SÍLVIO LUÍS MARANGON

Análise de métodos para programação de Contextualização

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia.

São Paulo
2006

SÍLVIO LUÍS MARANGON

Análise de métodos para programação de Contextualização

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia.

Área de Concentração:
Sistemas Eletrônicos

Orientador:
Prof. Dr. Márcio Lobo Netto

São Paulo
2006

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, de novembro de 2006.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Marangon, Silvio Luis
Análise de métodos para programação de contextualização / S.L. Marangon. -- ed.rev. -- São Paulo, 2006.
120 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1.Mecanismos de busca na Web 2.Recuperação da informacão 3.Mineração de dados 4.World Wide Web 5.Internet I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II.t.

AGRADECIMENTOS

A todos que colaboraram direta ou indiretamente na execução deste trabalho, em especial ao meu orientador Prof. Dr. Márcio Lobo Netto, os colegas do Laboratório de Sistemas Integráveis, em especial ao Prof. Dr. Sérgio Takeo Kofuji e ao Dr. Volnys Borges Bernal.

RESUMO

A localização de páginas relevantes na Internet em atividades como *clipping* de notícias[1], detecção de uso indevido de marcas ou em serviços anti-phishing[2] torna-se cada vez mais complexa devido a vários fatores como a quantidade cada vez maior de páginas na Web e a grande quantidade de páginas irrelevantes retornadas por mecanismos de busca.

Em muitos casos as técnicas tradicionais utilizadas em mecanismos de busca na Internet, isto é, localização de termos em páginas e ordenação por relevância, não são suficientes para resolver o problema de localização de páginas específicas em atividades como as citadas anteriormente.

A contextualização das páginas, ou seja, a classificação de páginas segundo um contexto definido pelo usuário baseando-se nas necessidades de uma atividade específica deve permitir uma busca mais eficiente por páginas na Internet.

Neste trabalho é estudada a utilização de métodos de mineração na Web para a composição de métodos de contextualização de páginas, que permitam definir contextos mais sofisticados como seu assunto ou alguma forma de relacionamento.

A contextualização de páginas deve permitir a solução de vários problemas na busca de páginas na Internet pela composição de métodos, que permitam a localização de páginas através de um conjunto de suas características, diferentemente de mecanismos de busca tradicionais que apenas localizam páginas que possuam um ou mais termos especificados.

---

[1] Serviço de coleta de notícias sobre uma instituição ou pessoa.

[2] Phishing é o ato de enviar e-mails em nome de uma empresa, tentando de alguma forma obter informações confidenciais do destinatário.

ABSTRACT

Internet services as news clipping service[3], anti-phising[4], anti-plagiarism service and other that require intensive searching in Internet have a difficult work, because of huge number of existing pages.

Search Engines try driver this problem, but search engines methods retrieve a lot of irrelevant pages, some times thousands of pages and more powerful methods are necessary to drive this problem.

Page content, subject, hyperlinks or location can be used to define page context and create a more powerful method that can retrieve more relevant pages, improving precision. Classification of page context is defined as classification of a page by a set of its feature.

This report presents a study about Web Mining, Search Engines and application of web mining technologies to classify page context.

Page context classification applied to search engines must solve the problem of irrelevant pages flood by allowing search engines retrieve pages of a context.

---

[3] Clipping is a media monitoring service.

[4] *Phishing* is the practice of distributing e-mail messages and Web sites that look like legitimate, usually for criminal purposes.

SUMÁRIO.

LISTA DE TABELAS
LISTA DE FIGURAS

1 INTRODUÇÃO 1

1.1 CONTEXTUALIZAÇÃO DE PÁGINAS NA WEB 2
1.2 SERVIÇOS DE BUSCA ESPECIALIZADOS 2
1.3 MÉTODOS PARA BUSCA NA WEB 4
1.4 OBJETIVO 6
1.5 ESCOPO 6
1.6 TRABALHOS RELACIONADOS 7
1.7 ORGANIZAÇÃO DO TRABALHO 8

2 MECANISMOS DE BUSCA NA WORLD WIDE WEB 10

2.1 PROCESSO DE RECUPERAÇÃO DE INFORMAÇÃO 10
2.1.1 Coleta dos documentos 12
2.1.2 Pré-processamento dos Textos 12
2.1.3 Modelos de Recuperação de Informação 13
2.1.4 Seleção dos atributos 17
2.1.5 Calculo do peso dos termos 18
2.1.6 Avaliação dos resultados 20
2.2 MECANISMOS DE BUSCA 22
2.2.1 Requisitos para um mecanismo de busca na Web 22
2.2.2 Tipos de mecanismos de busca 23
2.2.3 Tecnologias dos mecanismos de busca 23
2.2.3.1 Exploração dos Hyperlinks 24
2.2.3.2 Recuperação de Informação 24
2.2.3.3 Metabuscadores 25
2.2.3.4 Baseado em SQL 25
2.2.3.5 Pesquisa por conteúdo multimídia 25
2.2.3.6 Outras 26

2.2.4 Formas de consultas *26*
2.2.5 Métodos de aquisição de conteúdo *28*
2.3 ARQUITETURA DE UM MECANISMO DE BUSCA AUTÔNOMO 28
2.3.1 Sistema de Coleta *30*
2.3.2 Indexador *31*
2.3.3 Ranker *33*
2.3.4 Searcher *33*
2.4 *DISPATCHER* 34
3 MECANISMO DE BUSCA NUTCH 36
3.1 MECANISMOS DE BUSCA 36
3.2 NUTCH 37
3.3 ARQUITETURA NUTCH 39
3.3.1 Processo de coleta e indexação *40*
3.3.1.1 Iniciação 41
3.3.1.2 Coleta das páginas 41
3.3.1.3 Indexação das páginas 42
3.3.2 Processo de busca *42*
3.3.3 Estruturas de dados do Nutch *44*
3.3.4 Estruturas de dados do Lucene *45*
3.3.5 Campos do índice do Nutch *46*
3.3.6 Protocolos suportados pelo Nutch *47*
3.3.7 Consultas no Nutch *47*
3.3.8 Formatos de arquivos suportados pelo Nutch *49*
3.3.9 Cálculo de relevância no Nutch. *49*
3.3.9.1 Valor de relevância estático 49
3.3.9.2 Valor de relevância dinâmico do Lucene 50
4 MINERAÇÃO NA WEB 52
4.1 TAREFAS DE MINERAÇÃO NA WEB 55
4.1.1 Sumarização *55*
4.1.2 Classificação e Categorização *55*
4.1.2.1 Classificador Rocchio 56

4.1.2.2 Classificador Naive Bayes 58
4.1.2.3 Árvores de Decisão 59
4.1.2.4 Redes Neurais Artificial 60
4.1.2.5 Máquinas de Vetores Suporte (Support Vector Machines) 62
4.1.2.6 Vizinhos Mais Próximos (k Nearest Neighbor) 64
4.1.3 Agrupamento 65
4.1.3.1 Comunidades Web 66
4.1.3.2 Métodos de agrupamento baseados na análise da estrutura de hyperlinks 67
4.1.3.3 Métodos de agrupamento baseados na análise do texto 71
4.1.4 Métodos de ordenação por relevância em mecanismos de busca 74
4.1.4.1 Valor de relevância 74
4.1.4.2 Calculo do valor de relevância 76
4.1.4.3 Fonte de autoridade 77
4.1.4.4 Métodos de cálculo do valor de relevância baseados em conectividade 79
4.1.4.5 Problemas com a definição do valor de relevância 81

5 MÉTODOS DE CONTEXTUALIZAÇÃO 83

5.1 CLASSIFICAÇÃO DOS MÉTODOS 83
5.2 MÉTODOS DE CONTEXTUALIZAÇÃO 83
5.2.1 Consulta por termos 84
5.2.2 Programação por Análise de URL 84
5.2.2.1 Uniform Resource Locator (URL) 85
5.2.2.2 Contextualização pela URL 87
5.2.2.3 Métodos de análise de padrão em URL 90
5.2.3 Contextualização por referência direta 91
5.2.4 Contextualização pelo agrupamento de páginas 91
5.2.5 Agrupamento por referência direta 92
5.2.6 Contextualização pela categorização da página 92
5.3 FATORES A SEREM CONSIDERADOS PARA ESCOLHA DE UM MÉTODO 93

5.3.1 Treinamento *94*
5.3.2 Novas estruturas de dados *96*
5.3.3 Custo computacional *97*
6 PROPOSTA PARA PROGRAMAÇÃO DE CONTEXTUALIZAÇÃO **98**
6.1.1 Seqüência de aplicação dos métodos. *98*
6.1.2 Forma de Aplicação *99*
6.2 HISTÓRICO DE CONTEXTOS 100
6.3 INTEGRAÇÃO DE MÉTODOS DE CONTEXTUALIZAÇÃO NO NUTCH 101
6.3.1 Contextualização pela URL *101*
6.3.2 Contextualização por Agrupamento *103*
6.3.3 Contextualização pela Categoria *105*
6.3.4 Histórico de Contextos *106*
6.4 IMPLEMENTAÇÃO 107
6.4.1 Ambiente de testes. *107*
6.4.2 Testes realizados e resultados. *107*
6.4.2.1 Análise dos resultados. **108**
7 CONSIDERAÇÕES FINAIS. **109**
7.1 CONCLUSÃO. 110
7.2 CONTRIBUIÇÕES. 112
7.3 TRABALHOS FUTUROS. 112
REFERÊNCIAS **113**

LISTA DE FIGURAS

Figura 1 - O processo de recuperação de informação (YATES, R. B. RIBEIRO B. A., 1999). 11
Figura 2 - Modelo de espaço vetorial (SALTON, G.; WONG, A.; YANG C.S., 1975). 15
Figura 3 - Arquitetura de referência de mecanismos de busca (RISVIK, K. M.; AASHEIM, Y.; LIDAL, M., 2003)..... 29
Figura 4 - Arquitetura geral de um Sistema de Coletar 30
Figura 5 - Arquitetura geral do Indexador. 31
Figura 6 - Índice invertido..... 32
Figura 7 - Arquitetura geral do Ranker 33
Figura 8 - Arquitetura geral do Searcher..... 34
Figura 9 - Arquitetura geral do Dispatcher 35
Figura 10 - Diagrama de Dependências do Nutch (KHARE, R. et. al, 2004). 39
Figura 11 - Arquitetura Nucth. 40
Figura 12 - Processo de coleta e indexação do Nucth..... 41
Figura 13 - Processo de Busca. 43
Figura 14 - Árvore de Decisão. 60
Figura 15 - Rede Neural Artificial. 61
Figura 16 - Neurônio Artificial. 62
Figura 17 - Superfície de decisão com margem máxima (YANG, Y.; LIU, X., 1999) 63
Figura 18 - Conjunto de corte (FLAKE G. W.; GILES C. L.; COETZEE F. M., 2002) 69
Figura 19 - Grafos:(i) DBG(T,I,p,q) e (ii) CBG(T,I,p,q)(Reddy, P.K.; Kitsuregawa, M. 2001). 70
Figura 20 - Exemplo de rede RBF (TOMIYAMA, T. et al., 2003). 73
Figura 21 - Estrutura de um rede RBF (TOMIYAMA, T. et al., 2003). 73
Figura 22 - Hubs e Authorities (KLEINBERG, J. M, 1999) 78
Figura 23 - Esquema geral de uma URL (RFC1738). 85
Figura 24 - Sintaxe do esquema para http (RFC1738)..... 85
Figura 25 - Sintaxe para o esquema do nome de um computador (RFC1738) 85

Figura 26 - Forma general (NIC.BRb) 86
Figura 27 - Forma Reservada (NIC.BRb). 87
Figura 28 - Esquema genérico para análise..... 87
Figura 29 - Método Utilizado como Filtro. 98
Figura 30 - Uso do histórico..... 100
Figura 31 - Uso do histórico..... 101

LISTA DE TABELAS

Tabela 1 - Mecanismos de Busca: Licença e estado de desenvolvimento. 36
Tabela 2 - Mecanismos de busca: Características estendidas. 37
Tabela 3 - Informações sobre páginas armazenadas na WebDB (NUTCH 0.7.1 API). 44
Tabela 4 - Informações sobre hyperlinks armazenadas na WebDB (NUTCH 0.7.1 API) 45
Tabela 5 - Campos do índice do Nutch (NUTCH 0.7.1 API). 46
Tabela 6 - Campos extras do índice do Nutch (NUTCH 0.7.1 API) 47
Tabela 7 - Método x Treinamento. 95
Tabela 8 - Método x Novas estruturas de dados. 96
Tabela 9 - Método x Forma de Aplicação. 99

LISTA DE ABREVIATURAS E SIGLAS

API	Interface de Programação para Aplicativos.
EI	Extração de Informação.
HITS	Hyperlinks Induced Topic Search.
HTML	HyperText Makeup Language.
HTTP	HyperText Transfer Protocol.
Idf	Inverso da freqüência do termo nos documentos.
JSP	Java Server Page.
MD5	Algoritmo de hash de 128 bits.
NIC.BR	Núcleo de Informação e Coordenação do Ponto BR.
RBF	Radial Basis Function Network.
RI	Recuperação de Informação.
RNA	Rede Neural Artificial.
SQL	Structured Query Language.
STC	Suffix Tree Clustering.
SVM	Support Vector Machine.
TCP	Transmission Control Protocol.
Tf	Freqüência do termo no documento.
UDP	User Datagram Protocol.
URL	Uniform Resource Locator.
Webdb	Base de dados de páginas e hyperlinks do Nutch.

