Universidade Federal de Pernmbuco
Centro de Informática

Roberta de Souza Coelho

SAAL - Um Sistema para Armazenamento e Análise de Links da Web

Dissertação Apresentada ao Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Silvio Meira

Recife, junho de 2002

Agradecimentos

"Na vida há um constante intercâmbio entre os homens. Uns influenciam os outros. Cada um recebe de todos e todos recebem de cada um".

No desenvolvimento de um trabalho, como uma dissertação, este intercâmbio se torna ainda mais forte. Gostaria de agradecer a todos que colaboram para a conclusão deste trabalho.

Agradeço ao meu maravilhoso Deus e por estar sempre ao meu lado.

Aos tão queridos painho e mainha, por toda dedicação e amor que me transmitiram desde o jardim da infância até os dias de hoje.

Ao professor Silvio Meira, que possibilitou através do Radix, e de sua orientação o desenvolvimento esta pesquisa.

Ao professores Jorge Fernandes e Pedro Falcão que me incentivaram a realizar meu trabalho de mestrado na área de qualidade da resposta em Recuperação de Informação. E ao professor Jones Albuquerque que me ajudou bastante na conclusão deste trabalho.

Aos demais os professores e aos funcionários do Centro de Informática que indiretamente contribuíram para a conclusão deste trabalho.

Aos meus amados irmãos Renata e Jaime pelo amor, pelas cartinhas pelas palavras e pelos abraços.

Ao querido Uirá pelo seu amor e companheirismo ao longo destes anos, e por ter me ajudado bastante para conclusão deste trabalho.

A Magalí pela simpatia que constantemente esteve presente em todos os momentos que marcamos e remarcamos a reuniões.

Aos amigos do Radix ("Mobile") que foram de uma presença constante ao longo destes anos. Em especial aos amigos Nery e Ana Karla colaboraram ativamente no desenvolvimento deste trabalho, ao amigo Luciano que sempre foi muito prestativo, ao amigo Franklin por sua amizade, ao amigo João Batista que compartilhou as alegrias e os problemas da vida de um mestrando. Aos amigos que foram de uma ajuda tremenda ao realizar o processo de etiquetagem: Cynthia, Rodrigo, Mariano, Fernando Trinta e Tércio.

Resumo

O aumento do número de documentos disponíveis na *World Wide Web* (WWW) traz uma série de novos desafios para a área de Recuperação de Informação (RI). As páginas *Web* divergem em conteúdo e qualidade além de possuírem uma alta dinâmica. Em adição a estes desafios os engenho de busca estão constantemente lidando com usuários inexperientes e com páginas *Web* construídas com o intuito de manipular as funções de *ranking* dos engenhos de busca.

Estudos recentes têm mostrado que a performance dos engenhos de busca está longe da ideal. Apesar das evoluções tecnológicas, conseguidas até o momento, permitirem a coleta e o armazenamento de um número cada vez maior de páginas nas bases de índices dos engenhos de busca, a maioria destes sistemas enfrenta vários problemas no momento de classificar as páginas de acordo com a necessidade do usuário, em outras palavras, retornar para o usuário a informação que ele necessita.

A maioria dos engenhos de busca analisa as páginas *Web* como um documento texto simples, não levando em consideração a estrutura na qual a página *Web* está inserida. Diferentemente das coleções de documentos "flat", a WWW corresponde a uma coleção de documentos hipertexto que possuem informações auxiliares que vão além do conteúdo textual, tais como a estrutura dos *hiperlinks* e o texto dos *hiperlinks*. Estas informações são chamadas de informações "hiper", que em conjunto com as informações "texto" compõem o conjunto de informações que caracteriza uma página *Web*.

A inadequação de estratégias singulares no processo de recuperação de informações no ambiente *Web* constitui-se em um forte argumento para mostrar que as técnicas recuperação de informação tradicionais não são suficientes no momento de encontrar informações relevantes na *Web*.

Este trabalho propõe a utilização da estrutura de *links* da *Web* com o objetivo de produzir um peso de "importância" global para cada página *Web* indexada por um engenho de busca. Este peso, chamado "peso de autoridade", é integrado aos engenhos de busca, mais especificamente a função de *ranking* dos engenhos de busca que passa a utilizar estes pesos juntamente com pesos de similaridade textual, com o objetivo de melhorar a eficácia de recuperação do sistema.

Para calcular o "peso de autoridade" para cada página *Web* foi elaborado um algoritmo de análise de *links*, o *Global Hybrid Hyperlinked Inducted Topic Search* (GHHITS) que foi concebido a partir do estudo dos algoritmos de análise de *links* existentes.

Para validar o algoritmo em questão foi implementado o SAAL - Sistema para Armazenamento e Análise de *Links* - que propõe uma maneira eficiente de armazenar a estrutura de *links* da *Web*, e executar o algoritmo proposto sobre esta estrutura.

Por fim, são apresentados os resultados obtidos durante os testes que avaliaram a eficácia de recuperação de estratégias de busca que utilizaram o peso de autoridade como componente da função de *ranking*.

Neste trabalho é mostrado, portanto, como as informações estruturais podem ser utilizadas de forma a melhorar a qualidade da resposta retornada por um engenho de busca.

Abstract

The World Wide Web (WWW) is growing at phenomenal rates and creates many new challenges to the Information Retrieval (IR). Current estimates are that there are over then 150 million Web pages with a lifetime of less than one year. Moreover, the Web pages are extremely diverse in content and quality. In addition to these major challenges, search engines on the Web must also contend with inexperienced users and pages engineered to manipulate search engines ranking functions.

Recent studies report that the performance of actual search engines is far from being fully satisfactory. While technological progress has made it possible to deal reasonably with the size of the Web in terms of number of pages indexed, the big problem is just to properly classify pages in response to user's needs: in other words, to give the user the information s/he needs.

Most of the current search engines look at a Web page to evaluate, almost as a peace of text. Even with extremely sophisticated techniques like those already present in some search engine scoring mechanisms, this approach suffer from an intrinsic weakness: it does not take in to account the Web structure the page is part of.

Unlike "flat" document collections, the WWW is hypertext and provides auxiliary information on top of the text of Web pages, such as link structure and link text. That information is called "hyper information". So, the "overall information" of a Web page is composed by "textual information" and "hyper information".

In this dissertation, we take advantage of the link structure of the Web to produce a global "importance" ranking to every Web page, indexed by a search engine. This ranking called "authority weight" provides smooth integration with existing search engines, since it can be used in combination with the existing textual ranking score to improve the performance of the search engine.

In order to calculate this ranking we developed a link analysis algorithm called Global Hybrid Hyperlinked Inducted Topic Search (GHHITS). To validate this algorithm, it was developed a system called SAAL – a system to store and analyze the link structure of the Web.

Finally, we present the result of some tests that evaluate the retrieval performance of search strategies using the structural information in conjunction with textual

information. We show how the structural information is able to considerably improve the quality of information provided by a search engine.

Índice

Índice de Figuras

Índice de Tabelas

1

Introdução

Desde o surgimento da *Web* em meados dos anos 80 [Yan01], a sua popularidade e o número de informações disponíveis vem crescendo em ritmo acelerado. Um fator que favorece esse crescimento consiste na facilidade de se compartilhar conhecimento e idéias de uma forma eficiente e praticamente instantânea. A *Web*, atualmente, está se tornando um novo meio de divulgação de informações de alcance mundial e, conseqüentemente, um repositório universal de conhecimento.

A dinâmica, a abundância e a heterogeneidade das informações que compõem a *Web* trazem consigo novos desafios relacionados à obtenção dessas informações. A tarefa de encontrar documentos relevantes, entre os disponíveis, quase sempre, é entediante e difícil.

Diante deste cenário, surgiram alternativas para auxiliar o usuário a encontrar informações relevantes na *Web*, dentre elas podemos citar os sistemas de recuperação de informação para a Web, também chamados de engenhos de busca.

Neste capítulo, será feita uma descrição do contexto no qual este trabalho está inserido, apresentando a motivação e o problema a ser tratado. Em seguida, serão discutidos o objetivo do trabalho e a organização dos capítulos seguintes.

1.1 Descrição do Problema

Os engenhos de busca são compostos por subsistemas que empregam uma combinação de diferentes estratégias [YL96, AGM+00]: (i) **estratégias de *crawling* e indexação** – que correspondem ao processo de coleta e indexação de páginas da *Web* - o processo de indexação consiste na construção de estruturas chamadas índices responsáveis por armazenar os termos dos documentos, e realizar o mapeamento entre os termos consultados e os documentos que contém estes termos; (ii) **estratégias de armazenamento** – que abrangem as estruturas que irão armazenar as informações pertencentes aos documentos *Web* indexados; (iii) **estratégias de busca e ranqueamento** – que consistem em recuperar e ordenar os documentos retornados na resposta, de acordo com algum critério de relevância. Este processo de ordenação é chamado de ranqueamento.

Cada uma destas estratégias influencia, de forma significativa, a qualidade da resposta retornada pelo sistema. Apesar das evoluções tecnológicas, conseguidas até o momento, permitirem aos engenhos de busca coletarem e armazenarem um número cada vez maior de páginas em suas bases. A maioria dos usuários ainda encontra dificuldade em encontrar documentos que atendam às suas necessidades de informação.

Além das dificuldades relacionadas ao processo de busca já enfrentadas pelos sistemas de recuperação de informação tradicionais - sistemas responsáveis por realizar consultas em conjunto de dados relacionados semanticamente como em um Banco de Dados (BD) - tais como a sinonímia (dois termos com o mesmo significado), a polissemia (um termo com dois significados), que serão detalhadas no próximo capítulo – os engenhos de busca sofrem de problemas decorrentes do comportamento dos usuários da *Web*.

A maioria dos usuários dos engenhos de busca não possui uma idéia clara a respeito da sua necessidade de informação no momento da elaboração da consulta [BH98]. Segundo Marchionini [Mar92], os usuários dos engenhos de busca procuram atingir seus objetivos com a máxima satisfação, a partir do menor esforço cognitivo. Alguns estudos revelam evidências que confirmam este pensamento. Segundo pesquisas, cerca de 70% das consultas feitas contêm somente um termo [LG99a, LG99b].

As consultas que são expressas por um ou dois termos são chamadas de consultas por tópicos gerais. Para este tipo de consulta existe uma enorme quantidade de informação na *Web*. Nestes casos os engenhos de busca enfrentam um desafio de precisão: retornar em primeiro lugar apenas as páginas mais "importantes" para uma dada consulta. A precisão é a métrica mais utilizada para a avaliação de sistemas de recuperação de informação e consiste na fração dos documentos retornados que são relevantes.

As consultas para as quais não existem muitos documentos na *Web*, são chamadas de consultas por tópicos específicos. Em consultas deste tipo, os engenhos de busca enfrentam, além do desafio de precisão inerente à busca, um desafio de cobertura - ao tentar encontrar os documentos que respondem à consulta entre os milhares de documentos existentes na base.

Em consultas por tópicos gerais, que correspondem às consultas submetidas com maior freqüência aos engenhos de busca, os usuários não visitam todas os documentos retornados como resultado. A partir da análise de meio bilhão de consultas submetidas ao engenho de busca de propósito geral Altavista, Silverstein et. al. ([SHM+99], Tabela 7, página 10) concluiu que cerca de 85,2% dos usuários visitavam apenas a primeira página de resultados.

Segundo o comportamento do usuário detalhado acima, uma maneira eficiente de se melhorar a qualidade da resposta é, portanto, retornar em primeiro lugar, as páginas consideradas realmente relevantes para a consulta submetida.

Neste cenário, a estratégia de busca corresponde ao componente do engenho de busca que mais influencia na qualidade da resposta.

Visando minimizar o problema de precisão da resposta relacionado às consultas por tópicos gerais, os engenhos de busca incorporam às suas estratégias de busca outras fontes de informação que vão além do conteúdo [BM00, SEW]. Desta forma, os engenhos de busca procuram associar o conceito de "importância" às páginas *Web*, e a partir desta associação, recuperar entre os dez ou vinte primeiros documentos retornados, aqueles documentos realmente relevantes para a consulta submetida.

Dentre as fontes de informação que podem ser incorporadas às estratégias de busca, podem ser citadas:

- Ranking baseado na Classificação Humana: editores humanos são utilizados por

empresas como o Yahoo[1], para manualmente associar um conjunto de palavras-chave e categorias (conceitos semânticos) aos documentos *Web*. Uma vez que este processo é realizado por humanos, ele possui os seguintes problemas: (a) por ser bastante lento, este processo só pode ser aplicado a um subconjunto reduzido de páginas, não acompanhando, portanto, a alta taxa de crescimento da *Web*; (b) os termos associados às páginas podem ser incoerentes ou incompletos.

- Ranking baseado em Informações sobre a Utilização do Site: cada vez mais os engenhos de busca vem guardando informações de utilização do site, tais como, quais as consultas mais freqüentes, quais as URLs de resposta mais clicadas, ou até mesmo quanto tempo o usuário gasta em uma determinada URL resposta. Existem empresas especializadas em armazenar este tipo de informação, como a DoubleClick[2], que prestam este serviço para alguns *sites*. Os engenhos de busca podem fazer uso destas informações no momento de ordenar o conjunto de documentos retornados como resposta. De acordo com este procedimento, se uma página relevante não aparece entre as 20 primeiras, esta continuará sem aparecer, e consequentemente as páginas irrelevantes que aparecem entre as vinte primeiras continuarão aparecendo. Este efeito é chamado de efeito cascata.

- Ranking baseado na Conectividade: diferentemente de outras coleções textuais os documentos *Web* são conectados por *hiperlinks*. Os *hiperlinks* podem ter diferentes funções: navegacionais, comerciais ou informativas. Estas últimas embutem a opinião coletiva dos usuários da *Web*. Existem vários algoritmos na literatura, que propõem a análise da estrutura de *links* de uma coleção de documentos *Web* com o objetivo de extrair desta estrutura opinião coletiva dos usuários.

Dentre as fontes de informação citadas acima, a informação sobre a conectividade das páginas *Web* se mostrou mais promissora, visto que o *ranking* baseado na Conectividade embute a opinião coletiva dos usuários da *Web,* além de se mostrar imune ao efeito cascata e aos erros inerentes à utilização de humanos no processo de atribuição de pesos às páginas *Web*.

[1] *Web Directories*: www.yahoo.com

[2] www.doubleclick.com

1.2 Trabalho proposto

Tendo como motivação o cenário descrito anteriormente, constituem os objetivos deste trabalho:

(i) Uma análise dos principais algoritmos de análise de *links* encontrados na literatura [Kle97, BH98, CDR+98, Hav99].

(ii) Diante da inadequação dos algoritmos pré-existentes aos requisitos de tempo impostos pelos engenhos de busca comerciais, e do fato de alguns algoritmos possuírem descrições obscuras por fazerem parte do diferencial comercial destes sistemas, este trabalho propôs a elaboração de um algoritmo de análise de *links*, a partir das características positivas dos algoritmos estudados. Este algoritmo associa um conceito de importância às páginas *Web* para ser usado em conjunto com as informações textuais das páginas no momento do ranqueamento, com o objetivo de melhorar a qualidade da resposta do engenho de busca.

(iii) A implementação de um Sistema para Armazenamento e Análise da estrutura de *Links* (SAAL) para validar o algoritmo proposto. Este sistema corresponde a um módulo a ser acoplado a um engenho de busca baseado unicamente no paradigma textual.

(iv) Concepção de estratégias para minimizar os custo de tempo e espaço no processo de análise de *links.* Uma vez que é um requisito deste sistema ser utilizado em sistemas de busca comerciais os quais possuem fortes requisitos de tempo.

(v) Elaborar uma técnica de fusão de informação, para incorporar o novo paradigma de análise de *links* ao paradigma textual.

(vi) Avaliar o impacto na precisão da resposta ao se adicionar a informação sobre os *links* à estratégia de busca baseada apenas no paradigma textual.

1.3 Organização da Dissertação

Esta dissertação está organizada nos seguintes capítulos:

Capítulo 2 – Recuperação de informação

Aborda os principais conceitos, técnicas e modelos relacionados aos sistemas de recuperação de informação tradicionais, como também conceitos e técnicas específicos de sistemas de recuperação de informação para a *Web*.

Capítulo 3 – Algoritmos de Análise de *Links*

Descreve os principais algoritmos de Análise *de Links* encontrados na literatura, bem como alguns dos métodos de fusão de informação existentes.

Capítulo 4 – Proposta da Solução

Apresenta o problema, bem como a especificação da solução que consiste: (i) no algoritmo de análise *de links* proposto neste trabalho (GHHITS), o qual mescla algumas das principais características presentes nos algoritmos preexistentes, de forma a atender os requisitos de tempo impostos pelos sistemas de recuperação de informação para a *Web*; e (ii) no sistema para armazenamento e análise de *links* desenvolvido (SAAL).

Capítulo 5 – Implementação do Sistema

Descreve os desafios superados bem como as etapas realizadas no desenvolvimento do projeto, apresentando os aspectos técnicos envolvidos na implementação do sistema para armazenamento e análise de *links*.

Capítulo 6 – Avaliação da Eficácia de Recuperação

Apresenta o método de avaliação da eficácia de recuperação, bem como o sistema desenvolvido para a aplicação deste método e os resultados obtidos na avaliação as estratégias de busca propostas.

Capítulo 7 – Conclusão e Trabalhos Futuros

Apresenta conclusões sobre este trabalho, sua importância e suas contribuições. Além de serem apresentadas sugestões para pesquisas futuras.

2

Recuperação de Informação

A primeira solução sistemática para o problema de recuperação de informação em uma grande coleção de dados foi desenvolvida a cerca de 400 anos atrás, por bibliotecários que passaram a organizar os livros catalogando-os por autor e título [Yan01]. A próxima solução surgiu no século XVI, com a construção dos índices. Os índices correspondiam a listas de palavras-chave que apontavam para documentos. Um problema inerente a esta estrutura consistia em selecionar as palavras "apropriadas" para serem associadas a um documento.

Com o advento dos computadores, a partir da década de 50, surgiram os índices que armazenavam todos os termos pertencentes a um documento. Estes índices solucionaram o problema descrito acima, porém, outro problema surgiu: para a maioria das consultas, muitos documentos eram retornados, e poucos deles eram relevantes.

Um sistema de recuperação de informação (RI) deve ser capaz de representar, armazenar, organizar, recuperar e manter informações, e, sobretudo identificar a necessidade do usuário de tal forma que consiga recuperar os itens que melhor atendam a sua necessidade de informação [Kow97].

Neste capítulo, serão abordados os conceitos básicos, técnicas e modelos relacionados à recuperação de informação (RI).

2.1 Processo de Recuperação de Informação

Com o intuito de descrever o processo de recuperação de informação, a Figura 1 apresenta uma arquitetura simples e genérica de um sistema de recuperação de informação textual.

Antes de detalhar os módulos que fazem parte deste sistema, faz-se necessária a definição dos elementos que compõem a base de dados textual. A base de dados textual engloba: (i) os documentos a serem utilizados pelo sistema de RI; (ii) as operações a serem realizadas sobre o conteúdo textual destes documentos; (iii) e a visão lógica, que consiste na estrutura que armazenará o texto e nos elementos que poderão ser recuperados através desta estrutura [BYRN99].

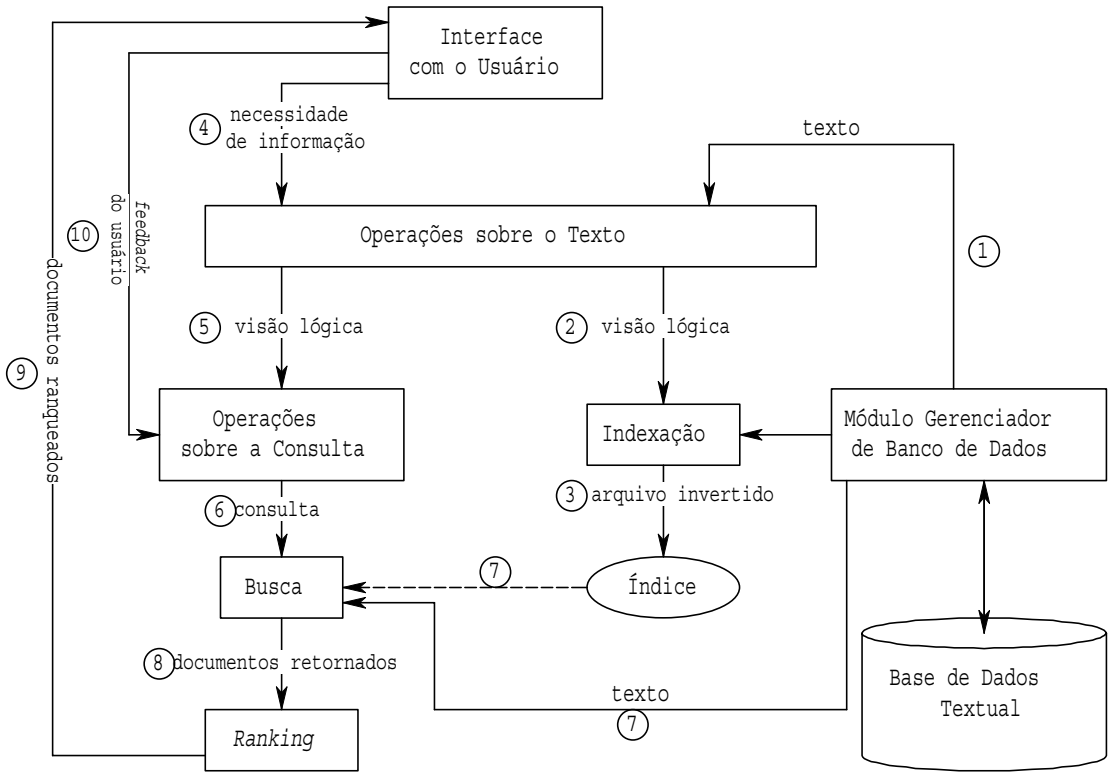

Figura 1: Processo de Recuperação de Informação - adaptado de [BYRN99].

As operações sobre o texto transformam os documentos iniciais em suas correspondentes visões lógicas (passos 1 e 2, Figura 1). A partir da visão lógica é construído um índice para o texto (passo 3, Figura 1). O índice corresponde a uma estrutura de dados crítica para um sistema de RI, uma vez que permite uma busca rápida sobre um grande volume de dados.

Existem várias estruturas de índice, porém a mais utilizada é a estrutura de arquivos invertidos, que será detalhada na Seção 2.2.1.

A construção e a manutenção dos arquivos invertidos corresponde a um processo bastante custoso no que diz respeito ao recurso tempo. Porém estes custos são amortizados na medida em que as consultas são submetidas ao sistema.

Após a construção de um índice para a base textual, dar-se-á início ao processo de recuperação de informação. Este se inicia quando o usuário submete uma consulta ao sistema (passo 4, Figura 1). São realizadas operações sobre a consulta que irão transformá-la numa representação inteligível ao sistema (passo 5, Figura 1). Esta consulta é processada (passos 6 e 7, Figura 1) e um conjunto de documentos é retornado em resposta a este processamento (passo 8, Figura 1).

Antes de serem retornados ao usuário, estes documentos serão ranqueados de acordo com uma medida de relevância (passo 9, Figura 1). Existem várias medidas que objetivam obter um valor de relevância para um documento. Na Seção 2.3 serão abordadas algumas delas.

Após os documentos terem sido retornados ao usuário, este examina o conjunto de documentos à procura de informações que atendam a sua necessidade de informação. Neste momento o usuário pode dar início a um ciclo de *feedback* (passo 10, Figura 1). Neste ciclo, o sistema utiliza os documentos selecionados pelo usuário como relevantes para alterar a consulta formulada inicialmente, e submetê-la novamente ao sistema.

2.2 Representação dos Dados

Um dos principais desafios inerentes ao processo de recuperação de informação consiste em acessar de forma eficiente o conteúdo dos documentos, bem como a relação existente entre eles. Estruturas de dados capazes de representar os documentos e localizá-los de forma eficiente são essenciais para o sucesso de um sistema de recuperação de informação.

A seguir, serão detalhadas as seguintes estruturas de acesso: (i) o arquivo invertido - a mais comumente utilizada tanto por sistemas de recuperação de informação, quanto por sistemas de banco de dados; (ii) a S*uffix Tree*; (iii) e o *Vertical Inverted File* (VIF) – estrutura elaborada para o engenho busca ***Rad***ix.

O engenho de busca ***Radix*** foi utilizado como estudo de caso, no decorrer deste

trabalho. O ***Radix*** consiste em um engenho de busca baseado unicamente no paradigma textual, que indexa as páginas pertencentes à Internet brasileira[3].

2.2.1 Arquivos Invertidos

Os arquivos invertidos são estruturas de dados que permitem encontrar de forma rápida quais documentos de uma coleção possuem um dado termo. Esta estrutura é composta por dois elementos: (i) o vocabulário - que consiste no conjunto de todas as palavras diferentes que ocorrem na coleção; (ii) e as listas de inversão - que são listas contendo todos os documentos da coleção onde ocorre um dado termo. A Figura 2 ilustra uma coleção de dados e o arquivo invertido correspondente à coleção.

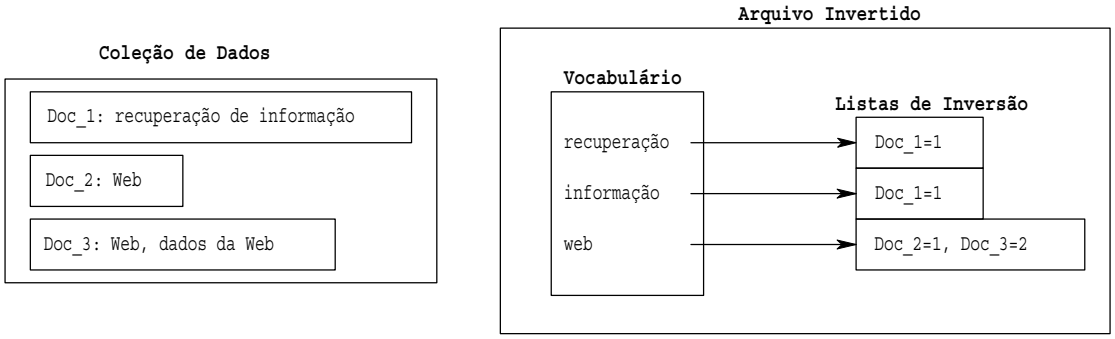

Figura 2: Uma coleção de dados exemplo e um arquivo invertido construído a partir dela.

As listas de inversão podem conter, além dos documentos onde ocorre o termo, outras informações que sejam úteis ao engenho de busca, como por exemplo, o número de ocorrências do termo no documento. Este valor é chamado na literatura [BYRN99, Kor97] de *Term-Frequency* (*tf*).

Um arquivo invertido é construído a partir do processamento de cada documento da coleção. Este processamento é chamado indexação, e é responsável por identificar todos os termos existentes na coleção, excluindo-se as palavras que aparecem com muita freqüência nos documentos da coleção (ex.: artigos, preposições e conjunções), e inseri-los no vocabulário juntamente com o ponteiro para a lista de inversão. As palavras ignoradas na construção do arquivo invertido são denominadas *stop-words* e um conjunto de *stop-words* corresponde a uma *stop-list*.

[3] www.radix.com

Quando uma consulta contendo mais de um termo é submetida a um sistema de recuperação de informação, o processo de busca é responsável por localizar a lista de inversão para cada termo da consulta. E dependendo dos operadores booleanos utilizados na formulação da consulta, a resposta será gerada a partir da interseção ou união dos conjuntos de documentos existentes em cada lista.

2.2.2 Suffix Trees

Os arquivos invertidos assumem que um texto pode ser representado através de uma sequência de palavras. Por esta razão consultas por frases são custosas de serem resolvidas. As *Suffix Trees* correspondem a um tipo de índice que permite responder a consultas por frases de forma eficiente.

As *Suffix Trees* [BYRN99] enxergam o texto como uma grande string, e associam um conjunto de *substrings* – formados por n caracteres seguidos do texto do documento – aos documentos. Na criação de uma *Suffix Tree*, cada posição do texto do documento corresponde a um ponto de referência que pode iniciar uma *substring*. Logo, uma *substring* pode iniciar em qualquer ponto do texto, podendo ser unicamente identificada por sua posição inicial.

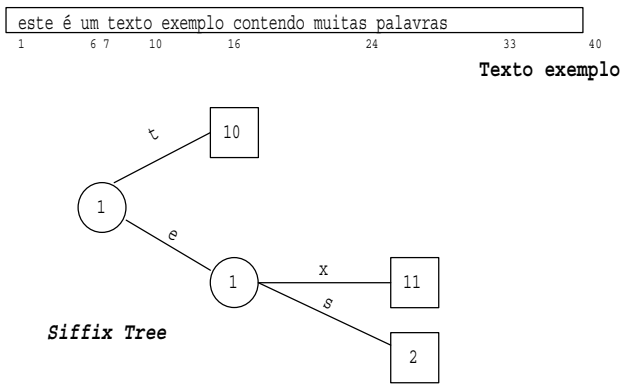

Figura 3: Uma *Suffix Tree* indexando termos do texto exemplo.

Uma *Suffix Tree* pode ser definida como uma árvore binária, não balanceada, formada por substrings. Os nós internos da árvore contêm caracteres da substring, e os nós externos contêm os ponteiros para posições no arquivo indexado. A Figura 3 mostra

um exemplo de uma *Suffix Tree* indexando os termos: *texto*, *exemplo* e *este*.

2.2.3 Vertical Inverted File (VIF)

Assim como os arquivos invertidos o *Vertical Inverted File* (VIF) [Mir02] também permite encontrar de forma rápida quais documentos de uma coleção possuem um dado termo. Além de permitir o acesso rápido a estas informações, o arquivo VIF permite um rápido acesso a informações sobre a posição dos termos no corpo do documento.

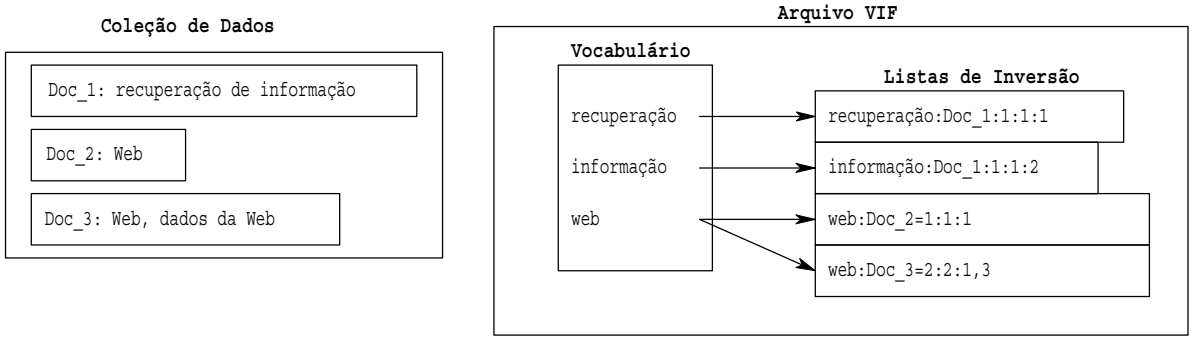

Figura 4: Campos que compõem o arquivo VIF.

A Figura 4 ilustra um arquivo VIF com seus respectivos campos: (i) o vocabulário - que consiste no conjunto de todas palavras diferentes que ocorrem na coleção; (ii) e as listas de inversão que englobam código do termo, código do documento, o valor de *term frequency*, tamanho da lista de posições, lista das posições que o termo aparece no documento.

2.3 Modelos de Recuperação de Informação

Tendo representado a coleção de dados e a necessidade de informação, o próximo passo na elaboração de um sistema de recuperação de informação consiste em determinar a relação de relevância existente entre os dois. A maneira convencional de se determinar esta relação consiste em computar para cada documento uma medida quantitativa que denota a similaridade ou probabilidade de relevância deste documento em relação a uma dada consulta. A medida de similaridade utilizada varia de acordo com o modelo de recuperação utilizado. A seguir, são detalhados alguns dos principais modelos de recuperação de informação.

2.3.1 Modelo Booleano de Recuperação de Informação

O Modelo Booleano de recuperação de informação [BYRN99] foi o primeiro método implementado para sistemas de RI. Ele se baseia em conceitos da teoria dos conjuntos e da álgebra booleana. As consultas são expressas através de expressões booleanas, as quais facilitam a associação de conceitos (consultas utilizando operadores do tipo AND) e as buscas por sinônimos (consultas utilizando operadores do tipo OR). Porém este modelo possui algumas desvantagens. Como a estratégia de busca se baseia no critério de decisão binária (o documento é relevante ou não relevante) não existe nenhuma noção de similaridade parcial entre o documento e a consulta, o que não permite que os documentos sejam ordenados por algum critério quando retornados em resposta a uma consulta. Apesar das consultas booleanas possuírem uma semântica precisa, a maioria dos usuários sente dificuldade em traduzir uma necessidade de informação para uma expressão booleana.

Estratégias de refinamento foram desenvolvidas com o intuito de tornar o processo de formulação de consultas mais fácil [BYRN99]. Estratégias que associam pesos a termos baseados na representatividade dos termos na consulta foram desenvolvidas com o objetivo de sanar o problema gerado pela utilização da decisão binária no processo de recuperação.

2.3.2 Modelo Vetorial de Recuperação de Informação

No Modelo Vetorial [BYRN99, Kor97] os documentos e as consultas são representados na forma de vetores do espaço n-dimensional, onde cada dimensão representa um termo. Por exemplo, seja um documento D e uma consulta C expressos da seguinte forma: D=(d1, d2, d3,..., dn) e C=(c1, c2, c3,..., cn), onde n corresponde ao número de termos pertencentes ao índice, e os valores das coordenadas correspondem a pesos que indicam a importância dos termos.

Existem várias fórmulas para se calcular o peso de um termo no modelo vetorial [BYRN99], porém a maioria delas corresponde à combinação das seguintes variáveis:

- Idf (*Inverse document frequency*) [Yan01]: Número de documentos na coleção,

que contém o termo presente na consulta.

- Tf (*Term frequency*): Número de ocorrências do termo no documento.

O modelo vetorial produz como resposta, uma lista de documentos ordenada por uma medida de similaridade entre o documento e a consulta. Várias medidas de similaridade foram propostas [Yan01], porém a mais utilizada corresponde o valor do cosseno do ângulo formado entre o vetor documento e o vetor consulta.

2.3.3 Modelo Booleano Estendido de Recuperação de Informação

O modelo Booleano Estendido incorpora as vantagens provenientes da utilização de operadores booleanos ao modelo Vetorial [Yan01]. Este modelo incorpora um indicador de restrição, chamado de valor-p, que quando utilizado em conjunto com os operadores booleanos determina o grau de restrição no qual os operadores devem ser aplicados.

2.3.4 Modelo Probabilístico de Recuperação de Informação

O modelo Probabilístico [BYRN99, Kor97] é similar ao modelo Vetorial na maneira de representação das consultas e documentos sob a forma de vetores, porém ao invés de recuperar documentos baseados no grau de similaridade destes com a consulta, o modelo probabilístico recupera documentos baseados na probabilidade de relevância do documento com relação à consulta. A idéia básica deste modelo consiste em associar um peso de relevância a cada um dos termos do índice, e calcular a probabilidade de relevância do documento através da soma dos pesos de relevância dos termos da consulta que aparecem no documento.

A probabilidade de relevância de um termo é calculada verificando-se a distribuição dos termos do índice entre os documentos relevantes de uma base de documentos classificados como relevantes ou não relevantes.

2.4 Recuperação de Informação na Web

Apesar da *Web* consistir em um conjunto de informações distribuído e descentralizado ela pode ser vista como uma grande coleção de documentos virtuais, na qual podem ser aplicados os modelos de representação das consultas, de documentos e os modelos de recuperação de informação elaborados por pesquisas na área de recuperação de informação tradicional.

Porém como grande parte das estratégias de recuperação tradicionais foram

concebidas baseadas em coleções estáticas de documentos, homogêneas e pequenas, algumas delas não podem ser aplicadas ao cenário da *Web* uma vez que os dados que compõem a *Web* possuem natureza dinâmica, distribuída e diversa em conteúdo, formato, contexto, propósito e qualidade [Hua98].

A *Web* é rica em novos tipos de informação antes ausentes nas coleções de dados tradicionais, tais como: os *hiperlinks* conectando documentos, estatísticas de utilização, *tags* HTML, hierarquias entre documentos (ex: Yahoo). Tais informações podem ser úteis na melhoria do processo de recuperação de informação que poderá, a partir da utilização destas novas informações, ir além do casamento de termos como ocorre na maioria dos sistemas de recuperação de informação tradicionais.

Além dos sistemas de recuperação de informação para a *Web*, outras alternativas foram desenvolvidas com o intuito de ajudar o usuário a desempenhar a tarefa árdua de encontrar informações relevantes na *Web*, dentre eles podemos citar os *Web Directories*.

Os *Web Directories*, como por exemplo o Yahoo!, consistem em um conjunto de *links* organizados em hierarquias. Apesar destes sistemas possuírem *links* de boa qualidade e relevantes ao contexto no qual estão inseridos, grande parte destes *links* torna-se desatualizada com o passar do tempo, pois como dito na Seção 1.1 o processo de indexação manual de páginas *Web* e não consegue acompanhar o ritmo de crescimento da *Web* que chega a mudar metade de seu conteúdo em 1 mês [PdFR00].

A seguir será dada uma visão geral acerca da evolução dos sistemas de recuperação de informação para a *Web* ao longo dos anos de 1994 a 2002. E em seguida, será mostrada a arquitetura C*rawler-Indexer* [BYRN99] – uma arquitetura centralizada de sistemas de recuperação de informação para a *Web*.

