 107–117.

[22] GARFIELD, E. New international professional society signals the maturing of scientometrics and informetrics. *The Scientist 9*, 16 (1995).

[23] GOFFMAN, W. A mathematical method for analyzing the growth of a scientific discipline. *J. ACM 18*, 2 (1971), 173–185.

[24] GRIMMETT, G. R., E STIRZAKER, D. R. *Probability and Random Processes.* Oxford University Press, August 2001.

[25] BHARAT, K., BRODER, A., HENZINGER, M., KUMAR, P., E VENKATASUBRAMANIAN, S. The connectivity server: fast access to linkage information on the web. In *WWW7: Proceedings of the seventh international conference on World Wide Web 7* (Amsterdam, The Netherlands, The Netherlands, 1998), Elsevier Science Publishers B. V., pp. 469–477.

[26] BHARAT, K., E HENZINGER, M. R. Improved algorithms for topic distillation in a hyperlinked environment. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1998), ACM, pp. 104–111.

[27] LEMPEL, R., E MORAN, S. The stochastic approach for link-structure analysis (salsa) and the tkc effect. *Comput. Netw. 33*, 1-6 (2000), 387–401.

[28] AMENTO, B., TERVEEN, L., E HILL, W. Does "authority" mean quality? predicting expert quality ratings of web documents. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2000), ACM, pp. 296–303.

[29] DONATO, D., LAURA, L., LEONARDI, S., E MILLOZZI, S. The web as a graph: How far we are. *ACM Trans. Interet Technol. 7*, 1 (2007), 4.

[30] BRODER, A., KUMAR, R., MAGHOUL, F., RAGHAVAN, P., RAJAGOPALAN, S., STATA, R., TOMKINS, A., E WIENER, J. Graph structure in the web. In *Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications networking* (Amsterdam, The Netherlands, The Netherlands, 2000), North-Holland Publishing Co., pp. 309–320.

[31] Brazil web application framework. http://www.experimentalstuff.com/Technologies/Brazil/index.html.

[32] Sun microsystems. http://www.sun.com.

[33] KUROSE, J. F., E ROSS, K. *Computer Networking: A Top-Down Approach Featuring the Internet.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[34] Apache license, versão 2.0. http://www.apache.org/licenses/LICENSE-2.0.html.

[35] Html 4.01 specification. http://www.w3.org/TR/html401.

[36] Mysql database server. http://www.mysql.com.

[37] Gnu general public license. http://www.gnu.org/licenses/gpl.html.