1 INTRODUÇÃO

A Web tornou-se o repositório universal do conhecimento e cultura humana, a qual tem permitido o compartilhamento de idéias e informações em escala nunca vista antes. Estima-se que o número total de documentos disponibilizados na Web seja superior a 11 bilhões de páginas (GULLI, A.; SIGNORINI. A, 2005). O seu sucesso é baseado no conceito de uma interface de usuário padronizada, que protege o usuário dos detalhes dos protocolos de comunicação, localização de máquinas e sistema operacional. Além disto, qualquer usuário pode criar sua própria página Web e fazer referência a qualquer outra página sem restrições (YATES, R. B.; RIBEIRO B. A., 1999).

Com o aumento da quantidade de informação disponível na Internet, surge o desafio de se encontrar a informação desejada que, freqüentemente, é uma tarefa difícil e tediosa. Para encontrar a informação desejada, o usuário pode seguir os *hyperlinks* apresentados até alcançar a informação, o que é ineficiente devido ao grande número de documentos na Web e a grande quantidade de informações de baixa qualidade (YATES, R. B.; RIBEIRO B. A., 1999).

Para solucionar estes problemas, foram desenvolvidos vários mecanismos para busca na Web. Estes mecanismos utilizam os mais variados métodos para organizar e classificar a informação. O termo "mecanismo de busca" é utilizado para definir uma grande variedade de serviços que provêem acesso aos recursos na Internet. O conceito de mecanismo de busca inclui: o mecanismo de aquisição de informação, o mecanismo de classificação e organização dos dados, o mecanismo de resposta e apresentação, e uma base de dados (PAGE, L.; BRIAN, S., 1998) (PAGE, L. et al., 1998).

Recuperação de dados, no contexto dos sistemas de Recuperação de Informação (RI), consiste principalmente em determinar quais documentos de uma coleção contém algumas palavras chaves fornecidas pelo usuário. Freqüentemente, isto não satisfaz as necessidades do usuário na procura por informações sobre um determinado assunto. Por outro lado, os sistemas de recuperação de dados, como sistemas de base de dados relacionais, lidam com dados de estrutura e semântica bem conhecida. Sistemas de recuperação de dados provêem uma solução para usuários de

sistemas base de dados, mas não resolvem o problema de recuperar informação sobre um assunto ou tópico (YATES, R. B.; RIBEIRO B. A., 1999).

Para fornecer a informação desejada pelo usuário, os sistemas de RI devem, de alguma forma, interpretar a informação contida nos documentos e classificá-los de acordo com sua relevância para a consulta do usuário (YATES, R. B.; RIBEIRO B. A., 1999). Com a popularização da Web, vários métodos foram desenvolvidos para solucionar estes problemas, como os trabalhos de Kleinberg, J. M. (1999) e Page, L. et al. (1998) que propõem métodos para qualificar páginas, analisando a estrutura de *hyperlinks* da Web, e o trabalho de Tomiyama, T. et al. (2003) para descoberta de comunidades na Web.

1.1 Contextualização de páginas na Web

Considerando-se a definição do termo *contexto* como: "conjunto; todo, totalidade" (FERREIRA, 2004), termo *contextura* como: "Ligação entre as partes de um todo; encadeamento, contexto" (FERREIRA, 2004) e o termo contextualização como: "o ato de vincular o conhecimento à sua origem e à sua aplicação" (DIEB); a programação de contextualização é definida como a classificação de uma página na Internet segundo um conjunto de suas características, ou seja, a classificação de seu assunto, seu agrupamento, domínio do sítio na Internet, etc.

Como a Internet tornou-se um dos meios de comunicação mais importantes atualmente e adquire cada vez mais importância comercial para as empresas, a contextualização pode ser útil para varias atividades empresariais como descoberta de uso indevido de marcas, uso indevido de obra autoral, serviço de *clipping* de notícias, entre outros.

1.2 Serviços de busca especializados

Muitas empresas utilizam a Internet tanto como meio para venda de seus produtos e serviços como meio para promoção e divulgação de suas marcas e

produtos. Porém, com a popularização da Internet, associado a sua facilidade de uso, abusos vem sendo cometidos, visando aos mais variados objetivos, como:

- Desvio de tráfego ou clientela;
- Incremento de tráfego em buscas;
- Fraudes com objetivos financeiros, políticos ou outros;
- Aumento de credibilidade do sítio por falsas parcerias;
- Difamar da empresa, produtos ou marca.

Indivíduos ou empresas que praticam atos com os objetivos citados anteriormente podem estar cometendo infrações como:

- Uso indevido de marca registrada;
- Uso indevido de obra autoral, como por exemplo: artigos, reportagens, livros, músicas, imagens, vídeos;
- Estelionato;
- Fraude;
- Concorrência desleal.

Algumas empresas fornecem serviços de busca especializados para auxiliar nestas tarefas, como: Turnitin (TURNITIN), que oferece serviços voltados para descoberta de plágio em trabalhos escolares; a iThenticate (ITHENTICATE), que fornece serviços voltados para descoberta de uso indevido de obra autoral, uso indevido de propriedade intelectual ou marca registrada de empresas e a Cyveillance (CYVEILLANCE), que fornece serviços como Anti-Phishing[5], detecção de uso não autorizado de marcas, logotipos e títulos de páginas, entre outros serviços.

Outro fator de interesse das empresas é o gerenciamento da imagem da empresa, de suas marcas e produtos na Internet. Este tipo de problema é tratado pelos serviços de *clipping* de notícias e anúncios na Internet, que consiste na busca por

[5] Phishing é o ato de enviar e-mail em nome de uma empresa, tentando de alguma forma obter informações confidenciais do destinatário do e-mail como senha e número de cartão de crédito.

notícias e anúncios de interesse da empresa. Diversas empresas fornecem serviços de *clipping* de notícias, como, por exemplo, a Info4 (INFO4) e ClipEx (CLIPEX). Estes podem oferecer serviços que podem abranger uma empresa, marca ou produto, ou também podem fornecer serviços de *clipping* que podem abranger um segmento de mercado, uma empresa ou um produto, o que permite à empresa avaliar sua posição no mercado e avaliar melhor suas estratégias.

Alguns dos problemas encontrados no *clipping* de notícias são: a identificação dos sítios de notícias e definição do contexto das notícias. A descoberta da notícia é geralmente realizada através da pesquisa em um conjunto de sítios previamente cadastrado, como sítios de jornais, revistas, agências de notícias e portais. A manutenção de uma base de dados com o histórico de notícias publicadas, que mantém uma cópia local das notícias e anúncios, procura evitar o problema causado devido à remoção das notícias e anúncios das páginas.

Serviços de pesquisa de preços na Internet também encontram o mesmo problema, pois estes necessitam saber quais são os sítios de vendas na Internet e a contextualização poderia auxiliar nesta tarefa.

1.3 Métodos para busca na Web

Muitos métodos foram propostos para auxiliar na procura por informações na Web, o que inclui métodos que exploram a estrutura de *hyperlinks* da Web para determinar a relevância de páginas, como PageRank (PAGE, L. et al, 1998) e HITS (KLEINBERG, J. M., 1999). Também foram propostos vários métodos para descoberta de agrupamentos e categorização de páginas na Web, e contextualização da consulta. Para descoberta de agrupamentos de páginas na Web foram propostos vários métodos que analisam da estrutura de *hyperlinsks*, utilizando métodos de análise de grafos, como fluxo em grafos (FLAKE, G. W.; GILES, C. L.; COETZEE, F. M., 2002) (IMAFUJI, N.; KITSUREGAWA, M. 2003), small-world (ADAMIC, L. 1999), grafos bipartidos (REDDY, P. K.; KITSUREGAWA, M. 2001). Para a contextualização da consulta, que procura identificar o assunto de interresse do usuário antes de realizar a seleção de documentos, existem as propostas de Loh, S.; Wives, L. K. e Frainer, A. S. (1997) e Mukherjea, S. e Foley, J. D.; (1995). Para

categorização das páginas existem vários trabalhos na área de Mineração de Texto, que procuram resolver o problema de categorização de documentos utilizando as mais variadas abordagens, como métodos estatísticos e aprendizado de máquina. Porém, estes métodos são genéricos e isoladamente não solucionam os problemas enfrentados para serviço de *clipping* de notícias e descoberta de uso indevido de obra autoral, uso indevido de propriedade intelectual ou marca registrada de empresas, pois são voltados para definir qual página é mais relevante ou popular para uma consulta, e não para seleção de páginas que se enquadrem em determinado contexto.

Para realizar uma tarefa como esta, uma alternativa é a utilização de vários métodos combinados, o que pode envolver:

- Mecanismos de busca na Internet;
- Métodos de classificação de relevância do documento (ranking);
- Análise de URLs;
- Técnicas de Mineração de Texto, como:
    - Categorização, que classifica as páginas por assunto, como por exemplo, a classificação automática de notícias de um jornal em categorias predefinidas como: política, economia, esportes, cultura e lazer;
    - Agrupamento, no qual as páginas são agrupadas em grupos relacionados por assunto, porém isto é realizado automaticamente e sem a definição prévia de categorias;
    - Sumarização, que gera um resumo automático sobre o conteúdo de uma página com a finalidade de auxiliar o usuário na seleção de páginas.
- Manutenção e uso de histórico:
    - Histórico de sítios classificados anteriormente;
    - Histórico de resultados para auxilio nas buscas do usuário.

Isoladamente estas técnicas, geralmente utilizadas em sistemas de recuperação de informação, podem não produzir resultados satisfatórios. Porém, o uso conjunto destas, compondo um método mais sofisticado, pode oferecer uma

alternativa mais adequada para auxiliar em problemas como os citados anteriormente.

A contextualização de páginas pode diminuir o volume de dados a serem analisados pelo usuário ou por um sistema, eliminando de forma automática páginas irrelevantes para o contexto. Por exemplo, um mecanismo de busca poderia selecionar automaticamente páginas de comentários (*blog*) em uma pesquisa para *clipping* de notícias. Em um sistema de detecção de violação de propriedade intelectual poderia aumentar a relevância de páginas associadas através de *hyperlinks* a uma página pertencente a um contexto de risco.

1.4 Objetivo

Mecanismos de busca para Internet geralmente utilizam duas técnicas básicas para busca de páginas:

a) Pesquisa em índice invertido para seleção de páginas;
b) Definição de relevância pela análise da estrutura de *hyperlinks*.

Outros métodos, como aqueles utilizados na área de mineração na Web, ainda são pouco explorados em mecanismos de busca.

Este trabalho pretende realizar a análise dos métodos provenientes das áreas de recuperação de informação e mineração na Web, que possam ser utilizados na contextualização de páginas da Web e estudar como tais métodos podem ser combinados em um processo de programação de contextualização de páginas da Web.

Também serão levantadas as funcionalidades adicionais necessárias para um mecanismo de busca suportar os métodos analisados.

1.5 Escopo

Relacionado ao objetivo apresentado na seção anterior este trabalho irá abranger as seguintes atividades:

- Levantamento do estado da arte em mecanismos de busca Web e Mineração da Web;
- Identificação de métodos adequados à contextualização de páginas na Web;
- Análise sobre a aplicação de métodos de contextualização de páginas em mecanismos de busca.

Esta fora do escopo deste trabalho a implementação de todos os métodos apresentados, bem como, análise comparativa entre estes.