2.4.1 Engenhos de busca: Evolução 1994 – 2002

A tecnologia associada aos engenhos de busca teve que se tornar escalável em um curto intervalo de tempo para acompanhar o ritmo acelerado de crescimento da *Web* [Kob99]. Em 1994 o World Wide *Web* Worm (WWWW), um dos primeiros engenhos de busca a serem construídos, possuía em suas bases de índices um conjunto de 110.000 páginas *Web* [BP98].

Em novembro de 1997, os maiores engenhos de busca da época possuíam de 2 a 100 milhões de documentos *Web* indexados [BP98]. Em janeiro de 2002 dois dos maiores engenhos de busca, o Google e o WiseNut atestavam possuir em seus índices

2.073.418.204 e 1.571.413.207 respectivamente. Ao passo que aumentavam as bases de índices também aumentava o número de consultas submetidas aos engenhos de busca. De março a abril de 1994 o WWW recebia em média 1.500 consultas por dia. Em 1997, o engenho de busca Altavista tratava mais de 20 milhões de consultas por dia.

O principal objetivo dos engenhos de busca da atualidade é solucionar os problemas associados à qualidade da resposta e à escalabilidade do sistema ante o extraordinário número de documentos *Web* que surgem a cada dia. Estima-se que o número de *hosts* conectados a Internet esteja dobrando a cada ano [Lyn97].

2.4.2 Desafios Enfrentados Pelos Engenhos de Busca

Os sistemas de recuperação de informação para a *Web* têm evoluído bastante, desde o seu surgimento em 1994. Porém, a maioria destes sistemas ainda sofre dos problemas clássicos sofridos pelos sistemas de informação tradicionais [LM00], quais sejam:

- Sinonímia: os sistemas não recuperam documentos que contenham termos sinônimos aos termos presentes na consulta. Ex: Os documentos que contém o termo 'automóvel' não são retornados quando uma consulta por 'carro' é submetida ao sistema.
- Polissemia: quando um termo possui mais de um significado. Neste caso o sistema não consegue distinguir qual significado está sendo solicitado pelo usuário, por exemplo, o termo 'Petri' pode corresponder às Redes de Petri uma subárea da Computação ou a lâminas de petri, material utilizado em laboratórios de análises clínicas.
- Estilos de Autoria (generalização da sinonímia): Dois documentos sobre um mesmo tópico, podem utilizar vocabulários diferentes, não necessariamente sinônimos, quando escritos por autores diferentes.

Em adição aos problemas clássicos, existem obstáculos específicos do ambiente *Web,* dentre eles podemos citar:

- Persuasão do Engenho de Busca [LM00]: De acordo com o comportamento do usuário de um engenho de busca, detalhado em [GS00], a maioria dos usuários

visualiza apenas as duas primeiras páginas resposta a uma consulta. Por este motivo, têm surgido várias empresas especializadas em estudar as funções de ranqueamento dos engenhos de busca, e utilizar técnicas que venham (manipular estas funções) a 'enganar' os engenhos de busca para posicionar as páginas dos seus clientes entre as primeiras paginas retornadas. Dentre as técnicas utilizadas para 'enganar' os engenhos de busca podemos citar: o *keyword spamming* que consiste em preencher a páginas com várias palavras da mesma cor que o background da página, ou inserir termos na meta *keyword* que não estão relacionados com o conteúdo real da página, mas que são tópicos muito consultados pela maioria dos usuários dos engenhos de busca.

- Perfil dos Usuários da *Web*: diferentemente dos usuários de sistemas de informação tradicionais, os quais possuem perfis semelhantes, os usuários dos sistemas de recuperação de informação para a *Web* (engenhos de busca) possuem diferentes motivações, necessidades, níveis de conhecimento e experiência. Tal diversidade de perfis faz com que os usuários expressem uma grande variedade de necessidades de informação, através de diversas formas e com diferentes critérios de satisfação.

Os desafios acima enumerados dificultam o processo de busca por informações na *Web* realizado pelos engenhos de busca.

2.4.3 Arquitetura Crawler-Indexer Centralizada

Muitos dos engenhos de busca utilizam uma arquitetura *Crawler-Indexer* centralizada [BYRN99, KT00]. Os *crawlers* são programas que executam localmente, enviando requisições HTTP a servidores *Web* remotos. As páginas *Web* retornadas em resposta a estas requisições http são enviadas pelo *crawler* para um servidor de indexação. O servidor de indexação é responsável por gerar uma representação lógica para as páginas e por gerar um índice centralizado. O servidor de *links* é responsável por fornecer aos *crawlers* os *links* dos documentos que devem ser atualizados ou indexados. Os *links* que são enviados aos *crawlers* correspondem aos *links* das páginas já indexadas ou a *sites* inseridos manualmente no sistema, sendo estes últimos chamados de *links* sementes.

A Figura 5 ilustra uma arquitetura simplificada de um engenho de busca.

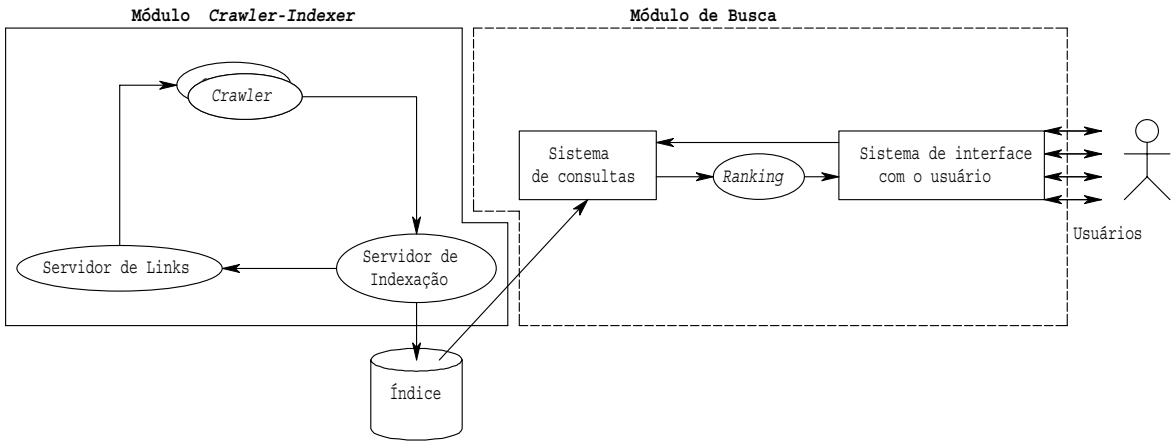

Figura 5: Arquitetura centralizada de um sistema de busca.

As caixas presentes da Figura 5, correspondem a uma representação de alto nível do sistema de consultas e o sistema de interface com o usuário. Ao serem explodidas, estas caixas podem originar um conjunto de processos, que não necessitam ser descritos neste momento.

Esta arquitetura é composta por dois módulos principais:

- **Módulo de Busca**: Módulo composto pelo sistema de consultas e pelo módulo de interface com o usuário. Responsável por receber a consulta do usuário e retornar uma lista de documentos resposta.

- **Módulo Crawler-Indexer**: Módulo composto pelo *crawler*, pelo servidor de indexação e servidor de *links.* Responsável por gerar o índice e mantê-lo atualizado.

Esta arquitetura possui alguns problemas dos quais podemos citar:

- **Atualização do Índice**: tendo em vista conteúdo dinâmico da *Web*, cerca de 25% das páginas da *Web* existem por um intervalo de tempo de apenas 20 dias [HOW00], a tarefa de manter o índice integralmente atualizado torna-se uma tarefa muito difícil e muitas vezes impraticável;

- **Qualidade da Resposta**: Com o número cada vez maior de documentos *Web* indexados, a tarefa de encontrar documentos relevantes através do casamento de termos torna-se uma tarefa cada vez mais difícil.

O problema de atualização dos índices pode ser amenizado a partir da elaboração de políticas de atualização, que tem por objetivo atualizar apenas um subconjunto das páginas do índice segundo um determinado critério de importância.

Como foi visto anteriormente, inerentes ao engenho de busca estão diferentes mecanismos de indexação, de armazenamento, e busca. Cada um destes mecanismos pode influenciar na qualidade da resposta do sistema.

Através de métricas de avaliação da eficácia em sistemas de RI pode-se medir o impacto causado, na qualidade da resposta, pela adição, remoção ou alteração de um mecanismo pertencente ao sistema de RI. A seguir serão mostrados os principais conceitos e métricas relacionados a avaliação de sistemas de RI.

2.5 Avaliação de Sistemas de RI

Os sistemas de RI podem ser avaliados sob um conjunto de aspectos [BYRN99], quais sejam:

- **Funcionalidade:** capacidade do sistema de desempenhar as funcionalidades por ele especificadas;
- **Tempo de resposta:** tempo gasto pelo sistema para responder a uma solicitação do usuário;
- **Espaço utilizado:** espaço em memória necessário para armazenar as estruturas de dados do sistema;
- **Eficácia de recuperação:** consiste em o quão preciso é o conjunto resposta retornado por um sistema de RI a uma dada consulta. Este aspecto está relacionado à capacidade do sistema retornar todos os documentos relevantes (existentes na coleção), e apenas estes, para uma dada consulta.

Há um *tradeoff* inerente às medidas de tempo gasto e espaço utilizado, o qual faz com que o sistema de RI tenha sempre que abrir mão de uma para conseguir o outra.

Duas questões importantes ao se tratar na avaliação de sistemas de recuperação são "O que avaliar?" e "Como avaliar?". Para responder a estas perguntas serão detalhados, a seguir, as métricas e os métodos utilizados na avaliação da eficácia de sistemas de RI. Para isto, faz-se necessário, inicialmente, uma breve discussão sobre o conceito de relevância.

2.5.1 Relevância

Um documento é considerado relevante a uma dada consulta se ele atende à necessidade de informação do usuário, que é expressa através de um conjunto de termos inerentemente vagos, que constituem uma consulta.

Relevância é um conceito cognitivo, por esta razão, diferentes usuários podem possuir diferentes conceitos de relevância. Todavia, a diferença entre os conceitos de relevância não é representativa a ponto de invalidar experimentos, que se realizam a partir do julgamento de relevância de documentos pertencentes a uma dada coleção, com o objetivo de calcular medidas de eficácia de recuperação. Voohees [Voo98] verificou que o julgamento de relevância de uma dada coleção por um grupo de usuários gera resultados equivalentes aos obtidos por esta tarefa realizada por um único usuário. Na maioria dos experimentos que necessitam do julgamento de relevância de usuários comuns, com necessidades de informações específicas ficam incumbidos do julgamento de relevância de um conjunto de documentos.

O julgamento de relevância pode ser feito pela classificação do documento como relevante ou não, ou pode consistir em um julgamento mais refinado no qual é associado um nível de relevância ao documento – excelente, bom, regular, irrelevante. Exemplos da utilização destes tipos de julgamentos podem ser encontrados em [ATH00] e [HCB+01].

2.5.2 Métricas de Avaliação da Eficácia em Sistemas de RI

As métricas de avaliação da eficácia em sistemas de RI correspondem a medidas que refletem a habilidade do sistema em satisfazer o usuário. Estas medidas respondem, portanto, à questão enunciada no início deste capítulo: "O que deve ser avaliado em um sistema de RI?". As principais métricas de avaliação da eficácia são: a precisão e a cobertura. A seguir, são apresentadas as características principais destas métricas.

Precisão

Esta métrica consiste na fração dos documentos retornados que são realmente relevantes. Ou seja, é o número de documentos relevantes do conjunto resposta retornado ao usuário (*Ra*) sobre o número de documentos do conjunto resposta (*A*) .

$$P = \frac{|Ra|}{|A|}$$

A Figura 6 mostra a representação gráfica da divisão dos documentos entre relevantes e retornados durante o processo de recuperação de informação.

Cobertura

O valor de cobertura corresponde à fração do conjunto dos documentos relevantes existentes na base. Ou seja, é o número de documentos relevantes do conjunto resposta retornado ao usuário (*Ra*) sobre o número de documentos relevantes àquela consulta que existem na coleção (*R*) .

$$C = \frac{|Ra|}{|R|}$$

Medir a cobertura absoluta é uma tarefa impraticável em coleções de dados imensas e dinâmicas como a *Web*. Neste cenário são propostas técnicas que procuram estimar a cobertura absoluta a partir do cálculo da cobertura relativa [TS92]. No Apêndice A são descritas duas técnica com este propósito: a técnica de *pooling* e a técnica de amostragem aleatória.

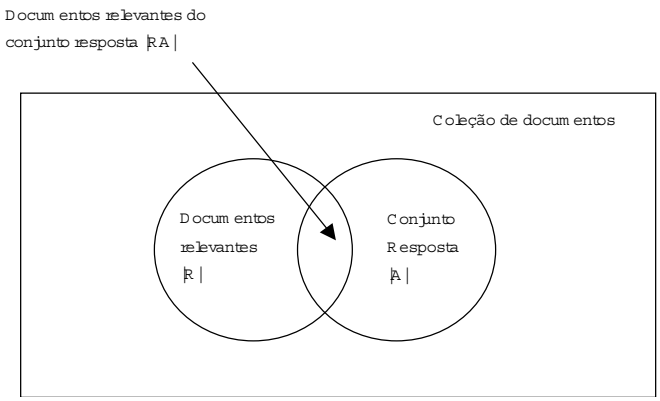

Figura 6 Representação gráfica da divisão dos documentos durante o processo de recuperação de informação.

As medidas de precisão e cobertura mostradas acima são aplicadas a uma única consulta. Para se avaliar a performance de um algoritmo que faz uso de um conjunto de consultas teste, são calculados os valores médios da precisão e da cobertura.

2.5.3 Single Value Summaries

Os valores médios de precisão e cobertura desempenham um importante papel na comparação da eficácia de diferentes sistemas de busca. Contudo, quando calculamos a precisão e a cobertura médias de um conjunto de consultas, podem passar despercebidas algumas anomalias do algoritmo em estudo quando aplicado a consultas específicas. Nestas situações podemos utilizar *Single Value Summaries* [BYRN99], medidas únicas que representam o comportamento da precisão e da cobertura ao longo do conjunto de consultas teste. Vejamos a seguir alguns exemplos destas variáveis.

Precisão Relativa

Teoricamente a precisão é uma métrica fácil de calcular. Após os documentos a lista de documentos resposta ser retornada, é preciso apenas verificar quais documentos desta lista são relevantes à consulta do usuário e calcular a proporção destes com relação ao total de documentos retornados. No entanto, na prática, esta tarefa torna-se muito difícil quando o conjunto resposta consiste em uma lista infindável de itens como acontece na maioria dos engenhos de busca de propósito geral. Nestes casos sugere-se a utilização da precisão relativa ou precisão@n [GP99], a qual consiste na precisão do conjunto dos n primeiros documentos retornados no conjunto resposta.

Como foi visto na Seção 1.1, cerca de 85,2% dos usuários visita apenas a primeira página de resultados. Tal comportamento justifica a utilização desta métrica em vários estudos que comparam a eficácia de sistemas de busca. Exemplos de estudos que utilizaram esta métrica são encontrados em [HCB+01].

Precisão Média TREC [4] *Style* (PMTS)

Esta métrica é uma das medidas utilizadas na *Text REtrieval Conference* (TREC) para avaliação de sistemas de recuperação de informação [HCT+99]. A precisão média TREC *Style* [HCB+01] é calculada dividindo-se o somatório dos valores da precisão - calculados em cada ponto onde um documento relevante é retornado - pelo número *n* de documentos relevantes existentes na coleção. Em casos onde o número de documentos relevantes na coleção não é conhecido, o valor de *n* passa a ser o número de elementos

[4] TREC (*Text REtrieval Conference*) - conferência anual de avaliação de sistemas de recuperação de informação.

avaliados para uma dada consulta. Esta medida passa a ser chamada precisão média TREC *Style* com ponto de corte n. Diferentemente da métrica original esta última não possuirá um componente referente à cobertura.

$$PMTS@n = \frac{1}{n} * \sum_{i=1}^{n} \frac{|Ra_i|}{|R_i|}$$

Onde:

- n é o número de documentos relevantes para uma dada consulta, existentes na coleção, ou o valor do ponto de corte nos casos em que o número de documentos relevantes não é conhecido.
- Ra_i corresponde ao número de documentos relevantes retornados no ponto onde o i-ézimo documento relevante é retornado.
- R_i consiste no número de documentos retornados na lista após o i-ézimo documento relevante ser retornado.

Desta forma, num processo de avaliação dos 7 primeiros documentos retornados por um sistema, se todos os 7 documentos retornados forem relevantes, a precisão média TREC-*Style* (PMTS@7) terá valor 1, e se apenas um dos documentos retornados for relevante o valor da PMTS@7 variará entre 1/7 e 1/49, dependendo da posição na qual o documento relevante foi retornado.

2.5.4 Métricas compostas

As métricas compostas para avaliação da eficácia de sistemas de RI foram criadas para representar em uma única medida o comportamento de duas ou mais medidas simples. Vejamos a seguir duas destas medidas.

Medida Harmônica ou Medida-F

A Medida Harmônica ou Medida-F [BYRN99] combina os valores de precisão e cobertura, de forma que o valor desta medida representa um equilíbrio entre ambas. A medida-f é computada como mostrado a seguir:

$$\text{Medida-F} = \frac{2}{(1/C(j)) + (1/P(j))}$$

Onde:

- C(j) é a cobertura calculada para a consulta j.
- P(j) é a precisão calculada para a consulta j.

A função F assume valores no intervalo [0,1]. Pode-se perceber que o máximo valor alcançado pela medida-f pode ser interpretado como a melhor combinação possível entre as medidas de precisão e cobertura.

Medida-E

Pode-se perceber que tanto a cobertura quanto a precisão possuem a mesma influência sob a medida-f. Porém, muitas vezes se quer aumentar uma medida em detrimento da outra. A medida-e [BYRN99] foi proposta com o objetivo de permitir que o usuário especificasse qual a métrica de maior interesse. A medida-e é definida a seguir:

$$\text{Medida-E} = \frac{1+b^2}{(b^2/C(j)) + (1/P(j))}$$

Onde:

- $b \in \mathbb{R}^+$

Desta forma, valores de b maiores que 1 indicam que está sendo dada uma maior "importância" à precisão, e valores de b menores que 1 indicam que uma maior "importância" está sendo dada à cobertura.

2.6 Métodos de Avaliação

Tradicionalmente os sistemas de recuperação são avaliados usando medidas de precisão e cobertura calculadas a partir de uma coleção de documentos previamente avaliados. Porém, no contexto da *Web,* não existe nenhum procedimento padrão de avaliação que possa ser aplicado diretamente, pelas seguintes razões:

1. A *Web* possui milhares de páginas, logo etiquetar o *corpus* inteiro corresponde a uma tarefa impossível. Não seria viável etiquetar apenas parte deste *corpus* uma vez que a maioria das consultas retorna um conjunto de páginas que está espalhado por toda *Web;*

2. Mesmo que fosse possível a criação de tal *corpus*, milhares de páginas surgem e desaparecem a cada dia, o que tornaria o tempo de vida útil deste *corpus* muito curto. Este *corpus* seria utilizado até o momento em que surgisse um grande número de novas páginas não contempladas pelo mesmo;

3. Existem coleções de documentos *Web* previamente etiquetados como os disponibilizados pelo TREC. Inicialmente estas *coleções* eram compostas apenas por texto, porém a partir do ano 2000 o TREC passou a disponibilizar coleções de documentos hipertexto. Muito se tem discutido acerca da representatividade destas coleções, na avaliação de sistemas que levam em consideração as informações contidas nos *links. U*ma vez que a maioria dos *links* existentes nestas coleções referencia documentos de mais alta qualidade do que os existentes na *Web*.

Tendo em vista as dificuldades de se aplicar diretamente um procedimento para avaliação de sistemas de busca para a *Web*, surge a necessidade da utilização de métodos para avaliação de sistemas desta natureza. O Apêndice B descreve quatro métodos de avaliação com suas características.

Os métodos de avaliação possuem uma série de características em comum, o que torna possível a enumeração do conjunto de passos que compõem a maioria destes métodos, quais sejam:

1. Escolha de um conjunto de consultas e de um número de páginas que deverá ser retornada nas buscas por cada consulta.

2. As consultas são submetidas aos engenhos de busca que participam da avaliação, é feita uma fusão das respostas oriundas de cada sistema.

3. O usuário através de um julgamento cego classifica as páginas segundo alguns critérios de relevância.

A maioria dos métodos de avaliação de sistemas de recuperação de informação para a *Web* se baseia no julgamento de relevância cego (*blind post-facto relevance judgements*) [CDG+98c], que possui este nome porque durante o julgamento os avaliadores não sabem qual sistema estão avaliando.

Hawking e Craswell [HCB+01] perceberam que o que diferencia um método de

outro são: (i) as consultas teste; (ii) os níveis de relevância considerados; (iii) o número e o nível de conhecimento dos avaliadores; (iv) e as métricas de avaliação utilizadas. Baseados nesta constatação, Hawking e Craswell elaboraram uma série de premissas que devem orientar estes métodos, a seguir são enumeradas algumas destas premissas:

- As consultas teste devem representar necessidades reais do usuário;
- Se há um módulo intermediário entre o sistema de RI e o usuário, o primeiro deve se comprometer a repassar as informações de forma integral ao sistema;
- Um grande número de consultas teste deve ser utilizado;
- As consultas teste devem representar uma grande variedade de necessidades de informação, tanto no que diz respeito ao assunto abordado quanto ao tipo de resultado requerido (se o usuário procura por um site, por um assunto, por uma resposta a uma pergunta, entre outros);
- Julgamento da resposta deve ser apropriado ao tipo de resultado solicitado;
- Todos os julgamentos relativos a uma consulta devem ser feitos por um único usuário em um curto intervalo de tempo;
- Os documentos a serem julgados devem ser apresentados da mesma forma que são visualizados em uma situação real;
- Os experimentos devem ser bem projetados e fáceis de serem reproduzidos.

Porém, algumas questões se mantêm em aberto no trabalho de Hawking e Craswell, são elas: "Como validar uma boa uma coleção de testes e quais métricas utilizar?". Buckley e Voohees [BV00] respondem esta questão. A partir da verificação do impacto causado pelo número de consultas testes e pelas métricas utilizadas, na confiabilidade dos resultados Buckley e Voohees sugerem uma conbinação ideal entre estes dois componentes do processo de avaliação. Para técnicas de avaliações que utilizam a Web como coleção de dados, Buckley e Voohees sugerem a utilização de 100 consultas teste, e da Precisão Média TREC *Style* (PMTS@n) com ponto de corte variando entre 10 e 20, uma vez que esta combinação mostrou taxas de erro aceitáveis (inferiores a 2,9%).

2.7 Considerações Finais

Nesse capítulo, foram apresentadas as principais técnicas, modelos e estruturas usadas em sistemas de recuperação de informação. Também foi fornecida uma visão geral acerca dos sistemas de recuperação de informação para a *Web*.

Foram apresentadas as métricas utilizadas com maior freqüência na avaliação de sistemas de RI: a cobertura e a precisão. Como a melhoria de uma destas métricas muitas vezes ocorre em detrimento da outra, os experimentos que utilizam estas duas métricas sempre procuram um valor de equilíbrio entre elas através da análise dos valores de métricas compostas como a medida-f ou a medida-e.

No capítulo seguinte serão apresentados aspectos relacionados à avaliação de sistemas de recuperação de informação.

3

Algoritmos de Análise de *Links*

Antes da era WWW, as estruturas de *links* já haviam sido estudadas pela biblioteconomia, uma área que estudava o comportamento das citações existentes entre documentos escritos [LM00]. Muitos trabalhos desta área objetivavam encontrar, dentre os artigos publicados em jornais científicos [PN76], quais os mais influentes em uma determinada área, como também encontrar o número de co-citações entre artigos, isto é, a freqüência com que ambos são citados por outros documentos [CDG+98a].

Uma das medidas mais conhecidas nesta área é o fator de impacto de Garfield ([Gar72], referenciado por [Kle97]). Este fator pode ser definido como o número de citações a um dado documento j em um intervalo de tempo de dois anos. Pinski e Narin ([PF76] referenciado por [Kle97]) partindo da observação de que nem todas as citações são igualmente importantes, elaboraram o seguinte conceito de influência: um documento é influente se ele é citado por documentos influentes.

Os *links* existentes entre páginas *Web* são similares às citações entre os artigos escritos, estudados pela biblioteconomia, porém, com importantes peculiaridades que merecem ser ressaltadas:

- As citações na literatura científica são estáticas e unidirecionais. Uma vez que um artigo é publicado, é inviável a inclusão de referências neste. Por esta razão, raramente, dois artigos citam um ao outro. Em contrapartida, as páginas *Web* freqüentemente referenciam páginas que surgem após sua criação. Por este motivo

a distância média entre duas páginas *Web* é relativamente pequena. De acordo com os estudos de Adamic [Ada99] 19 cliques, em média, separam quaisquer 2 páginas *Web,* escolhidas aleatoriamente;

- Os *links* entre páginas *Web* possuem diferentes utilidades em um site [CDG+99a]: (i) os ***links* estruturais** permitem ao usuário navegar pelas páginas de um site, ou de um domínio específico; (ii) os ***links* funcionais** conectam *sites* de diferentes domínios, que possuem conteúdos relacionados, e possivelmente relevantes, segundo a opinião do autor do documento origem do *link*; (ii) os ***links* de propaganda** que surgem quando uma página p faz propaganda de uma página q. Estes últimos podem ser provenientes de *banners* e demais formas de propaganda constantemente presentes na *Web*. Como *links* de propaganda se originam a partir do pagamento de mensalidades ou de acordos entre *sites*, os mesmos não refletem a opinião do usuário a respeito do *link* criado.

Os Algoritmos de Análise de *Links* analisam a estrutura de *links* de uma coleção de documentos *Web*, com o objetivo de extrair, desta coleção informações que podem ser utilizadas para vários propósitos [CDK+99, CDG+99b]. Constituem objetivos dos algoritmos de análise de *links*: (i) *topic destillation* - associar um peso de "importância" a páginas *Web* [Kle97, BH98, CDR+98, Hav99]; (ii) identificar comunidades na *Web* – as comunidades são definidas por um conjunto de páginas que se auto-referenciam e abordam um tópico específico [FLG00, GKR98a, KRR+99]; (iii) encontrar as páginas similares a uma página *Web* – funcionalidade "páginas parecidas" encontrada em alguns engenhos de busca [DH99]; (iv) identificar a reputação de uma página na *Web* [MR00, RM00, Men00]; elaborar políticas de *crawling* [DCL+00, CBD99]*;* (v) e classificar documentos *Web* [Cha99, Cha00]; (vi) associar contexto às consultas realizadas submetidas aos engenhos de busca [Low00, LAW00, CHR01].

Este capítulo irá focar nos algoritmos de *topic distillation,* responsáveis por associar um peso de "importância" às páginas *Web* através da análise da estrutura de *links*. Esta informação é utilizada pelos engenhos de busca no momento do ranqueamento da resposta com o objetivo de aumentar a eficácia de recuperação em consultas por tópico geral.

Vale a pena salientar que a utilização das informações contidas nos *links* se baseia na noção de que os *links* conectam documentos relacionados, que podem prover informações adicionais sobre o documento apontado. Contudo, no contexto da *Web*, os

*link*s conectam documentos de diversas qualidades e por várias razões como visto acima. Por esta razão os algoritmos de *topic distillation* podem muitas vezes inserir ruídos ou até mesmo degradar a performance de recuperação dos sistemas. Com o intuito de solucionar este problema, alguns algoritmos de *topic distilaion* utilizam heurísticas que extraem apenas os *links* funcionais de um subconjunto da *Web*. O Apêndice C traz uma coletânea de técnicas capazes de filtrar os *links* funcionais de uma amostra da *Web,* e algumas das dificuldades enfrentadas por elas.

Com o objetivo de simplificar a descrição dos algoritmos de *topic distillation*, primeiramente serão detalhadas as principais características presentes na estrutura de *links* da *Web*.

3.1 O Grafo de Links da *Web*

Uma das formas mais comuns de se representar o grafo Web, é através de um grafo direcionados, no qual as páginas correspondem aos nós e os *links* entre as páginas correspondem às arestas [KKR+99, CHK+01]. Dois *links* entre documentos *Web* podem pertencer a uma das configurações mostradas na Figura 7 [ERC+00].

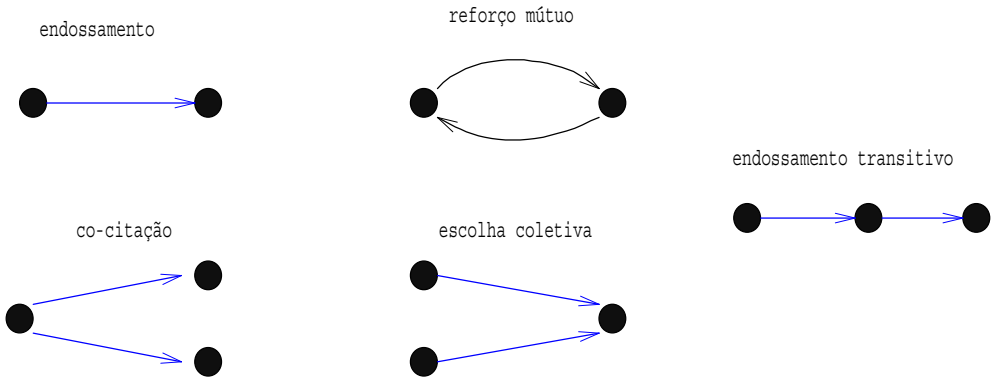

Figura 7: Padrões formados por dois *hiperlinks* da *Web* – adaptado de [ERC+00].

Segundo estudos realizados em abril do ano 2000 [ASG+00], o grafo da *Web* brasileira é composto por 11.546.360 nós (documentos) cada um contendo em média 6,74 *links*.

Cada documento *Web* contém um determinado número de *links* de saída (*forward links ou outlinks*), e um determinado número de *links* de entrada (*backlinks ou inlinks*). O número de *links* de saída corresponde ao número de *links* existente no corpo da página, mas não se pode saber com exatidão quantos *backlinks* uma página possui, a

menos que a *Web* inteira fosse indexada, tarefa que se torna inviável devido às proporções da *Web* e sua alta taxa de atualização.

O grafo *Web* pode ser representado sob a forma de uma matriz quadrada M onde cada entrada (i,j) desta matriz é igual a 1 se existe um *link* do documento i para o documento j, ou 0 caso contrário. Esta matriz é chamada matriz de adjacência ou matriz de transição. Algumas extensões desta matriz associam pesos aos arcos, desta forma cada entrada da matriz passa a conter um valor wij, ao invés do valor booleano descrito inicialmente.

A análise de um subgrafo da *Web* contendo cerca de 200 milhões de páginas e 1.5 bilhões de *hiperlinks* realizada por Broder et al. [BKM+00] revelou que a macro estrutura da *Web* pode ser representada a partir da Figura 8.

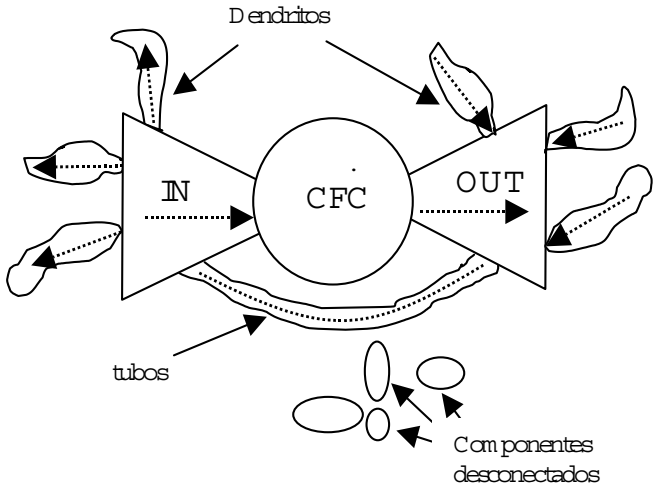

Figura 8: Representação do grafo *Web* – adaptado de [BKM+00].

Nesta representação, a *Web* pode ser naturalmente dividida em quatro partes: CFC, IN, OUT, e os componentes desconectados. A primeira parte consiste no coração da *Web*, onde todas as páginas podem alcançar umas às outras, através da estrutura de *hiperlinks* existente entre elas - este componente é chamado de componente fortemente conectado (CFC). Na parte IN, localizam-se as páginas que podem alcançar as páginas do CFC, mas não podem ser alcançadas pelas páginas dele – possivelmente as páginas que compõem esta parte correspondem a *sites* novos, que ainda não foram descobertos e por este motivo não possuem *backlinks*. Na parte OUT, são encontradas as páginas que são acessíveis pelo CFC, mas que não possuem *links* de volta para ele – estas páginas podem corresponder a *sites* corporativos que contém apenas *links* internos. Os

componentes desconectados correspondem às páginas pertencentes a intranets, que muitas vezes não apontam e não são apontadas por páginas pertencentes às demais partes.

Um usuário pode passar de qualquer nó na parte IN para qualquer nó na parte OUT passando através de nós em CFC. Os dendritos contêm os nós que são alcançados através de porções do conjunto IN, ou que podem alcançar porções do conjunto OUT. Quando os nós, que são alcançados através de porções do conjunto IN, podem alcançar porções do conjunto OUT, ocorre a formação dos tubos, os quais constituem passagens de uma porção em IN para uma porção em OUT, sem que para isto seja necessária a passagem pelo CFC.

3.2 Algoritmos de *Topic Distillation*

Os algoritmos de *Topic Distillation* associam pesos de importância às página*s Web* e são utilizados pelos engenhos de busca no momento do ranqueamento das páginas. A utilização deste peso possibilita os engenho de busca retornem as páginas que além de relevantes à consulta são também consideradas importantes para tópico consultado, segundo o algoritmo de *Topic Distillation*.

Estes algoritmos podem ser divididos em duas famílias: algoritmos dependentes da consulta e algoritmos independentes da consulta. A seguir serão abordados os principais algoritmos pertencentes a cada uma destas famílias.

3.2.1 Algoritmos Dependentes da Consulta

Esta família de algoritmos associa uma medida de qualidade a cada página pertencente a um subconjunto da coleção de documentos *Web* retornados em resposta a uma consulta.

Carrie e Kazman [Hen00] propuseram um algoritmo que combinava a análise da estrutura de *links* com a consulta do usuário. Para cada consulta o seguinte processamento era realizado: (i) um subgrafo contendo páginas que continham o tópico da consulta era construído - este conjunto inicial era formado pelas primeiras 200 páginas retornadas, através de meta-busca a um engenho de busca baseado em *matching* de termos; (ii) este conjunto era expandido passando a incluir a sua vizinhança – a vizinhança era formada por todas as páginas apontadas pelas páginas do conjunto inicial

e um número limitado de páginas que apontavam para as páginas do conjunto inicial (o número de *backlinks* era limitado em 50); (ii) as páginas eram ranqueadas de acordo com o seu número de *backlinks* no subgrafo expandido.

Como discutido anteriormente, algoritmos baseados no número de *backlinks* possuem o problema de atribuir o mesmo peso a cada um dos *backlinks*.

Partindo do pressuposto que nem todos os *backlinks* têm o mesmo nível de importância - se uma página A aponta um grande conjunto de documentos considerados relevantes na *Web*, isto significa que a opinião embutida em A se torna mais valiosa que a opinião contida em um site C que aponta para páginas que não são consideradas relevantes - Kleimberg [Kle97] implementou o algoritmo *Hyperlinked Inducted Topic Search* (HITS) que será detalhado a seguir.

A partir do HITS surgiram vários outros algoritmos de análise de *links*, com o objetivo de sanar algumas das deficiências do HITS originadas a partir da análise pura da estrutura de *links*. A seguir serão explicados o algoritmo HITS e suas extensões.

3.2.1.1 *Hiperlink Inducted Topic Search* (HITS)

O algoritmo *Hyperlinked Inducted Topic Search* (HITS) [Kle97, Kle99, GKR98b], proposto por Jon Kleinberg, baseia-se nos conceitos de *hub*, autoridade e na relação de reforço mútuo. Um *hub* é uma página que contém *links* para muitas outras páginas. De forma semelhante, uma autoridade é uma página que é apontada por muitas outras. A relação de reforço mútuo estabelece que um bom *hub* é aquele que possui *links* para boas autoridades e boas autoridades são aquelas referenciadas por bons *hubs*.

A Figura 9 representa graficamente os conceitos de *hub* e autoridade.

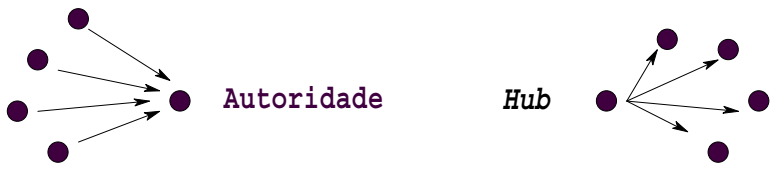

Figura 9: Autoridades e hubs.