1.6 Trabalhos relacionados

A World Wide Web Worm (McBryan O. A, 1994) foi um dos primeiros mecanismos de busca para Web, seguido por outros trabalhos acadêmicos, que procuraram tratar os vários aspectos relacionados aos mecanismos de busca. Em arquitetura, existem os trabalhos de Page, L. e Brian, S. (1998), Shawartz, C. (1998), Hu, W. C. (2001) e Risvik, K. M.; Aasheim, Y e Lidal, M. (2003) que exploram a idéia de um mecanismo de busca multicamadas. Porém, devido à grande importância comercial atual dos mecanismos de busca na Web, a disponibilidade de informações específicas sobre a arquitetura e métodos utilizados nos sistemas comerciais é pequena, já que são considerados de valor estratégico pelas empresas.

Para classificar as páginas segundo sua importância, destacam-se os trabalhos de Page et al (1998), que apresenta o algoritmo de classificação PageRank, utilizado pelo serviço de busca Google (GOOGLE) e o trabalho de Kleinberg, J. M. (1999), que apresenta o algoritmo HITS (*Hyperlink-Induced Topic Search*) para classificação de páginas, utilizado pelo serviço de busca Teoma (www.teoma.com) e também introduz os conceitos de páginas *autoridades* e *hubs*.

Para descoberta de comunidades Web, ou seja, agrupamentos de páginas relacionadas a um mesmo tópico ou interesse comum de seus criadores, existem vários trabalhos que exploram a estrutura de *hyperlinks* da Web e fortemente baseados na teoria de grafos, como os trabalhos de Flake, G.W.; Giles, C.L. e Coetzee, F.M. (2002) e Imafuji, N. e Kitsuregawa, M. (2003) que utilizam do

conceito de máximo fluxo e conjunto de corte para identificar comunidades Web. Também existem trabalhos que exploram o conceito de *small world*, como o trabalho de Adamic, L (1999), e trabalhos que exploram o conceito de grafos bipartidos, como Reddy, P.K.; Kitsuregawa, M. (2001) e Greco, G.; Greco, S. e Zumpano, E. (2002). Existem trabalhos que utilizam os algoritmos de classificação para descoberta de comunidades, como Gibson, D.; Kleinberg, J. e Raghavan, P. (1998), que utiliza o algoritmo HITS.

Para a contextualização da consulta, destacam-se as propostas de Loh, S.; Wives, L. K. E Frainer, A. S. (1997) e Mukherjea, S. e Foley, J. D.; (1995).

Para categorização das páginas existem vários trabalhos na área de mineração de texto que procuram resolver o problema de categorização de documentos utilizando as mais variadas abordagens, como métodos estatísticos e aprendizado de máquina. Alguns trabalhos propõem a utilização de técnicas de inteligência artificial, explorando técnicas de aprendizado de máquina tais como árvores de decisão, redes neurais artificiais e *Support Vector Machine* (SVM). Trabalhos que podem ser citados são: Tomiyama, T. et al (2003), o qual é baseado em lógica fuzzy e utiliza conjuntos conceituais fuzzy para descoberta de comunidades Web e Pirolli, P.; Pitkow, J. e Rao, R. (2005) que explora a similaridade entre textos, formação do texto e *hyperlinks* e Kinto, E. A. e Del-Moral-Hernandez, E. (2005) que utiliza SVM para classificar textos reduzidos. Também são populares alguns trabalhos que obtêm seus resultados a partir de pós-processamento dos resultados de outros mecanismos de busca comerciais como os metabuscadores.

Por fim, trabalhos têm sido desenvolvidos em sistemas de recuperação de informações, especialmente em ambientes controlados com um universo reduzido de documentos como, por exemplo, coleção de artigos científicos.

1.7 Organização do trabalho

Esta dissertação está estruturada da seguinte forma:

Capítulo 1 – Introdução.

Neste capítulo são apresentados os motivos que levaram a escrita deste trabalho, seu objetivo, seu escopo e trabalhos relacionados.

Capítulo 2 – Mecanismos de busca na World Wide Web.

Neste capítulo serão abordados tópicos relevantes à arquitetura dos mecanismos de busca e suas características, também são abordados tópicos de recuperação de informação e métodos de ordenação por relevância de páginas.

Capítulo 3 – Mecanismo de busca Nutch.

Neste capítulo é realizado um estudo de caso de mecanismo de busca, sendo descrito o mecanismo de busca Nutch.

Capítulo 4 – Mineração na Web.

Neste capítulo são apresentados métodos para mineração na Web, sendo apresentados métodos para categorização e agrupamento de páginas.

Capítulo 5 – Métodos de contextualização

Neste capítulo são descritos e analisados métodos para a contextualização de páginas.

Capítulo 6 – Proposta para programação de contextualização.

Neste capítulo é descrita a programação de métodos de contextualização em um mecanismo de busca na Web.

Capítulo 7 – Conclusão.

Neste capítulo serão apresentadas as conclusões.

Referências bibliográficas.

Por fim no último capítulo, as referências bibliográficas deste trabalho.

2 MECANISMOS DE BUSCA NA WORLD WIDE WEB

Os mecanismos de busca na Web são sistemas de Recuperação de Informação (RI) desenvolvidos especificamente para Web, que fornecem serviço de recuperação de informação na Web, ou seja, localização de páginas na Web que contenham informações específicas.

No contexto dos sistemas de RI (Recuperação de informação), recuperação de informação consiste principalmente em determinar quais documentos de uma coleção contém algumas palavras chaves fornecidas pelo usuário. Freqüentemente, isto não satisfaz as necessidades do usuário, que procuram informações sobre um determinado assunto. Sistemas de RI usualmente lidam com textos em linguagem natural, nem sempre bem estruturado e com semântica ambígua (YATES, R. B.; RIBEIRO B. A., 1999). Para fornecer a informação desejada pelo usuário, os sistemas de RI devem, de alguma forma, interpretar a informação contida nos documentos e classificá-los de acordo com sua relevância para a consulta do usuário (YATES, R. B.; RIBEIRO B. A., 1999).

Na próxima seção será descrito o processo de recuperação de informação e suas etapas. Nas seções seguintes, serão descritos os mecanismos de busca na Web e aspectos específicos destes, os quais não são tratados em RI.

2.1 Processo de recuperação de informação

Antes de iniciar o processo de recuperação de informação deve-se definir a base de dados de documentos. Os documentos da base de dados são processados para gerar a visão lógica de cada documento e posteriormente é realizada sua indexação (YATES, R. B.; RIBEIRO B. A., 1999). A Figura 1 exibe um modelo de um sistema de recuperação de informação. Diferentemente dos sistemas tradicionais, os sistemas de Recuperação de Informação para Web não possuem uma base de documentos própria. Portanto, estes sistemas devem possuir um mecanismo próprio de coleta de

documentos chamado sistema de coleta ou *crawler*, para obter os documentos e gerar uma base local de documentos.

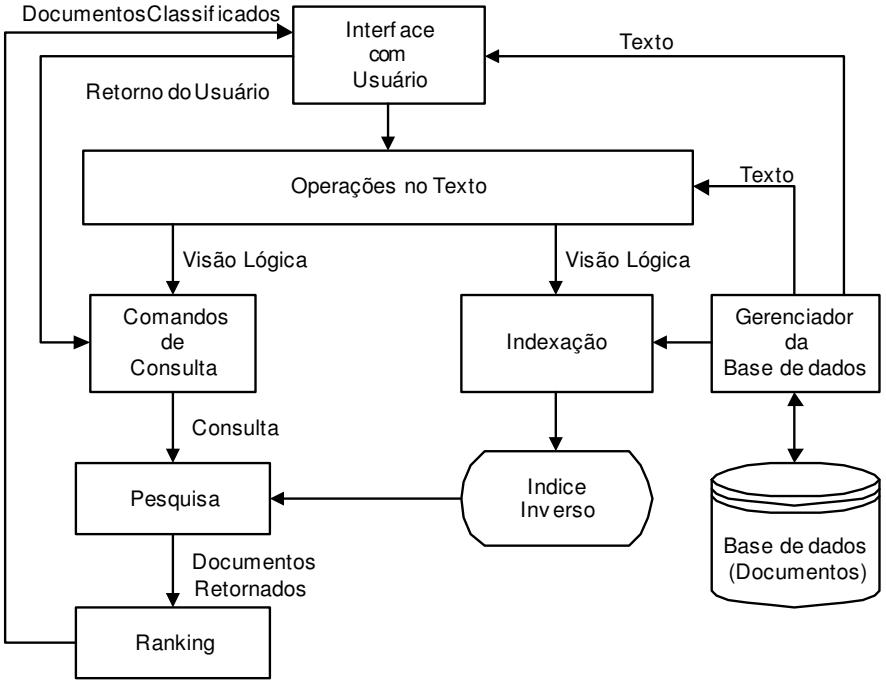

Figura 1 - O processo de recuperação de informação (YATES, R. B. RIBEIRO B. A., 1999).

O índice é uma estrutura crítica do sistema, pois possibilita que a busca seja realizada rapidamente em um grande volume de dados. O processo de recuperação de informação inicia-se com a submissão de uma consulta pelo usuário. Esta consulta é transformada para uma representação interna do sistema e, somente então, a consulta é processada para obter os documentos relevantes, indexados previamente. Antes dos documentos serem enviados ao usuário, estes são classificados e ordenados segundo um critério de relevância para a consulta específica. O usuário seleciona os documentos de seu interesse e inicia um ciclo de interação com o sistema. A consulta do usuário pode ser refinada segundo os documentos selecionados pelo usuário (YATES, R. B.; RIBEIRO B. A., 1999).

Os sistemas de recuperação de informação atuais diferem dos antigos em vários aspectos. Com o advento dos mecanismos de busca para Web, segundo Yates, R. B. e Ribeiro B. A. (1999) as principais diferenças são:

- A pesquisa atualmente é realizada no texto completo e não apenas nas citações bibliográficas;
- Sistemas modernos utilizam dados estatísticos para definir a relevância de um documento;
- O foco dos sistemas antigos era o usuário profissional intermediário, hoje, a maior parte das pesquisas é feita por usuários inexperientes;
- Maior número de fontes de informação disponível em rede, normalmente em hipertexto, e distribuída entre vários servidores.

Para que o processo de recuperação de informação possa ocorrer como descrito anteriormente, varias etapas preparatórias devem ser executadas, como:

- Coleta dos documentos;
- Pré-processamento dos textos;
- Definição do modelo de recuperação de informação a ser utilizado;
- Seleção dos atributos;
- Cálculo de peso dos termos;
- Avaliação dos resultados.

Nas próximas seções são descritas as principais etapas envolvidas na preparação da base de dados e índices de um sistema de recuperação de informação.

2.1.1 Coleta dos documentos

Nesta etapa é feita a coleta dos documentos relevantes.

2.1.2 Pré-processamento dos Textos

A etapa de pré-processamento compreende todas as atividades de preparação do texto para a representação interna utilizada no sistema. As principais tarefas da etapa pré-processamento do texto são:

a) **Análise léxica:** Na análise léxica são eliminados os sinais de pontuação, os dígitos numéricos, as letras maiúsculas são convertidas para minúsculas e é gerada a lista de termos correspondente ao texto;
b) **Eliminação de termos irrelevantes (stop words):** Nesta etapa são eliminados os termos que aparecem freqüentemente nos textos, dependente da língua utilizada. Estes termos, por serem muito comuns não são capazes de discriminar os textos e a sua remoção reduz o volume de dados a serem armazenados e analisados posteriormente. De acordo com Salton, G. (1983), esta técnica permite uma redução entre 30 e 50% no tamanho dos documentos;
c) **Normalização morfológica dos termos e remoção de afixos:** Na etapa de normalização morfológica todos os termos que possuem uma mesma raiz morfológica são substituídos pela sua raiz morfológica, que diminui o número de termos que representam os textos. Assim, todas as formas de um verbo regular da língua portuguesa são representadas por um único termo.

A Indexação semântica latente (*Latent Semantic Indexing*) é outra técnica utilizada no pré-processamento de documentos. Nesta técnica é analisada a correlação entre os termos de um conjunto de documentos, fazendo com que termos similares sejam representados em uma mesma dimensão, que representa um conceito, reduzindo o número de dimensões utilizadas no modelo de espaço vetorial e permitindo que documentos similares, que não compartilhem os mesmos termos, possam ser classificados em uma mesma categoria (KOSALA, R.; BLOCKELL, H., 2000).

2.1.3 Modelos de Recuperação de Informação

Os modelos conceituais de recuperação de informação são abordagens genéricas para o problema de recuperação de informação. Os modelos clássicos para representação de documentos desenvolvidos para a área de recuperação de

informação são os modelos booleano, espaço vetorial e probabilístico, sendo o modelo de espaço vetorial e suas variantes os mais utilizados (YATES, R. B.; RIBEIRO B. A., 1999). Além dos modelos clássicos existem vários outros modelos, como o modelo Fuzzy, o Lógico e o Extended Boolean (YATES, R. B.; RIBEIRO B. A., 1999). Nas próximas seções são descritos os três modelos clássicos.

a) Modelo booleano

O modelo booleano é baseado na lógica booleana. Neste modelo, os documentos são representados por um vetor de pesos {0,1} que indicam a presença ou ausência de um termo no documento. As consultas dos usuários são formadas por termos combinados com os operadores lógicos: conjuntivo (*AND*), disjuntivo (*OR*) e negação (*NOT*). Este modelo é do tipo combinação-exata (*exact-match*), ou seja, somente retornam documentos que combinam exatamente com a consulta do usuário. Uma alternativa aos sistemas combinação-exata (exact-match) são os sistema de melhor-combinação (*best-match*), como o modelo de espaço vetorial e o probabilístico (RUTHVEN, I.; LALMAS, M. 2003). O modelo booleano possui a vantagem de necessitar de pouco espaço para armazenamento e a principal desvantagem deste modelo é não prover combinação-parcial (YATES, R. B.; RIBEIRO B. A., 1999).

b) Modelo de espaço vetorial

No modelo de espaço vetorial, os documentos são representados por um vetor, como apresentado na Figura 2, no qual cada dimensão está associada a um termo e o peso do termo é o valor de coordenada desta dimensão (SALTON, G.; WONG, A.; YANG C.S., 1975).

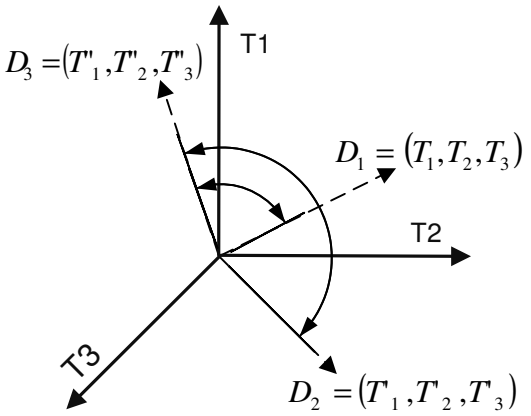

Figura 2 - Modelo de espaço vetorial (SALTON, G.; WONG, A.; YANG C.S., 1975).

Neste modelo, as consultas do usuário também são representadas como um vetor. A similaridade entre o vetor do documento e o vetor da consulta fornece o valor de relevância do documento para a consulta (RUTHVEN, I.; LALMAS, M. 2003). O angulo entre estes dois vetores define a similaridade entre um documento e uma consulta (SALTON, G. E BUCKLEY, C., 1988) e pode ser calculada utilizando-se a Equação 1:

$$sim(d_i, q_j) = \frac{\sum_{k=1}^{n} (Wdt_k . Wqt_k)}{\sqrt{\sum_{k=1}^{n} (Wdt_k)^2 . \sum_{k=1}^{n} (Wqt_k)^2}}$$

Equação 1- Função similaridade (Salton, G.; Buckley, C.,1988).

Sendo:

d_i - o vetor documento;

q_j - o vetor consulta;

Wdt_k - o peso do termo k do documento d_i;

Wqt_k - o peso do termo k da consulta q_j.

A função de similaridade também permite o calculo da semelhança entre documentos (RUTHVEN, I.; LALMAS, M. 2003). Algumas implementações do modelo de espaço vetorial podem utilizar 0 e 1 como valor de peso para os termos,

porém a forma mais comum é o uso de tf x idf (freqüência do termo no documento x o inverso da freqüência do termo nos documentos) (RUTHVEN, I.; LALMAS, M. 2003)[6].

O modelo de espaço vetorial possui como vantagens a simplicidade para implementação, a eficiência para computar a similaridade e comporta-se bem com coleções genéricas (CARDOSO, O. N. P., 2000).

c) **Modelo probabilístico**

No modelo probabilístico, os documentos e consultas também são representados como vetores de termos, sendo a função de similaridade do modelo de espaço vetorial substituída por uma função probabilística (RUTHVEN, I.; LALMAS, M. 2003).

Inicialmente proposto por Maron, M. E. e Kuhns, J. L., (1960), o modelo probabilístico é baseado no princípio probabilístico de ordenação (*Probability Ranking Principle*) e procura estimar a probabilidade de um documento ser relevante para um consulta do usuário (RUTHVEN, I.; LALMAS, M. 2003).

O princípio probabilístico de ordenação baseia-se na hipótese de que a relevância de um documento para uma consulta é independente de outros documentos, sendo que um documento é considerado relevante para uma consulta quando a probabilidade de um documento ser relevante $P(R/D)$ for maior do que a probabilidade deste ser irrelevante $P(\overline{R}/D)$, ou seja: $P(R/D) > P(\overline{R}/D)$ (RUTHVEN, I.; LALMAS, M. 2003).

Para cada documento é definida a similaridade é definida como:

$$sim(D,Q) = \frac{P(R/D)}{P(\overline{R}/D)}$$

Equação 2 – Função de Similaridade (YATES, R. B.; RIBEIRO B. A., 1999).

[6] Veja a seção 2.1.5 para mais detalhes.

Freqüentemente, o teorema de Bayes é utilizado para o desenvolvimento da função de similaridade e um processo iterativo de estimativas da probabilidade de relevância também é considerado para este modelo (YATES, R. B.; RIBEIRO B. A., 1999).

O modelo probabilístico possui como desvantagens a necessidade da estimativa inicial da probabilidade de relevância dos documentos e também não explorar fatores como a freqüência dos termos (*Tf*) nos documentos e inverso da freqüência do termo (idf) (YATES, R. B.; RIBEIRO B. A., 1999).

2.1.4 Seleção dos atributos

Antes de converter os documentos para um modelo de representação, deve ser realizada a seleção dos termos que devem ser utilizados para a representação dos documentos, pois utilizar todos os termos de um documento para sua representação pode prejudicar o desempenho do sistema, consumindo espaço de armazenamento e tempo de processamento e influenciando nos resultados de tarefas como categorização e agrupamento.

Como o tempo de processamento dos métodos de descoberta de conhecimento em texto depende do número de termos utilizados para representação dos documentos e também influenciam a capacidade destes de descriminar documentos, surge o problema de se determinar quais termos devem ser selecionados sem prejudicar os resultados e que este processamento seja realizado em um tempo de processamento aceitável.

Para realizar a seleção dos termos mais representativos vários métodos foram propostos. Entre este os mais comuns são:

a) Filtragem baseada no peso do termo

Na filtragem baseada no peso do termo, os termos que possuem peso inferior a um limiar predefinido pelo usuário são eliminados, não sendo utilizados na representação do documento.

b) Seleção baseada no peso do termo

Na seleção baseada no peso do termo, os *n* termos mais relevantes são selecionados para representar os documentos.

c) Seleção baseada em técnicas de processamento da linguagem natural

Técnicas de processamento de linguagem natural, como análise sintática e semântica, podem ser utilizadas para identificar os termos mais relevantes em um documento, atribuindo-se pesos maiores para termos de determinadas classes.

2.1.5 Calculo do peso dos termos

Existem várias formas para definir o peso de um termo, sendo uma das mais conhecidas a: $tf \times idf$ (*freqüência do termo no documento x inverso da freqüência do termo nos documentos*), definida para o modelo de espaço vetorial por Salton, G.; Wong, A. e Yang, C. S. (1975). A seguir são apresentadas as formas mais comuns de cálculo do peso dos termos.

a) **Freqüência absoluta do termo (*Tfa*)**

A Freqüência absoluta do termo é definida como o número de vezes que um determinado termo ocorre em um documento, não considerando a quantidade de termos no documento.

b) **Freqüência relativa do termo (*Tfr*)**

A freqüência relativa do termo normaliza o valor de relevância do termo, considerando a quantidade de termos de um documento. A freqüência relativa de um termo pode ser definida como:

$$Tfr(x) = \frac{Tfa(x)}{N(x)}$$

Equação 3 - Freqüência relativa do termo (Tfr)

Sendo:

Tfa(x) - a freqüência absoluta do termo x;

N(x) - quantidade de termos do documento x

c) **Freqüência de documentos.**

A freqüência de documentos é o número de documentos no qual o termo ocorre, permitindo distinguir um termo que ocorre em poucos documentos dos termos que ocorrem freqüentemente em muitos documentos. Termos que ocorrem em muitos documentos de uma coleção não possibilitam a diferenciação entre documentos.

d) Inverso da freqüência do termo na coleção (Inverse document frequency - Idf)

O inverso da freqüência de um termo (t_i) em uma coleção que possui *N* documentos é definido como:

$$idf_i = \log \frac{N}{n_i}$$

Equação 4 - Inverse document frequency

Sendo:

N - o número de documentos de uma coleção;

n_i - o número de documentos que contem o termo i.

e) Tfr x Idf

O uso de Tf x Idf como valor do peso do termo permite ao sistema de RI aumentar a importância de termos freqüentes em poucos documentos e que são relativamente raros na coleção de documentos como um todo, diminuindo a importância de termos que aparecem em muitos documentos (SALTON, G.; WONG, A. e YANG, C. S. 1975), o que é interessante pois termos de baixa freqüência no conjunto de todos documentos são mais discriminantes. Assim o peso do termo pode ser definido como:

$$Wi = Tfr_{ti,d} \times Idf_i$$

Equação 5 – Peso do termo.

Como os valores de pesos dos termos nestes métodos de cálculo dependem do número de documentos e do número de termos, o valor do peso de um termo é apenas válido para uma coleção de documentos enquanto esta não for alterada.

2.1.6 Avaliação dos resultados

A eficiência dos sistemas de recuperação de informação é geralmente avaliada utilizando-se métricas de precisão e abrangência (RIJSBERGEN, C. V., 1979). Outra métrica de avaliação é a *f-measure*. Estas métricas de avaliação são baseadas no conceito de relevância de um documento, o que pode ser considerado subjetivo, pois usuários diferentes poderiam avaliar diferentemente a relevância de um documento. Porém, estas diferenças na classificação da relevância não invalidam testes feitos utilizando-se coleções de documentos feitas especificamente para teste, que são formadas por documentos bem conhecidos e classificados adequadamente (SALTON, G. 1983).

a) **Abrangência**

Abrangência é definida como a proporção entre a quantidade de documentos relevantes recuperados e a quantidade total de documentos relevantes (YANG, Y.; LIU, X., 1999).

$$A = \frac{Rr}{Tr}$$

Equação 6 - Abrangência.

Sendo:

Rr - quantidade de documentos relevantes recuperados;

Tr - quantidade total de documentos relevantes.

b) **Precisão**

Precisão é definida como a proporção entre a quantidade de documentos relevantes recuperados e a quantidade total de documentos recuperados (YANG, Y.; LIU, X., 1999).

$$P = \frac{Rr}{Trec}$$

Equação 7 - Precisão.

Sendo:

Rr - o número de documentos relevantes recuperados;

Trec - o número de documentos recuperados.

c) F-measure

A avaliação dos resultados utilizando-se métricas precisão e abrangência isoladamente, pode resultar em uma avaliação incorreta dos resultados pois quanto maior a precisão do sistema, menor será a abrangência e quanto maior a abrangência, menor será a precisão (YANG, Y., 1999).

A métrica *F1-measure*, definida por Rijsbergen (1979), é uma forma comum de avaliar-se o desempenho de um sistema utilizando um valor único, balanceando os valores de precisão e abrangência (YANG, Y., 1999).

$$F1 = \frac{2AP}{(A+P)}$$

Equação 8 - F1 – measure (RIJSBERGEN, C.V,, 1979).

Uma forma mais geral de *F-measure* que permite diferentes pesos para precisão e abrangência é definido de acordo com YANG, Y. (1999):

$$f = \frac{(\beta^2 + 1).P.A}{\beta^2(P + A)}$$

Equação 9 - f – measure (YANG, Y., 1999)

Sendo:

β – parâmetro para diferenciar os pesos da precisão (P) e abrangência (A);

P – precisão;

A – abrangência.