O algoritmo HITS é utilizado para encontrar em meio a um subconjunto de páginas relacionadas a uma consulta, as páginas melhor referenciadas (autoridade) e as páginas que possuem o melhor conjunto de apontadores (*hub*s). O HITS possui os seguintes passos:

Passo 1 - Construção do Grafo Base: (i) o usuário submete uma consulta a um engenho de busca via meta-busca; (ii) um conjunto inicial de páginas, denotado por conjunto raiz, é retornado - o conjunto raiz pode não conter páginas autoridade, contudo uma vez que muitas das páginas pertencentes a este conjunto são relevantes ao tópico da consulta, pode-se esperar que algumas destas páginas apontem para páginas autoridade ou que algumas delas sejam apontadas por bons *hubs*; (iii) o conjunto raiz expandido, de forma a incluir as páginas que apontam e são apontadas por páginas contidas no conjunto raiz - este novo conjunto formado pelas páginas da resposta e sua vizinhança é chamado grafo base; (iv) o grafo base passa por um processo, chamado de limpeza do grafo, que elimina os arcos direcionados intrínsecos, ou seja, os arcos que conectam nós em um mesmo domínio da Internet, dando origem a um grafo direcionado contendo N nós.

Passo 2 - Cálculo dos Pesos de *Hub* e Autoridade: (i) associa-se a cada nó (página) do grafo base um peso de *hub* e um peso de autoridade, que é calculado de acordo com a relação de reforço mútuo através de sucessivas iterações das seguintes equações[5] (1) e (2); (ii) a computação dos pesos de *hub* e autoridade se repete até o momento em que os valores de autoridade e *hub* não mais variam acima de um valor pré-estabelecido entre sucessivas iterações. Neste momento diz-se que esta computação converge.

$$a[n] = \sum_{(n',n) \in N} h[n'] \qquad (1, [Kle97])$$

$$h[n] = \sum_{(n,n') \in N} a[n'] \qquad (2, [Kle97])$$

Passo 3 - Filtro de *Hubs* e Autoridades: após a convergência uma lista contendo as *c* páginas maiores autoridade e os *c* maiores *hubs* é retornada (neste algoritmo o valor de *c* varia entre 5 e 10).

A computação de *hubs* e autoridades converge para vetores com valores fixos, a medida em que o número de iterações aumenta, fazem-se necessárias algumas noções de álgebra linear [GL96]. A seção a seguir define os conceitos necessários para se

[5] O par ordenado (n,n') representa um arco partindo do nó n e chegando ao nó n', ambos pertencentes ao conjunto N de todos os nós do grafo.

comprovar a convergência deste algoritmo.

Características do algoritmo

Como a resposta deste algoritmo consiste apenas em retornar os maiores *hubs* e as maiores autoridades, apenas valor relativo dos pesos de *hub* e autoridade é levado em consideração. Na manipulação dos pesos é aplicada uma normalização para evitar o *overflow* dos pesos durante o processo iterativo. Contudo, é importante ressaltar que a normalização escolhida não afeta os resultados obtidos.

Após o conjunto raiz ser formado, o HITS consiste em um algoritmo baseado unicamente na análise da estrutura de *links*. Kleimberg [Kle97] verificou que a partir da computação descrita acima, o HITS retorna a coleção de autoridades e *hubs* mais densamente conectada contida no grafo base. Esta característica do algoritmo pode provocar o que ele chamou de difusão e generalização da resposta, que foi posteriormente denominado por Lempel [Lem99] como efeito da comunidade fortemente conectada (T*ightly Knit Community effect – TKC effect*).

O efeito TKC ocorre quando um conjunto de páginas recebe altos pesos de autoridade ou *hub* por pertencerem a um núcleo de páginas fortemente conectadas, não refletindo o conceito de importância associado à concepção deste algoritmo [Kle97].

Como exemplo, considere uma coleção C contendo as seguintes comunidades: (i) a comunidade *y*, contendo um pequeno número de *hubs* e autoridades, e onde cada *hub* aponta para a maioria das autoridades; (ii) e uma comunidade z contendo um maior número de *hubs* e autoridades que a comunidade *y*, porém cada *hub* aponta para um pequeno número de autoridades.

A comunidade z cobre o tópico de maior abrangência na coleção C, porém a comunidade y está densamente interconectada. O efeito *TKC* ocorre quando as páginas da coleção y obtêm maior peso de autoridade que as páginas da coleção z.

Os fatores que podem causar o efeito TKC são recorrentes em diferentes contextos e parecem refletir o interesse dos usuários das diferentes comunidades de usuários existentes na *Web*. Dentre estes fatores os seguintes podem ser enumerados: comercialização[GKR98a, Dav00], subtópicos "*Web-Centric*" [GKR98a, Lem99], fatores temporais [GKR98a].

- Comercialização: As propagandas realizadas na *Web* influenciam de forma considerável a estrutura de *links* da *Web*. Como descrito no capítulo 4, os *links* de

propaganda não carregam a opinião do usuário a respeito da qualidade da página apontada. Os *links* de propaganda existem em grande quantidade na *Web*, e por não serem filtrados no passo 1 do HITS acabam por influenciar na lista de páginas retornadas por este algoritmo. Nos casos em que os tópicos consultados possuem interesses individuais e comerciais, as páginas com maior peso de autoridade tendem a ser páginas mais fortemente relacionadas a interesses comerciais. Este fenômeno acontece com freqüência, e é o responsável pela presença de *sites* como www.altavista.com, www.yahoo.com, e www.microsoft.com como algumas das principais autoridades retornadas, independentemente do tópico consultado. Este problema pode ser solucionado a partir da remoção destas páginas do grafo base, através da utilização do um arquivo de "*stop-pages*".

- **Subtópicos "*Web–Centric*":** Para explicar este fenômeno torna-se necessário definir o conceito de generalidade de um tópico no ambiente *Web*. A generalidade de um tópico na *Web* está fortemente relacionada à forma na qual as páginas relativas a este tópico estão representadas na WWW. Desta forma, alguns tópicos podem parecer artificialmente gerais ou artificialmente específicos de acordo com o maior ou menor interesse, respectivamente, dos criadores das páginas a respeito destes tópicos. Subtópicos "*Web–Centric*" são comunidades de páginas da *Web* fortemente conectadas, e que dominam o peso de autoridade (ou *hub*) calculado sobre o grafo no qual esta comunidade está presente. O tópico desta comunidade pode representar conceitos mais gerais ou mais específicos que o tópico da consulta. Por exemplo, uma consulta por "Literatura Alemã" pode retornar páginas sobre "Literatura Européia", quando aquela não está bem representada na *Web* [GKR98a].

- **Fatores temporais:** a *Web* é uma coleção de dados dinâmica onde a cada dia surgem novos documentos ao passo em que outros deixam de existir. Esta característica da *Web* associa fatores temporais na computação dos pesos de autoridade e hub, os quais podem influenciar no conjunto das páginas autoridade e hub retornadas como resposta. Por exemplo, uma consulta pelo termo "Natal" pode tornar páginas autoridade contendo cartões natalinos ou sobre a cidade de Natal dependendo da época do Ano em que esta consulta é realizada

Em trabalhos subseqüentes, o algoritmo HITS é estendido [BH98, CDR+98, CDG+98b], passando a conter novas heurísticas que levam em consideração o conteúdo

das páginas do conjunto base, os servidores aos quais pertencem, dentre outras características intrínsecas aos documentos. Contudo a maioria destas extensões mantém o processo iterativo de computação dos pesos de *hub* e autoridade sendo estes pesos representados pelo autovetor principal originário da computação iterativa.

3.2.1.2 *Improved Algorithms for Topic Distillation in a Hyperlinked Enviroment* (IMPL)

O algoritmo IMPL originou-se a partir da análise do HITS e da detecção de algumas falhas originadas pelo desprezo do conteúdo das páginas no momento da computação dos pesos de *hub* e autoridade. Bharat e Henzinger [BH98] perceberam que o HITS retornava boas respostas para algumas consultas, porém, possuía uma baixa performance de recuperação para a maioria dos casos de testes. As prováveis causas para a baixa precisão da resposta, foram detectadas por Bharat e Henzinger [BH98] e são enumeradas a seguir:

- Relação de reforço mútuo entre *hosts*: Este comportamento acontece quando várias páginas de um ***host*** A apontam para uma única página em um ***host*** B. Isto irá inflacionar o peso de autoridade do site no ***host*** B, e conseqüentemente irá aumentar os pesos de hub das páginas do ***host*** A (Cenário 1, Figura 10). Uma situação oposta, mas com o mesmo efeito colateral ocorre quando um único documento em um ***host*** A aponta para vários documentos em um ***host*** B, o peso de Hub do documento presente no ***host*** A será inflacionado e conseqüentemente os pesos de autoridades das páginas apontadas presentes no ***host*** B (Cenário 2, Figura 10). Assumindo que as páginas de um mesmo *host* foram criadas por uma mesma pessoa ou organização, conferem peso de autoridade (ou hub) baseados na opinião de uma única entidade, o que não corresponde ao conceito de autoridade que se baseia na opinião coletiva dos usuários a respeito de uma página.

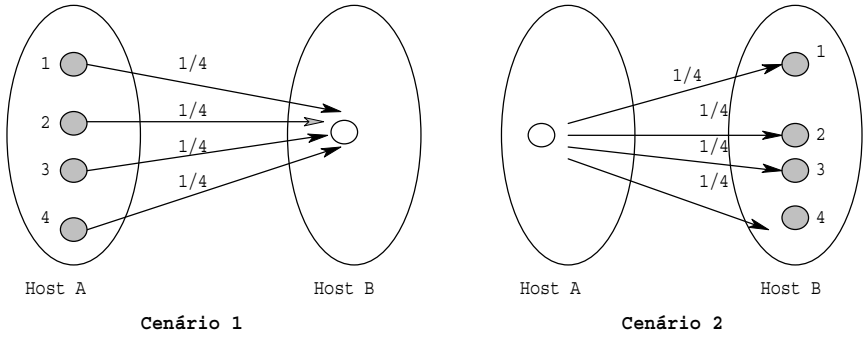

Figura 10: Relação de reforço mútuo entre *hosts*.

- *Links* gerados automaticamente: este comportamento se faz presente porque algumas ferramentas para construção de documentos *Web*, inserem *link*s automaticamente nas páginas que estão sendo geradas (ex: *link*s de copy right). Neste caso a hipótese de que o *link* embute a opinião do usuário não é real.

- *Topic drift*: ocorre no momento da expansão do grafo resposta, quando são inseridas páginas não relevantes ao tópico da consulta. Estas páginas podem estar melhor representadas (mais fortemente conectadas) que as páginas realmente relevantes para o tópico da consulta e por este motivo serem retornadas como as melhores autoridades pelo algoritmo HITS.

A essência do algoritmo IMPL está em incorporar análise de conteúdo juntamente a uma análise mais refinada da estrutura de *links* com o intuito de sanar os problemas enumerados acima.

Para solucionar o problema do reforço mútuo entre *hosts* todos os documentos de um dado *host* deveriam ter a mesma influência em um documento conectado a eles em outro *host*. Tal objetivo é alcançado atribuindo pesos aos arcos da seguinte forma:

Sendo k um número de *links* partindo de documentos em um *host* A para um documento em um *host* B, é atribuído um peso impl_aut igual a 1/k a cada um destes *links*. Este valor será utilizado para o cálculo do peso de autoridade da página presente no *host* B.

Em contrapartida, se *n links* de um único documento em um *host* A apontam para um conjunto de documentos em um *host* B, será atribuído a cada *link* um peso impl_*hub* igual a 1/*n*. Este valor será utilizado para o cálculo do peso de *hub* da página presente no *host* A.

A inclusão do peso do arco para o cálculo dos pesos de autoridade e *hub* acarreta

em modificações nas formulas (1) e (2) utilizadas pelo HITS, que podem ser representadas pelas fórmulas abaixo:

$$a[n] = \sum_{(n',n) \in N} h[n'] \times impl\_aut(n',n) \qquad (3, [BH98])$$

$$h[n] = \sum_{(n,n') \in N} a[n'] \times impl\_hub(n,n') \qquad (4, [BH98])$$

Onde:

- a é um vetor que armazena os pesos de autoridade e contém uma entrada para cada página do grafo base.
- h é um vetor que armazena os pesos de hub e contém uma entrada para cada página do grafo base.
- impl_aut e impl_hub são pesos conferidos aos *link*s.

Para solucionar o *topic drifft* e os problemas causados pela inclusão de *links* gerados automaticamente no grafo base foi necessária a incorporação de uma análise de conteúdo juntamente à computação dos pesos de *hub* e autoridade.

Bharat e Henzinger [BH98] oferecem duas alternativas para esta combinação: (i) eliminar os nós irrelevantes do grafo, (ii) regular a influência de um nó de acordo com sua relevância.

Peso de relevância

O peso de relevância é calculado a partir do cálculo de similaridade entre a consulta e o documento, como visto na Seção 2.3.2. O algoritmo em questão utiliza a seguinte medida de similaridade entre um documento e uma consulta:

$$similaridade(Q, D_j) = \frac{\sum_{i=1}^{t} (w_{iq} \times w_{ij})}{\left(\sum_{i=1}^{t} (w_{iq})^2 \times \sum_{i=1}^{t} (w_{ij})^2\right)^{1/2}} \qquad (5, [BH98])$$

Onde:

- Q é a consulta expandida contendo os primeiros 1000 termos de cada documento do conjunto raiz, ou de um subconjunto de documentos.

- Dj é um documento.
- $w_{ig} = tf_{ig} * idf_i$
- $w_{ij} = tf_{ij} * idf_i$
- tf_{ij} é a freqüência com que o termo i aparece no documento j.
- idf_i é a freqüência com que o termo i aparece nos documentos da base.

O peso de relevância pode ser utilizado para excluir nós da computação (podar nós) de acordo com algum limiar que pode ser um dos seguintes:

- Peso médio: são excluídas todas as páginas com peso inferior à média dos pesos de relevância.
- Peso médio do conjunto raiz: são excluídas todas as páginas com peso inferior a média dos pesos de relevância das páginas do conjunto raiz.
- Fração do peso máximo: são excluídas todas as páginas com peso inferior a uma fração do maior peso.

A poda dos nós da computação pode ocorrer uma única vez antes do início da computação dos pesos de *hub* e autoridade, ou de forma iterativa, onde a cada conjunto de iterações é excluído um conjunto de nós que não possuem o valor de relevância desejado.

Este peso também pode ser utilizado para regular a influência de um nó na computação dos pesos de autoridade e *hub* da seguinte forma:

$$a[n] = \sum_{(n',n) \in N} h[n'] \times impl\_aut(n',n) \times similaridade(n,q) \qquad (6, [BH98])$$

$$h[n] = \sum_{(n,n') \in N} a[n'] \times impl\_hub(n,n') \times similaridade(n,q) \qquad (7, [BH98])$$

Onde:

- similaridade(n,q) corresponde ao peso de similaridade do documento n com relação à consulta q.

Experimentos [BH98] mostraram uma melhoria significativa na precisão do conjunto resposta formado pelas páginas autoridade retornadas pelo IMPL, utilizando as heurísticas que amenizam a relação de reforço mútuo entre *hosts* e a heurística de poda

de páginas, em comparação às páginas autoridades retornadas pelo algoritmo HITS. O tempo de computação do IMPL é , em média, de 3 minutos para cada consulta.

3.2.1.3 Automatic Resource Compiler (ARC)

O Algoritmo ARC [CDR+98, CDG+98b] foi elaborado como parte do Projeto CLEVER do IBM *Almaden Research Center*. O ARC corresponde a uma extensão do algoritmo HITS, que além de analisar a estrutura de *links* existente entre as páginas, analisa o conteúdo das mesmas.

Este algoritmo possui os mesmos 3 passos utilizados pelo HITS para o cálculo dos pesos de *hub* e autoridade, porém com algumas peculiaridades no passo 2: (i) é atribuído um peso positivo w(p,q) a cada *link* de uma página p para uma página q – entrada da matriz de adjacência M contém o valor w(p,q). Este peso aumenta a medida em que são encontrados termos relativos à consulta na vizinhança do *link* e é calculado de acordo com a fórmula (8):

$$w(p,q) = 1 + n(t) \qquad (8,[CDR+98])$$

Onde:

- n(t) é o número de casamentos entre os termos da consulta e os termos da vizinhança do *link*.

Os valores de w devem ser maiores que 1, para que os valores de *hub* e autoridade cresçam com o número de iterações assim como no algoritmo HITS. Um termo pertence à vizinhança de um *link* se ele está contido na janela de bytes pertencente ao *link*. A janela de bytes consiste no número de bytes que devem ser considerados em ambos os lados do * *. Através de experimentos realizados no projeto CLEVER [CDR+98] concluiu-se que o tamanho ideal para esta janela é de 50 bytes.

3.2.1.4 *Spectral Filtering for Resource Discovering* (SFRD)

O algoritmo SFRD consiste na segunda fase do *Automatic Resource Compiler* (ARC) implementado no Projeto *CLEVER* do *IBM Almaden Research Center*.

Da mesma forma que o ARC o SFRD [CDG+98a] é baseado no conteúdo da página e no seu contexto, que consiste no conteúdo das páginas que apontam e são apontadas por ela. Este algoritmo trabalha com o conceito de janela de bytes, descrito na

seção anterior. Tal conceito é de grande valia, pois muitas vezes o texto presente nas redondezas do *link* está relacionado com o conteúdo da página apontada.

O algoritmo de análise de *links* SFRD é representado pelos seguintes passos:

Passo 1 - Construção do Grafo Base: neste passo foram incluídas várias heurísticas, não existentes no HITS, com o objetivo de melhorar a qualidade das páginas *hub* e autoridade retornadas pela análise de *links*, quais sejam:

- Páginas do mesmo site: para evitar a autopromoção, isto é, páginas que conferem peso de autoridade para páginas do mesmo site, esta heurística descarta os *link*s para páginas do mesmo site quando da construção do grafo inicial. Duas páginas são consideradas do mesmo site quando os IPs [Hun98] correspondentes:
  - ♣ Pertencem às classes A ou B e os dois octetos mais significativos coincidem;
  - ♣ Pertencem à classe C e os 3 octetos mais significativos coincidem;
  - ♣ Pertencem à classe D e todos os octetos coincidem.
- Heurística de cobertura: esta heurística objetiva retornar entre os 10 primeiros *hubs*, aqueles que apontem para o menor número de páginas em comum. Ao final de cada iteração que converge os seguintes passos são executados:
  1. Os melhores *hubs* são retornados;
  2. As autoridades que apontam para estes *hub*s são zeradas;
  3. Os valores de *hub* são computados novamente.
- Heurística de empacotamento: nesta heurística os *link*s intrínsecos são considerados como "empacotadores de autoridade". Para tanto a cada iteração os pesos de autoridade são distribuídos da seguinte forma: Se múltiplos documentos de um único site possuem peso de autoridade diferente de zero os pesos de autoridade são todos zerados com exceção da página com maior peso de autoridade.

Passo 2 - Cálculo dos Pesos de *Hub* e Autoridade: Cada entrada (p,q) da matriz de adjacência utilizada no algoritmo contém o valor w(p,q), calculado da seguinte forma:

$$w(p,q) = 0 \text{ se não há um link da página p para a página q} \quad (9,[SBD+98])$$
$$\text{ou } f(p,q) \text{ caso contrário.}$$

$$f(p,q) = 1 + \sum (tam\_janela - posição(ti)) + w(p,q) + eh\_raiz$$

Onde:

- tam_janela corresponde ao número de bytes antes e depois do *link*.
- ti é o termo pertencente à consulta.
- posição(ti) corresponde a distância do termo ti ao *link* ( ).
- eh_raiz: corresponde ao número de elementos do par (p,q) que pertence ao conjunto raiz.
- T é o conjunto dos termos que compõem a consulta.

O valor de afinidade f(p,q) corresponde à soma de 3 componentes: o primeiro é o valor *default* atribuído a todo *link* (geralmente 1), o terceiro varia com a presença de páginas no conjunto raiz, e o segundo corresponde à contribuição de cada termo da consulta, esta contribuição é proporcional à distância do termo a janela de bytes.

Passo 3 - Filtro de *Hubs* e Autoridades: Após a convergência e retornada uma lista contendo as 5 maiores autoridades e os 5 maiores *hub*s.

3.2.1.5 DiscoWeb

Este algoritmo foi desenvolvido a partir da constatação de que nem sempre o autovetor principal (autovetor gerado após a convergência HITS) contém o melhor ranking, pois como o grafo é composto de vários subgrafos desconexos, o autovetor principal pode "perder" vários nós relevantes que faziam parte de grafos mais fracamente conectados.

O algoritmo DiscoWeb [DGK+99a, DGK+99b] computa ¼ dos autovetores correspondentes ao grafo base - grafo composto pelas páginas da resposta juntamente com páginas que apontam e são apontadas por estas páginas. Estes autovetores são analisados, e são utilizadas heurísticas para encontrar as comunidades de páginas que aparecem em um ou mais autovetores, e posteriormente atribuir valores de *ranking* locais e globais a cada comunidade existente em um ou mais autovetores. Logo, se uma

determinada página pertence a comunidades que aparecem em vários autovetores, mas não são contempladas pelo autovetor principal, possuirão um valor de *ranking* baseado em nos valores de *ranking* computados para cada autovetor.

3.2.2 Algoritmos Independentes da Consulta

Os algoritmos independentes da consulta ou globais associam uma medida de qualidade intrínseca a cada página da coleção, sem levar em consideração informações textuais das páginas ou informações originadas a partir de uma consulta específica.

Um dos algoritmos globais mais simples consiste em considerar o somatório dos *backlinks* de uma página como medida de qualidade. Logo, quanto mais *backlinks* apontarem para a página maior será seu ranking. Uma desvantagem deste algoritmo é que ele não consegue distinguir entre uma página apontada por um grande número de páginas de baixa qualidade e uma página apontada pelo mesmo número de páginas de alta qualidade. É, portanto, muito simples inflacionar o peso de qualidade uma página neste algoritmo, basta criar um procedimento automático para construção de *links* para uma dada página.

Para solucionar este problema Brin e Page [PBM+98] desenvolveram a medida de qualidade global chamada PageRank, que será detalhada a seguir.

3.2.2.1 PageRank

O algoritmo PageRank [BMP+98, PBM+98, Hav99] associa uma medida de qualidade as páginas tomando como base no número de *links* que apontam para ela, porém levando em consideração que um *backlink* nem sempre confere o mesmo grau de importância. Esta medida de qualidade é chamada de valor PageRank. Ela é calculada por um algoritmo de mesmo nome, que implementa um conceito de propagação de pesos, no qual uma página possuirá um alto ranking se o somatório dos *rankings* de seus *backlinks* for alto. Esta idéia é capturada pela seguinte fórmula matemática:

$$R(p) = d \cdot \frac{1}{T} + (1-d) \sum_{i=1}^{k} \frac{R(p_i)}{C(p_i)} \qquad (10, [PBM+98])$$

Onde:

- R é um vetor que contem os pesos de ranking para todas as páginas do grafo *Web*.
- k é o número de *backlinks* da página p.
- p_i representa os *backlinks* de p.
- $C(p_i)$ são os *forwardlinks* da página p_i.
- d é um fator de normalização utilizado para manter constante o somatório dos valores de ranking de todas as páginas em 1. O valor de d é calculado empiricamente através de experimentos.
- T corresponde ao número total de páginas *Web* indexada.

Desta forma, uma página que possui poucos *backlinks* com alto PageRank pode possuir maior PageRank que uma página que possua vários *links* provenientes de páginas que não são muito "importantes" no cenário *Web*. O valor de PageRank(p) é computado iterativamente, e é inicializado com valor 1/T. Este cálculo pode ser visto como a computação do autovetor principal da matriz de adjacência M, onde cada entrada (i,j) desta matriz é igual a 1 se existe um *link* do documento i para o documento j, ou 0 caso contrário. A Figura 9 mostra um exemplo da propagação simplificada do peso de PageRank da página através do algoritmo PageRank.

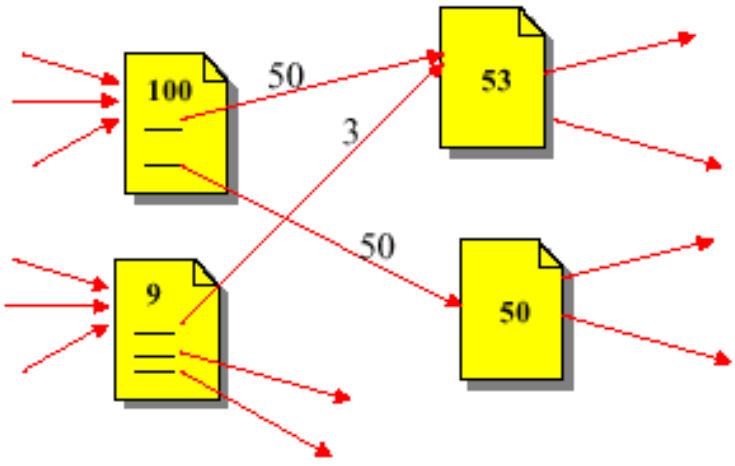

Figura 11: Propagação dos pesos PageRank.

O valor de inicialização não influenciará no valor final obtido na computação, mas influenciará no número de iterações necessárias para a convergência. Como este

algoritmo está apenas interessado no valor de PageRank relativo, um critério de convergência que pode ser empregado consiste em parar a computação quando a ordem relativa das primeiras n páginas não variar, onde n é empiricamente especificado.

O algoritmo PageRank descrito acima pode ser explicado através do modelo de comportamento de usuários da *Web*, conhecido como modelo do "*random surfer*". Este modelo representa o comportamento do usuário da *Web* através de duas ações: o usuário navega na *Web* clicando em sucessivos *links* de forma randômica, ou escolhendo páginas ao acaso dentre as páginas que compõem todo conjunto de páginas da *Web*. Segundo este modelo a fórmula (10) pode ser interpretada da seguinte forma: o usuário tem duas possibilidades de chegar na página p, uma através de um de seus *backlinks* com uma probabilidade (1-d), e a segunda através de uma escolha aleatória de uma página com uma probabilidade d.

Porém, este modelo não consegue representar o comportamento um usuário real, onde os *links* visitados não são escolhidos de forma aleatória. Um exemplo simples são as páginas de *bookmarks* ou páginas de jornais que são visitadas freqüentemente. Para modelar tal comportamento o PageRank permite a utilização do vetor E, chamado de vetor de personalização, que associa um peso extra às páginas *Web* visitadas freqüentemente. Esta nova variável é representada matematicamente na fórmula a seguir:

$$R(p) = d.E(p) * + (1-d) \sum_{i=1}^{k} \frac{R(p_i)}{C(p_i)} \qquad (11, [PBM+98])$$

Logo, quanto maior o valor de E(p) maior será o valor do PageRank de p. Neste caso, páginas que não contenham muitos *backlinks* podem, caso tenham um alto valor E(p), receber um maior PageRank que páginas muito referenciadas, e conseqüentemente influenciar no valor de PageRank das páginas por ela apontadas. O vetor de personalização pode conter características de um único usuário ou de um grupo de usuários da *Web*.

Como o PageRank se baseia na opinião coletiva embutida na estrutura de *links* da *Web*, sua eficácia depende largamente da cobertura do grafo sobre o qual este algoritmo será executado. O PageRank é utilizado por um dos maiores sistemas de busca da atualidade, o engenho de busca Google, que computa o PageRank para todo subgrafo da *Web* indexada.

Por ser um algoritmo de análise de *links* independente da consulta, o PageRank

não possui a fase de construção do grafo base que é parte inerente do algoritmo HITS e das suas extensões. Contudo, o PageRank não é aplicado sobre o grafo da *Web* de forma direta, são realizados alguns processamentos sobre este grafo, os quais garantem a convergência do algoritmo (a redução no tempo de processamento), e evitam que algumas páginas do grafo acumulem pesos de PageRank. Os processamentos realizados no grafo *Web* são enumerados a seguir:

- Processo de Limpeza do Grafo: Neste processo serão excluídas todas as páginas que não apontam para nenhuma outra. Este processo de limpeza é repetido diversas vezes, de forma a retirar iterativamente as páginas que possuem o número de *links* de saída igual a zero. Após o cálculo dos pesos de PageRank, estes são propagados para as páginas que foram excluídas pelo processo de limpeza através do mesmo número de iterações que foram necessárias para a retirada destas páginas.
- Adição de *Link*s de Saída Fictícios: Como forma alternativa ao processo de limpeza, as páginas com número de *links* de saída igual a zero podem passar a apontar artificialmente para todas as páginas do grafo *Web*.

O Apêndice F mostra, através de conceitos da algebra linear, como estes processamentos realizados no grafo *Web* iluenciam na convergência do algoritmo PageRank.

3.2.3 Comparação entre os Algoritmos de *Topic Distillation*

O PageRank corresponde a um algoritmo de análise de *links run-time,* pois, uma vez que é aplicado off-line para todo o grafo da *Web* indexada, não necessita de um tempo adicional para a análise de *links* no momento da consulta.

Esta vantagem, no entanto, pode acarretar alguns problemas, pois páginas não relevantes para uma dada consulta podem possuir maior valor de PageRank que páginas relevantes para a consulta. Tal problema é facilmente resolvido através de uma análise do conteúdo dos documentos posteriormente à análise de *links*. A análise de conteúdo tem como função associar um peso de similaridade, entre os termos da consulta e o documento, ao ranking das páginas. O Google utiliza o PageRank para análise de *links* e posteriormente realiza a análise do conteúdo das páginas resposta, para só então, atribuir

um peso final a cada documento.

Já o algoritmo HITS e seus derivados são aplicados on-line para cada consulta, e realizam a análise da estrutura de *links* do subconjunto da *Web* correspondente ao subgrafo resposta o qual contém apenas as páginas do conjunto resposta e as páginas de sua vizinhança – as páginas da vizinhança correspondem às páginas que apontam e são apontadas apenas por páginas do conjunto resposta, podendo também ser incluídos na vizinhança as páginas com mais deum nível de indireção com relação às páginas do conjunto resposta. O HITS e suas extensões não podem ser considerados algoritmos de análise de *links run-time,* uma vez que as implementações correntes levam alguns minutos para computar o ranking das páginas para subgrafos resposta contendo em média 10000 nós. Intervalo de tempo que está bem aquém do tempo de resposta requerido pelos engenhos de busca iterativos que é de poucos segundos.

Os algoritmos que estendem o HITS com análise do conteúdo realizam esta análise no momento da computação dos pesos de autoridade, através da inclusão de pesos de similaridade na própria fórmula para computação dos pesos de *hub* e autoridade (Fórmulas 6 e 7). Após a computação deste valor, as páginas da resposta são ranqueadas apenas pelo peso de *hub*/autoridade que já encapsula informações sobre o conteúdo da página.

Para o PageRank uma página é autoridade se ela é apontada por páginas que também são autoridade. Este cenário se faz presente em comunidades colaborativas, existentes em subconjuntos da *Web*.

Já o algoritmo HITS e seus derivados são apropriados para encontrar autoridades em comunidades competitivas, onde uma página autoridade em um dado tópico não aponta diretamente para as demais páginas autoridades neste tópico por questões de concorrência entre as páginas. Esta família de algoritmos implementa a seguinte relação: uma página é considerada boa autoridade se é apontada por páginas anônimas que apontam para outras páginas autoridade.

Os cenários competitivos estão cada vez mais presentes entre as comunidades da *Web*, podendo ser presenciado entre as páginas de engenhos de busca, entre páginas de montadoras de automóveis, entre lojas em geral, ou mesmo entre portais que querem manter o usuário nas páginas do seu site, para acumular *page-views* - hoje em dia, o número de *page-views* corresponde a uma das métricas para análise do sucesso e da popularidade de um dado site.

Da mesma forma que o IMPL o algoritmo ARC também estende o algoritmo

HITS com análise textual. O ARC analisa o grafo de vizinhança do conjunto resposta com distância igual a dois e atribui pesos aos arcos, baseado na medida de similaridade existente entre os termos da consulta e o texto presente nas redondezas do *link* no documento origem. Existem 3 diferenças entre o ARC e o IMPL, quais sejam: (i) o IMPL utiliza uma consulta expandida ao invés da consulta original; (ii) no IMPL a medida de relevância é calculada levando em consideração o documento inteiro e não apenas o texto nas proximidades do *link*; (iii) no IMPL o arco possui dois pesos associados, um que é utilizado no cálculo do peso de autoridade e o outro que é usado no cálculo dos pesos de *hub*.

3.3 Fusão de Informação

"Duas cabeças pensam melhor que uma" esta é a premissa básica da fusão de informação. As pesquisas realizadas na área de fusão de informação em RI têm investigado várias maneiras de se combinar diferentes estratégias de recuperação de informação, e têm mostrado que a fusão tem um efeito positivo na eficácia de recuperação independentemente de quais estratégias estão sendo combinadas. Alguns pesquisadores [FNL98] observaram que a inclusão de um componente "fraco" na função ainda pode resultar em um ganho de eficácia de recuperação, o que sugere que a fusão pode produzir um todo maior que a soma das partes.

A capacidade da fusão de amenizar as características negativas de uma estratégia, ao passo em que reforça as características positivas, oferece novas perspectivas acerca do descobrimento de melhores estratégias de busca na área de RI.

Descobrir a melhor maneira de se encontrar informações na *Web*, constitui-se numa tarefa difícil, senão impossível. Uma vez que ainda não foi descoberta uma fórmula única para recuperação de informações na *Web*, podemos nos beneficiar a partir de investigações sobre como combinar da melhor forma as técnicas existentes.

Existem inúmeros tipos de fusão, que são empregados de acordo com a situação de recuperação. Tomando como base a situação de recuperação na qual a estratégia de fusão será aplicada podemos classificar as estratégias segundo as seguintes perspectivas: fusão de fontes de informação de uma única coleção de dados (fusão de dados); fusão de múltiplas coleções de dados (fusão de coleções); e fusão de múltiplos paradigmas de recuperação (fusão de paradigmas).

As potenciais vantagens da fusão em recuperação de informação para a *Web* são bastante significativas. A baixa interseção existente entre as bases indexadas pelos engenhos de busca, faz da meta-busca a estratégia ideal para aumentar a cobertura da resposta. A fusão dos resultados de diferentes engenhos requer estratégias de fusão de coleções. Há inúmeros sistemas de busca para a *Web* que podem melhorar sua eficácia de recuperação a partir da fusão de informações provenientes da localização do termo no documento, da formatação do termo, do tamanho da fonte (fusão de dados).

O *corpus Web* é rico em múltiplas fontes de informação tais como *links*, conteúdos textuais, estatísticas de acesso, cada uma destas fontes podendo ser usada para complementar uma a outra. Este mesmo *corpus* engloba documentos de diversas qualidades, vocabulários e tamanhos, e possui usuários que divergem quanto a sua experiência, seus interesses e suas necessidades de informação. Outra característica dos usuários da *Web*, descrita na Seção 1.1, é que estes são tipicamente inexperientes e impacientes, submetem consultas curtas e imprecisas e apenas avaliam as URLs entre os 10 e 20 primeiros resultados da busca.

Tais características tornam a recuperação de informações na *Web* uma tarefa difícil. A fusão de paradigmas pode ocorrer na combinação da busca baseada no conteúdo textual com a busca baseada na estrutura de *links*, com o objetivo de melhorar a eficácia de recuperação dos sistemas de busca para a *Web*.

Ao longo da próxima seção vamos nos ater as estratégias empregadas na fusão de paradigmas por fazerem parte do objetivo deste trabalho.

3.3.1 Fusão de Paradigmas

Nas seções anteriores (4.3.1, 4.3.2), já foram vistas várias instâncias de fusão de paradigmas, algumas propondo a propagação de informações textuais através da estrutura de *hiperlinks* [CDR+98], outras incorporando o texto existente nas proximidades dos *links* a computação dos pesos de *hub* e autoridade [BH98], ou até mesmo incorporando uma estratégia puramente textual com um conjunto de estratégias de recuperação de informação tradicional [PBM+98, Hav99].

Contudo, nenhuma das fusões de paradigmas, anteriormente mencionadas, define formalmente um método de fusão, ao invés disto, estas fusões aplicam os paradigmas individualmente, em seqüência, ou em paralelo integrando-os de maneira *ad-hoc*.

Na literatura, alguns estudos objetivam impor um certo formalismo na combinação de dos paradigmas textual e estrutural. O primeiro formalismo para fusão de informações foi descrito no trabalho de Fox [FNL88]. Fox estende o modelo vetorial, o qual passa a incorporar, além das características relativas ao conteúdo do documento, características extrínsecas, tais como o nome do autor e informações sobre a estrutura de *links* que circunda o documento. Neste caso a medida de similaridade entre um documento e uma consulta passa a ser calculado a partir da combinação linear de métricas de acordo com a equação que segue:

$$SIM(D,Q) = a*(\text{content-similarity}) + b*(\text{external-attributes-similarity}) + c*(\text{citation-similarity}) \quad (12, [FNL88])$$

Onde:

- D é o documento para o qual se está calculando a similaridade;
- Q é a consulta realizada;
- a,b,c são pesos ótimos calculados a partir de métodos de regressão linear [FNL98].

Xiloan et al [ZG00] propõe a combinação de um conjunto de métricas qualitativas calculadas separadamente com o valor de similaridade retornado pelo algoritmo de busca baseado no modelo vetorial para a obtenção de um ranking final para cada documento *Web*, obtido da seguinte forma:

$$sd = sr*(ap*td + bp*ad + cp*id + dp*rd + ep*pd + fp*cd) \quad (13, [ZG00])$$

Onde:

- sd é o ranking final do documento d;
- sr é o valor de similaridade retornado pelo algoritmo de busca baseado no modelo vetorial;
- ap, bp, cp, dp, ep e fp são medidas de qualidade obtidas para cada documento, incluindo atualidade, disponibilidade, autoridade, popularidade, coesão, resistência ao ruído. Estas medidas estão normalizadas, variando entre zero e um e estão descritas em [ZG00];
- td, ad, id, rd, pd e cd são pesos que representam a importância da métrica

correspondente no momento da busca.