2.2 Mecanismos de busca

2.2.1 Requisitos para um mecanismo de busca na Web

Existem algumas características esperadas ou desejadas em um mecanismo de busca na Web para que estes respondam de forma efetiva ao desafio de localizar na Web a informação desejada pelo usuário. Em Hu, W. C. et al. (2001), são identificados alguns dos requisitos para um mecanismo de busca:

a) Efetividade em localizar e qualificar os documentos;
b) Eficiência dos algoritmos de busca e "ranking" do sistema;
c) Sistemas não tendenciosos no acesso as páginas;
d) Resultados úteis e expressivos;
e) Dados sempre atualizados;
f) Abrangência (Quantidade de documentos indexados);
g) Adaptação do sistema às consultas dos usuários.

2.2.2 Tipos de mecanismos de busca

Em relação à interação com o usuário, basicamente existem dois tipos de mecanismos de busca:

a) **Listas classificadas** *(ClassifiedLists)* (PAGE, L. et al., 2001).

Neste tipo de mecanismo de busca, os documentos são classificados e apresentados em listas, previamente organizadas em uma estrutura hierárquica de classes. Os mecanismos de listas classificadas também são chamados de Sistemas de Diretórios. Exemplos destes mecanismos são Yahoo (YAHOO) e Open Directory Project (OPEN).

b) **Mecanismo de busca baseado em consulta** *(Query-based engines)* (PAGE, L. et al., 2001).

Mecanismo de busca baseado em consulta utiliza o texto fornecido pelo usuário para realizam pesquisa na base de dados e retornam uma lista ordenada documentos, que contenham o texto fornecido. Exemplos destes mecanismos são Altavista (ALTAVISTA), Google (GOOGLE) e Nutch (NUTCH).

Considerando-se a forma como obtém as informações sobre os documentos e os classifica, os mecanismos de busca podem ser classificados como:

a) **Autônomos:** Todas as tarefas são realizadas automaticamente pelo computador;
b) **Não-autônomos:** Pessoas realizam as tarefas de localizar e classificar os documentos.

2.2.3 Tecnologias dos mecanismos de busca

Considerando vários aspectos tecnológicos, segundo (HU, W. C. et al., 2001), as tecnologias de pesquisa utilizadas pelos mecanismos, podem ser agrupadas em seis diferentes categorias:

a) Exploração dos Hyperlinks;
b) Recuperação de Informação;
c) Metabuscadores;
d) Baseado em SQL;
e) Pesquisa por conteúdo multimídia;
f) Outras.

A seguir será apresentada uma breve descrição destas técnicas.

2.2.3.1 Exploração dos Hyperlinks

Hyperlinks podem ser uma importante fonte de informação para os algoritmos de categorização. Esta técnica é baseada na identificação de dois importantes tipos de páginas para um determinado tópico:

- Páginas *autoridade*, que fornece importante fonte de informação para um tópico;
- Páginas *hub*, que fornece uma coleção de hyperlinks para páginas autoridade.

Maiores detalhes são apresentados na seção 4.1.4.4.1.

2.2.3.2 Recuperação de Informação

Técnicas provenientes da área de Recuperação de Informação são amplamente utilizadas na pesquisa na Web. *Relevance feedback* e *data clustering* são duas das técnicas mais utilizadas em mecanismos de busca:

- Na técnica *Relevance Feedback* (retroalimentação por relevância), no qual os itens selecionados do resultado de uma consulta são utilizados para construir uma consulta mais precisa;
- Data Clustering, que procura melhorar o resultado de uma consulta através do agrupamento de páginas similares.

2.2.3.3 Metabuscadores

Metabuscadores são mecanismos de busca que utilizam vários outros mecanismos de busca simultaneamente para realizar uma pesquisa mais ampla e retornam os resultados ordenados e formatos para o usuário. Como é difícil catalogar todos os documentos disponíveis na Web, a idéia destes mecanismos de busca é permitir pesquisas em um conjunto de documentos muito maior do que seria possível para um único serviço de busca realizar e aumentar as probabilidades de se encontrar o documento desejado. Tipicamente, os metabuscadores não compilam ou catalogam os documentos. Exemplos destes mecanismos são Metacrawler (METACRAWLER) e Kartoo (KARTOO).

2.2.3.4 Baseado em SQL

Este trata a Web como uma grande base de dados, na qual as páginas são registros e uma linguagem similar ao SQL é utilizada para fazer as consultas.

2.2.3.5 Pesquisa por conteúdo multimídia

Com a difusão de grande quantidade de conteúdo multimídia na Internet, vários mecanismos de busca foram desenvolvidos para permitir a procura por documentos multimídia de áudio, imagens e vídeo. Estes geralmente utilizam o nome do documento ou meta dado para pesquisa. Um exemplo de mecanismo de busca que

analisa o conteúdo para realizar a consulta é o Speechbot (SPEECHBOT), que utiliza reconhecimento de discurso para converter áudio em texto.

2.2.3.6 Outras

Aqui são incluídos os métodos que não se enquadram em nenhum dos grupos citados anteriormente, como: pesquisa focada em documentos XML ou tópico, uso de linguagem natural para interface com o usuário ou qualquer outra tecnologia para tornar os sistemas de busca mais eficientes.

2.2.4 Formas de consultas

Do ponto de vista dos usuários de mecanismos de busca existem duas formas de se realizar uma busca na Web. Para mecanismos de busca do tipo Lista Classificada, as pesquisa são feitas através da seleção de classes e subclasses em uma estrutura de arvore, na qual os documentos estão organizados, até se encontrar o documento desejado. Para os mecanismos de busca baseados em consulta, o usuário elabora uma consulta, geralmente um conjunto de palavras chaves, a submete ao mecanismo de busca, que retorna uma lista de documentos ordenados pela sua relevância. Porém, para os desenvolvedores de mecanismo de busca, as consultas dos usuários podem ser vistas de uma forma mais genérica, como identificado por (KLEINBERG, J. M, 1999). As consultas podem ser de três tipos básicos:

a) Consulta específica;
b) Consulta ampla;
c) Consulta por página similar.

Estas formas de consulta apresentam vários desafios para um mecanismo de busca, como:

a) **Para as consultas específicas:** O problema de se identificar páginas que contém uma informação específica, as quais geralmente são poucas, em meio a um grande número de páginas com informações similares;
b) **Para as consultas amplas:** O problema encontrado, quando tratando este tipo de consulta, é o grande número de páginas relevantes encontradas, o que é difícil de ser tratado por uma pessoa;
c) **Para as consultas por página similar:** Quando utilizando este tipo de consulta surgem problemas como: relacionar uma página de referência à outra página similar e mesmo diferente entendimentos por parte dos usuários a respeito do que seja uma página similar.

Os mecanismos de busca também podem fornecer mecanismos que auxiliem na formulação das consultas, sendo os mais utilizados tesauros e relevance feedback:

a) **Tesauros:** Segundo definido em (FERREIRA, A. B. H., 2004), tesauro é um "vocabulário controlado e dinâmico de descritores relacionados semântica e genericamente, que cobre de forma extensiva um ramo específico de conhecimento".

   Os tesauros fornecem relacionamentos do tipo:

   - Termo mais abrangente ↔ Termo mais específico.
   - Termo padrão ↔ Termos sinônimo.

   Tais relacionamentos permitem aos usuários identificarem os termos mais apropriados para uma consulta ou definirem consultas mais abrangentes ou específicas.

b) ***Relevance Feedback*** **(Retroalimentação por Relevância):** Nesta técnica, os documentos de uma consulta considerados mais relevantes pelo usuário serão utilizados para construir uma nova consulta mais precisa.

2.2.5 Métodos de aquisição de conteúdo

Existem basicamente duas formas para um mecanismo de busca incluir documentos a sua base de dados, que são (PAGE, L. et al., 2001):

- Através de submissão do documento;
- Ou através do uso de agentes autônomos, também chamados robôs.
-

Mecanismos de busca do tipo Lista de Classificados geralmente utilizam o primeiro método (submissão de documento). Porém, este método apresenta alguns problemas, pois utilizam listas mantidas por pessoas (PAGE, L.; BRIAN, S., 1998):

- Apenas cobrem tópicos populares eficazmente;
- Não são capazes de cobrir todos os tópicos "exóticos";
- São difíceis e caros de manter;
- Base de documentos cresce lentamente.

Mecanismos de busca autônomos geralmente utilizam o segundo método (agentes autônomos), que também podem apresentar alguns problemas como:

- Grande volume de dados para serem coletados e inspecionados;
- Problemas com a análise do conteúdo, devido a páginas com erros.

2.3 Arquitetura de um mecanismo de busca autônomo

Os mecanismos de busca autônomos são compostos por uma variedade de subsistemas, que podem variar em número, função, implementação e nomenclatura. Para descrever a arquitetura dos mecanismos de busca baseado em consulta de forma satisfatória, é apresentado, na Figura 3, um modelo de referência da arquitetura de um mecanismo de busca.

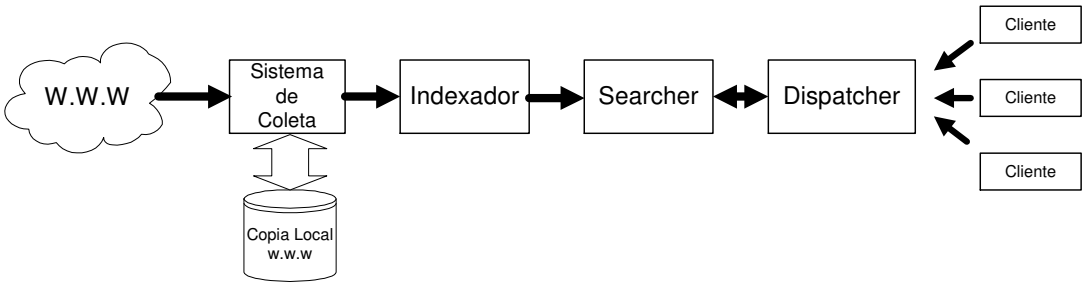

Figura 3 - Arquitetura de referência de mecanismos de busca (RISVIK, K. M.; AASHEIM, Y.; LIDAL, M., 2003).

O modelo de referência da arquitetura de um mecanismo de busca é composto basicamente por quatro módulos principais:

a) O Sistema de coleta é o modulo que realiza a coleta de documentos da Web para permitir que sejam pesquisáveis;
b) O Indexador constrói um índice dos documentos a partir dos documentos adquiridos pelo Sistema de coleta;
c) O *Searcher* recebe do *Dispatcher* a consulta do usuário, então realiza a pesquisa no índice gerado pelo Indexador e retorna os resultados para o *Dispatcher*;
d) *Dispatcher* recebe a consulta do usuário e a distribui para vários *Searcher*, após receber os resultados dos *Searcher*, compila este em uma única lista, ordenando os resultados pelo valor de relevância, e então retorna o resultado ao usuário.

Na literatura sobre mecanismos de busca, existe uma grande variação na nomenclatura utilizada para os módulos, bem como, na sua funcionalidade e nos seus submódulos. A seguir será apresentada uma descrição um pouco mais detalhada de cada um destes módulos.

2.3.1 Sistema de Coleta

O Sistema de Coleta é o componente do sistema que agrega os documentos da Internet, ou seja, faz uma cópia local destes. Isto é normalmente executado em paralelo por várias máquinas.

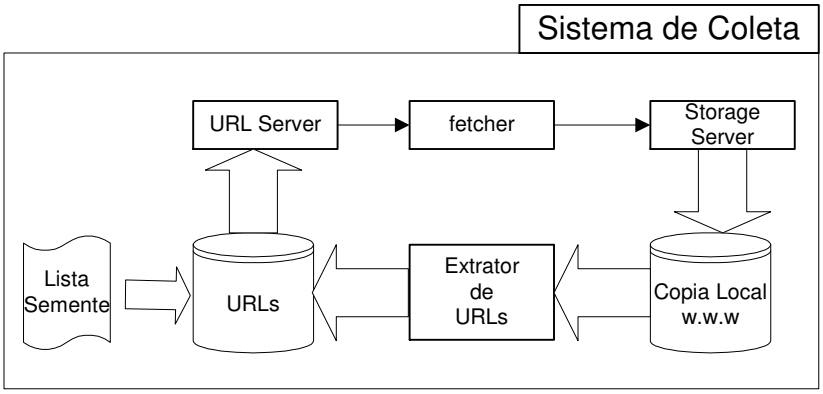

Figura 4 - Arquitetura geral de um Sistema de Coletar

O Sistema de coleta pode ser divido em quatro componentes, como apresentado na Figura 4, cada um com uma atividade específica:

- **URLServer:** Extrai as URLs de uma base de dados e utiliza um ou mais *Fetchers* para realizar a transferência do documento.
- **Fetcher:** Também chamado *spider*, é o módulo que efetivamente executa a transferência e cópia local dos documentos.
- **StoreServer:** Componente que armazena os documentos no repositório, normalmente os documentos são compactados. Para todo documento armazenado é associado a um identificador de documento.
- **Extrator de URLs:** Realiza as seguintes tarefas: extrai as URLs dos documentos, converte URLs relativas para URLs absolutas, faz a associação entre URL, documento e texto ancora, e os armazena na base de dados de URLs.