Estes pesos são calculados a partir da razão entre a melhoria na eficácia de recuperação obtida pela incorporação desta métrica sozinha, e a soma das melhorias obtidas pela incorporação de todas as métricas qualitativas.

Um terceiro princípio, descrito por Yang [Yan01], propõe combinar os resultados das buscas em três sistemas baseados em paradigmas diferentes. O primeiro baseado no conteúdo do documento, o segundo baseado no peso de autoridade do documento calculado a partir da aplicação do HITS ao grafo resposta, e o terceiro relacionado à estrutura semântica existente na árvore de diretórios do engenho de busca Yahoo. Yang propõe que estes três paradigmas sejam combinados da seguinte forma:

Rank=(SimTextual+AUTORIDADE+SimYahoo) * num (14,[Yan01])

Onde:

- Rank é o valor do ranking final do documento.
- num é o número de estratégias que retornou o documento em questão.

3.4 Considerações Finais

O objetivo principal deste capítulo foi descrever os principais algoritmos de análise de *links*, mais especificamente algoritmos de *topic distillation,* que objetivam filtrar as páginas "autoridade" existentes no conjunto resposta.

Também foram mostradas, neste capítulo, as técnicas de Fusão de Informação que serão de grande utilidade nos capítulos posteriores, os quais mostrarão de que forma o conceito de "importância" obtido através da análise de *links* é incorporado à estratégia de recuperação baseada na similaridade textual, de maneira a melhorar a eficácia de recuperação do sistema.

4

Especificação do SAAL

Este capítulo apresenta o algoritmo de análise de *links* proposto neste trabalho, como também a especificação do sistema de armazenamento e análise de *links* (SAAL) elaborado para validar este algoritmo. O SAAL associa um peso de "importância" às páginas *Web*, através da análise da estrutura de *links*, para, posteriormente, utilizá-lo no processo de ranqueamento da resposta de um sistema de recuperação de informação para a *Web*.

A seguir, são apresentadas uma descrição mais detalhada do problema enfrentado pelos engenhos de busca, e a solução adotada, neste trabalho, para atacar este problema.

4.1 Apresentação do Problema

Como foi visto na Seção 1.1, quando consultas por tópicos gerais (ou tópicos populares) são submetidas aos engenhos de busca, estes enfrentam um desafio de precisão ao tentar encontrar, entre as milhares de URLs retornadas, as URLs mais "importantes", para serem apresentadas nos primeiros lugares da lista resposta.

Segundo estudos sobre o comportamento dos usuários da *Web* mostrados na Seção 1.1, em situações onde são retornadas "milhares" de URLs como resposta, o usuário apenas analisa as 10 ou 20 primeiras.

A Figura 12 ilustra a situação descrita acima, nela é mostrada uma tela que

contém o resultado retornado pelo ***Radix*** para a consulta "futebol".

Figura 12: Lista de documentos retornados para a consulta futebol.

O ***Radix*** - engenho de busca que indexa as páginas em português, baseado unicamente no paradigma textual - foi utilizado como estudo de caso, no decorrer deste trabalho.

Segundo pesquisas, cerca de 70% das consultas submetidas aos engenhos de busca contêm somente um termo [LG99a, LG99b]. Para verificar se este comportamento se faz presente no engenho de busca utilizado no estudo de caso, foi realizada uma análise de todas as consultas submetidas ao engenho de busca ***Radix*** entre os meses de setembro, outubro e novembro de 2001. A Figura 13, contém o número de termos por consulta da amostra analisada. Do total de consultas da amostra, 13% possuíam 1 só termo e 51% possuíam 2 termos. Somados, estes dois tipos de consulta correspondem a mais de 50% das consultas submetidas a este engenho de busca.

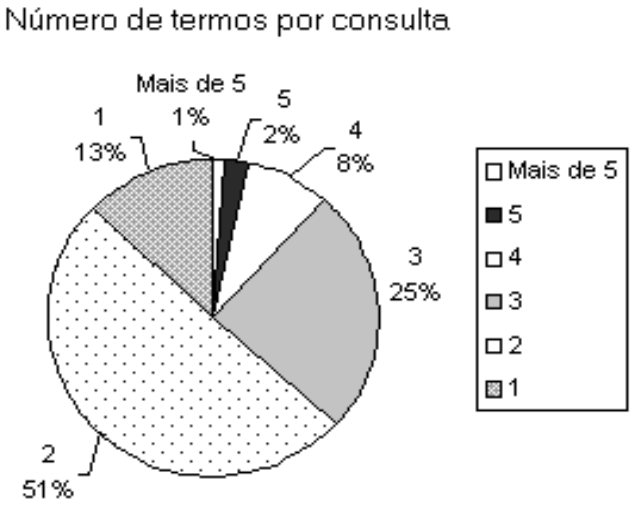

Figura 13: Número de termos por consulta.

Tendo visto que a maioria dos usuários expressa sua necessidade de informação através de consultas com poucos termos, pode-se concluir que o desafio de precisão é um problema, freqüentemente, enfrentado pelos engenhos de busca.

Defini-se consultas por tópicos gerais como sendo as consultas que contém de 1 a 2 termos, e que por esta razão retornam muitas páginas como resposta. Algumas consultas fogem a esta definição simplificada, como por exemplo, uma consulta por "Mar de Bellingshausen" – mar que banha a parte ocidental da Antártica, que apesar de ser uma consulta formada por poucos termos, quando submetida a um engenho de busca não retorna muitas páginas no conjunto resposta.

Ao se analisar o log de acesso do engenho de busca ***Radix***, percebeu-se que a maioria das consultas submetidas se repetiam e correspondiam a consultas simples com poucos termos e que retornavam muitas páginas como resposta - consultas por tópicos gerais.

A seguir será mostrada a solução, proposta neste trabalho, para enfrentar o desafio de precisão, constantemente presente em consultas por tópicos gerais.

4.2 Solução Proposta

A solução proposta neste trabalho, para o problema descrito acima, foi: (i) analisar a estrutura de *links* da *Web*, para, a partir desta análise, produzir um peso de "importância" global para cada página *Web* indexada; (ii) e alterar componente de

Ranking do engenho de busca, para que este passe a raquear as páginas utilizando, além de critérios textuais, este critério de "importância". Esta alteração tem por objetivo fazer com que o engenho de busca retorne em primeiro lugar as páginas que além de relevantes, são consideradas importantes para tópico consultado.

Com o propósito de analisar a estrutura de *links* da *Web*, foi elaborado o GHHITS, um algoritmo de análise de *links* que associa um peso de "importância" às páginas *Web*, chamado de "peso de autoridade". Para validar este algoritmo, foi elaborado um sistema para armazenamento e análise dos *links* da *Web* (SAAL). Este sistema aplica o algoritmo de análise de *links* GHHTIS ao grafo da *Web* indexada e disponibiliza o peso de autoridade para ser utilizado pelo engenho de busca no momento do ranqueamento das páginas.

4.3 Algoritmo Proposto: *Global Hybrid* HITS (GHHITS)

O GHHITS consiste em um algoritmo de análise de *links run-time*, que incorpora algumas das técnicas do algoritmo HITS e de suas extensões, detalhados na Seção 3.2.1.

O algoritmo GHHITS, assim como o HITS, trabalha com três estruturas básicas:

- Vetor autoridade: um vetor que armazena os pesos de autoridade, possuindo uma entrada para cada página do subconjunto da *Web* que participa da computação;
- Vetor *hub*: um vetor que armazena os pesos de *hub*, que também possui uma entrada para cada página.
- Grafo *Web*: estrutura que armazena todos os *links* pertencentes a um subgrafo da *Web*.

Porém, contrariamente ao HITS, o qual se aplica apenas ao subgrafo resposta, necessitando ser computado a cada consulta submetida ao engenho de busca, o GHHITS se aplica a todo o grafo da *Web* indexada. Desta forma, o GHHITS associa pesos gerais de autoridade e *hub* para cada página do grafo *Web* o que o torna mais robusto às técnicas de *spamming* baseadas em link, detalhadas no Apêndice C.

No momento da consulta, como os valores de autoridades foram pré-

computados, não é requerido nenhum tempo adicional para a análise de *links*. Por esta razão o GHHITS é considerado um algoritmo *run-time* de análise de *links,* atendendo, assim, aos requisitos de tempo dos engenhos de busca comerciais que é de poucos segundos.

Este algoritmo utiliza os vetores de personalização InitAut e InitHub responsáveis por representar a opinião de usuários, sobre o nível de importância que deve ser conferido a uma página ou a um conjunto de páginas durante o cálculo dos pesos de *hub* e autoridade.

O algoritmo GHHITS é composto pelos passos descritos a seguir:

Passo 1 - Construção do Grafo Base: o grafo base é construído a partir do subgrafo da *Web* indexada.

Passo 2 - Cálculo dos Pesos de Hub e Autoridade: (i) associa-se a cada nó (página) do grafo base um peso de *hub* e um peso de autoridade, que são inicializados com o valor 1; (ii) em seguida os pesos de *hub* (h) e autoridade (a) são calculados de acordo com a relação de reforço mútuo, através de sucessivas iterações das equações mostradas a seguir.

$$a[n] = \text{InitAut}[n] + \sum_{(n',n) \in N} h[n'] \times \text{impl\_aut}(n',n) \quad (15)$$

$$h[n] = \text{InitHub}[n] + \sum_{(n,n') \in N} a[n'] \times \text{impl\_hub}(n,n') \quad (16)$$

Onde:

- a é um vetor que armazena os pesos de autoridade e contém uma entrada para cada página do grafo base.
- h é um vetor que armazena os pesos de *hub* e contém uma entrada para cada página do grafo base.
- N conjunto de todos os nós pertencentes ao grafo da *Web* indexada.
- n e n' são nós do grafo.
- impl_aut e impl_*hub* são pesos conferidos aos arcos que objetivam minimizar a relação de mútuo entre os *hosts* adotado pelo algoritmo IMPL descrito na Seção 3.2.1.2.

- InitAut e InitHub vetores de personalização.

Esta computação termina quando o número de pesos de autoridade diferentes de zero permanece constante ao longo de 10 iterações. Este critério de parada foi obtido empiricamente através de experimentos descritos na Seção 5.5. O critério de parada utilizado pelo GHHITS difere do critério utilizado pelo HITS e suas extensões. O HITS utiliza o critério de convergência (a diferença entre os pesos de *hub* e autoridade entre sucessivas iterações).

O número de pesos de autoridade diferentes de zero permanece constante ao longo de 10 iterações e o número de páginas que permanecem na mesma ordem ao longo de 10 iterações é constante.

4.3.1 Características do algoritmo

Da mesma forma que o algoritmo HITS, o GHHITS está sujeito a fatores que podem causar o efeito TKC, são eles:

Comercialização

As propagandas realizadas na *Web* influenciam de forma considerável a estrutura de *links* da *Web* como um todo. Este problema é mais grave no HITS uma vez que é mais fácil influenciar a estrutura de *links* localmente (no subgrafo resposta) do que em todo o grafo da *Web* indexada. Este problema é solucionado no Passo 1 do algoritmo, a partir da utilização de heurísticas de eliminação de *links* comerciais descritas a seguir.

Fatores temporais

Como dito anteriormente, os fatores temporais correspondem a páginas e *links* que influenciam a estrutura de *links* da *Web* de forma passageira. O impacto causado por estas influências se reduz à medida que estas páginas deixam de existir ou estes *links* são removidos. Porém, estas influências podem se manter "artificialmente" em bases indexadas por engenhos de busca que ainda não sofreram atualização. Estas influências não mais representam características da *Web*. Percebe-se, portanto, que a política de atualização da base de índices é de muita importância para que os algoritmos de análise de *links* reflitam a opinião atual dos usuários da *Web*.

O GHHITS está sujeito a um fenômeno semelhante ao subtópico "*Web-Centric*", descrito na Seção 3.2.1, o qual podemos chamar de comunidades "*Web-Centric*". Este fenômeno ocorre comunidades de páginas da *Web* fortemente conectadas passam a dominar os pesos de *hub* ou autoridade. Neste caso porém não se presencia a especialização ou a generalização da resposta uma vez que os pesos de *hub* e autoridade são calculados sobre todo o grafo da *Web* indexada.

4.3.2 Heurísticas de Limpeza

O GHHITS aplica duas heurísticas sobre o grafo da *Web* indexada, com o objetivo de eliminar os fatores que podem causar o efeito TKC. Estas heurísticas, chamadas heurísticas de limpeza, atuam extraindo do grafo *Web* os *links* não funcionais.

As heurísticas de limpeza podem ser divididas em duas famílias, heurísticas de pré-processamento e heurísticas de pós-processamento, de acordo com o momento em que são aplicadas à estrutura de *links* da *Web*.

As heurísticas de pós-processamento têm a vantagem de armazenar o grafo de *links* da *Web* de forma integral. No entanto, necessitam ser aplicadas antes de cada execução do algoritmo de análise de *links*. Já as heurísticas de pré-processamento extraem os *links* não funcionais no momento do armazenamento da estrutura de *links* da *Web*, reduzindo o espaço ocupado por esta estrutura.

A seguir, são detalhadas as heurísticas aplicadas ao grafo *Web*: a heurística de pré-processamento que elimina os *links* intrínsecos do grafo *Web* e a heurística pós-processamento que elimina os *links* comerciais.

Heurística de Eliminação de Links Intrínsecos

O GHHITS utiliza a estratégia de eliminação de *links* intrínsecos empregada pelo SFRD [CDG+98a], com algumas adaptações. Para evitar a autopromoção, isto é, páginas que conferem peso de autoridade para páginas do mesmo *site*, esta heurística descarta os *links* intrínsecos durante a construção do grafo *Web*. De acordo com esta heurística, duas páginas são consideradas do mesmo *site* quando:

1. O hosts são similares: quando o domínio contido na URL é o mesmo.
2. Os IPs são correspondentes:
   - Quando as duas páginas pertencem às classes IP A ou B e os dois octetos

mais significativos coincidem.

- Quando as duas páginas pertencem à classe IP C e os 3 octetos mais significativos coincidem.

- Quando as duas páginas pertencem à classe IP D e todos os octetos coincidem.

Durante o processo de limpeza do grafo são aplicadas as heurísticas enumeradas acima na seguinte ordem: se a página A possui *host* similar ao *host* da página B, B é descartado, caso contrário, os IPs das páginas são verificados e caso sejam considerados similares, de acordo com os critérios acima, B é descartado. Tal procedimento evita que durante a limpeza sejam analisados todos os IPs das páginas contidas no grafo.

Heurística de Eliminação de *Links* Comerciais

Esta heurística consiste em eliminar do grafo as páginas que possuem número de *backlinks* superior a *n*, onde *n* é calculado empiricamente. Foram realizados experimentos em uma base de 5 milhões de páginas indexadas. Estes experimentos mostraram que 100% das páginas que possuíam mais de 1000 *backlinks* eram provenientes de *banners*.

4.4 Arquitetura do Sistema para Armazenamento e Análise e de *Links* (SAAL)

Como visto na Seção 2.4.3, muitos engenhos de buscam utilizam a arquitetura *Crawler*-I*ndexer* centralizada, dentre eles pode ser citado ***Radix***, utilizado como estudo de caso neste trabalho.

O SAAL estende a arquitetura *Crawler*-I*ndexer* centralizada a partir da inclusão de um módulo para armazenamento e análise de *links.* A Figura 14 ilustra: (i) a arquitetura simplificada de um engenho de busca que não armazena nem analisa as informações estruturais da *Web*; (ii) e a arquitetura simplificada de um engenho de busca que utiliza o SAAL para desempenhar estas respectivas tarefas.

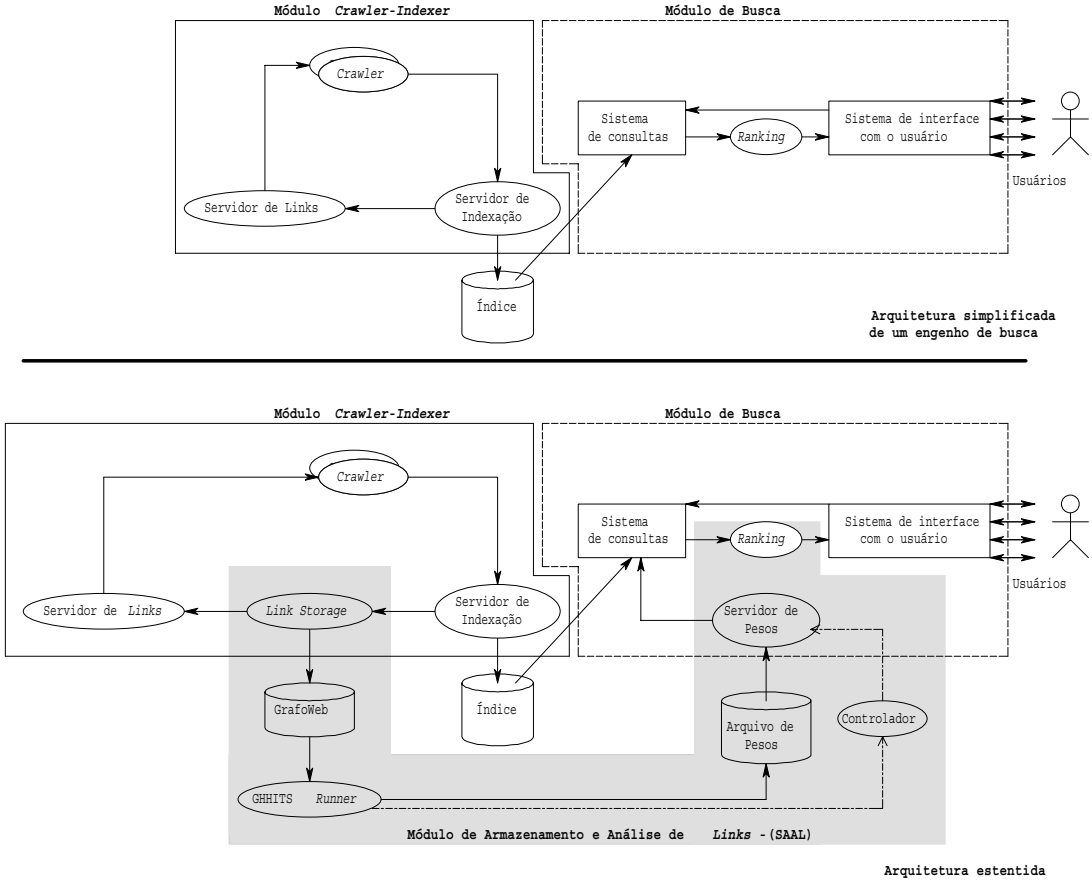

Figura 14: Arquitetura simplificada de um engenho de busca que utiliza do SAAL.

Uma visão mais detalhada sobre cada componente pertencente ao módulo de análise e armazenamento de *links* é apresentada nas seções que seguem.

4.4.1 Link Storage

O *Link Storage* é responsável por receber e armazenar informações sobre a estrutura de *links* existente entre as páginas *Web* indexadas – informações que são freqüentemente desprezadas pelos engenhos de busca em geral.

Este componente recebe, do Servidor de Indexação, uma lista de *links* no seguinte formato:

<url _origem>|<link_destino_1>|<link_destino_2>|<link_destino_3>| ... |<link_destino_n>

Onde:

- url _origem é a URL da página indexada
- *link*_destino_1..n correspondem aos *links* encontrados no corpo da página

indexada.

O *Link Storage* também é responsável por aplicar as heurísticas de eliminação de *links* intrínsecos do grafo enumeradas na Seção 4.3, às listas de *links* que são recebidas, armazenando no Grafo *Web* apenas os *links_destino* que atendem aos requisitos impostos pelas heurísticas de limpeza.

4.4.2 Grafo *Web*

Representar um pequeno grafo através de uma estrutura de dados é uma tarefa trivial. Porém, representar um grafo contendo milhões de nós e milhões de arestas, de modo a armazenar, recuperar e atualizar os dados relativos ao grafo de maneira eficiente consiste em um desafio de engenharia.

Várias pesquisas e produtos comerciais, se utilizam das informações contidas na estrutura de *links* da *Web*, de maneira *ad-hoc*, e na maioria das vezes ineficiente [BBH+98]. Uma forma eficiente de armazenar e recuperar o grafo da *Web* foi elaborada por Baraht et al [BBH+98], que construíram o servidor chamado *Connectivity Server*, que fornece as informações sobre a estrutura de *links* existente entre as páginas indexadas pelo engenho de busca Altavista. O *Conectivity Server* armazena o grafo da *Web*, em tabelas de um banco de dados.

O Grafo *Web* corresponde ao componente do SAAL que contém a estrutura de *links* da *Web* indexada. Esta estrutura é armazenada em estruturas do tipo VIF descritas na Seção 2.2.2. O VIF, assim como as demais estruturas de índice descritas na Seção 2.2, foi concebido para suportar grandes quantidades de dados, e recuperá-los de forma eficiente.

Esta estrutura foi escolhida entre as demais estruturas de armazenamento, pois estudos mostraram que os arquivos VIF possuem tempo de acesso e atualização menores que os respectivos tempos despendidos em arquivos invertidos comuns e em bancos de dados[6] [Mir02].

[6] Inicialmente o Grafo Web foi armazenado em arquivos invertidos comuns, mas como os experimentos demonstraram que os tempos de acesso e atualização do VIF eram menores, o Grafo Web foi alterado passando a ser armazenado em estruturas do tipo VIF. Como o engenho de busca ***Radix*** utilizado como estudo de caso deste trabalho utiliza estruturas do tipo VIF para indexar as informações pertencentes às páginas Web, os procedimentos de inserção, atualização e remoção de elementos, já haviam sido implementados.

Ao final do processo de indexação, o Grafo *Web* contém as páginas que foram indexadas, e a partir deste momento dá-se início ao processo de análise dos *links* realizado pelo GHHITS *Runner*.

4.4.3 GHHITS Runner

Este componente implementa o algoritmo GHHITS detalhado na Seção 4.3, sendo responsável por executá-lo sobre um subconjunto da *Web* indexada (Grafo *Web*). Também serão executadas, por este componente, as heurísticas de limpeza de pós processamento descritas na Seção 4.3

Após as heurísticas de limpeza terem sido aplicadas no subgrafo da *Web* indexada, o algoritmo iterativo de análise de *links* é então executado sobre este subgrafo, gerando como resposta o Arquivo de Pesos.

4.4.4 Arquivo de Pesos

O Arquivo de Pesos[7] gerado pelo algoritmo de análise de *links*, corresponde a um arquivo binário que contém o peso de autoridade e o peso de *hub* para cada página participante da computação do GHHITS.

4.4.5 Servidor de Pesos

O servidor de pesos consiste em um servidor remoto que atende às requisições do engenho de busca. Estas requisições estão na forma de um *array* (*array_request*) contendo identificadores de documentos para os quais estão sendo solicitados os respectivos pesos de *hub* e autoridade. A resposta a esta requisição consiste em um *array* bidimensional (*array_response*) de mesmo tamanho do *array request*. A primeira dimensão do *array response* varia entre zero e o maior índice do *array request,* e a segunda dimensão varia entre zero e um.

A Figura 15 indica o significado dos valores contidos no *array request* e no *array response.*

[7] Apesar dos pesos de *hub* não terem sido utilizados neste trabalho, eles foram incluídos no Arquivo de Pesos para viabilizar pesquisas futuras sobre a utilização destes pesos no ranqueamento das páginas.

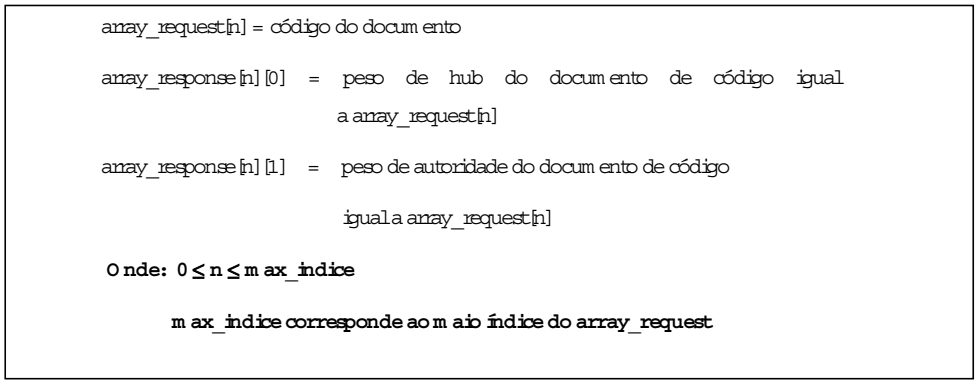

Figura 15: Semântica do *array_request* e do *array_response.*

4.4.6 Controlador

Este componente é responsável por receber informações de controle e repassá-las ao Servidor de Pesos. Estas informações de controle compreendem as informações de *start*, *stop* e *reload* do Servidor de Pesos. A diretiva *reload* é utilizada para alertar ao Servidor de pesos que um novo Arquivo de Pesos foi gerado e por esta razão este sistema deve recarregar os pesos de *hub* e autoridade que estão armazenados na memória principal.

4.4.7 Ranking

Para que os pesos de autoridade passassem a ser usados pelo engenho de busca no momento do ranqueamento da resposta, foram necessárias alterações no componente *Ranking* deste sistema.

A fórmula de ranqueamento presente no componente *Ranking* foi alterada, passando a combinar os paradigmas textual e estrutural. Estes paradigmas estão representados na fórmula de ranqueamento através das medidas de similaridade textual e importância, descritas a seguir:

(1) Medida de similaridade: obtida através do algoritmo de busca baseado no paradigma textual. Este algoritmo procura associar um peso de similaridade entre um documento e uma consulta. No engenho de busca ***Radix***, utilizado como estudo de caso, este algoritmo utiliza o modelo booleano estendido acrescido de heurísticas de proximidade entre os termos, para associar um peso de similaridade entre os documentos da base e a consulta do usuário. A representação dos documentos é feita a partir de todos os termos do

conteúdo do título e da URL do documento, excluindo-se as *stop-words.* Como dito na seção 2.3 a medida de similaridade associa uma probabilidade de relevância aos documentos, por esta razão, muitas vezes, esta medida é chamada de medida de relevância.

(2) Medida de importância: obtida através do algoritmo de busca baseado no paradigma estrutural. Esta medida corresponde ao peso de autoridade calculado através do algoritmo GHHITS.

A fórmula de ranqueamento passou a ser representada através da seguinte fórmula de fusão de informação:

Ranking = (a * Norm_SimTextual + b*Norm_AUTORIDADE) (22)

Onde:

- Ranking corresponde ao valor final atribuído ao documento.
- Norm_SimTextual é o valor de similaridade originado pelo algoritmo de busca baseado no paradigma textual, normalizado pelo maior valor de similaridade que pode ser retornado por este algoritmo.
- Norm_AUTORIDADE: corresponde ao peso de autoridade originado pelo algoritmo GHHITS, baseado no paradigma estrutural, normalizado pelo maior valor de autoridade que pode ser retornado por este algoritmo.
- a e b correspondem a constantes cujos valores foram calculados empiricamente, através de experimentos realizados no Capítulo 6.

4.5 Diagramas de Sequência do Sistema

Nesta seção, são apresentados os diagramas de sequência UML [BRJ99] que melhor especificam e representam a dinâmica do sistema em questão.

A Figura 16 apresenta o diagrama de sequência UML que descreve o processo de indexação, coleta e análise da estrutura de *links*. No processo de indexação, os *crawlers* são entidades responsáveis por: coletar páginas *Web;* realizar o *parsing* destas páginas extraindo os termos e os *links* de cada página, enviando-os para o Servidor de Indexação. A tarefa do Servidor de Indexação é armazenar os termos das páginas nas bases de índices e repassar os *links* coletados para o *Link Storage* que irá armazená-los em estruturas persistentes (Grafo *Web*). Após este processo ser finalizado, o GHHITS

Runner inicia a análise dos *links* do Grafo *Web,* que corresponde a um processamento iterativo, que executa um número predefinido de vezes. Ao fim da análise de *links*, o Arquivo de Pesos é gerado. Este arquivo contém os pesos de *hub* e autoridade das páginas que participaram da computação do GHHITS.

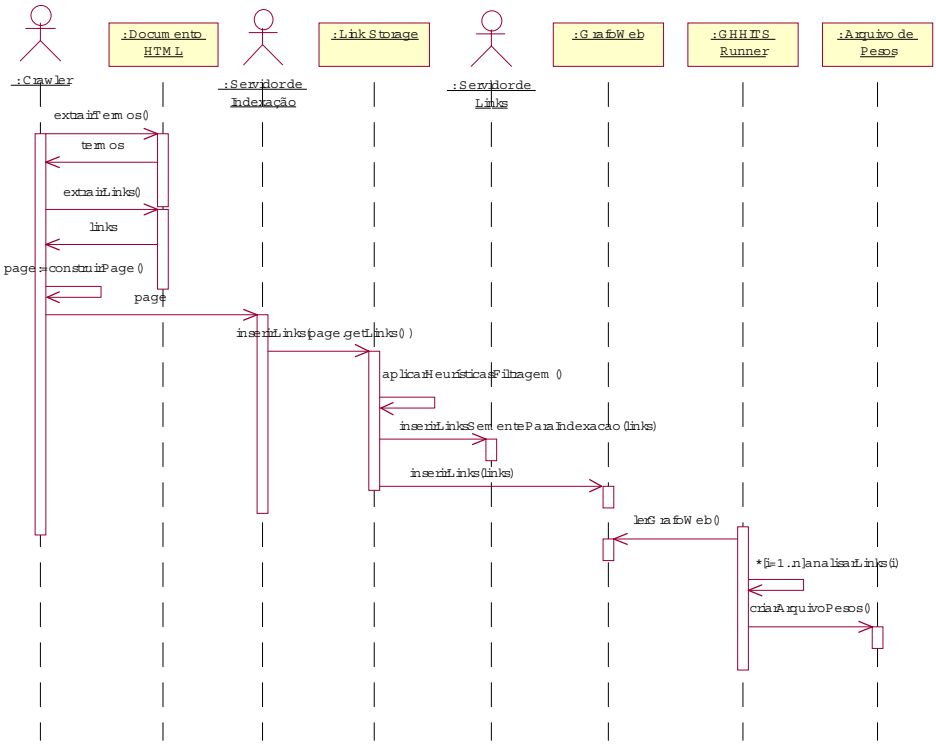

Figura 16: Diagrama de seqüência ilustrando os passos que compõem o processo de coleta de links.

A Figura 17, apresenta, através de um diagrama de sequência UML, as interações que ocorrem durante a busca. Quando uma consulta chega ao módulo de busca do ***Radix*** ("Sistema de Consultas"), este inicializa a busca por documentos que atendam à consulta em sua base de índices ("Índice"). À medida em que este sistema lê os códigos de página existentes no Índice, são enviadas requisições ao Servidor de Pesos. Os dados da requisição são enviados na forma de um *array* contendo os códigos de páginas lidos. O Servidor de Pesos é responsável por analisar os códigos de página, e retornar os respectivos pesos de *hub* e autoridade. A resposta à requisição consiste no *array* bidimensional explicado na Figura 15. O componente de *Ranking* do engenho de busca é responsável por utilizar os pesos de autoridade em conjunto com a medida de similaridade textual do documento para a obtenção de um novo peso que é utilizado como critério de ordenação no momento de apresentar as páginas do resultado da

consulta submetida.

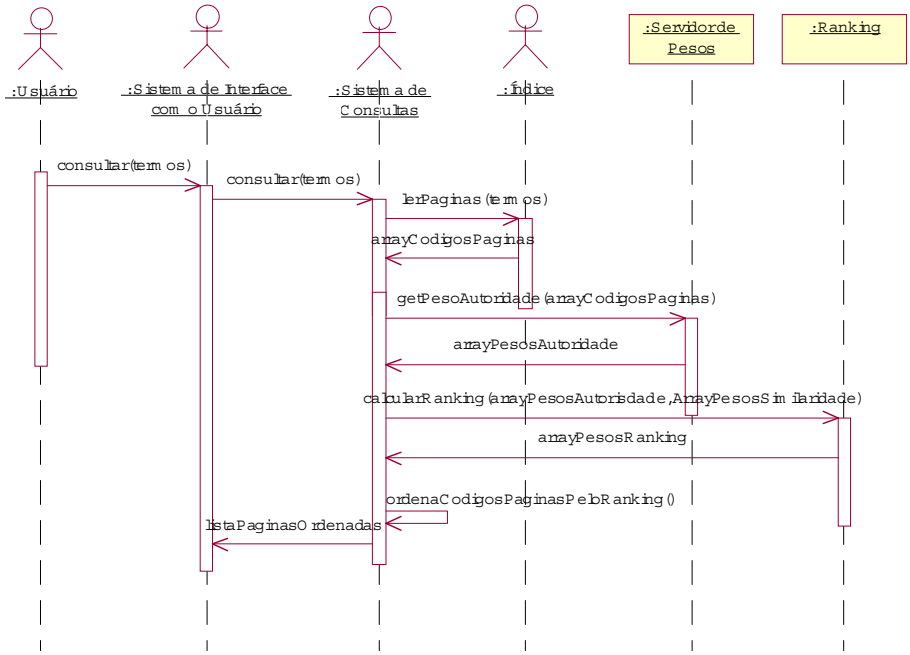

Figura 17: Diagrama de seqüência ilustrando as ações que são desempenhadas no momento da busca após a adição do módulo de análise de links ao sistema.

Na Figura 18 é apresentado o diagrama de seqüência para o processo de atualização dos pesos de *hub* e autoridade executado periodicamente. O processo de atualização ocorre sempre que um novo Arquivo de Pesos é gerado. Neste momento o servidor de pesos recebe uma requisição de *reload,* e passa a carregar os novos pesos de *hub* e autoridade em memória.

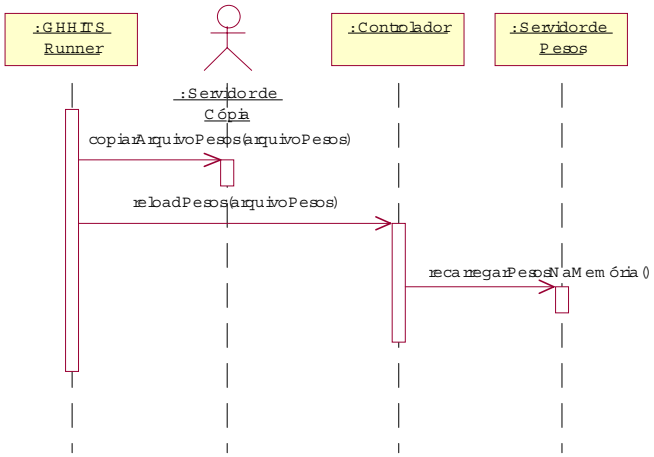

Figura 18: Diagrama de sequencia ilustrando o *reload* dos pesos de *hub* e autoridade.

4.6 Considerações Finais

Neste capítulo foi visto que os usuários de engenhos de busca encontram grande dificuldade na formulação de consultas, formulando consultas simples contendo de 1 a 2 termos, que levam a respostas com baixa precisão.

A partir da análise do desafio de precisão enfrentado principalmente em consultas por tópicos gerais, foi proposto um algoritmo de análise de *links* com o objetivo de associar um peso de "importância" às páginas *Web* para ser utilizado juntamente com o paradigma textual no momento do ranqueamento.

O objetivo desta fusão de paradigmas é retornar entre as primeiras páginas da lista de respostas as páginas que além de relevantes à consulta submetida são páginas consideradas "importantes" para o tópico consultado, melhorando, desta forma, a qualidade da resposta fornecida pelo engenho de busca.

A solução proposta consistiu: (i) na elaboração de um algoritmo para análise da estrutura de *links* (GHHITS), que mescla características dos principais algoritmos de *topic distillation* existentes na literatura; (ii) e na construção do Sistema para Armazenamento e Análise de *Link*s (SAAL) com a finalidade de validar o algoritmo proposto.

Os aspectos relativos à implementação deste sistema são apresentados no capítulo seguinte.

5

Projeto e Implementação do SAAL

Após terem sido mostrados, nos capítulos anteriores, a motivação geral do trabalho, os conceitos que dão suporte a este e a especificação do Sistema para Armazenamento e Análise de *Links* (SAAL), neste capítulo será apresentada uma descrição detalhada da implementação deste sistema.

Um dos objetivos que se desejou atingir na implementação do SAAL foi a construção de um sistema que pudesse ser facilmente acoplado a qualquer engenho de busca. Durante a implementação deste sistema foi utilizada a linguagem de programação *Java,* orientada a objetos e independente de plataforma. Este sistema foi projetado em módulos com forte coesão e baixo acoplamento com relação ao sistema de busca ao qual o SAAL foi adicionado.

Outro objetivo que demandou bastante esforço para ser alcançado foi a elaboração de uma estratégia de computação que não requisitasse recursos computacionais que crescessem linearmente ao número de páginas indexadas pelo sistema. Para tanto foi utilizada a estratégia de computação baseada em blocos, detalhada neste capítulo.

Nas próximas seções, serão apresentados os detalhes relativos à implementação do SAAL, assim como algumas características, e decisões tomadas durante o desenvolvimento.

5.1 Linguagem de Programação

Para implementação do SAAL, foi escolhida a linguagem de programação *Java*, como já citado anteriormente. Essa foi a linguagem escolhida por ser voltada para aplicações em rede, além de oferecer várias vantagens: independência de plataforma, extensibilidade, tratamento de exceções e robustez.

A comunicação entre os módulos do SAAL e o sistema de busca é feita via *Sockets*. *Socket* é uma objeto *Java* que permite a comunicação entre duas máquinas através do protocolo TCP/IP [Hun98]. Maiores informações sobre a tecnologia utilizada podem ser encontradas no *site* da ***Sun***[8].