2.3.2 Indexador

O Indexador analisa os documentos presentes no repositório, convertendo cada um em um conjunto de registro de ocorrência de palavras. Cada registro contém uma palavra e informações relevantes sobre esta, como: posição no documento, tamanho aproximado da fonte e capitalização. O Indexador então armazena estes registros na base de dados de índice, ordenados pelo identificador do documento. O Indexador pode incorporar as funções do Extrator de URLs, evitando que os documentos sejam analisados duas vezes. A Figura 5 apresenta a arquitetura do indexador.

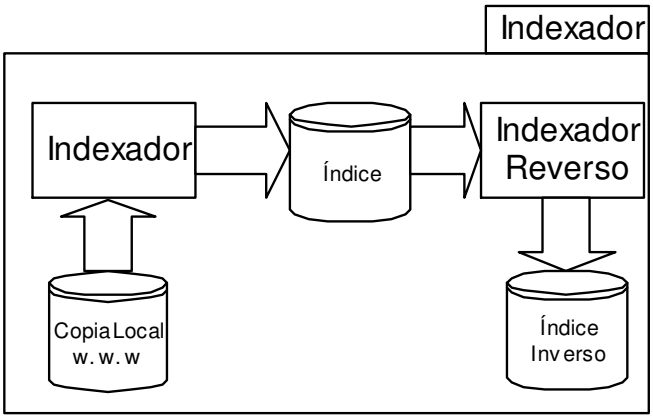

Figura 5 - Arquitetura geral do Indexador.

A partir dos dados armazenados no índice, os quais estão ordenados pelo identificador do documento, o Indexador reverso gera o índice invertido, que associa palavras aos documentos que as contém, como exemplificado na Figura 6. Este também gera um léxico para pesquisa, o qual é composto por uma lista de palavras e sua localização no índice invertido.

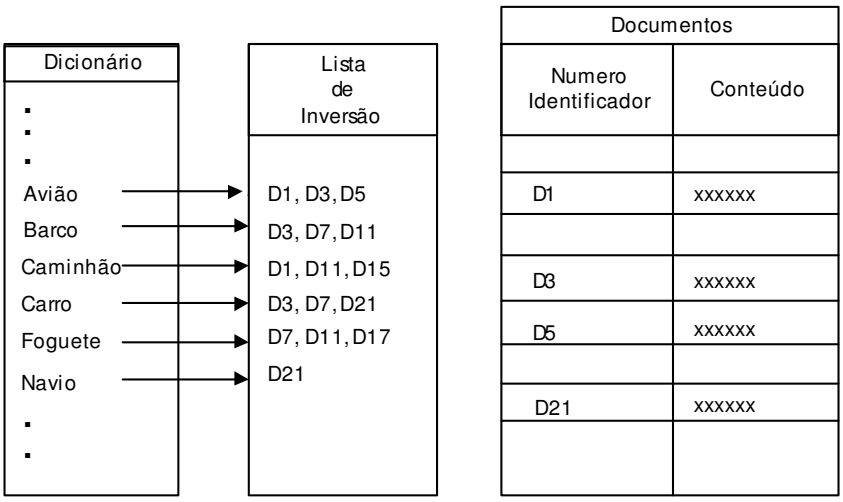

Figura 6 - Índice invertido.

Duas técnicas geralmente utilizadas em sistemas de recuperação de informação e também utilizadas pelos mecanismos de busca na Web são:

- A não inclusão de palavras comuns, como artigos e preposição, no índice invertido, que também são excluídas das consultas. Para isto é mantida uma lista de palavras comuns;
- A utilização de algoritmos de normalização de palavras (*stemming algorihm*), que agrupam palavras morfologicamente relacionadas, como todas as formas de um verbo regular da língua portuguesa, em um único termo no índice invertido.

A vantagem de se utilizar estes métodos é a redução do tamanho dos arquivos de índice. Os algoritmos de derivação de palavras também permitem a generalização nas consultas na recuperação dos dados. Estas transformações no texto reduzem a complexidade da representação do documento e permitem mover a visão lógica do documento do texto completo para os termos de indexação (YATES, R. B.; RIBEIRO B. A., 1999). Outro fator importante a ser considerado é o maior custo computacional envolvido no uso do texto completo.

2.3.3 *Ranker*

O *ranker* calcula o valor de relevância para um determinado documento. Este pode utilizar vários métodos para definir a relevância de um documento, como: as citações feitas em outros documentos ou informações de uso[7]. A Figura 7 apresenta o relacionamento do ranker com outros componentes do sistema.

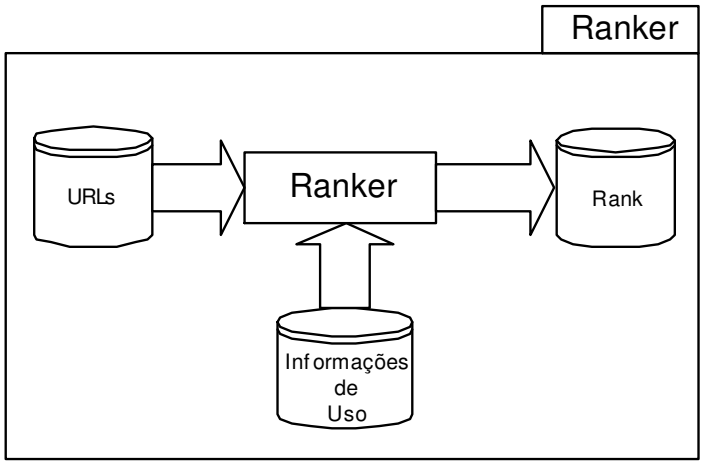

Figura 7 - Arquitetura geral do Ranker

2.3.4 *Searcher*

O *Searcher* realiza a pesquisa para uma dada consulta. Para isto, utiliza o léxico gerado pelo Indexador, o índice invertido e os valores de relevância para gerar uma lista de documentos relevantes à consulta do usuário, como mostra a Figura 8.

[7] Veja o seção 4.1.4.1 para mais detalhes.

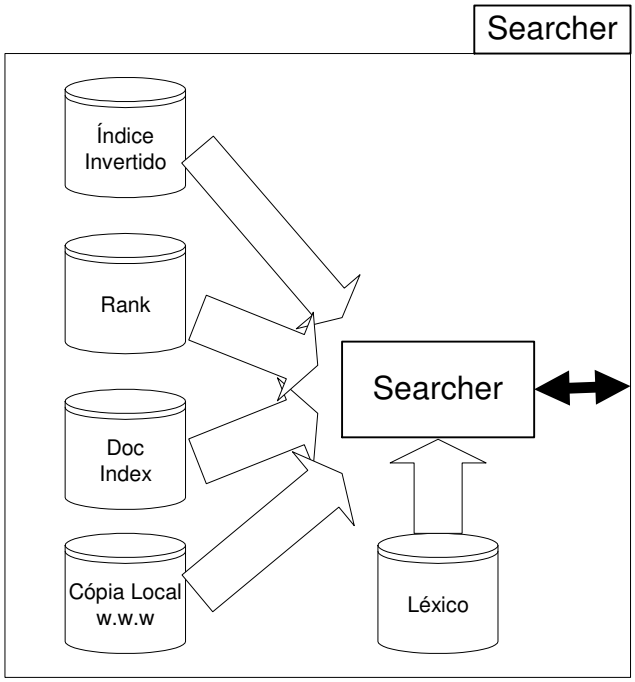

Figura 8 - Arquitetura geral do Searcher

Normalmente, são utilizados vários *Searchers* para tratar uma consulta, sendo função do *Dispatcher* consolidar os resultados em uma única lista e retorná-la ao usuário.

2.4 *Dispatcher*

O *dispatcher* é o componente que realiza a interface entre o usuário e o *searcher*, como apresentado na Figura 9, possuindo três funções básicas:

a) Distribuir as consultas do usuário entre os vários *searchers*, o que inclui a conversão de representação da consulta do usuário para representação interna;
b) Receber os resultados dos *searchers* e consolidá-los em uma lista única;
c) Realiza a interface com o usuário para consulta e apresentação dos resultados.

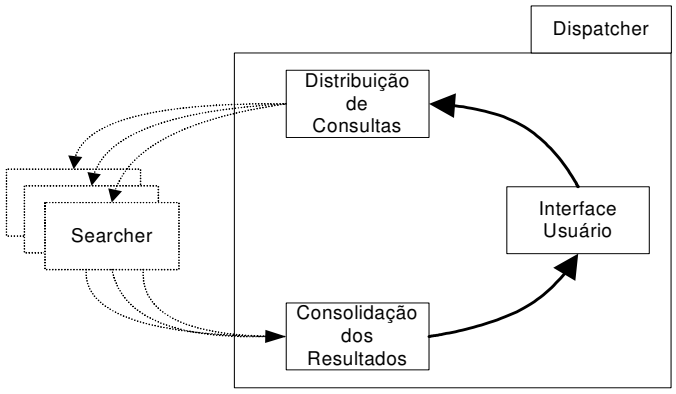

Figura 9 - Arquitetura geral do Dispatcher

Em Yates, R. B. e Ribeiro B. A. (1999) é observado que a interface com o usuário deve suportar dois aspectos distintos do processo de acesso à informação: a busca e recuperação dos dados e a análise e síntese dos resultados. Yates, R.B. e Ribeiro, B.A. (1999) também sugerem alguns princípios de projeto para a interface com o usuário:

- Oferecer retorno informativo;
- Reduzir a necessidade de memorização pelo usuário;
- Prover interfaces alternativas para o usuário.

A interface deve contemplar as necessidades e expectativas tanto de usuários experientes quanto de usuários inexperientes. Portanto, deve possuir duas ou mais formas de apresentação da interface. Deve ser provida ao usuário uma interface intuitiva, que reduza a necessidade de aprendizado e memorização, fornecendo funcionalidade como: geração automática de listas de documentos selecionados; sugestão de novas consultas, baseadas no contexto das consultas anteriores e nos documentos selecionados pelo usuário; pesquisa em um subconjunto de documentos, que poderiam ser os documentos retornados em uma consulta previa ou documentos de um tipo determinado, como documentos de texto, música ou figuras.

3 MECANISMO DE BUSCA NUTCH

3.1 Mecanismos de busca

Existem diversos projetos de mecanismos de busca de código aberto (*open source)*, os principais: WebGlimpse (WEBGLIMPSE), Ht://Dig (HTDIG), Swish-e (SWISH-E), MnogoSearch (MNOGOSEARCH) e Nutch (NUTCH). Estes mecanismos de busca apresentam diferentes estados de desenvolvimento e diferentes licença de uso, como apresentado na Tabela 1, adaptado de (KHARE, R. et. al., 2004):

Tabela 1 - Mecanismos de Busca: Licença e estado de desenvolvimento.

	Licença	**Estado de Desenvolvimento**
WebGlimpse	Comercial (*Permite o uso não lucrativo*).	Ativo
Ht://Dig	GPL	Inativo
SWISH-E	GPL/LGPL	Inativo
MnogoSearch	Comercial(Windows) GPL(Unix)	Ativo
Nutch	Apache	Ativo

Além disso, estes mecanismos de busca apresentam várias características próprias e que são relevantes para a escolha de um sistema de busca que ofereça suporte para desenvolvimento e testes de sistemas de programação de contextualização. A seguir é apresentada uma tabela adaptada de (KHARE, R. et. al., 2004), comparando algumas destas características:

Tabela 2 - Mecanismos de busca: Características estendidas.

	Coleta de dados	Análise de hyperlinks para Relevância	Sistema de armazenamento	Outras características
WebGlimpse	Sistema de arquivo local	Não	Sistema de arquivos local	Perl/C Unix
Ht://Dig	Internet	Não	Sistema de arquivos local	C++ Unix
SWISH-E	Internet	Não	Sistema de arquivos local	CGI/Perl/C
MnogoSearch	Internet e sistema de arquivo local	Sim	SQL	CGI/Perl/C Window/Unix
Nutch	Internet e sistema de arquivo local	Sim	Sistema próprio local ou distribuído	JSP/Java Multiplataforma

O mecanismo de busca Nutch foi escolhido como mecanismo de busca para teste e desenvolvimento, pois apresenta algumas características que o torna a opção mais interessante, como: possuir código aberto e não comercial, desenvolvimento ativo, boa documentação, mantém uma base de dados de URLs de fácil acesso entre outras características que serão apresentadas nas próximas seções.