5.2 Ambiente e Plataforma de Desenvolvimento

O ambiente de desenvolvimento adotado para a implementar o conjunto de programas que compõem o SAAL foi o JDK (*Java Development Kit*) 1.3 da IBM. As conexões com o banco de dados são realizadas através de JDBC (*Java Database Connectivity*) – uma API para acesso a bancos de dados, que permite que uma aplicação *Java* estabeleça conexões com um SGBD, envando comandos SQL e recebendo os resultados destes comandos.

A seguir é apresentada a estrutura, utilizada pelo SAAL, para a manutenção da estrutura de *links* da *Web*.

5.3 Estrutura de Armazenamento

A implementação do GHHITS é bastante simples, se a magnitude da *Web* é momentaneamente esquecida. Porém sabe-se, a priori, que a *Web* comporta um grande número de documentos, e que por esta razão, os algoritmos que são executados sob este conjunto requerem um maior cuidado na utilização de estruturas de dados para armazenar o espelho da *Web*.

Neste trabalho, a estrutura de *links* da *Web* é representada na forma de um grafo direcionado, onde um nó representa uma URL e uma aresta representa um *link* entre

[8] http://java.sun.com/

duas URLs.

O armazenamento do grafo *Web* é feito através de duas estruturas: (i) uma que realiza o mapeamento entre a URL e seus *links* de saída (*forwardlinks*); (ii) e outra estrutura que realiza o mapeamento entre uma URL e o conjunto de URLs que apontam para ela (*backlinks*).

Nesta representação do grafo *Web*, um arco direcionado partindo de um nó A e chegando a um nó B, aparece duas vezes: uma vez na estrutura que contém os *forwardlinks* do nó A, e outra vez na estrutura que armazena os *backlinks* do nó B. Esta redundância na representação dos arcos facilita o processo de busca por *forwardlinks* e *backlinks* de uma dada página.

A Figura 19 abaixo mostra a representação através de arquivos VIF do seguinte subconjunto da Web: a página da UOL (www.uol.com.br), possui links apontando para as páginas da VocêSA(www.uol.com.br/vocesa.html), SuperInteressante (www.uol.com.br/ vocesa.html) e Veja (www.uol.com.br/veja.html). O arquivo de VIF *forward* realiza o mapeamento entre a página da UOL e seus *forwardlinks,* e o arquivo VIF *backward* realiza o mapeamento de uma página com seus respectivos *backlinks*. Além de realizar estes mapeamentos estas estruturas podem guardar outras informações relacionadas ao *link,* como os pesos impl_aut e impl_hub utilizados para minimizar a realação de reforçõ mútuo entre *hosts* presente no cálculo dos pesos de hub e autoridade.

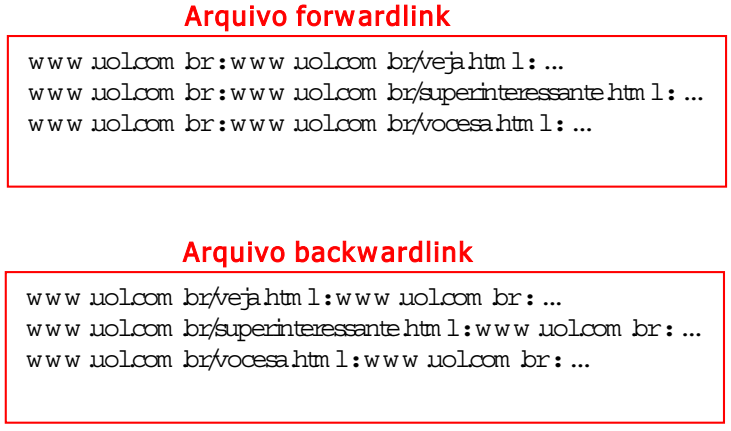

Figura 19: Forma simplificada dos arquivos VIF *forward* e *backward*.

Como as URLs possuem o tamanho médio de 80 bytes, armazenar a própria URL como um nó do grafo seria bastante custoso. Por esta razão foi utilizada uma estrutura que realiza o mapeamento entre a URL e um número inteiro que corresponde

ao seu identificador. A Figura 1920 ilustra os arquivos VIF que armazenam as informações sobre os *forwardlinks e backlinks* das páginas do grafo *Web,* respectivamente.

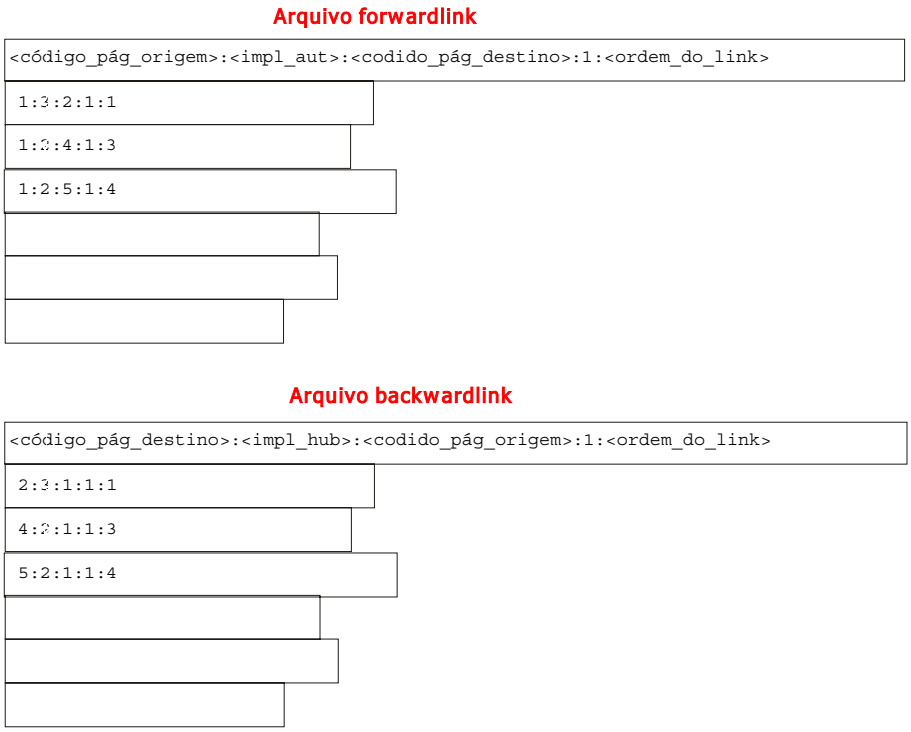

Figura 20: Arquivos VIF que representam o grafo Web.

Como pode ser visto, os arquivos mostrados contêm os mesmos campos do arquivo VIF utilizado para armazenar as informações sobre o conteúdo dos documentos, descrito na Seção 2.2.2. Porém, foram associadas outras semânticas[9] a cada campo que compõe este arquivo. Os valores de impl_*hub* e impl_aut correspondem aos valores explicados na Seção 4.3.1, que objetivam minimizar a relação de reforço mútuo entre os *hosts*.

[9] A constante 1 que precede o campo <ordem_do_link> não tem utilidade neste algoritmo, sendo este um campo reservado para uso futuro.

5.4 Estratégias de Implementação do GHHITS

Nesta seção são descritas duas estratégias de implementação do algoritmo GHHITS. Inicialmente será explicada a estratégia chamada GHHITS *Naive*, durante a qual são explicitadas as estruturas de dados necessárias à computação dos pesos de *hub* e autoridade.

Posteriormente, é apresentada uma estratégia de computação mais eficiente, chamada de GHHITS *Escalável*, a qual reduz substancialmente os requisitos de memória principal, utilizando-se da estratégia de *join* orientado a blocos [Hav99]. A estratégia de *join* orientado a blocos foi adotada neste trabalho por ter sido aplicada para a solução de um problema semelhante – gerenciamento da memória utilizada durante a computação do algoritmo PageRank. Estratégias similares são comumente utilizadas, por comunidades científicas, para otimizar processamentos baseados na multiplicação entre matrizes e vetores ([Tol97, IY99] referenciados por [Hav99]).

Uma vez que, o GHHITS é executado sobre um subgrafo da *Web,* representado na forma de grandes estruturas de dados armazenadas em disco, a análise de custo deste algoritmo é feita com base nos custos de I/O. Avaliações empíricas do tempo de processamento também são apresentadas para as estratégias de implementação descritas a seguir.

5.4.1 GHHITS *Naive*

Conceitualmente, o processo de análise de *links* pode ser representado através de sucessivas multiplicações entre a Matriz de Links e um vetor de pesos de "importância". A Figura 21 ilustra este processo. Os Apêndices D e E apresentam mais detalhes a respeito do processo de análise de *links* através da multiplicação entre matizes e vetores.

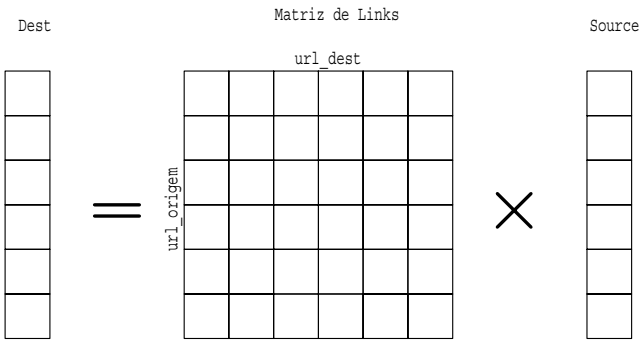

Figura 21: Multiplicação entre a Matriz de *links* e o vetor de pesos de importância ("Source").

Como mostrado na seção anterior, a estrutura de *links* da *Web,* é armazenada em dois arquivos VIF. Por esta razão, a matriz de adjacência, ilustrada na Figura 21, possui duas versões: (i) matriz de *forwardlinks* - uma matriz quadrada M onde cada entrada (i,j) desta matriz é igual a 1 se existe um *link* do documento i para o documento j, ou 0 caso contrário – representada pelo arquivo VIF *forward*; e a matriz de *backlinks* - uma matriz quadrada M onde cada entrada (i,j) desta matriz é igual a 1 se existe um *link* do documento j para o documento i, ou 0 caso contrário – representada pelo arquivo VIF *backward*.

A estratégia GHHITS *Naive*, representa os dois vetores ("Dest" e "Source"), ilustrados na Figura 21, através de dois *arrays* do tipo *double*, contendo uma entrada para cada página do grafo.

O algoritmo iterativo para o cálculo dos pesos de *hub* e autoridade pode ser expresso através do pseudo código mostrado na Figura 22 e na Figura 23.

```
iterateNaive(forwardLinks,backwardLinks,k)
begin

        k:naturalnumber
        authorityWeightArray:arrayofnweights
        hubWeightArray:arrayofnweights
        forwardLinks:structurewhichstoretheforwardlinksofeachpage
        backwardLinks:structurewhichstorethebackwardlinksofeachpage.

        authorityWeightArray = inicializeWeight(1);

        for i=0... k{
            calculateTotalWeights(forwardLinks,authorityWeightArray,hubWeightArray);
            calculateTotalWeights(backwardLinks,hubWeightArray,authorityWeightArray);
            normalize(authorityWeightArray);
            normalize(hubWeightArray);
        }

end
```

Figura 22: Corpo do algoritmo GHHITS *Naive.*

```
calculateTotalWeights(Links,Dest,Source)
begin
        Dest = inicializeWeight(0);
        Links: structure which store the forward links or back links of each page.

    (a) while(Links.eof()){
    (b)     Links.read(url_origem ,n,url_dest1,url_dest2,url_dest3,... ,url_destn);
            For j= 1... n
                 Dest[url_destj] = Desti[url_destj] + Source[url_origem ];
             }
        }

        write Dest to disk

end
```

Figura 23: Método do GHHITS *Naive* responsável pelo cálculo dos pesos de *hub* e autoridade.

A partir da análise deste pseudo-código - linha (a) da Figura 23 - pode-se perceber que a matriz de *links* ("Links") é percorrida 2 vezes por cada iteração, uma vez para calcular os valores dos pesos de *hub* a partir dos valores de autoridade, e uma segunda vez no cálculo dos pesos de autoridade a partir dos pesos de *hub* calculados no passo anterior. Assumindo que a memória principal é grande o bastante para armazenar os *arrays* contendo os pesos de *hub* e autoridade, o custo de I/O (C) para cada iteração da implementação mostrada acima é dada por:

$$C = 2x\,|Links|$$

Se a memória principal for grande o bastante para armazenar apenas o *array* Dest, nós assumimos que o *array* Source será armazenado em disco, o que acarretará no seguinte custo de I/O:

$$C = 2x\,|Links| + 2x\,|Dest| + 2x\,|Source|$$

O *array* Source é lido seqüencialmente da memória secundária durante o cálculo dos pesos do *array* Dest - linha (b) da Figura 23. *O array* Dest precisa ser escrito em disco para atuar como *array* Source no passo subseqüente da iteração Estes dois passos acontecem duas vezes ao longo das iterações: (i) no início da iteração o vetor de autoridades desempenha o papel do *array* Dest e o vetor de pesos de *hub* desempenha o papel do *array* Source; (ii) na segunda parte da iteração o vetor de *hub* desempenha o papel do *array* Dest e o vetor de pesos de autoridade desempenha o papel do *array* Source.

Apesar de muitas máquinas possuírem uma quantidade de memória principal suficiente para armazenar os *arrays* Source e Dest, processos de indexação que incorporam milhões de páginas, a cada dia, às bases do engenho de busca, irão claramente resultar em estruturas (vetores e matrizes) que podem chegar a exceder a quantidade de memória principal da maioria dos computadores.

A estratégia de computação, descrita a seguir, possibilita a execução do algoritmo GGHITS de análise de *links* em um subgrafo da *Web*, sem que seja necessária uma quantidade de memória principal proporcional ao número de páginas participantes do grafo.

5.4.2 GHHITS Escalável

Existe uma similaridade entre o cálculo do GHHITS e o operador relacional *join.* Sejam Dest, Source e Links tabelas de um banco de dados relacional, ilustradas na Figura 24.

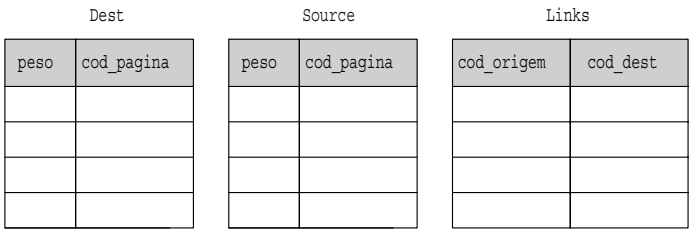

Figura 24: As estruturas Dest, Source e Links como tabelas de um BD.

De acordo com esta representação, o cálculo dos pesos de autoridade dos pesos de autoridade de uma página de código igual a x consiste em guardar em cada entrada do *array* Dest o somatório dos valores do *array* Source que obedecem ao seguinte critério:

```
select  sum (Source.peso)
 from  Source, Links
 where Links.cod_dest =x  and
        Source.cod_pagina = Links.cod_origem
```

Apesar desta analogia não ser exata, a principal técnica de *join* – o *join* orientado a blocos - pode ser utilizada para controlar o tamanho do *array* Dest durante a computação dos pesos de hub e autoridade.

A estratégia GHHITS Escalável permite que só sejam armazenados na memória principal os valores que estejam sendo utilizados pela computação. Nesta estratégia, o vetor Destino (que hora corresponde ao vetor de pesos de autoridade, hora ao vetor de pesos de *hub*) que na versão *Naive* ficava todo em memória, é particionado em β blocos, cada um contendo um número *D* de páginas, como ilustrado de forma abstrata na Figura 25 e na Figura 26.

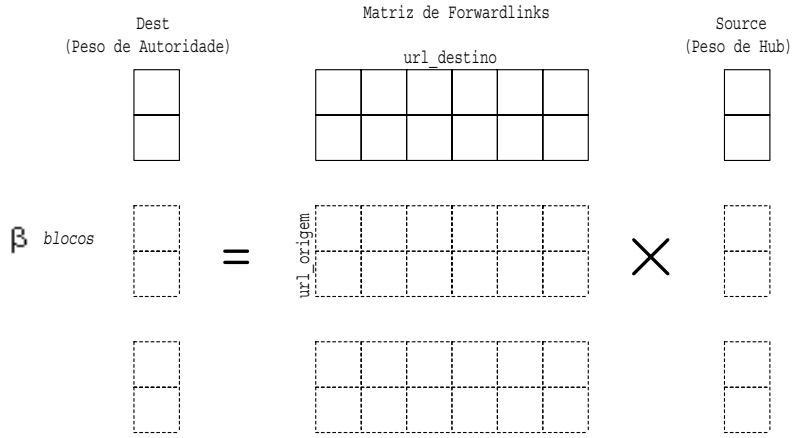

Figura 25: Cálculo dos pesos de autoridade.

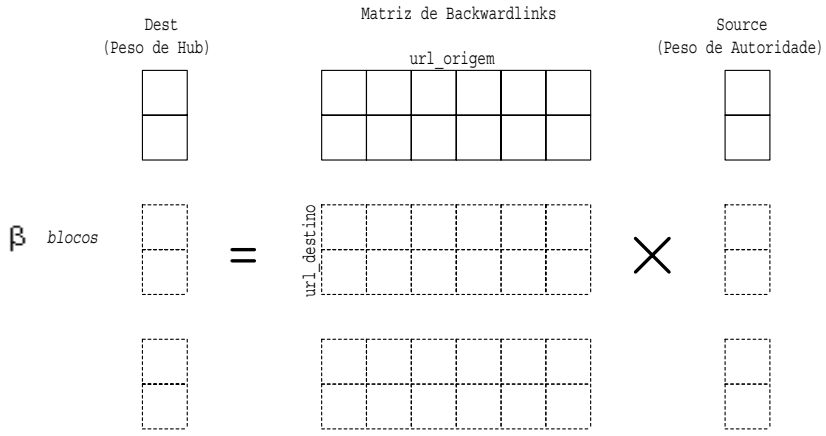

Figura 26: Cálculo dos pesos de hub.

Fisicamente a estrutura de *Links Forward* e a estrutura de *Links Backward* correspondem a arquivos binários particionados. A Figura 27 mostra graficamente como está representado o arquivo de *Links Forward.*

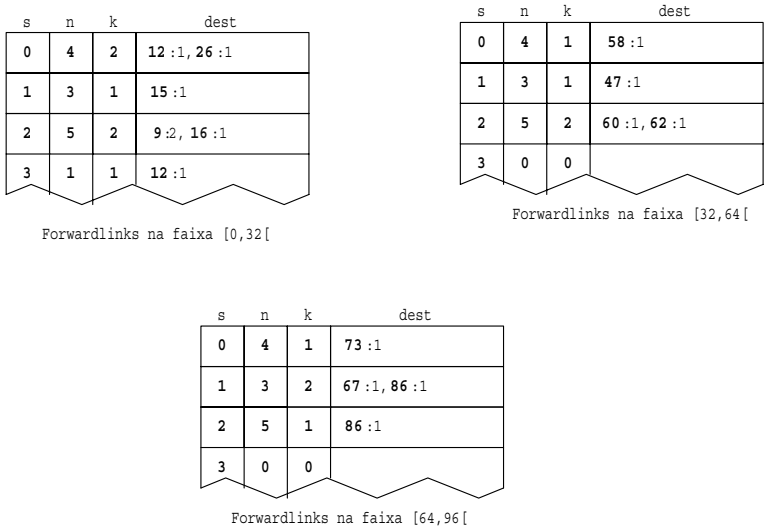

Figura 27: Estrutura de links particionada.

Vemos que cada linha do bloco possui os seguintes campos: **s** - nó origem; **n** - número total de *links* existentes no corpo de s (*forwardlinks*); **k** - número de *forwardlinks* de s que possuem código pertencente à faixa (ex.: [0,32[); **dest-***forwardlinks* de s que possuem código pertencente à faixa (ex.: [0,32[). O campo **s** varia de 0 ao maior código de página, em cada um dos blocos.

O algoritmo GHHITS Escalável pode ser expresso através do seguinte pseudo código:

```
iterate(G,k)
begin

   G :collection of n linked pages
   K :natural number
   authorityWeightArray :array of n weights
   hubWeightArray :array of n weights
   forwardLinks: structure which store the forward links of each page
   backwardLinks: structure which store the backward links of each page.
   IsAuthority :boolean value which is true, when the authority weight is being
                calculated or false otherwise

authorityWeightArray = inicializeAuthorityWeight(1);
writeOnDiskAuthorityWeightArray();

for i=0... k{

    hubWeightArray = readFromDiskHubWeightArray();
    IsAuthority=true;
    calculateTotalWeights(forwardLinks,hubWeightArray,IsAuthority);

    authorityWeightArray = readFromDiskAuthorityWeightArray();
    IsAuthority=false;
    calculateTotalWeights(backwardLinks,authorityWeightArray,IsAuthority);

    normalize(authorityWeightArray);
    normalize(hubWeightArray);
}
end
```

Figura 28: Corpo do algoritmo GHHITS Escalável.

```
calculateTotalWeights(Links,Source,IsAuthority)
begin

      Links_i: structure which store the forward links of each page on the block i.

      for i= 0 ..NUM_BLOCKS{
          Dest_i= 0;
          while(Links_i.eof()){
                Links_i.read(source,n,k,dest1,dest2,dest3,... ,destk);
                For j= 1... k
                      Dest_i[destj] = Dest_i[destj] +Source[source];
                }
                if IsAuthority
                      writeOnDiskAuthorityWeightBlock(Dest_i);
                else
                      writeOnDiskHubWeightBlock(Dest_i);
                fi
        }
end
```

Figura 29: Método do GHHITS Escalável responsável pelo cálculo dos pesos de *hub* e autoridade.

O *array* Dest é representado através de *arrays* do tipo *double*. Cada *array* contém um número de entradas igual à quantidade de nós do grafo. E após todos os blocos de arquivo de forward terem sido percorridos, o procedimento se repetirá, só que desta vez o

array Dest conterá os pesos de *hub* e os blocos Links_i percorridos corresponderão aos blocos do arquivo de backward.

A matriz de *links* será percorrida 1 vez a cada vez que o *array* Source é percorrido. Sendo ε o acrécimo no espaço de armazenamento ocasionado pelo particionamento, sendo definido de tal forma que a igualdade $\Sigma i|Links\_i| = |Links| \times (1+\varepsilon)$ seja satisfeita, o custo de I/O da implementação do GHHITS Escalável pode ser dada por:

$$C = 2x (|Links| \times (1+ \varepsilon)) + 2x |Dest| + 2\beta x |Source|$$

Na prática ε mostrou-se razoavelmente pequeno como mostrado em experimentos realizados neste trabalho e detalhados a seguir, e ilustrados na Tabela 1.

5.4.3 Comparação entre as Estratégias

As estratégias descritas foram aplicadas a um subconjunto da *Web* contendo 3.742.983 páginas, que corresponde a um subconjunto das páginas indexadas pelo engenho de busca ***Radix*** ao longo de três meses - setembro, outubro e novenbro de 2001. Este subconjunto foi adotado neste expefimento por representar um subconjunto representativo das páginas que compõem a Web brasileira - de acordo com as estimativas calculadas na seção 6.1 este subconjunto corresponde a 11,14% da Web brasileira.

Este subconjunto de páginas indexadas dá origem a um grafo composto por 85.648.933 nós, que correspondem às páginas efetivamente indexadas e aos *links* existentes no corpo destas páginas.

Como descrito na Seção 4.3.2, antes da execução do algoritmo de análise de *links* este grafo passa por um processo de limpeza onde são aplicadas as heurísticas detalhadas na seçaõ referenciada.

Este processo de limpeza é necessário para evitar a autopromoção entre *links* do mesmo site, e evitar a influência de *links* comerciais na computação dos pesos de *hub* e autoridade.

Após o processo de limpeza, o subconjunto de páginas indexadas reduziu-se a 1.384.398, o qual deu origem a um grafo contendo 2.882.853 nós - o que corresponde a 3,37% do total de *links* coletados. À estes nós resultantes são atribuídos novos ids de

forma seqüencial variando de 0 a 2.882.852, que ao final do processamento são mapeados para o universo de códigos de página utilizados pelo engenho de busca ao qual este módulo foi inserido.

O grafo resultante da limpeza é armazenado nos arquivos VIF ilustrados na Figura 19 e na Figura 20. Os identificadores dos nós origem, dos nós destino e o número de *links* de saída são representados por números inteiros contendo 32 bits. O tamanho da estrutura de *links*, após o processo de limpeza é de 14,64 MB. Os vetores de *hub* e autoridade, deste grafo contendo 2.882.853 nós, ocupam juntos aproximadamente 44 MB.

Foi realizado um experimento com o objetivo de verificar o acréscimo no espaço de armazenamento provocado pela estratégia de particionamento deste grafo Web. Neste experimento foi verificado o espaço utilizado por utilizando três tipos de particionamento diferentes: 1 bloco (GHHITS Naive), 2 blocos e 4 blocos. A Tabela 1 ilustra os resultados deste experimento, os quais constataram um pequeno acréscimo no espaço de armazenamento, provocado pelo particionamento. Vale a pena salientar que estas estruturas armazenadas em disco são compactadas, e por esta razão não representam o tamanho das estruturas que serão colocadas na memória principal durante a execução do GHHITS.

Número de blocos	Tamanho da estrutura de *links forward*	Tamanho da estrutura De *links backward*	Tamanho Total	ε
1	12,02 MB	2,62 MB	14,64 MB	0 MB
2	12,14 MB	2,63 MB	14,77 MB	0,13 MB
4	12,36 MB	2,64 MB	15,00 MB	0,36 MB

Tabela 1- Resultados obtidos no experimento de análise do acréscimo no espaço de armazenamento provocado pela estratégia de particionamento.

Outro custo introduzido pelo particionamento é o tempo gasto para percorrer o *array* Source β vezes para se calcular todos os valores do *array* Dest. Experimentos realizados neste trabalho mostraram que o acréscimo no tempo necessário para percorrer o vetor Source β vezes é pequeno.

Estes experimentos consistiram em medir o tempo de execução por iteração do GHHITS utilizando três tipos de particionamento: 1 bloco (GHHITS *Naive*), 2 blocos e 4 blocos. Cada tipo de particionamento sendo testado sob três diferentes configurações de

memória física: 512 MB, 32 MB e 24 MB. Estas configurações foram escolhidas por simularem os seguintes cenários: (i) na configuração de memória física com 512 MB, todas as estratégias possuem memória suficiente para armazenar os seus vetores Dest em memória com folga para armazenar estruturas auxiliaes; (ii) 32 MB corresponde ao tamanho aproximado do vetor Dest da estratégia Naive; (iii) a configuração de memória contendo 24MB simula a situação onde a quantidade de memória principal é insuficiente para armazenar o vetor Dest da estratégia Naive - a qual armazena no vetor Dest uma entrada para cada página pertencente ao Grafo.

A Figura 30 mostra o tempo requerido por cada iteração do GHHITS, para cada um dos três tipos de particionamento submetidos a três diferentes configurações de memória.

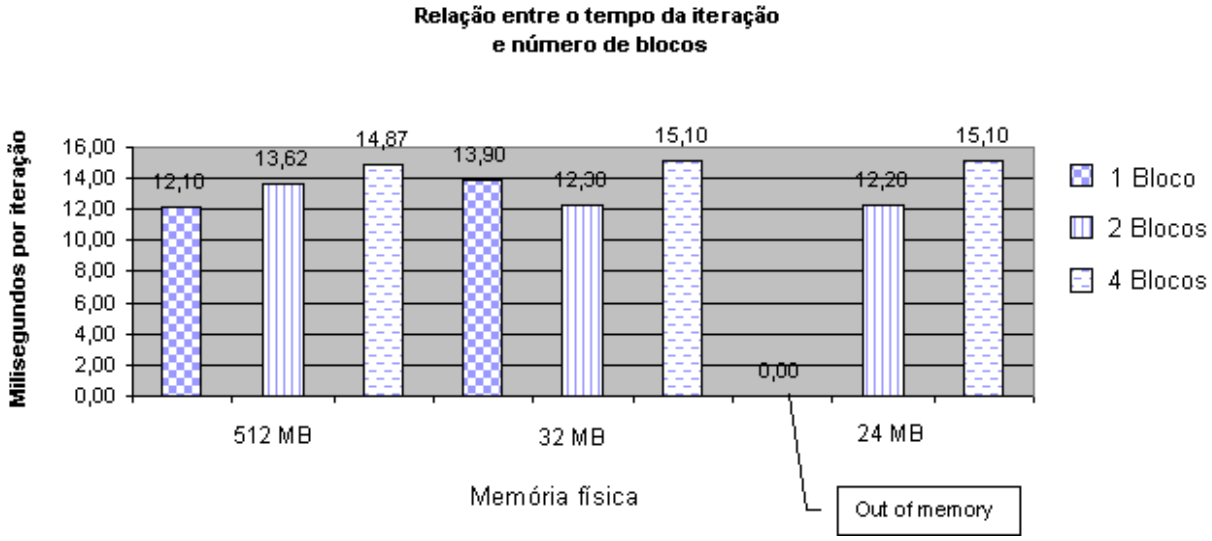

Figura 30: Relação entre o tempo gasto por iteração, o número de blocos e o tamanho da memória disponível.

A partir da análise deste gráfico, pode-se perceber que a utilização de muitos blocos degrada a performance, pois muito tempo é gasto em operações de I/O. Em contrapartida, a utilização de apenas um bloco realiza menos operações de leitura e escrita de blocos, mas não é escalável, já que o consumo de memória cresce proporcionalmente ao número de páginas que passam a participar da computação.

Assim sendo, a estratégia de particionamento mais eficiente é aquela que procura equilibrar o número de operações e I/O com a quantidade da memória utilizada, de forma que a memória disponível é suficiente para armazenar pelo menos um bloco. Neste experimento a estratégia que se mostrou mais eficiente foi a estratégia que particiona o *array* Dest em dois blocos.

5.5 Análise da convergência

A ordem global induzida pelo vetor de pesos de autoridade nos apresenta uma visão intuitiva da taxa de convergência.

O GHHITS foi executado sobre o grafo *Web* descrito na seção anterior, e em um grafo contendo a metade do número de páginas deste grafo. A cada 25 iterações a ordem entre as páginas foi comparada. A Figura 31 mostra um gráfico contendo a comparação entre as ordenações.

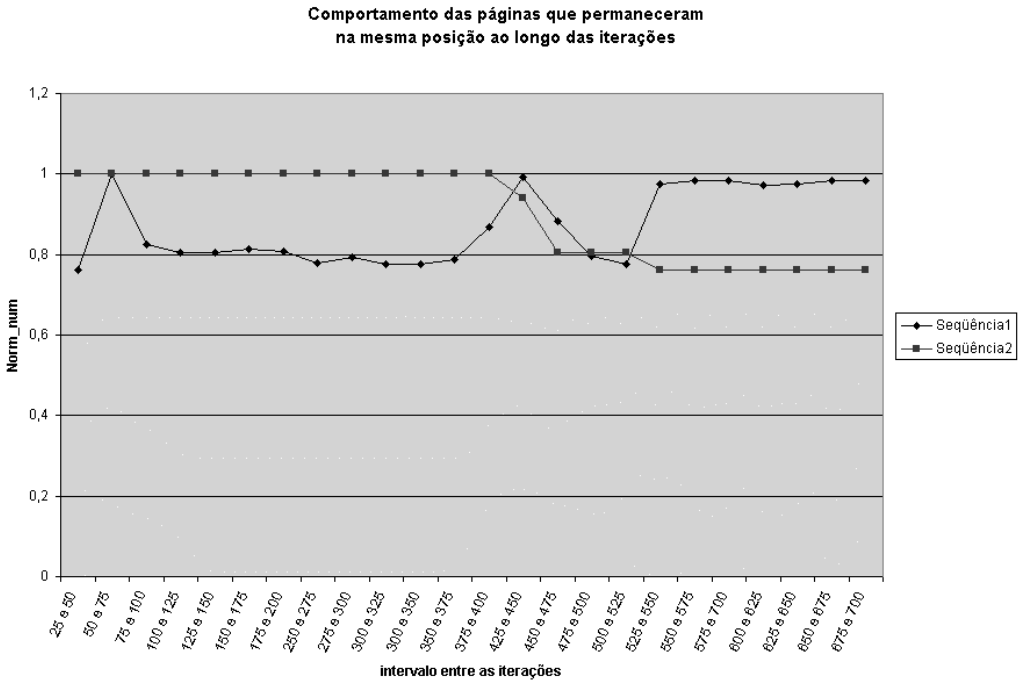

Figura 31: Número de páginas na mesma ordem induzida entre as iterações. A Seqüência 2 corresponde ao comportamento das páginas do grafo contendo 2.882.853 nós, e a Seqüência 1 corresponde ao comportamento das páginas em um grafo com 1.440.003 nós. O valor de Norm_num corresponde ao número de páginas que permaneceram na mesma colocação entre as iterações, normalizado pelo maior valor deste número obtido neste experimento.

Pode ser observado, através deste gráfico, que a partir da iteração 550 o número de páginas na mesma posição, permaneceu constante nas duas execuções do algoritmo GHHITS.

Também foi analisado o número de páginas com peso de autoridade diferente de zero a cada iteração - Figura 32. Percebe-se que, na medida em que a as iterações vão sendo executadas, o número de páginas com peso de autoridade diferente de zero passa a

ser quase constante a cada iteração a partir da iteração 550. O que é indício que neste momento foi atingido um estado estável.

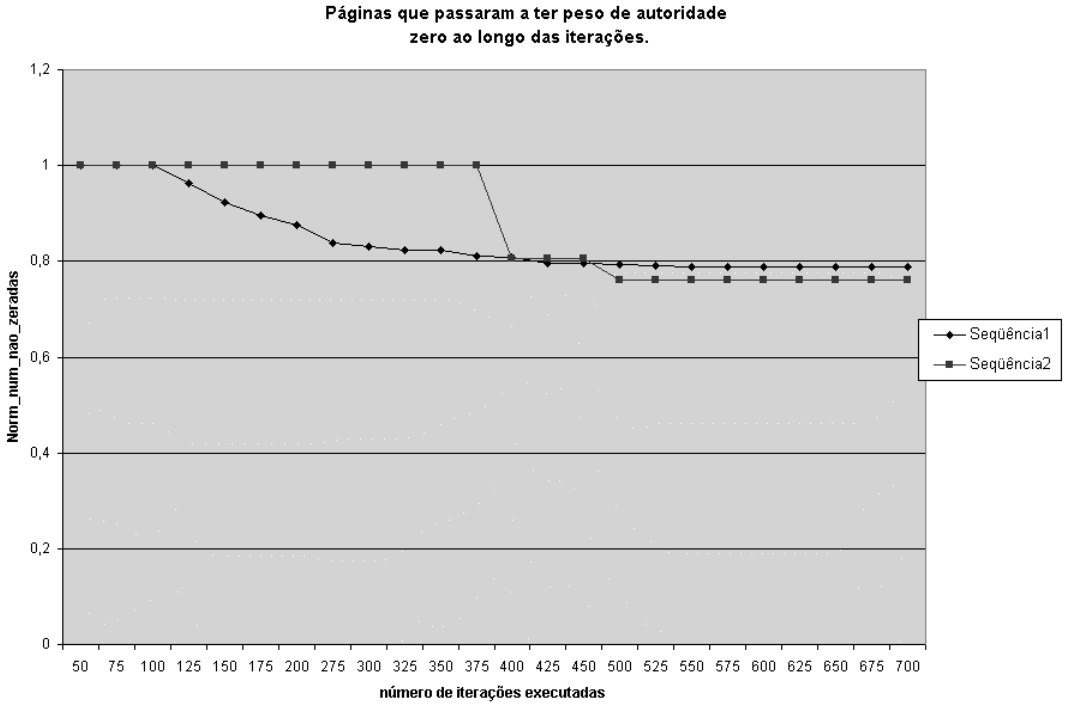

Figura 32: Número de páginas que passam a ter peso zero ao longo da iterações. O valor de Norm_num_nao_zeradas corresponde ao número de páginas que possuem pesos de autoridade diferentes de zero, normalizado pelo maior valor deste número obtido neste experimento.

Em um grafo contendo 2.882.853 nós, apenas 270.465 nós possuíam pesos de autoridade maior que zero, após a convergência, isto é apenas 9,38% das páginas do grafo *Web* analisado possuem peso de autoridade a ser utilizado como um componente do processo de ranqueamento, as demais páginas possuem peso de autoridade igual a zero.

A computação do GHHITS é finalizada quando os seguintes critérios de parada são atendidos:

(1) O número de pesos de autoridade diferentes de zero assim como o número de páginas na mesma ordem permanecem constantes ao longo de 10 iterações.

(2) Ou quando o número de iterações ultrapassar 1.000.

Este último critério impede que, sob alguma configuração do grafo de entrada, esta computação chegue a convergir, e por este motivo continue a computação dos pesos

de *hub* e autoridade indefinidamente.

Como foi visto na seção anterior, nas configurações, nas quais foi realizado o experimento, o tempo médio gasto por iteração é de aproximadamente 15s. Para executar as 1000 iterações seriam necessários apenas 15.000 segundos que equivalem a 4,17 dias.

5.6 Servidor de Pesos

Os pesos originados a partir do processamento de análise de *links* deverão ficar acessíveis ao engenho de busca, que utilizará os mesmos no momento de ranquear as páginas da resposta. O processamento do GHHITS descrito na seção anterior armazena os pesos de *hub* e autoridade em um arquivo binário contendo os seguintes campos por linha: <peso_autoridade><peso_*hub*>. Neste arquivo o número da linha corresponde ao código da página no universo e códigos do engenho de busca ao qual o SAAL foi adicionado.

Como neste tipo de aplicação, o tempo de resposta é crucial, optou-se por usar uma *cache* de pesos. Essa *cache* armazena em memória as 300.000 páginas com maior peso de autoridade, e é realimentada à medida que requisições por outras páginas cheguem.

Quando uma requisição é feita ao servidor de pesos, este sistema procura os pesos de autoridade e *hub* em sua *cache*, caso não exista ele procura estes pesos no arquivo que contém todos os pesos de autoridade e *hub* pré-calculados. Esta *cache* é implementada segundo a estratégia LRU (*Last Recently Used*), os pesos de autoridade solicitados a mais tempo, dão lugar aos pesos de autoridade solicitados mais recentemente. Caso os pesos de autoridade e *hub* não existam no arquivo, é retornado zero para o engenho de busca.

Como forma de otimizar esta consulta que é feita via *socket*, as requisições são feitas por blocos de N páginas, e a resposta à requisição consiste em um *array* contendo os pesos para as páginas solicitadas. O tempo de resposta médio para a uma requisição de pesos de autoridade de um bloco contendo 500 páginas é de aproximadamente 223ms.

5.7 Considerações Finais

Nesse capítulo, foram apresentados os aspectos práticos sobre a implementação do Sistema de Armazenamento e Análise de *Links* (SAAL) e sobre a incorporação do mesmo ao engenho de busca ***Radix*** utilizado como estudo de caso. Como pôde ser visto, o sistema foi desenvolvido de forma modular e facilmente extensível. Para isso, foram adotadas a abordagem orientada a objetos e a linguagem de programação *Java*. Detalhes de implementação e decisões tomadas durante o processo de implementação foram apresentados e discutidos. O próximo capítulo avalia o impacto na precisão da resposta causado pela utilização do peso de autoridade em conjunto com o peso de similaridade textual.

6

Avaliação da Eficácia de Recuperação

A avaliação da eficácia de recuperação de informação de um engenho de busca é de fundamental importância para a validação das técnicas e tecnologias desenvolvidas assim como para supervisionar a qualidade do serviço que é oferecida pelo engenho de busca.

Neste trabalho foi construído um sistema de avaliação de eficácia de recuperação de engenhos de busca, para que fosse avaliado o impacto na precisão da resposta causado pela fusão do paradigma de análise de *links* (GHHITS) com o paradigma textual.

Foram realizados os seguintes tipos de experimentos, que utilizaram a fórmula de fusão de informação mostrada na Seção 4.5, com variações apenas no valor dos coeficientes *a* e *b*:

(1) Estratégia Textual (**TEXT**);

(2) Peso de Autoridade (**AUT**);

(3) 0.85 x Estratégia Textual + 0.15 x Peso de Autoridade (**0.85TEXT_0.15AUT**)

(4) 0.75 x Estratégia Textual + 0.25 x Peso de Autoridade (**0.75TEXT_0.25AUT**)

(5) 0.65 x Estratégia Textual + 0.35 x Peso de Autoridade (**0.65TEXT_0.35AUT**)

A coleção de testes utilizada consistiu em um conjunto contendo 3.742.983 documentos, corresponde ao subconjunto das páginas indexadas pelo engenho de busca ***Radix*** - detalhado na Seção 5.4.3. Durante a avaliação foram utilizadas as métricas, descritas na Seção 2.5.3: precisão@10, precisão@7, precisão@5, precisão PMTS@10, precisão PMTS@7, precisão PMTS@5.

Adicionalmente, foi realizado um estudo que definimos, neste trabalho, como estudo da "aceitabilidade da estratégia". A aceitabilidade da estratégia corresponde ao número de consultas para as quais uma estratégia foi considerada melhor que outra, em termos do valor da precisão@10. Diz-se que uma estratégia **A** possui maior aceitabilidade que uma estratégia **B**, quando a estratégia **A** possui um maior número de consultas com maior valor da precisão@10, com relação à estratégia **B**.

A motivação para este estudo específico foi detectar os casos em que a estratégia possuía um alto valor de precisão relativa média, mas, em contrapartida, possuía um baixo valor da precisão relativa para maioria das consultas. Este estudo também foi realizado por Chakrabarti et. al. [CDG+98c], em experimentos que realizaram a comparação entre um conjunto de estratégias de busca. O comportamento detectado por este estudo também pode ser detectado através da análise do valor da mediana da precisão@10.

Nesse capítulo, inicialmente, são definidos o método de avaliação utilizado e a ferramenta de avaliação desenvolvida. Posteriormente, são mostrados os resultados obtidos na avaliação das estratégias de busca enumeradas acima, além da discussão acerca destes resultados.

6.1 Método de Avaliação

Neste trabalho, foi elaborado um método de avaliação que procurou atender às premissas que devem orientar métodos de avaliação de sistema de RI, enumeradas na Seção 2.6. A seguir, são descritos: o propósito, as consultas teste, a técnica de avaliação e as métricas utilizadas por este método de avaliação. Chamamos o método de avaliação proposto neste trabalho, e descrito a seguir, de **método Radix** de avaliação de estratégias de busca.

Propósito

O objetivo deste método é comparar a eficácia de recuperação das 5 estratégias de busca enumeradas no início deste capítulo. Uma estratégia de busca é definida como uma implementação da fórmula de fusão de paradigmas (Fórmula 22, Seção 4.4.7), a partir da utilização de diferentes valores para os coeficientes a e b.

Consultas teste

Neste estudo foram selecionadas as 113 consultas. Estas consultas correspondem a união das consultas pertencentes ao conjunto das 100 consultas submetidas com maior freqüência ao engenho de busca ***Radix*** entre os meses de outubro, novembro e dezembro de 2001. A Figura 33 mostra o número de termos por consulta desta amostra.

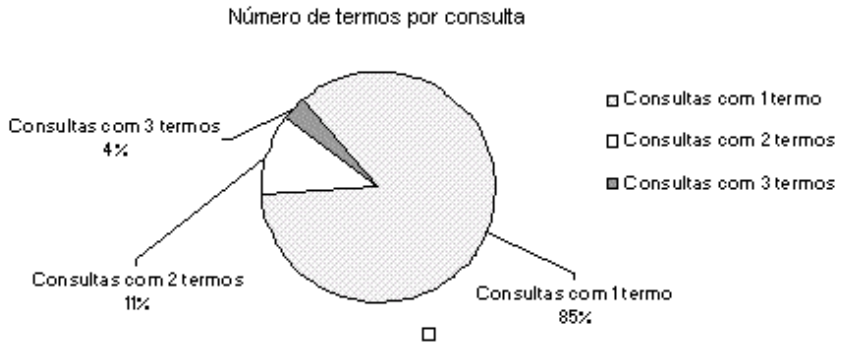

Figura 33: Número de termos por consulta no conjunto de consultas teste.

Técnica de avaliação

Para avaliarmos a inclusão do paradigma estrutural ao processo de busca, foi utilizada a técnica de julgamento de relevância cego. Para aplicar esta técnica, foi desenvolvida uma ferramenta de avaliação semi-automática da eficácia de recuperação de um sistema de RI.

O sistema permite o cadastro de *n* estratégias de busca para serem avaliadas em paralelo. A Figura 34 ilustra os passos que são realizados pelo sistema de avaliação, em um cenário composto por duas estratégias sendo avaliadas.

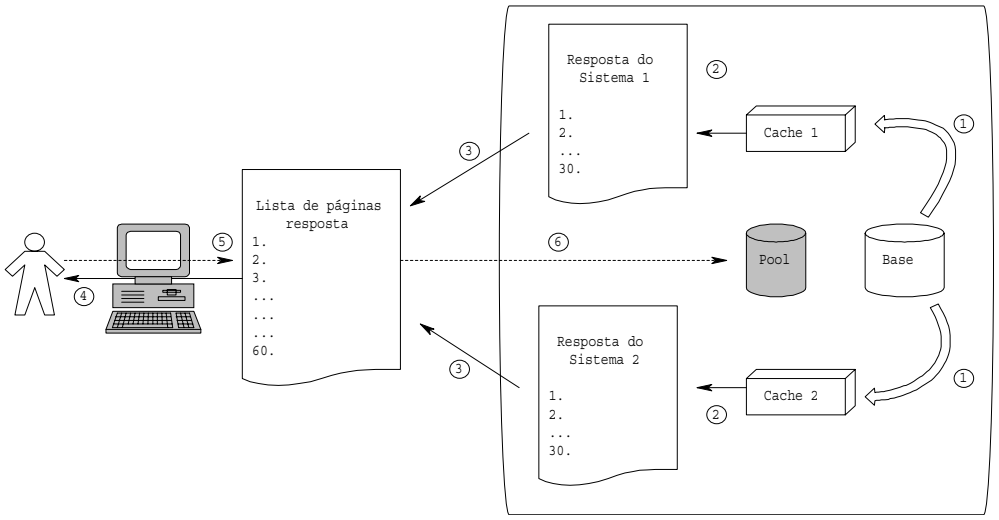

Figura 34: Sistema de avaliação da eficácia de recuperação.

Os passos ilustrados na Figura 34 são descritos a seguir:

Passos 1 e 2 – Pré-Processamento das Consultas Teste: as consultas teste são submetidas as duas estratégias de busca avaliadas. Para cada consulta submetida, as *n* primeiras URLs retornadas por cada estratégia são inseridas em uma base temporária – foi utilizada uma base temporária, ao invés da avaliação *on-line*, para evitar que os avaliadores esperassem o intervalo de tempo necessário para submeter a consulta a *n* estratégias de busca.

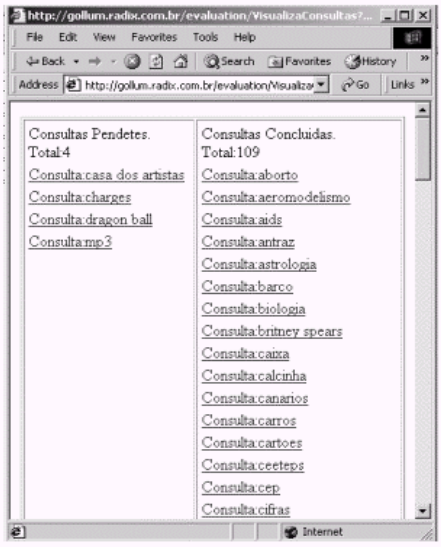

Figura 35: Tela do sistema de avaliação que exibe as consultas teste ao usuário.

Passos 3 e 4 – Apresentação da Lista de Páginas: o usuário submete uma das consultas teste (ver Figura 35). É retornada, então , uma lista contendo as URLs resposta das duas estratégias avaliadas, ordenadas de forma aleatória. A tela responsável por apresentar a lista resposta contém 3 colunas: a primeira contendo as páginas não avaliadas, a segunda contendo as páginas julgadas como relevantes, e a terceira contendo as páginas julgadas como irrelevantes (ver Figura 36).

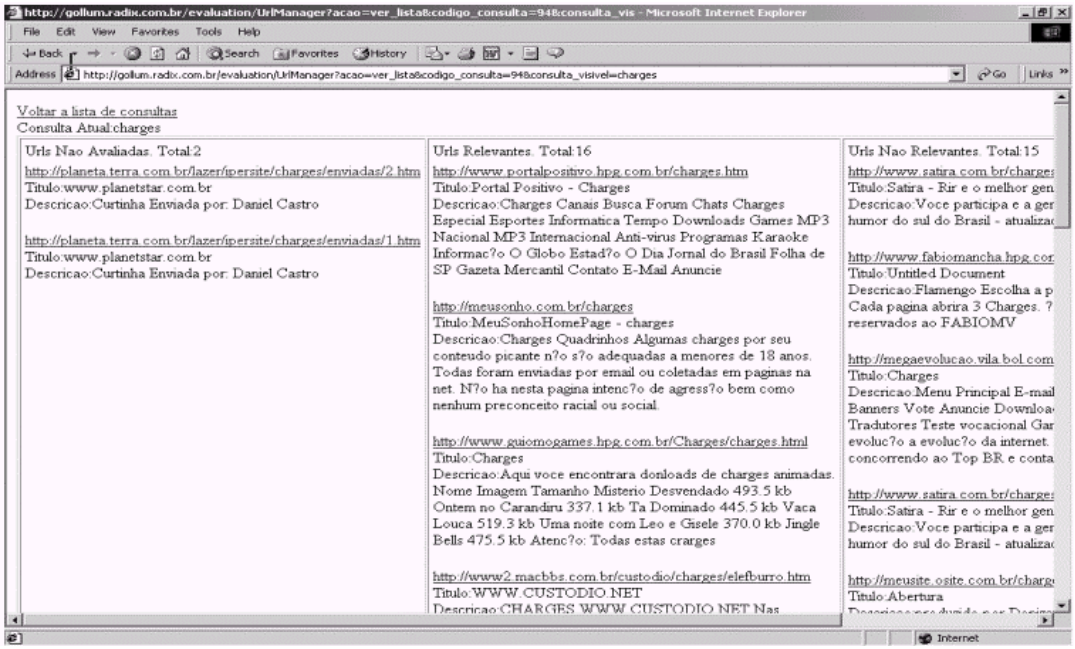

Figura 36:Tela do sistema de avaliação que mostra as URLs a serem avaliadas.

Passos 5 e 6 – Julgamento de Relevância Cego: o usuário seleciona uma das páginas retornadas e classifica-a como relevante ou irrelevante. A tela de avaliação (ver Figura 37) contém: o site a ser avaliado, 3 links - relevante, irrelevante e não avaliar agora, este último é necessário caso o usuário deseje interromper a avaliação. Este julgamento é armazenado pelo sistema no Pool – uma base que posteriormente será utilizada para o cálculo das métricas. Segundo estudos mencionados na Seção 2.6, o critério de relevância binário é suficiente para avaliar sistemas de recuperação de informação.

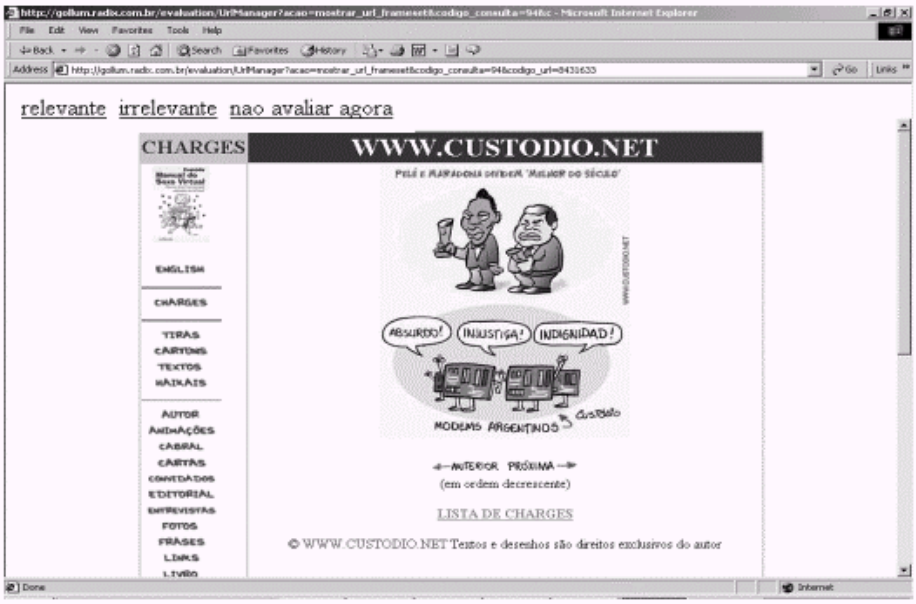

Figura 37: Tela de julgamento de relevância.

Avaliadores Humanos

Participaram da avaliação 10 voluntários, todos eles profissionais da área de informática e com experiência na utilização de engenhos de busca para a *Web*. Cada avaliador ficou responsável por avaliar, individualmente, um número pré-determinado de consultas (5 a 20) durante um período máximo de 5 dias.

Métricas e técnicas utilizadas

Foram utilizadas as seguintes métricas: precisão@10, precisão@7, precisão@5, precisão PMTS@10, precisão PMTS@7, precisão PMTS@5.

Base de documentos

A base de dados sobre a qual foram aplicadas todas as estratégias avaliadas, corresponde a um subconjunto das páginas indexadas pelo engenho de busca ***Radix*** contendo 3.742.983 documentos, detalhado na Seção 5.4.3. Todas as páginas contidas nesta base (com exceção das páginas que não possuíam *links* extrínsecos) participaram da computação dos pesos de *hub* e autoridade. Nesta computação todas as entradas dos vetores InitAut e InitHub foram iguais a 0.

Para calcularmos a representatividade desta amostra podemos compará-la ao tamanho da *Web* brasileira. Sabemos que não é possível determinar o tamanho exato da

Web brasileira, pois a cada dia surgem novas páginas ao passo em que outras deixam de existir. Todavia, podemos estimar o seu valor da forma descrita a seguir.

Uma consulta à página de estatísticas de domínios DNS da *Web* brasileira cadastrados na FAPESP – Fundação de Amparo a Pesquisa do Estado de São Paulo (http://www.registros.br), na ocasião da avaliação dos sistemas (janeiro de 2002), revelou que havia 413.952 domínios cadastrados com DNS válido. Um estudo da *Web* brasileira realizado por B. Ribeiro-Neto et. al.[ASG+00] constatou que um servidor da *Web* brasileira possui, em média, 70 documentos e que o número de servidores por domínio é de 1,16. Tomando como base estes valores pode-se estimar o número de documentos existentes na *Web* em dezembro de 2001 era de 1,16x70x413.952 ≅ 33.612.902 documentos da *Web* brasileira.

Logo, o subconjunto da *Web* contendo 3.742.983 documentos utilizado nesta avaliação corresponde a 11,14% da *Web* brasileira.

6.2 Avaliações Realizadas

Foi avaliado o impacto na precisão da resposta, causado pela fusão do paradigma textual com o paradigma de análise de *links* (GHHITS).

O paradigma textual utilizado neste experimento correspondeu ao paradigma empregado pelo engenho de busca ***Radix***, utilizado como estudo de caso. Este paradigma é baseado no modelo booleano estendido incluindo medidas de proximidade entre os termos. A estratégia de recuperação textual deste sistema não será detalhada, uma vez que estas informações fogem do escopo deste trabalho, não sendo, portanto, necessárias à execução da avaliação em gestão.

6.2.1 Análise da Precisão com Limites

Nesta seção serão mostrados os valores das métricas precisão@10, precisão@7, precisão@5,PMTS@10, PMTS@7 e PMTS@5 para todas as estratégias de busca avaliadas. Os resultados obtidos pelas estratégias que utilizam o peso de autoridade são comparados com os resultados alcançados pela estratégia textual. Estas comparações são realizadas através das seguintes medidas de comparação absoluta (CA) e comparação relativa (CR).

Comparação absoluta (CA):

CA-TEXT = valor_métrica(estratégia_avaliada) - valor_métrica(TEXT)

Comparação relativa (CR)

$$\text{CR-TEXT} = \frac{\text{CA-TEXT}}{\text{valor\_métrica(TEXT)}}$$

Experimento com Estratégia Textual –TEXT

Os resultados encontrados para a estratégia **TEXT** a partir do julgamento cego entre os sistemas são ilustrados na Tabela 2:

	precisão@5	PMTS@5	precisão@7	PMTS@7	precisão @10	PMTS@10

Tabela 2 Precisão média com limite para TEXT.

TEXT	0,482	0,365	0,474	0,347	0,478	0,332

Experimento com Peso de Autoridade – AUT

Os resultados encontrados para a estratégia **AUT** a partir do julgamento cego entre as estratégias são ilustrados na Tabela 3:

	precisão@5	PMTS@5	precisão@7	PMTS@7	precisão @10	PMTS@10
AUT	0,309	0,208	0,294	0,188	0,284	0,166
CA-TEXT	-17,3%	-15,7%	-18,0%	-15,9%	-19,4%	-16,6%
CR-TEXT	-35,92%	-43,07%	-38,00%	-45,77%	-40,52%	-50,05%

Tabela 3: Precisão média com limites para AUT.

A estratégia baseada apenas no peso de autoridade provocou quedas muito grandes nos valores da precisão em comparação aos valores da estratégia puramente textual. Este resultado já era esperado uma vez que ao ranquear as páginas apenas por informações estruturais, perdem-se as características das páginas importantes de serem analisadas, tais como: o *tfidf*, a distância entre os termos, a distância entre o termo consultado e o início do documento, características estas que são levadas em consideração pelos algoritmos de recuperação textuais.

Experimento com Fusão de Informação - 0.85TEXT_0.15AUT

Os resultados encontrados para a estratégia **0.85TEXT_0.15AUT** a partir do julgamento cego entre os sistemas são ilustrados Tabela 4.

	precisão@5	PMTS@5	precisão@7	PMTS@7	precisão@10	PMTS@10
0.85TEXT_0.15AUT	0,525	0,405	0,516	0,384	0,516	0,368
CA-TEXT	4,2%	4,0%	4,2%	3,7%	3,9%	3,6%
CR-TEXT	8,78%	10,95%	8,87%	10,62%	8,13%	10,74%

Neste experimento verificou-se que todos os valores das precisões relativas melhoraram com relação aos respectivos valores de precisão obtidos para o experimento TEXT. As precisões relativas que mostraram maior melhoria segundo a comparação absoluta foram a precisão@5 e a precisão@7, ambas provocaram uma melhoria de 4,2%. Já a métrica PMTS@10 provocou uma melhor melhoria relativa (10,74%).

Experimento com Fusão de Informação - 0.75TEXT_0.25AUT

Os resultados encontrados para a estratégia **0.75TEXT_0.25AUT** a partir do julgamento cego entre os sistemas são ilustrados Tabela 5.

	precisão@5	PMTS@5	precisão@7	PMTS@7	precisão@10	PMTS@10
0.75TEXT_0.25AUT	0,536	0,428	0,531	0,407	0,523	0,381
CA-TEXT	5,3%	6,3%	5,7%	6,0%	4,5%	4,9%
CR-TEXT	11,06%	17,13%	11,96%	17,23%	9,42%	14,64%

Tabela 5: Precisão média com limites para - 0.75TEXT_0.25AUT.

Para este experimento verificou-se que todos os valores das precisões relativas

Tabela 4: precisão média com limites para - 0.85TEXT_0.15AUT

melhoraram com relação aos respectivos valores de precisão obtidos para o experimento TEXT. A métrica que mostrou maior melhoria segundo as comparações absoluta e relativa foi a PMTS@10, que provocou uma melhoria de 6,3% e 17,13%, respectivamente.

Experimento com Fusão de Informação - 0.65TEXT_0.35AUT

Os resultados encontrados para a estratégia **0.75TEXT_0.25AUT** a partir do julgamento cego entre os sistemas são ilustrados na Tabela 6.

	precisão@5	PMTS@5	**precisão@7**	**PMTS@7**	**precisão@10**	**PMTS@10**
0.65TEXT_0.35AUT	0,504	0,397	0,504	0,381	0,509	0,366
CA-TEXT	2,1%	3,2%	3,0%	3,4%	3,2%	3,4%
CR-TEXT	4,39%	8,66%	6,38%	9,84%	6,65%	10,13%

Tabela 6: Precisão média com limites para - 0.65TEXT_0.35AUT

Para este experimento verificou-se que todos os valores das precisões relativas melhoraram com relação aos respectivos valores de precisão obtidos para o experimento TEXT. As métricas que mostraram maior melhoria segundo a comparação absoluta foram a PMTS@7 e PMTS@10, ambas provocaram uma melhoria de 3,4%. Já a métrica PMTS@10 provocou a melhor melhoria relativa (10,13%).

Todas as estratégias que utilizaram a fusão de paradigmas, **0.85TEXT_0.15AUT, 0.75TEXT_0.25AUT, 0.65TEXT_0.35AUT,** conseguiram uma significativa melhora da precisão@10 (precisão relativa aos 10 primeiros documentos retornados), nestes experimentos, de acordo com a comparação relativa, a precisão melhorou em 8,13%, 9,42%, 6,65% respectivamente, e de acordo com a comparação absoluta, a precisão melhorou em 3,9%, 4,5%, 3,2% respectivamente com relação ao paradigma puramente textual **TEXT**.

A seguir, são mostrados alguns gráficos que ilustram as variações nos valores das precisões para as estratégias avaliadas.

Os valores da precisão relativa entre os cinco, sete e dez primeiros resultados para as estratégias avaliadas neste experimento são ilustrados na Figura 38, Figura 39 e Figura 40 respectivamente.

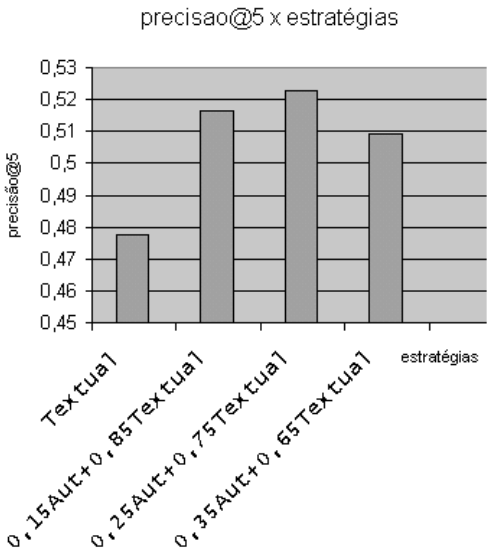

Figura 38: Gráfico contendo a variação dos valores da precisão relativa nos 5 primeiros documentos retornados em quatro estratégias avaliadas.

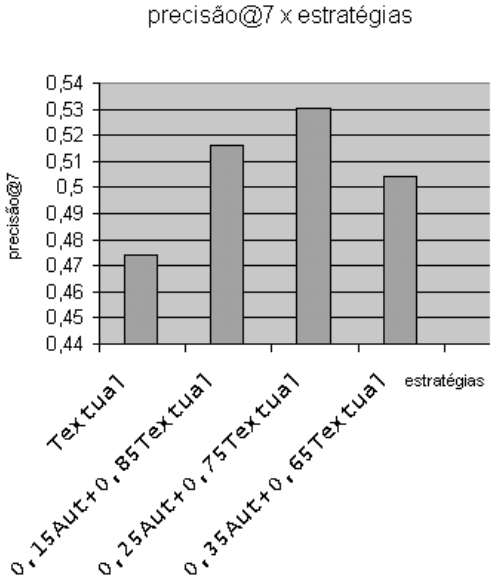

Figura 39: Gráfico contendo a variação dos valores da precisão relativa nos 7 primeiros documentos retornados em quatro estratégias avaliadas.

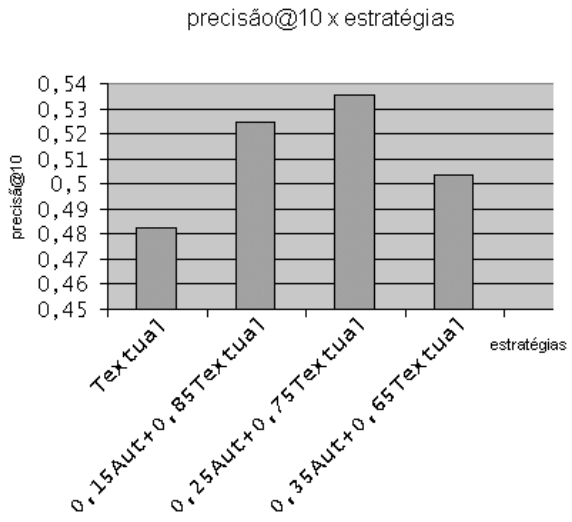

Figura 40: Gráfico contendo os valores da precisão relativa entre os 10 primeiros documentos retornados pelas estratégias avaliadas.

Ao se fazer uma análise destes gráficos pode-se perceber que após o valor do multiplicador do peso de autoridade ultrapassar 0,25 os valores de todas as precisões relativas avaliadas sofreram uma significativa queda. A partir de tal constatação pode-se concluir que para as configurações sobre as quais este teste foi executado o valor ideal para o coeficiente do peso de autoridade na fórmula de fusão adotada é de 0,25.

6.2.2 Aceitabilidade da Estratégia

Além do valor da precisão relativa, também foi analisada a aceitabilidade da estratégia - o número de consultas para as quais a estratégia avaliada possuiu maior valor da precisão@10 que a estratégia textual (TEXT). Através desta medida, pode-se mensurar o grau de satisfação do usuário do engenho de busca que utiliza cada uma destas estratégias. Através desta análise podem ser detectados os casos em que a estratégia possui um alto valor médio da precisão relativa, mas, em contrapartida, possui um baixo valor da precisão relativa para maioria das consultas.

A seguir são realizadas comparações dois a dois entre a estratégia TEXT e as três estratégias que utilizando as informações contidas nos *links* acarretaram uma significativa melhora no valor da precisão@10 com relação a estratégia TEXT.

No conjunto das consultas teste, a estratégia 0.85TEXT_0.15AUT possuiu maior valor de precisão@10 em 31,76% das consultas, provocando uma melhora de 18,97% em relação a estratégia TEXT de acordo com a comparação relativa. A Tabela 7 ilustra estes

dados.

Estratégia	% de consultas	CA_TEXT	CR_TEXT
Empate	49,41%	-	-
TEXT	18,82%	-	-
0.85TEXT_0.15AUT	31,76%	12,94%	18,97%

Tabela 7: Número de consultas com maior valor de precisão@10 para as estratégias TEXT e 0.85TEXT_0.15AUT. Para o cálculo dos valores das comparações relativa (CR_TEXT) e absoluta (CA_TEXT) o número de consultas empate foi adicionado às duas estratégias.

No conjunto das consultas teste, a estratégia 0.75TEXT_0.25AUT possuiu maior valor de precisão@10 em 35,29% das consultas, provocando uma melhora de 29,08% em relação a estratégia TEXT de acordo com a comparação relativa. A Tabela 8 ilustra estes dados.

stratégia	% de consultas	CA_TEXT	CR_TEXT
Empate	48,24%	-	-
TEXT	16,47 %	-	-
0.75TEXT_0.25AUT	35,29%	18,82%	29,08%

Tabela 8: Número de consultas com maior valor de precisão@10 para as estratégias TEXT e 0.75TEXT_0.25AUT.

No conjunto das consultas teste, a estratégia 0.65TEXT_0.35AUT possuiu maior valor de precisão@10 em 29,41% das consultas, provocando uma melhora de 11,66% em relação a estratégia TEXT de acordo com a comparação relativa. A Tabela 8 ilustra estes dados.

Estratégia	% de consultas	CA_TEXT	CR_TEXT
Empate	49,41%	-	-
TEXT	21,18%	-	-
0.65TEXT_0.35AUT	29,41%	8,23%	11,66%

Tabela 9: Número de consultas com maior valor de precisão@10 para as estratégias TEXT e 0.65TEXT_0.35AUT.

Através da análise destes dados foi constatado que as estratégias que obtiveram

maior valor médio da precisão@10 que a estratégia TEXT, possuíram um maior valor da precisão relativa para maioria das consultas com relação a estratégia TEXT.

6.3 Discussão dos Resultados

As avaliações mostraram que as estratégias 0.85TEXT_0.15AUT, 0.75TEXT_0.25AUT e 0.65TEXT_0.35AUT obtiveram uma significativa melhora na precisão relativa entre 10 primeiros documentos retornados (precisão@10) com relação a estratégia puramente textual (TEXT). De acordo com a comparação relativa, estes experimentos provocaram melhoras no valor da precisão@10 de 8,13%, 9,42%, 6,65% respectivamente. Isto nos revela que o uso das informações estruturais contidas na estrutura de *links* da *Web* atingiu os objetivos esperados.

As três estratégias enumeradas acima não chegaram a atribuir um coeficiente maior que 50% para o peso de autoridade (b>=0,5), pois o objetivo desta fusão de paradigmas foi utilizar o GHHITS como estratégia auxiliar ao ranqueamento baseado no paradigma textual, e não com o intuito de substituir este paradigma.

A única estratégia analisada na qual foi atribuído um valor de b maior que 0,5 foi a estratégia AUT (b=1, a=0). Por utilizar as informações estruturais contidas nos *links* como única fonte de informação, desprezando o conteúdo da página, esta estratégia mostrou quedas expressivas nos valores da precisão da resposta com relação à estratégia TEXT. Como dito anteriormente, este resultado já era esperado uma vez que ao ranquear as páginas apenas por critétios estruturais, perdem-se características das páginas importantes de serem analisadas, tais como: o *tfidf*, a distância entre os termos, a distância entre o termo consultado e o início do documento; características estas que são levadas em consideração pelos algoritmos de recuperação textuais.

Este comportamento justifica a afirmação dita no Capítulo 3, que diz que os algoritmos de *topic distillation* podem, muitas vezes, inserir ruídos ou até mesmo degradar a performance de recuperação dos sistemas.

Caso o algoritmo GHHITS levasse em consideração o conteúdo dos *links* (texto âncora) poderia ser atribuído um valor de b > 0,5, pois neste caso o peso de autoridade conteria informações textuais além de informações estruturais.

Através da análise da aceitabilidade das estratégias pode-se perceber que as estratégias que possuíram maior precisão@10 foram as mesmas que possuíram maior

"aceitabilidade", em comparação com a estratégia TEXT. De acordo com a comparação relativa, as estratégias 0.85TEXT_0.15AUT, 0.75TEXT_0.25AUT e 0.65TEXT_0.35AUT obtiveram uma melhora de 18,97%, 29,08% e 11,66% respectivamente com relação a estratégia TEXT.

Dentre todas as estratégias analisadas, a estratégia 0.75TEXT_0.25AUT foi a que apresentou os melhores resultados tanto na análise da "aceitabilidade" quanto com relação aos valores da precisão@10. A partir do momento em que o valor de b na fórmula de fusão ultrapassou 0,25, os valores da precisão@10 e da "aceitabilidade" começaram a cair. Tal comportamento pode ser devido ao fato de que quando é atribuído a b um valor maior que 0,25 está sendo dada muita "importância" a informações que devem ser consideradas auxiliares no momento do ranqueamento das páginas. Esta justificativa recorre aos mesmos argumentos utilizados para justificar a queda brusca nos valores da precisão quando se utilizou apenas as informações estruturais no momento do ranqueamento (AUT).

Uma vez que as informações sobre o conteúdo das páginas carregam as informações sobre a "relevância" (segundo a medida de similaridade textual) e as informações estruturais incorporam a noção de "autoridade" às páginas *Web*, chegamos à conclusão que as estratégias com valor de b maior que 0,25 retornavam páginas importantes mas que não eram relevantes ao tópico consultado, e as estratégias que possuiam b menor que 0,25 retornavam páginas relevantes (segundo a medida de similaridade textual) mas que não eram importantes para a consulta submetida.

A estratégia 0.75TEXT_0.25AUT representa, portanto, o equilíbrio entre as medidas de importância e relevância da fórmula de fusão, no cenário avaliado neste experimento. Atingindo desta forma o objetivo deste trabalho que corresponde a alterar a resposta do engenho de busca de forma que este passe a retornar páginas que além de relevantes são consideradas "importantes" para o tópico consultado.

6.4 Considerações Finais

Nesse capítulo foram apresentados experimentos realizados com o objetivo de verificar impacto na eficácia de recuperação de um sistema de recuperação ao passar a utilizar a informação contida nos links, sob a forma do peso de autoridade. Para a realização destes experimentos foi necessária a elaboração de um método de avaliação

que procurou atender às premissas, enumeradas na Seção 2.6. Chamamos o método proposto de **método Radix** de avaliação de estratégias de busca.

As métricas utilizadas nestes experimentos foram: a precisão relativa dos 10 primeiros documentos retornados e a precisão média TREC-Style dos 10 primeiros documentos retornados. Estes experimento mostraram que houve uma significativa melhora nos valores destas mátricas quando informações relativas a estrutura de links foram utilizadas em conjunto com informações textuais no processo de recuperação de informação.

7

Conclusões e Trabalhos Futuros

Como foi apresentado nesta dissertação, as características da Web - distribuição, heterogeneidade e dinamicidade - tornam a tarefa de encontrar documentos relevantes, quase sempre entediante e difícil. Para auxiliar o usuário a encontrar informações relevantes na *Web,* surgiram alternativas dentre as quais podemos citar os engenhos de busca, descritos ao longo deste trabalho.

Quando consultas por tópicos populares (tópicos que estão presentes em um grande conjunto de páginas *Web*) são submetidas aos engenhos de busca, a lista resposta retornada por estes sistemas muitas vezes é imensa. Nestes casos, o usuário do engenho de busca não visualiza todas estas páginas retornadas, restringindo sua análise aos 10 ou 20 primeiros documentos retornados.

Nestes casos seria interessante para o engenho de busca retornar nas primeiras posições da lista de documentos retornados, os documentos mais "importantes" sobre o tópico consultado.

Ante a incapacidade dos algoritmos de recuperação de informação puramente textuais, de associar um conceito de "importância" as páginas *Web*, os engenhos de busca vêm procurando por outras fontes de informação para serem utilizados conjuntamente com as informações textuais no momento do ranqueamento, e capazes de associar este conceito de "importância" às páginas *Web.* Dentre estas informações podem ser citadas: as meta *tags* presentes no documento; os *hiperlinks; a*s informações

associadas manualmente aos documentos por um conjunto de editores humanos contratados por diretórios *Web* (ex: Yahoo); e estatísticas de acesso, coletadas a partir do *log* de acesso dos engenhos de busca.

Dentre estas informações, a que se mostrou mais promissora foi a informação contida na estrutura de *hiperlinks*, uma vez que esta informação pode embutir a opinião dos usuários a respeito das páginas *Web* apontadas.