3.2 Nutch

Nutch é um mecanismo de busca para Web, sendo um projeto de código aberto, desenvolvido em linguagem Java e que utiliza a biblioteca para indexação e recuperação de informação Apache Lucene. Lucene fornece uma API para indexação e pesquisa em texto.

Ambos, Nutch e Lucene, são projetos da Apache Software Foundation, fazendo parte do grupo Jakarta de desenvolvimento de programas em Java para servidores.

As principais características da biblioteca do Lucene (CUTTING, D., 2004; Apache Lucene Project, 2005a; Apache Lucene Project, 2005b) são:

- **Coleta de documentos:** Lucene concentra-se na indexação incremental de texto e recuperação de textos. A coleta dos documentos fica a cargo da aplicação. No caso do Nutch, este implementa o sistema de coleta de páginas na Web;
- **Fontes de dados:** Lucene faz a abstração da fonte de dados (arquivos, registros, etc.), o que permite aos desenvolvedores incluírem facilmente novas fontes de dados, além de arquivos;
- **Formato de arquivos:** Lucene suporta vários formatos de arquivos, além de texto puro e HTML, e possui um mecanismo de filtros que permite a inclusão de suporte para outros formatos de arquivos;
- **Indexação incremental:** Lucene permite a adição de novos documentos a um índice, sem a necessidade de reindexação de todos os documentos;
- **Rótulo de conteúdo:** Lucene suporta a rotulação do conteúdo considerando um documento como uma coleção de campos, formados pelo par <*rotulo, valor*>. Assim os termos presentes em um documento podem ser rotulados. Por exemplo, um termo presente no título pode ser rotulado com *title* e um termo no corpo do texto como *body*. Isto permite a realização de consultas específicas por campos;
- **Processamento de termos irrelevantes (*stop-word*):** O Lucene fornece suporte à remoção de termos irrelevantes através de um mecanismo de filtragem dos dados de entrada;
- **Normalização morfológica:** O Lucene não realiza normalização morfológica. Este fornece um mecanismo de filtragem dos dados de entrada que permite a inclusão de métodos de normalização morfológica. O nutch também não utiliza normalização morfológica;

- **Consultas:** Lucene suporta vários tipos de consulta, que inclui consultas booleanas, com diferenciação de campos, com busca aproximada dos termos e outras;
- **Concorrência:** Lucene permite que várias consultas sejam realizadas simultaneamente nos índices e também permite que consultas sejam realizadas enquanto o índice é atualizado;
- **Idioma:** O mecanismo de filtro do Lucene permite que os desenvolvedores incluam suporte para um idioma específico.

3.3 Arquitetura Nutch

O Nutch possui uma arquitetura modular que utiliza uma *API* para *"plug-in"* que permite a inclusão de módulos como, por exemplo, módulos para análise de documentos, protocolos de comunicação e processamento das consultas.

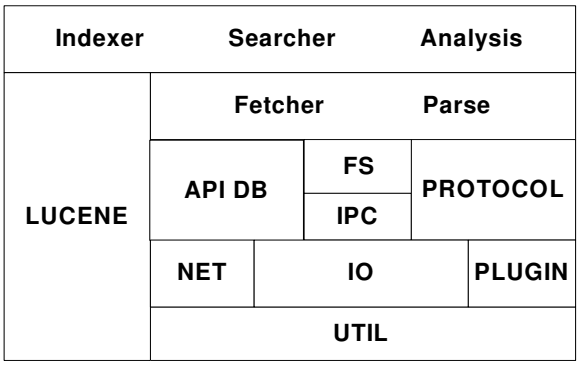

Figura 10 - Diagrama de Dependências do Nutch (KHARE, R. et. al, 2004).

Os principais componentes do Nutch, como apresentados na Figura 10, são (KHARE, R. et. al., 2004):

a) **Fetcher (Sistema de coleta):** responsável pela coleta de páginas e a extração de hyperlinks;
b) **Indexer (Indexador):** responsável pela indexação do texto das páginas;

c) **Searcher:** A partir de uma consulta de um usuário, o Searcher realiza a recuperação das páginas mais relevantes, gera o sumário e retorna os resultados para o usuário.

d) **Base de Dados (DB):** Possui dois componentes, como apresentado na Figura 11:

- WebDB: que armazena informações sobre a estrutura de hyperlinks da Web;
- Segmentos: que armazenam os documentos e arquivos de índices.

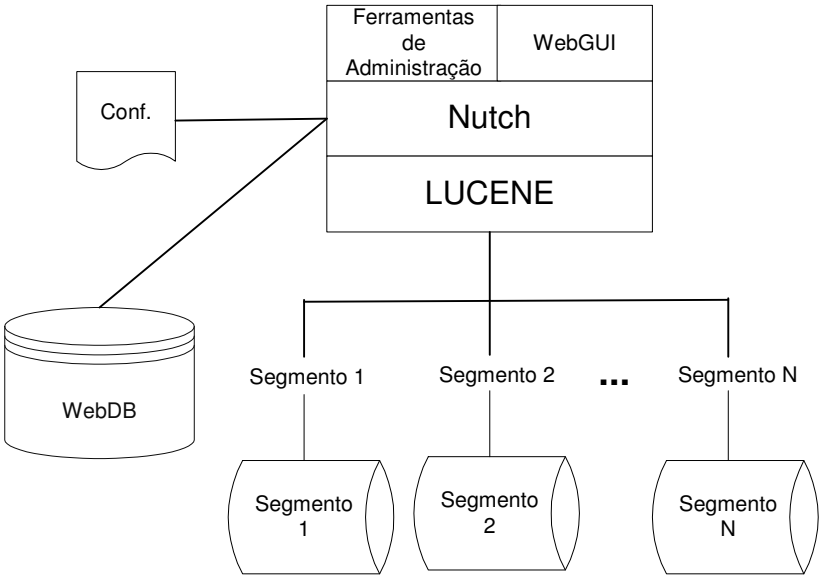

Figura 11 - Arquitetura Nucth.

3.3.1 Processo de coleta e indexação

O processo de coleta e indexação de páginas do Nutch pode ser dividido em três etapas:

a) Iniciação;

b) Coleta de páginas;

c) Indexação das páginas.

A Figura 12 apresenta um diagrama do processo de coleta e indexação.

3.3.1.1 Iniciação

A iniciação é composta pela criação da WebDB, que armazena a URL das páginas a serem coletadas[8] e inclusão de algumas URLs na WebDB para que o processo de coleta possa iniciar.

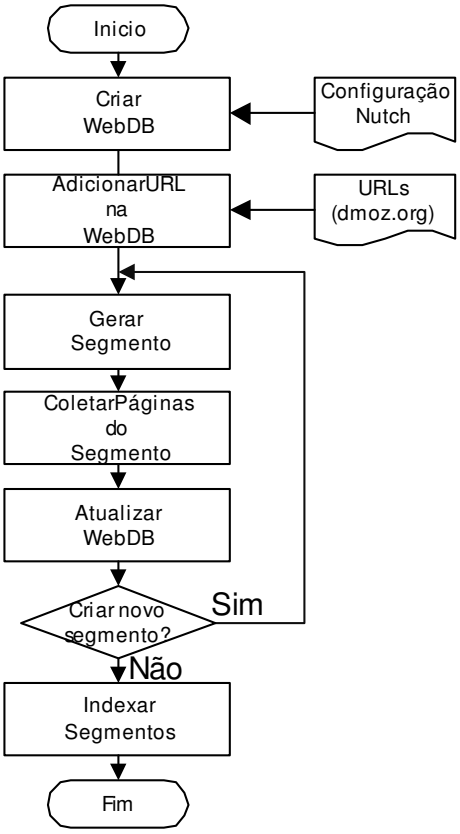

Figura 12 - Processo de coleta e indexação do Nucth.

3.3.1.2 Coleta das páginas

O Nutch utiliza particionamento por documento, ou seja, um servidor contém todos os registros de um determinado documento. Assim, as consultas devem ser enviadas para todos os servidores independentemente dos termos utilizados na consulta (KHARE, R. et. al., 2004). O Nutch constrói vários pequenos índices,

[8] Veja a seção 3.3.3 para mais detalhes.

chamados segmentos de índice, e periodicamente os une em um único índice. Para cada novo documento adicionado é criado um novo segmento de índice, que posteriormente é unido a outros segmentos (KHARE, R. et. al., 2004).

Para melhorar a utilização de espaço em disco, o Nutch remove arquivos duplicados dos segmentos. Para isto, o Nutch gera uma lista de todas as páginas de todos os segmentos ordenadas pelo valor de hash md5 da página. Assim as páginas duplicadas podem ser identificadas e as cópias mais velhas são removidas.

3.3.1.3 Indexação das páginas

O Lucene realiza a indexação do texto para o Nutch. O Nutch utiliza o recurso de indexação de múltiplos campos do Lucene para permitir consultas com palavras e frases por campos, como por exemplo, pesquisar termos na URL, no texto ancora, no título ou no texto do corpo da página (KHARE, R et. al., 2004).

O Lucene provê uma estrutura completa para indexação de texto, porém não possui algumas funcionalidades necessárias a um mecanismo de busca como, por exemplo, um sistema de coleta de páginas na Web, uma base de dados de *hyperlinks* ou interface com o usuário. Adicionalmente, o Nutch implementa uma base de dados de hyperlinks para suporte a coleta de páginas e métodos de análise de hyperlinks, e uma base de dados de páginas para armazenamento das páginas coletadas para futura indexação, sumarização e suporte a outras atividades (KHARE, R. et. al., 2004).

3.3.2 Processo de busca

O processo de busca, como apresentado na Figura 13, inicia com a consulta do usuário. A interface com o usuário do Nutch é uma página JSP (*Java Server Page*), que analisa a consulta do usuário em formado de texto e a repassa para o modulo de busca do Nutch no servidor Web.

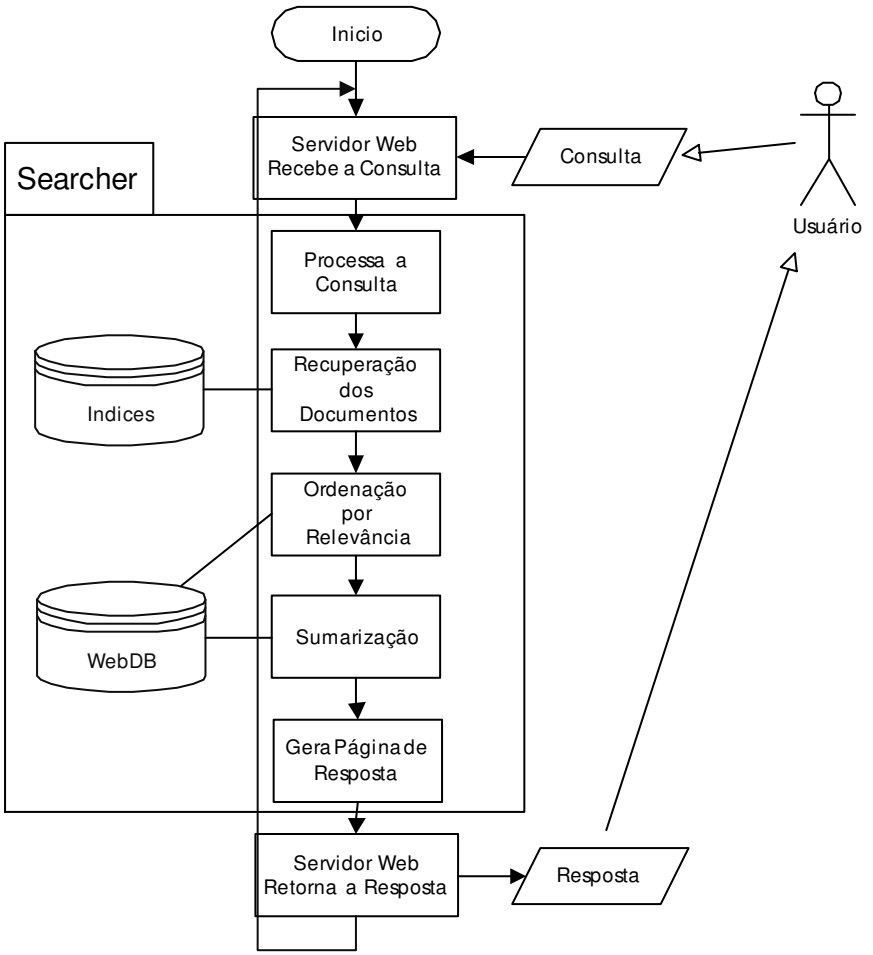

Figura 13 - Processo de Busca.