O objetivo principal deste trabalho, apresentado nos Capítulos 5 e 6, foi a elaboração de um algoritmo de análise de *links* capaz de associar um conceito de importância às páginas *Web,* e atender aos requisitos de tempo impostos pelos engenhos de busca comerciais. Para validar o algoritmo proposto, foi desenvolvido o Sistema para Armazenamento e Análise dos *Links* da *Web* (SAAL).

Para verificar o impacto na qualidade da resposta causado pela adição do peso de "importância" ao ranqueamento das páginas, foi realizado um estudo de caso que consistiu em acoplar o SAAL ao engenho de busca ***Radix,*** e realizar experimentos que tentaram encontrar a melhor combinação entre o peso de similaridade textual (preexistente no engenho de ***Radix***) e o peso de autoridade.

Nestes experimentos foram elaboradas e testadas quatro estratégias de fusão de paradigmas, com o objetivo de encontrar aquela que conduzisse a um maior impacto na qualidade da resposta. Este impacto foi calculado através de métricas de avaliação da eficácia em sistemas de RI, descritas na Seção 2.5.2: precisão@10, precisão@7, precisão@5, precisão PMTS@10, precisão PMTS@7, precisão PMTS@5.

Nas próximas seções, serão apresentadas as contribuições deste trabalho bem como as dificuldades e os problemas enfrentados durante o seu desenvolvimento. Para finalizar, serão mostradas as limitações da abordagem proposta e os trabalhos futuros que podem ser realizados a partir dela.

7.1 Contribuições

Todos os objetivos almejados neste trabalho, enumerados no Capítulo 1, foram alcançados:

(i) Foi realizado um estudo dos algoritmos de análise de *links* dependentes e independentes de consulta encontrados na literatura (Capítulo 3).

(ii) Os conhecimentos adquiridos nesse período foram de extrema importância para o desenvolvimento deste trabalho, uma vez que forneceram o embasamento teórico necessário para a elaboração do algoritmo GHHITS (Capítulo 4).

(iii) Foi implementado o Sistema para Armazenamento e Análise de *Links* – SAAL para validar o algoritmo proposto (Capítulo 4).

(iv) Foram concebitas estratégias de otimização do processamento que posteriormente foram aplicadas ao SAAL (Capítulo 5).

(v) Foi elaborada um fórmula de fusão (Capítulo 4).

(vi) Um sistema para avaliação da eficácia de recuperação foi implementado (Capítulo 5). Este sistema poderá ser utilizado posteriormente para calcular a precisão relativa entre sistemas diferentes ou entre o mesmo sistema quando aplicado a bases de dados ou estratégias de busca diferentes.

As estratégias que utilizaram a fusão de paradigmas - 0.85TEXT_0.15AUT, 0.75TEXT_0.25AUT, 0.65TEXT_0.35AUT - conseguiram uma significativa melhora da precisão@10. Isto nos revela que o uso das informações estruturais contidas na estrutura de *links* da *Web* atingiu os objetivos esperados.

Além das contribuições enumeradas acima, que fizeram parte do objetivo do trabalho, outra importante contribuição deste trabalho, consistiu na aplicação do algoritmo GHHITS (uma extensão do algoritmo HITS) a todo grafo da *Web* indexada e a análise de seu comportamento, uma vez que na literatura o HITS e suas extensões somente eram aplicados a um subconjunto de páginas formado a partir do conjunto resposta (grafo base).

Segundo palavras do próprio Kleimberg inventor do HITS em resposta a um e-mail enviado pela autora deste trabalho: "*hub*s e autoridades têm outro sentido quando aplicamos o algoritmo para todo grafo".

Pode ser citada ainda como contribuição deste trabalho o estudo detalhado das características de um subconjunto da *Web* contendo 3.742.983 milhões de páginas (11,14% da *Web* brasileira em janeiro de 2002). Neste subconjunto verificaram-se as seguintes características na estrutura de *links*:

- 1.384.398 páginas (27,69 % das páginas) possuíam *links* externos ao seu site;

- 35,13% páginas possuíam *backlinks* de *sites* diferentes do site ao qual pertence;
- As páginas com maior peso de autoridade são páginas que possuem baixa profundidade no site, geralmente, são as páginas home do *site.*;
- As páginas com maior peso de autoridade no Grafo Web analisado neste trabalho são páginas ".com". O Apêndice G traz uma lista contendo as100 maiores autoridades e os 100 maiores *hubs* do grafo *Web* analisado neste trabalho.

O conhecimento adquirido a partir da análise da estrutura de *links* deste subconjunto da *Web* indexada pelo engenho de busca ***Radix***, fornece uma visão geral da estrutura de *links* da *Web* brasileira.

Foi realizado um estudo das consultas submetidas ao engenho de busca ***Radix***, a partir do qual foi possível analisar o comportamento dos usuários e definir estratégias para melhorar a eficácia de recuperação do sistema. Como foi constatado que 52% das consultas submetidas ao sistema possuíam de 1 a 2 termos, optou-se por utilizar algoritmos de análise de *links* que segundo a literatura demonstraram ser comprovadamente eficazes quando utilizados para melhorar a eficácia de recuperação de consultas por tópicos gerais (de forma simplificada consideramos consultas gerais como sendo consultas compostas por 1 ou 2 termos). Este conhecimento também forneceu indícios quanto ao comportamento dos usuários da *Web* brasileira.

Durante toda a fase de desenvolvimento do SAAL, procurou-se construir um sistema de fácil manutenção, bem como possível de ser acoplado a qualquer engenho textual através de poucas adaptações. Para tanto, ele é composto por módulos, de modo a facilitar as tarefas referentes à programação e aos testes de cada um deles.

Outra contribuição deste trabalho é a possibilidade de se utilizar os pesos de autoridade nas políticas de indexação das páginas. Como o conceito de autoridade está relacionado à opinião coletiva dos usuários da *Web,* ele pode ser considerado um bom critério no momento de se escolher as páginas que devem ser indexadas ou atualizadas [NW01, DCL+00, CBD99].

7.2 Dificuldades Encontradas

A dificuldade inicial encontrada durante a realização dessa dissertação foi a falta

de documentação detalhada sobre os algoritmos de análise de *links* utilizados por engenhos de busca comerciais. Esta dificuldade foi devida à forma como as funções de ranqueamento são mantidas em segredo pelos engenhos de busca uma vez que fazem parte do diferencialdestes sistemas.

Contudo, pôde ser encontrada uma vasta literatura a respeito de sistemas não comerciais que detalhavam os passos seguidos na manipulação e posterior utilização das informações contidas na estrutura de *links* da *Web*.

Em adição ao problema acima citado, outra dificuldade derivou do processo chamado calibragem da função de *ranking*, que consiste em encontrar os valores de a e b da Fórmula (22) que fazem com que o sistema retorne o maior número de documentos relevantes para a maioria das consultas. Durante este processo foram avaliadas apenas cinco alternativas de *ranking* pelas razões que seguem:

- **Custo associado a etiquetagem:** executar um processo de etiquetagem manual das páginas consiste em um processo muito custoso pois necessita de voluntários dispostos a etiquetar manualmente um conjunto de páginas como relevantes ou irrelevantes.

- ***Measure Stability*:** o alto custo associado ao processo de avaliação das estratégias de forma que atendesse aos requisitos de *measure stability* definidos por Buckley e Voohees [BV00] e detalhados na Seção 3.6.2.

- **Julgamento Cego:** após a avaliação das cinco estratégias enumeradas no Capítulo 6 deste trabalho, não puderam ser realizados outros experimentos a não ser que fossem repetidas as primeiras avaliações, pois caso contrário, este experimento estaria violando o conceito de julgamento cego [CDG+98c], no qual se deve assegurar que a avaliação de todos os sistemas participantes deve ser realizada ao mesmo momento.

7.3 Limitações e Trabalhos Futuros

O trabalho aqui apresentado fornece uma solução adequada ao problema de precisão enfrentado pelos engenhos de busca com maior acuidade em consultas por tópicos gerais. Esta solução consistiu na utilização das informações contidas na estrutura de *links* em conjunto com as informações textuais no processo de recuperação de informações na *Web*.

No entanto, esta solução apresenta limitações, algumas delas relacionadas às restrições do modelo inicial proposto, outras relacionadas à necessidade de uma análise mais aprofundada do comportamento do algoritmo sob determinadas condições.

A seguir são enumeradas estas limitações juntamente com proposta de trabalhos futuros. Além disso, são enumerados trabalhos futuros relacionados a tarefas que, apesar de simples, não foram realizadas por estarem além do escopo deste trabalho.

7.3.1 Comunidades "*Web Centric*":

Após a convergência do algoritmo HITS, Lempel [LM00, BRR+01] comprovou que apenas as comunidades mais conectadas dominam o peso de autoridade. Como visto na Seção 4.4 o GHHITS está sujeito a mesma limitação.

Como solução para esta limitação, o módulo de processamento do GHHITS deverá ser alterado de tal forma a contemplar os seguintes passos:

1. Executar o GHHITS até a convergência.
2. Após a convergência, as páginas com peso de autoridade superior a MaxPesoAutoridade/2 são retiradas do grafo.
3. São atribuídos pesos de autoridade e *hub* para cada página extraída do grafo. Estes pesos devem ser inversamente proporcionais ao número da iteração do GHHITS. Esta proporcionalidade é representada da seguinte forma:

$$\text{RankAut}(p_j) = \text{PesoAutoridade}(p_j) * 1/N$$

Onde:

- N é o número da rodada.
- p_j é página excluída do grafo na rodada N.

4. Repetir os passos anteriores até que tenham sido atribuídos pesos de autoridade e *hub* para todas as páginas do grafo.

7.3.2 Analisar a Convergência de GHHITS

O sistema atual utiliza um critério de parada diferente do utilizado pelo algoritmo HITS. Como trabalho futuro poderia ser realizada uma análise mais

aprofundada do algoritmo GHHITS aplicado para todo o grafo da *Web* indexada, de forma a associar propriedades a este que possibilitasse prever o número de iterações necessárias para a convergência, assim como é previsto pelo PageRank, e pelo HITS e suas extensões.

Esta análise englobaria um estudo das características da estrutura de *links* da *Web* brasileira e do comportamento do algoritmo aplicado a este grafo. Esta análise deverá ser feita utilizando propriedades da álgebra linear relacionadas à computação de autovalores e autovetores.

7.3.3 Retornar Páginas não Indexadas na Resposta

Este trabalho consiste em uma nova forma de utilizar os pesos de *hub* e autoridade calculados pelo GHHITS, permitindo que páginas que não foram efetivamente indexadas sejam retornadas como resposta a uma consulta submetida ao engenho de busca. Chamamos de páginas efetivamente indexadas aquelas que foram visitadas pelos crawlers e enviadas para o servidor de indexação o qual é responsável por extrair os termos e os links da página Web.

Como visto na Seção 4.5, no momento em que uma consulta está sendo processada pelo Sistema de Consultas, este envia um conjunto de requisições ao Servidor de Pesos contendo os códigos das páginas que foram lidos pelo sistema. A resposta a esta requisição consiste em um *array* bidimensional contendo os pesos de hub e autoridade das páginas cujos códigos estavam presentes na requisição.

Como trabalho futuro, o *array_request* deveria ser alterado para conter apenas os códigos das 100 páginas melhor ranqueadas segundo o paradigma textual. O Servidor de pesos também seria alterado para passar a retornar o conjunto dos códigos das páginas com maior peso de autoridade referenciadas pelas 100 páginas contidas no *array_request*.

Os códigos de página retornados pelo Servidor de Pesos não corresponderiam necessariamente a páginas indexadas pelo sistema. Estas páginas poderiam ser mostradas em uma área da página resposta, contendo o seguinte texto descritivo: "Estas são as páginas consideradas autoridade pelas páginas da resposta."

7.3.4 Personalização das Autoridades e *Hubs*

Como visto na Seção 4.3 o algoritmo GHHITS utiliza os vetores InitAut e InitHub para representar a opinião de usuários, sobre o nível de importância que deve ser conferido a uma página ou a um conjunto de páginas durante o processo de análise de *links*.

Porém, nos experimentos realizados, neste trabalho, os vetores InitAut e InitHub possuíram todas as entradas iguais a 0, não sendo, portanto, verificadas as propriedades de personalização da resposta provocada pela utilização destes vetores.

Propõe-se como trabalho futuro, a utilização de diferentes valores de entradas para os vetores InitAut e InitHub. Tais valores confeririam mais ou menos importância a algumas páginas *Web* de acordo com a opinião um usuário ou uma comunidade de usuários. Através da execução do HHITS utilizando estes vetores seria possível analisar as propriedades de personalização deste algoritmo.

7.3.5 Comparação entre o HITS e o GHHITS

A utilização dos pesos de autoridade gerados pelo algoritmo GHHITS, quando combinadas com informações textuais, provocou uma significativa melhora na eficácia de recuperação do engenho de busca utilizado como estudo de caso.

No entanto, seria interessante realizar como trabalho futuro, uma comparação entre a qualidade da resposta gerada pelo algoritmo HITS (ou alguma de suas extensões) aplicado ao grafo resposta e o algoritmo GHHITS aplicado a todo o grafo da *Web* indexada.

Caso a qualidade da resposta do HITS (ou alguma de suas extensões) se mostre muito melhor que a provocada pela utilização do GHHITS. O HITS poderia ser utilizado em *batch* durante o processo de construção de uma *cache* contendo as consultas submetidas com mais frequência ao engenho de busca.

Neste cenário, o GHHITS continuaria sendo utilizado para as consultas que não pertencessem a cache, uma vez que o mesmo corresponde a um algoritmo de análise de *links run-time*.

7.3.6 Associar Contexto à Consulta

Como mencionado nas Seções 3.2.1.3 e 3.2.1.4 o conteúdo presente nos *links* e

nas suas proximidades (*janela de bytes*) podem conter informações que caracterizam a página melhor do que o seu próprio conteúdo. A estrutura de armazenamento de *links* implementada neste trabalho já permite o armazenamento e atualização eficientes das informações contidas nos *links* entre as páginas *Web*.

A estrutura do arquivo VIF contendo estas informações é mostrada na Figura 41.

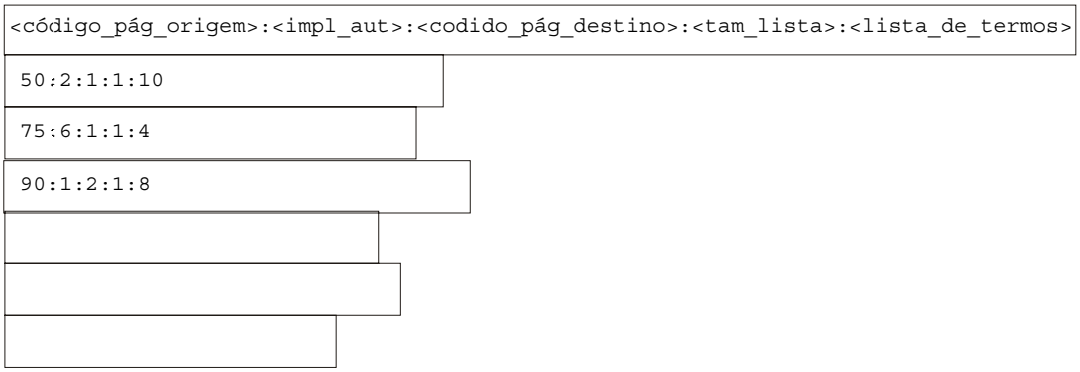

Figura 41: Arquivo VIF contendo o texto do link.

A lista de códigos representada na estrutura acima deverá armazenar dois valores: o primeiro corresponde à posição do termo no link, e o segundo corresponde ao código do termo. Estas informações devem ser armazenadas em um número inteiro contendo 32 bits, da forma mostrada na Figura 42.

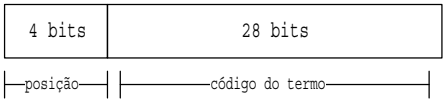

Figura 42: Representação do código no arquivo pertencente ao VIF.

Os processos de coleta e disponibilizarão destas informações devem ser alterados para passar a considerar estas informações. O processo de coleta necessita apenas que o *crawler* passe a enviar as informações sobre os termos do link - os termos do link correspondem aos termos existentes no interior o <a href> , ou nas suas redondezas de acordo com o conceito de janela de bytes explicado anteriormente - contendo as seguintes informações para o módulo de processamento:

<url_origem>|<link_destino1>:termo,posicao:termo,posicao|<link_destino1>:termo,posicao:termo,posicao |<link_destino1>:termo,posicao:termo,posicao

O processo de disponibilização consiste em inverter a estrutura descrita na Figura 41 de forma que a mesma passe a realizar o mapeamento do código do termo para códigos das páginas. Esta estrutura corresponde ao arquivo VIF original que será utilizado no processo de busca, da seguinte forma:

1. Quando uma consulta chega a interface do sistema, este repassa a consulta simultaneamente para dois sistemas de busca: um sistema que contém em sua base apenas os termos provenientes do conteúdo dos *link* que apontam para a página com exceção dos termos pertencentes a *stop-list*, e outro sistema que possui uma base contendo todos os termos da página com exceção dos termos pertencentes a *stop-list.*
2. É realizada a fusão das respostas retornadas por estes dois sistemas, e o resultado desta fusão é retornado para o usuário.

Desta forma, a lista de páginas resposta conterá além das páginas que possuem os termos da consulta em seu conteúdo, ou no texto do *link* das páginas que apontam para ela.

7.4 Considerações Finais

Nesse capítulo foram apresentadas as considerações finais sobre o trabalho desenvolvido. O trabalho em questão consistiu em propor e desenvolver um Sistema para Armazenamento e Análise de Links (SAAL) com o propósito de associar um peso de autoridade as páginas Web a partir da análise da estrutura de links. E utilizar esta informação em conjunto com o peso de similaridade textual com o objetivo de melhorar a eficácia de recuperação de um engenho de busca.

Referências Bibliográficas

[Ada99] L. A. Adamic, "The Small World Web", Proc. of ECDL99, 1999.

[AGM+00] A. Arasu, H. Garcia-Molina, J. Cho, A. Paepcke, and S. Raghavan, "Searching the Web", ACM Transactions on Internet Technology, 2001.

[ASG+00] E. Alonso, A. Silva, P. Golgher, R. Barra, A. Laender, B. Ribeiro-Neto, N. Ziviani, "Um Retrato da Web Brasileira", XXVII Simposio Brasileiro de Sistemas Integrados de Software and Hardware (SEMISH'2000), 2000.

[ATH00] B. Amento, L. Terveen, and W. Hill, "Does 'authority' mean quality? Predicting expert quality ratings on Web documents", Proc. of the 23rd Annual International ACM/SIGIR Conference, pp. 296-303, 2000.

[BBH+98] K .Bharat, A. Broder, M. Henzinger, P. Kumar, and S. Venkatasubramanian, "The Connectivity Server: Fast Access to Linkage Information on the Web", Proc. of 7th WWW Conference, pp. 469-477, 1998.

[BH98] K. Bharat and M. R. Henzinger, "Improved Algorithms for Topic Distillation in Hyperlinked Environments", Proc. of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pp. 104-111.

[BKM+00] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tompkins, and J. Wiener, "Graph Structure in the Web", Proc of the 9th WWW Conference, 2000.

[BM00] K. Bharat and George Mihaila, "When Experts Agree: Using Non-Affiliated Experts to Rank Popular Topics", Proc of the 9th WWW Conference, 2000.

[BMP+98] S. Brin, R. Motwani, L. Page, T. Winograd, " What can you do with the Web in your pocket?", Data Engineering Bulletin, 1998.

[BP98] S. Brin, L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", Proc. of 8th WWW Conference, pp. 107-117, 1998.

[BRJ99] G. Booch, J. Rumbaugh, I. Jacobson. "Unified Modeling Language – User Guide", Addison-Wesley, 1999.

[BRR+01] A. Borodin, J. S. Rosenthal, G. O. Roberts, P. Tsaparas, "Finding Authorities and Hubs From Link Structures on the World Wide Web" (commended paper), Proc. of the 10th World Wide Web Conference, 2001.

[BV00] C. Buckley and E. Voorhees, "Evaluating evaluation measure stability", Proc. of International ACM SIGIR (SIGIR'00), 2000.

[BYRN99] R. Baeza-Yates, B. Ribeiro-Neto, "Modern Information Retrieval", Addison Wesley, 1999.

[CBD99] S. Chakrabarti, M. van den Berg, and B. Dom, "Focused crawling: a new approach to topic specific Web resource discovery", Proc. of of 8th WWW Conference, 1999.

[CDG+98a] S. Chakrabarti, B. Dom, D. Gibson, S.R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins, "Spectral filtering for resource discovery", ACM SIGIR workshop on Hypertext Information Retrieval on the Web, 1998.

[CDG+98b] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, S. Rajagopalan, "Automatic resource list compilation by analyzing hyperlink structure and associated text", Proc. 7th International World Wide Web Conference, 1998.

[CDG+98c] S. Chakrabarti, B. Dom, D. Gibson, S.R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, "Experiments in topic distillation.", ACM SIGIR workshop on Hypertext Information Retrieval on the Web, 1998.

[CDG+99a] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, "Hypersearching the Web", Scientific American, June 1999.

[CDG+99b] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, "Mining the link structure of the World Wide Web", IEEE Computer, August 1999.

[CDK+99] S. Chakrabarti, B. E. Dom, R. Kumar, P. Raghavan, S. Rajagopalan, A. S. Tomkins, D. Gibson, and J. M. Kleinberg, "Mining the Web's link structure", IEEE Computer, 32(8), pp. 60-67, 1999.

[CDR+98] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleimberg, "Automatic resource compilation by analyzing hyperlink structure and associated text", Proc. 7th International World Wide Web Conference, 1998.

[Cha00] S. Chakrabarti. "Data mining for hypertext: A tutorial survey", ACM SIGKDD Explorations, 1(2), pp. 1-11, 2000.

[Cha99] S. Chakrabarti, "Recent results in automatic Web resource discovery", ACM computing survey, 1999.

[CHK+01] D. Callaway, J. Hopcroft, J. Kleinberg, M. Newman, S. Strogatz, "Are randomly grown graphs really random?", Physical Review E 64, (041902), 2001.

[CHR01] N. Craswell, D. Hawking, S. Robertson, "Effective site finding using link anchor information", Proc. of the 24th Annual International Conference on Research and Development in Information Retrieval, 2001.

[Dav00] B. D. Davison, "Recognizing nepotistic links on the Web", Artificial Intelligence for Web Search, Technical Report WS-00-01, pp. 23-28, AAAI Press, 2001.

[DCL+00] M. Diligenti, F. Coetzee, S. Lawrence, C. Lee Giles, and M. Gori, "Focused crawling using context graphs". Proc. of 26th International Conference on Very Large Databases (VLDB 2000), 2000.

[DGK+99a] B. D. Davison, A. Gerasoulis, K. Kleisouris, Y. Lu, H. Seo, W. Wang, and B. Wu, "DiscoWeb: Applying Link Analysis to Web Search", Poster Proc. of the Eighth International World Wide Web Conference, pp. 148-149, 1999.

[DGK+99b] B. D. Davison, A. Gerasoulis, K. Kleisouris, Y. Lu, H. Seo, J. Tian, S. Wang, W. Wang, and B. Wu, "An Early DiscoWeb Prototype at TREC8", Proc. of the Eighth Text REtrieval Conference (TREC-8), 1999.

[DH99] J. Dean and M. R. Henzinger, "Finding Related Web Pages in the World Wide Web", Proceedings of the 8th International World Wide Web Conference (WWW8), pp. 389-401, 1999.

[ERC+00] K. Efe, V. V. Raghavan, C. H. Chu, A. L. Broadwater, L. Bolelli, and S. Ertekin. "The shape of the Web and its implications for searching the Web". International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, 2000.

[FLG00] G. W. Flake, S. Lawrence, and C. L. Giles, "Efficient identification of Web communities", Proc. of the Sixth International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD-2000), pp. 150-160, 2000.

[FNL88] E. Fox, G Nunn, and W. Lee, "Coefficients of Combining Concept Classes in a Collection ", Proc. of the Eleventh International Conference on Research and Development in Information Retrieval, pp. 291-308, 1988.

[Gar72] E.Garfield, "Citation Analysis as a tool in journal evaluation", Science 178, pp. 471-479, 1972.

[GKR98a] D. Gibson, J. Kleinberg, P. Raghavan. "Inferring Web communities from link topology", Proc. 9th ACM Conference on Hypertext and Hypermedia, 1998.

[GKR98b] D. Gibson, J. Kleinberg, P. Raghavan, "Structural Analysis of the World Wide Web", Position paper at the WWW Consortium Web Characterization Workshop, 1998.

[GL96] G.H.Golub, C.F.V. Loan, "Matrix Computations", 3rd edition, 1996.

[GP99] M. Gordon, P. Pathak, "Finding information on the world wide Web: The retrieval effectiveness of search engines", Information Processing and Management, 35 (2), pp.141-180, 1999.

[GS00] C. Gurrin & A. Smeaton, "Connectivity Analysis Approach to Increasing Precision in Retrieval from Hyperlinked Documents", Proc. of the 8th Text Retrieval Conference, 2000

[Har94] D. Harmman, "Overview of the second text retrieval conference (TREC-2)", ARPA Workshop on Human Language Technology, pp. 328-334, 1994.

[Hav99] T. Haveliwala, "Efficient computation of PageRank", Technical report, Stanford University, 1999.

[HCB+01] D. Hawking, N. Craswell, P. Bailey, K. Griths, "Measuring search engine quality", Journal of Information Retrieval, published by Kluwer Academic, November 2001.

[HCT+99] D. Hawking, N. Craswell, P. Thistlewaite, and D. Harman, "Results and challenges in Web search evaluation", Proc. of the Eighth International World Wide Web Conference, pp. 243-252, 1999.

[Hen00] M. R. Henzinger, "Link Analysis in Web Information Retrieval", to appear in the IEEE Data Engineering Bulletin, November 2000.

[Hua98] L. Huang, "A Survey on Web Information Retrieval Technologies", 1998.

[Hun98] C. Hunt, "TCP/IP Administration", O`Reilly, 1998.

[IY99] E. J. Im, K. Yelick, "Optimizing sparse matrix vector multiplication on smps", Proc. of the Ninth SIAM Conference on Parallel Processing for Scientific Computing, 1999.

[KKR+99] J. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, "The Web as a graph: Measurements, models and methods". Invited

survey at the International Conference on Combinatorics and Computing, 1999.

[Kle97] J. Kleinberg, "Authoritative sources in a hyperlinked environment", Proc. 9th ACM-SIAM Symposium on Discrete Algorithms, 1998. Extended version in Journal of the ACM 46(1999). Also appears as IBM Research Report (RJ 10076), May 1997.

[Kle99] J. Kleinberg. Hubs, "Authorities, and Communities". ACM Computing Surveys 31(4), December 1999.

[Kob99] M. Kobayashi, "Information retrieval on the Web: Selected topics", IBM Research, Tokyo Research Laboratory, IBM Japan, Dec. 16, 1999.

[Kor97] R. R. Korfhage, "Information Storage and Retrieval". Wiley Computer Publishing, 1997.

[Kow97] G. Kowalski "Information Retrieval Systems. Theory and Implementation", Kluwer Academic Publishers, 1997.

[KRR+99] S.R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, "Trawling the Web for emerging cyber-communities", Proc. of the 8th World Wide Web Conference, 1999.

[KT00] M. Kobayashi and K. Takeda, "Information retrieval on the Web", IBM Research Report, (RT0347), April 2000

[KT99] J. Kleinberg, A. Tomkins, "Applications of linear algebra to information retrieval and hypertext analysis", tutorial survey at the ACM Symposium on Principles of Database Systems, 1999.

[Law00] S. Lawrence, "Context in Web Search", IEEE Data Engineering Bulletin, 23(3), pp.25-32, 2000.

[Lem99] R. Lempel, "Finding Authoritative Sites on the WWW (and other hyperlinked media) by Analyzing the Web's Link-Structure", MSc Thesis, Technion, Israel Institute of Technology, 1999.

[LG99a] S. Lawrence and C. L. Giles, "Accessibility of information on the Web", the journal Nature:, Nature, Vol. 400, pp. 107-109, 1999.

[LG99b] S. Lawrence, C. L. Giles, "Searching the Web: general and scientific information access," IEEE Communications Magazine, vol.37, no.1, pp. 116-122, Jan. 1999.

[LM00] R. Lempel and S. Moran, "The stochastic approach for link-structure analysis (salsa) and the tkc effect", Proc. of the 9th World Wide Web Conference, pp. 387-401, 2000.

[Low00] D. Lowe, "Improving Web Search Relevance: Using Navigational Structures to Provide a Search Context", (best paper at conference) Proc. of AusWeb2K - The Sixth Australian World Wide Web Conference, Cairns, 2000.

[Lyn97] C. Lynch, "Searching the Internet", Sci Am, pp. 52-56. 1997.

[Mar92] G. Marchionini, "Interfaces for End-User Information Seeking", Journal of American Society for information Science. 43 (2), pp. 156-163. 1992.

[Men00] A. Mendelzon, "Data (and Links) on the Web", slides from a presentation at the 11th NEC Research Symposium, Stanford University, 2000.

[Mir02] O. G. De Miranda. "Vertical Inverted File", trabalho de mestrando (em curso), no Centro de Informática da UFPE.

[MR00] A. O. Mendelzon, D. Rafiei, "What do the Neighbours Think?", Computing Web Page Reputations IEEE Data Engineering Bulletin, 2000.

[NW01] M. Najork, J. L. Wiener. "Breath-first Search Crawling Yields High-quality Pages", Proc. of the 10th World Wide Web Conference, 2001.

[NZJ01] A.Y. Ng, A. X. Zheng, M. I. Jordan, "Link analysis, eigenvectors, and stability", Proc. Int. Joint Conf. Artificial Intelligence, 2001.

[PBM+98] L. Page, S. Brin, R. Motwani, T. Winograd, "The PageRank citation ranking: Bringing order to the Web", Stanford Digital Library Working Paper, 1998.

[PdFR00] C. I. P. S. Pádua, N. M. da Fonseca, R. S. F. Resende, "Aspectos Dinâmicos dos Documentos da Web Brasileira", SEMISH , 2000.

[PN76] G. Pinski, F. Narin, "Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics", Proc. and Management 12, 1976.

[RM00] D. Rafiei and A. O. Mendelzon, "What is this Page known for? Computing Web Page Reputations", Proc. of the 9th International World Wide Web Conference, 2000.

[SEW] Search Engine Watch. URL: www.searchenginewatch.com.

[SHM+99] C. Silverstein, M. Henzinger, H. Marais and M. Moricz, "Analysis of a very large Web search engine query log", SIGIR Forum, 33(1), pp.6-12, 1999.

[Tol97] S. Toledo, "Improving the memory-system performance of sparse-matrix vector multiplication", IBM Journal of Research and Development, volume 41. IBM, 1997.

[TS92] J. Tauge-Sutcliefe, "The Pragmatics of Information Retrieval experimentation revisited", Information Processing and Management, 28:467-490, 1992.

[Voo98] E. M. Voohees, "Variations in relevance judments and the measurement of retrieval effectiveness", Proc. of SIGIR'98, pp. 315-323, 1998.

[Yan01] K. Yang, "Literature Review", Doctoral Dissertation Committee: Robert M. Losee, Gary Marchionini, Gregory B. Newby ,Paul Solomon, Ellen Voorhees. School of Information and Library Science University of North, January 2001

[YL96] B. Yuwono, D. Lee, "Search and Ranking Algorithms for Locating Resources on the World Wide Web," Proc. of the Twelfth International Conference on Data Engineering, 1996.

[ZG00] X. Zhu and S. Gauch, "Incorporating quality metrics in centralized/distributed information retrieval on the WWW", 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 288-295, 2000.

[Zob98] J. Zobel, "How Reliable are de Results os Large-Scale Information Retrieval Experiments", Proc. of Annual ACM-SIGIR Conference on Research and Development on Information Retrieval Melbourne, 1998.

[Ada99] L. A. Adamic, "The Small World Web", Proc. of ECDL99, 1999.

Apêndice A

Técnicas para Avaliação da Eficácia de Recuperação em Sistemas de RI

As técnicas para avaliação da eficácia em sistemas de RI, correspondem às formas utilizadas para medição dos valores enumerados na Seção anterior. Nesta Seção será respondida a questão enunciada na Seção 2 sobre como deve ser avaliado um sistema de RI.

As técnicas de avaliação da eficácia de sistemas de RI, podem ser classificadas em dois tipos: (1) técnicas que utilizam coleções de teste; (2) e técnicas que utilizam a *Web* como base de testes.

Uma coleção de testes consiste em um conjunto de documentos estático e bem definido, juntamente com um conjunto de consultas e um conjunto de julgamentos de relevância. Uma consulta corresponde a um conjunto de termos que representa a necessidade de informação do usuário.

O julgamento de relevância especifica quais documentos deveriam ser retornados em resposta a cada uma das consultas e quais não deveriam. As avaliações baseadas em tais coleções possibilitam o cálculo preciso das medidas de precisão e cobertura e podem ser reproduzidas, uma vez que a coleção de testes não muda ao longo do tempo.

O cálculo preciso da cobertura requer um conhecimento detalhado de todos os documentos contidos na coleção. Em sistemas que utilizam a *Web* como base de testes, este conhecimento da base torna-se impraticável, ante ao grande conjunto de dados disponíveis na *Web*, inviabilizando o cálculo preciso da cobertura. Nestes casos, os sistemas utilizam técnicas que objetivam chegar a um valor de cobertura próximo ao real, dentre elas podem ser citadas a amostragem aleatória e a técnica de *pooling,* que serão explicadas nas subseções seguintes.

1. Técnica de Amostragem Aleatória

A técnica de amostragem aleatória propõe que o processo de análise da relevância dos documentos não retornados, necessário ao cálculo da medida de

cobertura, seja realizado apenas em um subconjunto (amostra) da coleção de testes. Os resultados obtidos, através desta amostragem, devem, posteriormente, ser generalizados para toda a coleção. Por exemplo, se é realizado o julgamento de relevância em 5% dos documentos não retornados, e deste conjunto 20 documentos foram identificados como relevantes. Aplicando esta proporção para toda coleção, conclui-se que no total de documentos não retornados existiam 400 documentos relevantes.

O sucesso desta técnica está diretamente relacionado à escolha de uma amostra representativa, a qual deverá preservar a diversidade de documentos contidos na coleção. Uma amostra representativa em coleções muito grandes pode, muitas vezes, vir a ser também uma coleção muito grande ou em contrapartida pode conter pouquíssimos ou nenhum documento relevante.

2. Técnica de *Pooling*

Esta técnica foi proposta na *Text REtrieval Conference* (TREC), para a avaliação dos sistemas de RI a ele submetidos. A técnica de *pooling* se baseia na construção de um pool de documentos, que corresponde a uma coleção de documentos formada a partir da análise de relevância nos *n* primeiros documentos retornados quando um conjunto de consultas é submetido a cada sistema participante do TREC. O número *n* de documentos retornados e julgados é denominado de profundidade do *pool.* O valor da profundidade do *pool,* varia de acordo com o método de avaliação utilizado. Nas avaliações realizadas na TREC, o valor da profundidade do *pool* é igual a 100, em outros trabalhos [CDG+98,BH98] o valor de *n* varia entre 10 e 20 documentos.

Tomando-se como exemplo 20 sistemas participantes do TREC, inicialmente cada sistema é submetido a um conjunto de tópicos[10] e os 100 primeiros documentos retornados por cada sistema são julgados como relevantes ou não e inseridos no *pool*. O número de todos os documentos relevantes para uma dada consulta obtido neste processo corresponde ao número total de documentos relevantes do *pool*.

Harmman [Har94] afirma que o *overlapping* entre os sistemas participantes do

[10] Tópicos correspondem a necessidades de informação do usuário e que podem ser expressos através de consultas que utilizem vocabulários diferentes, mas que representem a mesma necessidade de informação.

pool é baixo. Um dos principais fatores, que dão suporte a esta afirmação, é o grande número de diferentes conjuntos de termos que podem ser utilizados para expressar uma necessidade de informação. Isto faz com que com que diferentes conjuntos de documentos sejam recuperados por cada sistema.

O valor de cobertura obtido através da técnica de *pooling,* para o k-ésimo sistema participante da construção do *pool* é calculado da seguinte forma:

$$Cobertura_k = \frac{|RA_k|}{\bigcup_{j=1}^{N} F\_ranking_p\ (RA_j)}$$

Onde:

- RA_k é conjunto de documentos relevantes retornados pelo k-ésimo sistema participante da construção do pool.
- $|RA_k|$ corresponde ao número de elementos deste conjunto.
- N é o número de sistemas participantes da construção do pool.
- p é a profundidade do pool.
- F_ranking é a função que retorna todos os documentos relevantes com posição menor ou igual a p, retornados pelo sistema j.

Após a construção do *pool*, qualquer sistema que não participou deste processo pode fazer uso dele para o cálculo de cobertura, desde que use os mesmos tópicos utilizados para a construção do *pool*, e a mesma coleção de testes. O valor de cobertura para sistemas não participantes da construção do pool é calculado da seguinte forma:

$$Cobertura\ y = \frac{|RA|}{\left|RPool_x\right|}$$

Onde:

- RA é o conjunto de documentos relevantes retornados pelo sistema y.
- |RA| o número de elementos do conjunto RA.
- x corresponde a consulta teste.
- RPool é o número de documentos relevantes e existentes no pool para consulta x.

Vale a pena salientar que esta técnica possui alguns problemas, quais sejam: (i) os documentos relevantes que não são retornados entre os 100 primeiros, são considerados irrelevantes por não estarem incluídos no *pool*; (ii) a confiabilidade dos resultados apresentados pelo TREC foi questionada em vários trabalhos dentre pode ser citado o trabalho realizado por Zobel [Zob98] que mostrou que há uma probabilidade de que o cálculo da cobertura para sistemas não participantes da construção do *pool* seja subestimado.

Apêndice B

Métodos de Avaliação da Eficácia de Recuperação de Sistemas de RI para a Web

1. Método utilizado por Brian, Loren e Will

Propósito:

Brian, Loren e Will estudaram a influência de métodos de análise de *link* na qualidade da resposta retornada por um único engenho de busca [ATH00].

Consultas teste:

Um estudo das consultas submetidas ao engenho de busca Magellan [ATH00] entre os anos de 1997 e 2000 mostrou que 42% das consultas submetidas eram relacionadas a entretenimento. Por esta razão neste estudo foram selecionados 5 tópicos sobre entretenimento.

Técnica de avaliação:

O conjunto resposta para cada um destes tópicos foi obtido a partir de diretórios do engenho de busca Yahoo[11] relacionados ao assunto abordado por estes tópicos. Numa primeira fase, foram selecionados 40 usuários de nível universitário. A tarefa destes usuários foi selecionar, via *browser*, os 15 documentos que melhor caracterizassem um tópico, isto é, os documentos que contivessem um conteúdo útil e compreensível para um usuário que deseja aprender sobre o tópico. Na segunda fase, foram alocados de 3 a 4 *experts* sobre cada tópico, tendo como tarefa, associar pesos de relevância aos 15 documentos selecionados, na fase anterior, numa escala entre 1 (o pior) e 7 (o melhor). As URLs foram apresentadas em ordem diferente para cada avaliador. As páginas com peso de relevância entre 5 e 7 foram considerados relevantes, e as demais foram consideradas irrelevantes para propósitos de cálculo.

Métricas e técnicas utilizadas:

[11] *Site*: http://www.yahoo.com

Foram utilizadas as seguintes métricas: precisão@5, precisão@10.

2. Método utilizado por Soumen Chakrabarti et al.

Propósito:

Comparação da eficácia entre 3 sistemas de recuperação de informação [CDG+98a].

Consultas teste:

Foram utilizados 26 tópicos extraídos do *log* do engenho de busca Altavista[12].

Técnica de avaliação:

Os 26 tópicos foram submetidos aos engenhos de busca analisados no experimento, as 10 primeiras respostas provenientes de cada sistema foram inseridas em uma única lista resposta e posteriormente ordenadas em ordem alfabética de acordo com seus títulos. Esta lista foi submetida a 26 voluntários os quais classificaram as páginas retornadas de acordo com a seguinte escala de relevância: ruim, regular, boa, fantástica. As páginas boas e fantásticas foram classificadas como relevantes e as demais como irrelevantes para propósitos de cálculo. O peso de relevância associado a um documento consistiu no peso associado a este documento pela maioria dos avaliadores.

Métricas e técnicas utilizadas:

precisão@10 e cobertura relativa.

3. Método utilizado por Bharat e Henzinger

Propósito:

Bharat e Henzinger [BH98] fizeram uma análise comparativa da influência na qualidade da resposta de 8 algoritmos de análise de *link.* Estes algoritmos de análise de *link* eram baseados nos conceitos de página autoridade (uma página que é muito referenciada por outras páginas) e página *hub* (uma página que possui em seu conteúdo

[12] *Site*: http://www.altavista.com

uma lista de páginas autoridade). Estes conceitos que são explicados detalhadamente na Seção 4.3.1.

Consultas teste:

Foram utilizados 28 tópicos extraídos do *log* de acesso do engenho de busca Altavista[13] entre os anos de 1997 e 2000. Uma base foi construída para armazenar os conjuntos resposta a estas consultas.

Técnica de avaliação:

Os 28 tópicos foram submetidos à base construída utilizando-se cada um dos 8 algoritmos. As 14 primeiras respostas provenientes de cada algoritmo foram inseridas de forma aleatória em um única lista resposta e submetidas à avaliação por 3 ou 4 voluntários os quais classificaram as páginas retornadas de acordo com a seguinte escala de relevância: irrelevante, relevante, relevante e bom *hub* ou relevante e boa autoridade.

As páginas que foram classificadas ao mesmo tempo como relevante e bom *hub* ou como relevante e boa autoridade foram consideradas relevantes para efeitos de cálculo. O peso de relevância associado a um documento foi o peso associado a este documento pela maioria dos avaliadores.

Métricas e técnicas utilizadas:

precisão@5, precisão@10 e cobertura relativa em listas de 5 e 10 documentos retornados

4. Método utilizado por Hawking e Craswell

Propósito:

Hawking e Craswell [HCB+01] fizeram uma análise comparativa da eficácia de 20 engenhos de busca e 8 algoritmos de análise de *link* baseados nos conceitos de página *hub* e página autoridade.

Consultas teste:

Foram utilizados 54 tópicos extraídos do *log* de acesso dos engenhos de busca

[13] *Site*: http://www.altavista.com

Altavista e Electric Monk[14].

Técnica de avaliação:

Os 54 tópicos foram submetidos a cada um dos engenhos de busca. As primeiras 20 respostas válidas retornadas para uma dada consulta, por cada engenho avaliado, foram inseridas de forma aleatória em uma única lista resposta e submetidas à avaliação por um voluntário e classificou-as como: irrelevantes ou relevantes.

Métricas e técnicas utilizadas:

Precisão média TREC-*Style* e precisão@20

A seguir é mostrada uma tabela contendo uma comparação entre métodos de avaliação de eficácia de recuperação de sistemas de busca detalhados acima: Brian, Loren e Will, Bharat e Henzinger, Chackrabarti et al e Hawking e Craswell.

Tabela B1: Comparação entre métodos de avaliação de eficácia de recuperação de sistemas de busca. Notas: [a] - deste conjunto de documentos são retirados os inacessíveis e os documentos não escritos em inglês. [b] - Os conceitos relativos a páginas *hub* e páginas autoridade são explicados na Seção 4.3.1.

	Brian, Loren e Will	**Bharat e Henzinger**	**Chackrabarti et al.**	**Hawking e Craswell**
Detalhes				
· Número de engenhos avaliados	1	1	3	20
· Número de alternativas de ranking para um mesmo engenho	3	8	1	1
· Ano das consultas submetidas	1999	1999	1999	1999
Necessidades de informação				
· Genuínas	sim	sim	sim	sim
· Abrangência	entretenimento	geral	geral	geral
· Como a consulta foi gerada	extraídas de *logs* da *Web*	extraídas de *logs* da *Web*	extraídas de *logs* da *Web*	extraídas de *logs* da *Web*
· Número de consultas utilizadas	5	28	26	54
Resposta				

[14] *Site*: http://electricmonk.com/.

· Documentos avaliados por consulta	15 primeiros	14 primeiros [a]	10 primeiros (5 *hub*s e 5 autoridades)[b]	primeiros 20 documentos acessíveis
· Utilização de operadores booleanos	não	não	não	não
· Utilização da técnica de *pooling* para o cálculo da cobertura relativa.	não	sim (profundidade=10)	sim (profundidade=10)	não
Como as páginas foram julgadas				
· Por quem	universitários sem conhecimentos específicos sobre RI	não especificado	universitários sem conhecimentos específicos sobre RI	universitários sem conhecimentos específicos sobre RI
· Pela pessoa que elaborou a consulta	Não	não	não	não
· Número de avaliadores	40	3	37	6
· Mais de um avaliador por consulta	sim	não	sim	não
· Conceito de relevância conjunto	opinião da maioria	opinião da maioria	opinião da maioria	apenas 1 usuário
·Julgamento cego	sim	sim	sim	sim
· Proibido seguir *hiperlinks* da página	não	não	não	sim
· Apresentação	*browser*	*browser*	*browser*	interface textual
· Níveis de relevância	primeira fase: binário - (relevante, não relevante); segunda fase: 7 níveis em escala crescente de relevância.	3 níveis (não relevante, relevante, relevante e bom *Hub* ou Autoridade)[b]	4 níveis em escala crescente de relevância	binário (relevante, não relevante)
· Níveis considerados realmente relevantes, para efeitos de cálculo, quando existe uma escala de níveis	5,6,7	-	3,4	-
· Medidas de eficácia utilizadas	precisão@5 e precisão@10	precisão@5 e precisão@10 e cobertura relativa em listas de 5 e 10 documentos retornados	precisão@10 e cobertura relativa em listas de 5 e 10 documentos retornados	Precisão média TREC-*Style* e precisão@20

Apêndice C

Heurísticas de Filtragem

Na construção do grafo *Web*, procura-se detectar os casos em que as páginas são apontadas por outras razões além do mérito. Este processo corresponde à aplicação de um conjunto de heurísticas ao grafo *Web*, e é chamado de processo de filtragem.

Várias características do ambiente *Web* dificultam o processo de filtragem [Dav00], dentre elas podem ser citadas:

- **Menu de Navegação**: os menus de navegação são utilizados pelos *sites* para facilitar a navegação dos usuários. Existem *sites* que utilizam um menu navegacional, e separam o seu conteúdo em URLs com *host-names* distintos. Intencionalmente ou não, como conseqüência desta estrutura, muitos *links* escapam de heurísticas simples de reconhecimento de *links* intrínsecos, baseadas no *matching* de *host-name* entre URLs. Por exemplo o site da www.tucows.com.br possui um menu navegacional que aponta para páginas com *host-names* diferentes, dentre estes *links* podem ser citados: news.tucows.com.br, e help.tucows.com.br.

- ***Spamming* baseado em *links***: alguns *Web* designers por saberem que determinados engenhos de busca levam em consideração a estrutura de *links* entre as páginas, no momento do ranqueamento, constroem várias páginas apontando umas para as outras. Este procedimento pode inflacionar o peso de ranking das páginas em engenhos de busca que levam em consideração informações estruturais no momento do ranqueamento. Contudo, este tipo de *spamming* apesar de possível, é bem mais difícil de ser realizado que o *spamming* baseado no conteúdo textual explicado na Seção 2.4.2.

A seguir são enumerados dois exemplos de heurísticas simples e seus problemas [Dav00]:

Marcar *links* entre páginas com o mesmo *host-name* como internas: Esta heurística exclui *links* entre *home pages* de usuários diferentes em um mesmo site e preserva *links* entre diferentes *hots* em um mesmo domínio;

Marcar páginas contendo o mesmo domínio como intrínsecas: Esta heurística

elimina *links* entre *home pages* de usuários diferentes, como no caso acima. Como também elimina *links* entre departamentos de uma mesma organização, ao menos que estes *sites* estejam em servidores diferentes.

Ambas soluções permitem que *links* de propaganda pertencentes a diferentes domínios afetem a análise de *links*. Existem várias alternativas para combater estes problemas, dentre elas podem podemos citar as *stop-pages* que correspondem a listas contendo as páginas que supostamente correspondem a *links* de propaganda por possuírem uma quantidade excessiva de *links* de entrada.

Apêndice D

Definição de autovalores e autovetores

Seja M a matriz simétrica n x n, um autovalor de M é o número λ para o qual existe algum vetor ω,obedecendo a seguinte propriedade: $M\omega = \lambda\omega$. O conjunto de todos vetores ω para um autovalor λ constitui um subespaço de R^n, a dimensão deste subespaço é chamada de multiplicidade do autovalor λ. [GL96].

Uma matriz M possui no máximo n autovalores distintos. Pode-se denotar os autovalores associados a Matriz M por λ_1 (M), λ_2 (M), λ_3 (M),...,λ_n (M), ordenados decrescentemente de acordo com o seu valor absoluto, e sendo citado um número de vezes igual a sua multiplicidade. Para cada autovalor distinto é escolhida uma base ortogonal do autoespaço correspondente, de tal forma que para cada autovalor λ_i (M) é obtido um autovetor ω_i (M) que quando ordenados decrescentemente obedecem a mesma ordenação estabelecida para os autovalores correspondentes: ω_1 (M), ω_2 (M),...,ω_n (M).

Por questões de simplicidade assumiu-se que:

$$\left|\lambda_1 (M)\right| > \left|\lambda_2 (M)\right|$$

A partir do que foi assumido acima, nos referimos a ω_1 (M) como o autovetor principal, e todos os demais autovetores como autovetores não principais.

Apêndice E

Análise da convergência do algoritmo HITS

Para provar que a computação de *hubs* e autoridades converge à medida que o número de iterações aumenta [Kle97, KT99], temos:

Sendo G=(V,E) o grafo contendo as n páginas do grafo base e os *links* entre elas. Sendo M a matriz de adjacência $M_{n \times n}$, onde cada entrada (i,j) é igual a 1 se existe um *link* partindo do nó i e chegando ao nó j, ou 0 caso contrário, e a e h os vetores que armazenam os pesos de autoridade e *hub* respectivamente e que e contém uma entrada para cada página do grafo base. As equações (1)(2) para o cálculo dos pesos de *hub* e autoridade podem ser reescritas da seguinte forma:

$$a^{(t)} = M^T h^{(t-1)} = (M^T M) a^{(t-1)}$$

$$h^{(t)} = M a^{(t)} = (M M^T) h^{(t-1)}$$

Onde:

- t é o identificador da iteração corrente.
- t-1 é o identificador da iteração anterior.
- $a^{(t)}$ é um vetor que contém os pesos de autoridade das páginas do grafo na iteração t. Este corresponde a um vetor unitário[15] na direção do vetor $(M^T M)\, a^{(t)}$
- $h^{(t)}$ é um vetor que contém os pesos de hub das páginas do grafo na iteração t. Este corresponde a um vetor unitário na direção do vetor $(M\ M^T)h^{(t)}$.
- M^T é a transposta da matriz de adjacência.
- k é o identificador da iteração.

A Álgebra linear [GL96, NZJ01] estabelece que se M é uma matriz simétrica n x n e v é um vetor não ortogonal ao autovetor principal ω1 (M), então um vetor unitário da direção de Mk x v converge para o autovetor principal se k aumentar sem limites. Também (como corolário), se M não possui entradas negativas, o autovetor principal

[15] Um vetor unitário é aquele que possui o módulo igual a 1.

não possuirá entradas negativas.

Conseqüentemente, se $h^{(t-1)}$ é um vetor não ortogonal ao autovetor ω1 ($M\ M^T$) o vetor unitário $h^{(t)}$ converge para o autovetor principal. De forma similar, podemos mostrar que sendo $a^{(t-1)}$ não ortogonal ao autovetor principal ω1 ($M^T\ M$), o vetor unitário $a^{(t)}$ converge para o autovetor principal.

- De acordo com experimentos realizados por Kleimberg [Kle97], com o valor de c variando entre 5 e 10 este algoritmo converge em 20 iterações. Quando os vetores são inicializados com 1 [1,1,1,1....], estas iterações sucessivas correspondem ao *power method* [Lem99], procedimento utilizado para a obtenção do autovetor principal de uma matriz. Neste caso o vetor a é o autovetor da matriz ($M^T\ M$) e de forma semelhante o vetor h é o autovetor da matriz ($M\ M^T$).

Apêndice F

Análise da convergência do algoritmo PageRank

De forma simplificada o PageRank também pode ser expresso através do cálculo do autovetor principal da matriz M quadrada, estocástica[16], correspondente ao grafo direcionado G=(V,E), onde V corresponde a todas as páginas que possuem pelo menos um *link* de saída, e E corresponde a todos os *links* entre estas páginas.

Se existe um *link* partindo da página j para a página i, existirá uma entrada m_{ij} na matriz M com valor $w_{ij,}$ calculado da seguinte forma:

$$w_{ij} = \begin{cases} 0 & \text{não há um link da página } i \text{ para a página } j \text{ ou} \\ 1/N_j & \text{caso contrário.} \end{cases}$$

Onde:

Nj corresponde ao número de *links* contidos na página j.

Uma iteração do algoritmo PageRank corresponde a multiplicação entre a matriz M com o vetor *Rank,*(M x *Rank*). O vetor *Rank* possui uma entrada para cada página do grafo. Segundo o *power method* [Lem99], repetições sucessivas desta multiplicação dará origem ao autovetor principal *Rank** da matriz M [GL96]. Como a matriz M é uma matriz estocástica, o autovalor principal desta matriz é 1 [PBM+98, GL96], o que leva a seguinte igualdade:

$$Rank^* = M \times Rank$$

O desvio desta igualdade corresponde a uma estimativa de erro, que pode ser utilizada para se analisar a taxa de convergência do algoritmo. Após um determinado número e iterações o valor módulo do *Desvio* se aproxima de zero, isso significa que a computação está próxima da convergência.

$$Desvio\,i = Rank_{i+1} - Rank\,i$$

Outra forma de se analisar a convergência deste algoritmo é através da ordem

[16] A matriz estocástica possui o somatório dos valores de cada coluna igual a 1.

global induzida pelo vetor de pesos PageRank, a qual apresenta uma visão intuitiva da taxa de convergência.

Como o algoritmo PageRank está interessado na ordem induzida entre as páginas a análise da convergência pode ser feita através de histogramas contendo a diferença entre as posições das páginas em duas ordenações, originadas por iterações distintas.

Ao se analisar a mudança na ordem das páginas a medida em que sucessivas iterações vão sendo realizadas, o PageRank está mais interessado com as páginas que aparecem entre as primeiras posições. A página menos importante ou a segunda menos importante segundo esta ordem induzida, muito provavelmente são páginas irrelevantes, não devendo, portanto, participar da análise. Esta análise de convergência leva em consideração apenas as páginas que aparecem entre as posições 1 e 1.000.000 em pelo menos uma das iterações analisadas. Experimentos [PBM+98, Hav99, NZJ01] demonstraram que o PageRank converge em 100 iterações.

Apêndice G

Características do Grafo *Web* Analisado

Tamanho do Grafo Web	Pesos de *autoridade* diferentes de zero	% do grafo com valor de autoridade diferente de zero	Média	Mediana	Valor de autoridade Máximo	Valor de autoridade Mínimo
2.490.368	58254	2,34%	0.001586	1.15E-06	0.001586	1.2E-303

Tabela G. 1: Características do pesos de autoridade calculados para as páginas pertencente ao Grafo *Web* utilizado neste experimento.

AUTORIDADES	
http//games.tucows.com	0.466631546
http//tukids.tucows.com	0.463950168
http//pda.tucows.com	0.463494525
http//ispcentral.tucows.com	0.424176002
http//advertise.tucows.com	0.422874283
http//www.tucows.com	0.421853198
http//shop.tucows.com	0.419324227
http//news.tucows.com	0.419310215
http//kickoff.tucows.com	0.418126062
http//www.domaindirect.com	0.418125287
http//lwn.tucows.com	0.416816721
http//content.tucows.com	0.414445713
http//www.opensrs.org	0.414239305
http//submit.tucows.com/contactus.html	0.414139548
http//about.tucows.com	0.414116891
http//submit.tucows.com	0.414116891
http//mailman.tucows.com	0.4107901
http//radio.tucows.com	0.409348117
http//www.themeoftheday.com	0.408500346
http//openxrs.tucows.com	0.391584251
http//partner.tucows.com	0.391525536
http//www.analogx.com	0.231172172
http//www.coffeecup.com	0.226257705

http//www.bluedomino.com	0.206301103
http//www.mcafee.com	0.181659901
http//www.webgenie.com	0.152086614
http//www.tucows.com/privacy.html	0.145884559
http//www.netscape.com	0.144973188
http//tucows.idirect.com/warn.html	0.140660479
http//www.americansys.com	0.136817151
http//legacysoftware.cjb.net	0.124134327
http//www.vanillasoftware.com	0.124046573
http//www.bee.net/dutch/index.html	0.123965804
http//www.emulive.com	0.123775719
http//www.webroot.com	0.123324183
http//www.acdsystems.com	0.121654977
http//www.microsoft.com/ie	0.120183526
http//www.ulead.com	0.120154205
http//www.rudenko.com/3dwb.html	0.119996909
http//www.offsitelabs.com	0.11967346
http//www.microsoft.com/Windows/IE/WebAccess/default.asp	0.115707822
http//www.desksoft.com	0.11567379
http//www.multimania.com/kinkodev	0.113483471
http//perso.wanadoo.fr/pierre.g/indexgb.html	0.111749006
http//members.xoom.com/m507	0.111579054
http//www.workgroupmail.com	0.111372679
http//www.speedresearch.com/stockwatch	0.111274519
http//www.desoft.co.uk	0.111262565
http//www.vista.ru	0.111239854
http//www.rudenko.com/shellnet.html	0.10740662
http//hp.vector.co.jp/authors/VA002416/teraterm.html	0.105996238
http//www.zip.com.au/~roca/ttssh.html	0.105857013
http//html.tucows.com	0.103623854
http//www.filetopia.com	0.103468476
http//sh39.net	0.103137292
http//open-sft.com/ariadne	0.103104243
http//www.sdisw.com	0.102996731
http//www.matchware.net	0.102863344
http//www.disco-software.com	0.102822983
http//www.icq.com	0.100547161
http//www.gizmosoft.com/merchant.html	0.098993687
http//www.broadscape.com	0.098940669
http//www.itserv.com	0.098551896
http//www.pixel-group.com	0.098235267
http//news.tucows.com/windows/tucows07182000.html	0.097433884
http//www.centered.com/index.html	0.097433884
http//home.wxs.nl/~ruurdb	0.096779953
http//www.geocities.com/SiliconValley/Network/1027	0.096768492
http//office.tucows.com	0.096679849
http//www.vandyke.com	0.096182063
http//www.tucows.com/color95.html	0.09612124
http//www.symantec.com	0.095199871
http//www.tucows.com/search.html	0.094856167
http//www.infoseccorp.com	0.094695107

http//www.sitemonster.com	0.09467563
http//www.cypressnet.com	0.094635761
http//www.nearsoftware.com/alligator	0.094586119
http//www.e-motional.com/creator.htm	0.094574727
http//www.wegroup.org	0.094565446
http//www.bigredh.com	0.094544351
http//www.kburra.com	0.094484372
http//www.illumix.com	0.094416659
http//www.tucows.com/preview/48941.html	0.091165002
http//www.xtreeme.com/sitexpert	0.090464171
http//www.icqroaming.com	0.090376558
http//www.soft-trade.com/?a=avapp&appid=220&	0.090363901
http//www.funduc.com	0.090303922
http//www.netcplus.com	0.090164698
http//gpsoft.hypermart.net	0.090160972
http//www.maccasoft.com	0.090096563
http//www.firstexplorer.com	0.090070687
http//www.logipole.com	0.090032506
http//www.f-secure.com	0.09001774
http//www.mochasoft.dk	0.088426506
http//www.ifi.uio.no/~staalesc/PGP	0.086574193
http//web.mit.edu/network/pgp.html	0.086574193
http//home.netscape.com/download/1019100/10002-en-win32-4.08-base-128_qual.html	0.086429836
http//www.netturbo.com	0.08625686
http//www.htmlworkshop.com	0.086214671
http//www.metaproducts.com	0.086172693

Tabela G. 2: Maiores autoridades do grafo Web utilizado neste trabalho.

AUTORIDADES .br	
http//acd.ufrj.br/~cracky/w3e	0.016130027
http//www.estado.com.br	0.011578548
http//www.oglobo.com.br	0.011312194
http//www.jb.com.br	0.011067497
http//www.cade.com.br	0.009709214
http//www.megavision.com.br	0.009182885
http//www.dpnet.com.br	0.008824302
http//www.dou.gov.br	0.008770862
http//www.ibm.com.br/~aviram/software/bip	0.00806867
http//www.dgabc.com.br	0.007545971
http//www.diariodonordeste.com.br	0.007470031
http//www.diario.com.br	0.007273676
http//www.zeek.com.br	0.007263292
http//www.gazeta.com.br	0.007164592
http//www.atarde.com.br	0.007118455
http//www.jornaldocommercio.com.br	0.007111951
http//www.opopular.com.br	0.007111951
http//www.opovo.com.br	0.006985383
http//www.acritica.com.br	0.006985383

http//www.jt.com.br	0.006701486
http//www.estaminas.com.br	0.006667031
http//www.santa.com.br	0.006643124
http//www.folhaweb.com.br	0.006635213
http//www2.uol.com.br/JC	0.0066099
http//www.redegazeta.com.br	0.0066099
http//www.mdnet.com.br/agazeta	0.0066099
http//www.diariopopular.com.br	0.0066099
http//www.oliberal.com.br	0.0066099
http//www.imesp.com.br	0.006478421
http//www.atribuna.com.br	0.006158476
http//www.openline.com.br/~jpb	0.005933819
http//www.openline.com.br/~onorte	0.005782993
http//www.openline.com.br/~db	0.005782993
http//www.folhadamanha.com.br	0.005782993
http//www.unicamp.br	0.005777801
http//www.in.gov.br	0.005756987
http//www.puc-rio.br	0.00574194
http//www.tribunadonorte.com.br	0.005406104
http//www.zh.com.br	0.005406104
http//www.tribunademinas.com.br	0.005406104
http//www.an.com.br	0.005406104
http//www.cnpq.br	0.00520438
http//www.unb.br	0.005130901
http//www.usp.br	0.005094688
http//www.correiobraziliense.com.br	0.005076758
http//www.ufrgs.br	0.005061992
http//www.agestado.com.br	0.004907298
http//www.ufg.br	0.004825733
http//www.ufes.br	0.004800419
http//www.ufsc.br	0.004800419
http//www.uol.com.br/fsp	0.004677486
http//www.cpopular.com.br	0.004588102
http//www.diariodopovo.com.br	0.004581036
http//www.otempo.com.br	0.004581036
http//www.uol.com.br/odia	0.004579197
http//www.ufrj.br	0.004564863
http//www.conectiva.com.br	0.004555371
http//www.mct.gov.br	0.004545597
http//www.correioweb.com.br	0.004544215
http//www.diariodesorocaba.com.br	0.004520835
http//www.ufpi.br	0.00450382
http//www.ufscar.br	0.004478376
http//www.ufv.br	0.004455875
http//www.ufmg.br	0.004358137
http//www.linuxmall.com.br	0.004323331
http//www.ufop.br	0.004281493
http//www.ufpe.br	0.004277274
http//www.folha.com.br	0.004263682
http//www.surf.com.br	0.00422946
http//www.ita.cta.br	0.004211881

http//www.mec.gov.br	0.004162731
http//www.uff.br	0.004154926
http//www.ufrn.br	0.004125393
http//www.pucsp.br	0.004107111
http//www.gazeta-oam.com.br	0.004092791
http//www.ufba.br	0.004074766
http//www.itau.com.br	0.004059297
http//www.primeiramao.com.br	0.004034335
http//www.ime.usp.br/~massaro/loser	0.004025897
http//www.dcc.unicamp.br/~guazzibe/blackbook	0.004025897
http//www.if.ufrj.br/~lsk/akount/index.html	0.004025897
http//www1.cptec.inpe.br/~sene	0.004025897
http//www.momentus.com.br/users/hook/kpackviewer.html	0.004025897
http//www.monamb.furg.br/pgwebforum	0.004025897
http//www.ufmt.br	0.004024139
http//www.mobile.com.br/palm/games	0.003870149
http//www.esw.com.br	0.003870149
http//www.bradesco.com.br	0.003826905
http//www.enter-net.com.br/diariodaamazonia	0.003803525
http//www.ufpb.br	0.003786825
http//www.correiodopovo.com.br	0.003730222
http//www.idg.com.br	0.003630374
http//www2.uol.com.br/observatorio	0.00361216
http//www.ufsm.br	0.003592756
http//www.minc.gov.br	0.003585794
http//www.unesp.br	0.003581154
http//www.lancenet.com.br	0.003561736
http//www.epoca.com.br	0.003523319
http//www.pucpr.br	0.003522089
http//www.fazenda.gov.br	0.003430679

Tabela G. 3: Maiores autoridade ".br" do grafo *Web* analizado. A maior autoridade ".br" apareceu na colocação 647 da lista contendo todas as autoridades. As colocações superiores foram ocupadas por páginas ".com".

Tamanho do Grafo	Pesos de *hub* diferentes de zero	% do grafo com valor de hub diferente de zero	Média	Mediana	Valor de *hub* Máximo	Valor de *hub* Mínimo
2.490.368	68573	2,75%	0.003351	0.003951	0.005302	3.6E-306

Tabela G. 4: Características do pesos de *hub* calculados para as páginas pertencente ao Grafo *Web* utilizado neste experimento.

HUBS

http//tucows.silcons.com.br/adnload/143826_48701.html	0.005204029
http//tucows.alternex.com.br/winnt/adnload/1297_28645.html	0.00519981
http//tucows.atarde.com.br/winnt/adnload/3900_30094.html	0.00519981
http//tucows.alternex.com.br/win2k/adnload/38448_29037.html	0.00519981
http//tucows.alternex.com.br/adnload/5029_29892.html	0.00519981
http//www2.carrier.com.br/win2k/adnload/149778_49716.html	0.005195802
http//tucows.triang.com.br/win2k/adnload/149778_49716.html	0.005195802
http//tucows.vcnet.com.br/win2k/adnload/149778_49716.html	0.005195802
http//www.tucows.uai.com.br/win2k/adnload/149778_49716.html	0.005195802
http//tucows.sigmabbs.com.br/win2k/adnload/149778_49716.html	0.005195802
http//www.tucows.via-rs.com.br/win2k/adnload/149778_49716.html	0.005195802
http//tucows.yawl.com.br/win2k/adnload/149778_49716.html	0.005195802
http//tucows.infolink.com.br/win2k/adnload/149778_49716.html	0.005195802
http//tucows.canal13.com.br/win2k/adnload/149778_49716.html	0.005195802
http//tucows.cmg.com.br/win2k/adnload/149778_49716.html	0.005195802
http//tucows.mma.com.br/win2k/adnload/149778_49716.html	0.005195802
http//tucows.mpc.com.br/win2k/adnload/149778_49716.html	0.005195802
http//tucows.urbi.com.br/win2k/adnload/149778_49716.html	0.005195802
http//news.telesc.net.br/win2k/adnload/149778_49716.html	0.005195802
http//tucows.vsp.com.br/win2k/adnload/149778_49716.html	0.005195802
http//rav.mdnet.com.br/adnload/83_28478.html	0.005195591
http//tucows.silcons.com.br/adnload/144221_48889.html	0.005195591
http//tucows.alternex.com.br/adnload/83_28478.html	0.005195591
http//tucows.alternex.com.br/winnt/adnload/1194_28476.html	0.005195591
http//tucows.alternex.com.br/winnt/adnload/7156_28448.html	0.005191372
http//tucows.alternex.com.br/adnload/10448_28509.html	0.005191372
http//tucows.alternex.com.br/winnt/adnload/19749_30100.html	0.005191372
http//tucows.mdnet.com.br/adnload/61228_29483.html	0.005191372
http//tucows.alternex.com.br/winnt/adnload/143827_48701.html	0.005191372
http//tucows.atarde.com.br/win2k/adnload/70861_29681.html	0.005191372
http//linuxberg.onlinet.com.br/conhtml/adnload/8029_1148.html	0.005190739
http//tucows.sti.com.br/win2k/adnload/150195_49761.html	0.005187364
http//tucows.mps.com.br/win2k/adnload/150195_49761.html	0.005187364
http//www2.carrier.com.br/win2k/adnload/150195_49761.html	0.005187364
http//tucows.twin-net.com.br/win2k/adnload/150195_49761.html	0.005187364
http//tucows.fortalnet.com.br/win2k/adnload/150195_49761.html	0.005187364
http//tucows.amazon.com.br/win2k/adnload/150195_49761.html	0.005187364
http//tucows.claretianas.com.br/win2k/adnload/150195_49761.html	0.005187364
http//tucows.rio.com.br/win2k/adnload/150195_49761.html	0.005187364
http//tucows.cba.zaz.com.br/win2k/adnload/150195_49761.html	0.005187364
http//tucows.telepar.net.br/win2k/adnload/150195_49761.html	0.005187364
http//tucows.dglnet.com.br/win2k/adnload/150195_49761.html	0.005187364
http//tucows.mma.com.br/win2k/adnload/150195_49761.html	0.005187364
http//tucows.atarde.com.br/win2k/adnload/37773_29413.html	0.005187153
http//tucows.mdnet.com.br/adnload/79_28474.html	0.005187153
http//tucows.netfly.com.br/adnload/48941_48093.html	0.005187153
http//rav.mdnet.com.br/adnload/61616_28496.html	0.005187153
http//tucows.netfly.com.br/win2k/adnload/37773_29413.html	0.005187153
http//tucows.svn.com.br/tucows/winnt/adnload/61792_30367.html	0.005187153
http//tucows.netfly.com.br/win2k/adnload/38527_29571.html	0.005187153
http//rav.mdnet.com.br/win2k/adnload/72973_29239.html	0.005187153

http//tucows.netfly.com.br/winnt/adnload/1149_28430.html	0.005187153
http//tucows.atarde.com.br/win2k/adnload/38527_29571.html	0.005187153
http//tucows.atarde.com.br/winnt/adnload/61792_30367.html	0.005187153
http//tucows.netfly.com.br/winnt/adnload/1801_29331.html	0.005187153
http//tucows.atarde.com.br/win2k/adnload/38807_28539.html	0.005187153
http//linuxberg.onlinet.com.br/kdehtml/adnload/31987_2946.html	0.00518652
http//linuxberg.infolink.com.br/x11html/adnload/9139_3464.html	0.00518652
http//linuxberg.infolink.com.br/conhtml/adnload/31499_1480.html	0.00518652
http//www.linuxberg.infolink.com.br/conhtml/adnload/8108_1253.html	0.00518652
http//www.linuxberg.infolink.com.br/conhtml/adnload/50458_1704.html	0.00518652
http//linuxberg.infolink.com.br/x11html/adnload/9419_3712.html	0.00518652
http//www.linuxberg.infolink.com.br/conhtml/adnload/8594_2019.html	0.00518652
http//www.linuxberg.infolink.com.br/x11html/adnload/10272_4511.html	0.00518652
http//www.linuxberg.infolink.com.br/conhtml/adnload/8254_1507.html	0.00518652
http//linuxberg.onlinet.com.br/conhtml/adnload/8733_2251.html	0.00518652
http//www.linuxberg.infolink.com.br/conhtml/adnload/8265_1525.html	0.00518652
http//linuxberg.infolink.com.br/conhtml/adnload/8267_1528.html	0.00518652
http//www.linuxberg.infolink.com.br/x11html/adnload/9850_3940.html	0.00518652
http//linuxberg.infolink.com.br/conhtml/adnload/8112_1259.html	0.00518652
http//linuxberg.onlinet.com.br/conhtml/adnload/8489_1831.html	0.00518652
http//linuxberg.onlinet.com.br/conhtml/adnload/8853_1954.html	0.00518652
http//www.linuxberg.infolink.com.br/conhtml/adnload/8112_1259.html	0.00518652
http//www.linuxberg.infolink.com.br/conhtml/adnload/73518_41206.html	0.005186098
http//linuxberg.inx.com.br/gnomehtml/adnload/77578_31552.html	0.005186098
http//linuxberg.onlinet.com.br/conhtml/adnload/73518_41206.html	0.005186098
http//linuxberg.matrix.com.br/gnomehtml/adnload/77578_31552.html	0.005186098
http//linuxberg.yawl.com.br/gnomehtml/adnload/77578_31552.html	0.005186098
http//linuxberg.inx.com.br/x11html/adnload/9117_36963.html	0.005186098
http//linuxberg.yawl.com.br/gnomehtml/adnload/77578_49732.html	0.005186098
http//linuxberg.inx.com.br/x11html/adnload/9117_36965.html	0.005186098
http//linuxberg.tecsat.com.br/x11html/adnload/9117_33738.html	0.005186098
http//linuxberg.tecsat.com.br/x11html/adnload/9117_36965.html	0.005186098
http//linuxberg.yawl.com.br/gnomehtml/adnload/77578_41800.html	0.005186098
http//linuxberg.unisys.com.br/conhtml/adnload/73518_41206.html	0.005186098
http//linuxberg.infolink.com.br/x11html/adnload/9117_36964.html	0.005186098
http//linuxberg.unisys.com.br/gnomehtml/adnload/77578_41800.html	0.005186098
http//linuxberg.uninet.com.br/gnomehtml/adnload/77578_49732.html	0.005186098
http//linuxberg.pop-sc.rnp.br/conhtml/adnload/73518_49003.html	0.005186098
http//linuxberg.uol.com.br/x11html/adnload/9117_36964.html	0.005186098
http//www.linuxberg.infolink.com.br/x11html/adnload/9117_36965.html	0.005186098
http//linuxberg.tecsat.com.br/x11html/adnload/9117_36963.html	0.005186098
http//tucows.cba.zaz.com.br/adnload/149906_49762.html	0.005183145
http//tucows.mariodias.com.br/adnload/149776_49716.html	0.005183145
http//ln1.sigmabbs.com.br/adnload/149906_49762.html	0.005183145
http//tucows.yawl.com.br/winme/adnload/138569_29874.html	0.005183145
http//tucows.rionet.com.br/adnload/149906_49762.html	0.005183145
http//tucows.hotlink.com.br/adnload/149776_49716.html	0.005183145
http//tucows.plugnet.com.br/adnload/149906_49762.html	0.005183145
http//tucows.unisys.com.br/winme/adnload/146135_49416.html	0.005183145

Tabela G. 5: Maiores hubs do grafo Web utilizado neste trabalho. Como o engenho de busca utilizado como estudo de caso indexa apenas as páginas em português não há necessidade de uma taela apenas com as páginas ".br".