Se estiver executando em um único servidor o Nutch traduz a consulta do usuário em uma consulta do Lucene que, por sua vez, retorna uma lista de *hits* (conjunto de informações sobre as páginas relevantes) que são convertidas em uma página *html*. Se o Nutch estiver sendo executado em vários servidores a consulta é distribuída entre os vários servidores. Os resultados retornados por estes são consolidados em uma única lista, que é processada e enviada ao usuário (KHARE, R. et. al., 2004).

3.3.3 Estruturas de dados do Nutch

A **WebDB** é uma base de informações sobre a Web mantida pelo Nutch. Armazena informações sobre todas as páginas conhecidas e os *hyperlinks* que estas páginas contêm. Estas informações servem de suporte a tarefas como a coleta de páginas e análise dos *hyperlinks* para ordenação por relevância.

As funcionalidades fornecidas pela API de acesso a WebDB são:

- Adição e remoção de páginas;
- Recuperação de páginas pela URL, MD5;
- Adição e remoção de *hyperlinks*;
- Recuperação de todas as URL;
- Recuperação de *hyperlinks* de uma página e do *hyperlinks* de páginas que apontam para uma determinada página;
- Número de páginas e URLs na WebDB.

Para tornar o acesso a estas informações mais rápido, a WebDB é formada por quatro arquivos de dados. Os quatro arquivos armazenam:

a) Páginas ordenadas por URL;
b) Páginas ordenadas por MD5;
c) Hyperlinks ordenados por URL;
d) Hyperlinks ordenados por MD5.

A replicação dos dados sobre páginas e *hyperlinks* aumenta o espaço necessário em disco para o armazenamento dos dados, porém diminuem o tempo de acesso as informações.

Na WebDB são armazenadas algumas informações relacionadas a cada página. A Tabela 3 apresenta as informações mantidas para cada página:

Tabela 3 - Informações sobre páginas armazenadas na WebDB (NUTCH 0.7.1 API).

Campo	**Descrição**
URL	A URL da página. Esta é a chave primária.
ID	O identificador (hash MD5 do conteúdo) da página.

DATE	Data em que a página deve ser coleta novamente.
RETRIES	Número de falhas tentando coletar a página.
INTERVAL	Período entre coletas em dias.
SCORE	Valor de relevância.
NEXTSCORE	Utilizado durante o calculo do valor de relevância.

A Tabela 4 apresenta as informações armazenadas para cada *hyperlinks*.

Tabela 4 - Informações sobre hyperlinks armazenadas na WebDB (NUTCH 0.7.1 API)

Campo	**Descrição**
VERSION	Versão desta entrada.
FROM_ID	O identificador (*hash* MD5) da página de origem, onde esta o hyperlink.
DOMAIN_ID	O identificador (*hash* MD5) do domínio da página da página de origem.
TO_URL	URL do destino do *hyperlink*.
ANCHOR	Texto ancora do *hyperlinks*.
TARGET_HAS_OUTLINK	Indica se o documento alvo possui *hyperlinks* de saída.

3.3.4 Estruturas de dados do Lucene

O Lucene é composto por quatro componentes fundamentais, que são (CUTTING, D., 2004):

a) **Índice:** Lucene utiliza índice invertido, ou seja, para cada termo é associada uma lista de documentos;
b) **Documento:** Um documento é uma seqüência de campos;
c) **Campo:** Um campo é uma seqüência nomeada de termos, composta por um par <atributo, valor>, sendo o atributo um nome de campo, como: URL, título, etc.;
d) **Termo:** Um termo é uma seqüência de caracteres. A mesma seqüência de caracteres em dois campos diferente são consideradas como termos diferentes.

Os índices do Lucene podem ser compostos de múltiplos segmentos de índices. Cada segmento é um índice completamente independente.

Os documentos indexados são identificados por um número inteiro. Este número é único apenas internamente a um segmento.

Cada segmento contém os seguintes elementos (CUTTING, D., 2004):

a) O nome dos campos utilizados nos índices;
b) Os campos: Para cada documento é armazenado um conjunto de campos, que são utilizados para armazenar informações auxiliares sobre os documentos, como: URL e título;
c) Dicionário de termos: Que contém todos os termos utilizados em todos os campos indexados;
d) Freqüência dos termos: Para cada termo do dicionário é armazenada a lista de documentos que o contém e a freqüência do termo no documento;
e) Posição dos termos: Para cada termo do dicionário é armazenada a posição deste em cada documento;
f) Fator de normalização: Para cada termo do dicionário é armazenado um valor de normalização para o calculo da relevância do termo.

3.3.5 Campos do índice do Nutch

O Nutch cria um índice para cada um dos campos apresentados na Tabela 5.

Tabela 5 - Campos do índice do Nutch (NUTCH 0.7.1 API).

Campo	Descrição
anchor	Termos do texto âncora
content	Termos presentes no corpo do texto
host	Termos presentes no domínio
title	Termos presentes no título das páginas
url	Termos presentes nas URLs

Utilizando o *plugin index-more,* também é possível incluir no índice os campos apresentados na Tabela 6:

Tabela 6 - Campos extras do índice do Nutch (NUTCH 0.7.1 API)

Campo	**Descrição**
date	Data de criação do documento
contentLenght	Tamanho do documento
contentType	Tipo do conteúdo do documento

3.3.6 Protocolos suportados pelo Nutch

O nutch suporta os seguintes protocolos *file, http e ftp* para coleta de documentos (NUTCH 0.7 API).

3.3.7 Consultas no Nutch

O Nutch permite consultas por palavra e por campos, ou seja, é possível especificar em qual campo a busca será realizada. Ao receber uma consulta, esta é analisada e expandida para os vários campos do Nutch, quando o campo não é especificado (NUTCH 0.7 API). A expansão da consulta é feita para realizar-se a busca nos campos anchor, content e URL. Assim, uma consulta simples para os termos "*carro esporte*" é expandida para:

url:(+carro +esporte +"carro esporte"~p^a)^x
anchor:(+carro +esporte +"carro esporte"~q^b)^y
content:(+carro +esporte +"carro esporte"~p^c)^z

Sendo:

url, anchor e content - campos do índice.
^ - indica o peso do campo na consulta.
~ - indica o valor para busca aproximada.

O Nutch também permite vários tipos de consultas, as quais são baseadas nas consultas do Lucene, como apresentado em (NUTCH 0.7 API):

a) **Query-basic:** Realiza a consulta nos campos *anchor*, *content* e *url*;
b) **Query-more:** Realiza consultas incluindo a data, tamanho e tipo do documento;
c) **Query-site:** Realiza busca pelo campo *site*;
d) **Query-url:** Realiza busca pelo campo *url*.

Antes de realizar a pesquisa para uma consulta, o Lucene transforma uma consulta textual do usuário em uma representação interna, a qual permite a pesquisa. O Lucene suporta vários tipos de consulta, que incluem (APACHE LUCENE 1.4 API):

a) **Booleanas:** Permite o uso de expressões lógicas com os operadores AND, OR e NOT. Esta forma de consulta permite agregar várias consultas em uma consulta mais sofisticada. Por exemplo: carro AND prateado;
b) **Frases:** Permite a busca por frases nos texto. Por exemplo: "edit distance";
c) **Faixa de valores:** Busca documentos que contenham termos que estão dentro de uma faixa especificada. Por exemplo: [carro TO carroça], os termos entre os termos carro e carroça;
d) **Wildcards:** Uma forma de busca aproximada, que permite a substituição de um único carácter ou de vários. Por exemplo: ja?a, que pode retornar java e jaca;
e) **Prefixo:** Permite a busca por termos que iniciem por um prefixo determinado pelo usuário. Por exemplo: java*, que retornaria termos iniciando com java como java e javali;
f) **Fuzzy:** Permite a busca por termos utilizando o algoritmo de distância de Levenshtein. Por exemplo: árvore~0.5, que retorna termos semelhantes à árvore com até três caracteres de diferença.

3.3.8 Formatos de arquivos suportados pelo Nutch

O Nutch é capaz de analisar e indexar os seguintes formatos de arquivo (NUTCH 0.7 API): *txt, html, javascript, ms-power point, ms-word, ms-excel, ms-powerpoint, pdf, rss, rtf, mp3 e flash swf*.

3.3.9 Cálculo de relevância no Nutch.

O Nutch calcula o valor de relevância estático das páginas utilizando um algoritmo similar ao PageRank[9] (KHARE, R. et. al., 2004). Para definir a relevância de uma página é realizado o seguinte procedimento (NUTCH 0.7 API):

a) O cálculo do valor de relevância dinâmico da página em relação a uma determinada consulta, utilizado o esquema do Lucene[10];
b) O cálculo do valor final de relevância, multiplicando o valor de relevância dinâmico da página para a consulta pelo valor de relevância estática da página, calculado previamente.

3.3.9.1 Valor de relevância estático

O valor de relevância estático pode ter seu calculo distribuído entre vários processos. O processo de cálculo da relevância é executado em um laço iterativo composto por três etapas executadas repetidamente, sendo o número de repetições definido pelo usuário. Estas três etapas são (NUTCH 0.7 API):

a) ***InitRound***: Etapa de preparação, na qual a base de *urls* é dividida entre os vários processos;
b) ***ComputeRound***: Etapa na qual é calculada a contribuição ao valor de relevância de cada *hyperlink* da página;

---

[9] Veja a seção 4.1.4.4.2 para mais detalhes.

[10] Veja a seção 3.3.9.2 para mais detalhes.

c) ***CompleteRound***: Etapa na qual os resultados são consolidados, somando-se a contribuição à relevância de todas as referências, utilizando o fator $d = 0.85$[11].

3.3.9.2 Valor de relevância dinâmico do Lucene

O Lucene calcula e armazena em seus índices os valores de *tf* e *idf* para os termos das páginas, o que permite a aplicação de métodos que utilizem estes valores. O valor de relevância para uma consulta, chamado *score* no Lucene, é calculado utilizando-se a expressão (Apache Lucene 1.4 API):

$$score(q,d) = \sum_{t \in q} tf(t,d) * idf(t) * b(t,q) * b(t,d) * lNorm(ft,n) * c(q,d) * qNorm(t,q)$$

Equação 10 - Valor de relevância do termo no Lucene (Apache Lucene 1.4 API).

Sendo:

tf(t,d) – freqüência do termo *t* no documento *d*;

idf(t) – freqüência inversa do termo, que mede a freqüência com que o termo *t* aparece no índice;

b(t,q) – fator de relevância do termo *t* para o consulta *q*;

b(t,d) – fator de relevância do termo *t* para o documento *d*;

lNorm(ft) – Valor de normalização para o campo *ft*;

c(q,d) – fração dos termos da consulta *q* presentes do documento *d*;

qNorm(t,q) – fator de normalização da consulta, que permite comparar o valor de relevância entre consultas.

A implementação padrão do *score* no Lucene utiliza apenas os fatores *tf(t,d), ifd(t), c(q,d) e lNorm(ft)*, os quais são implementados como:

---

[11] Veja seção 4.1.4.4.2 para mais detalhes.

- $tf(t,d) = \sqrt{f(t)}$, sendo *f(t)* a freqüência do termo no documento. Assim, quanto mais freqüente é o termo no documento, maior é a relevância do termo;
- $ifd(t) = \log\left(\frac{n}{df(t)+1}\right)+1$, sendo *n* o número de documentos e *df* a freqüência do termo nos documentos. Assim, quanto maior a ocorrência do termo entre os documentos, menor a relevância do termo;
- $c(q,d) = \frac{nt(q,d)}{mt(q)}$, sendo *nt(q,d)* o número de termos da consulta que o documento contem e *mt(q)* o número de termos da consulta. Assim, documentos que contem mais termos da consulta, terão relevância maior;
- $lNorm(tf) = \frac{1}{\sqrt{numTerms}}$, sendo *numTerms* o número de total de termos em um determinado campo. Assim, um termo em um campo com menos termos é mais relevante.

O esquema para cálculo do valor de relevância adotado pelo Lucene permite que o valor de relevância seja personalizado para aplicações específicas.

Este capítulo apresentou o mecanismo de busca Nutch, utilizado para implementação e testes dos métodos para programação de contextualização. A programação de contextualização no Nutch é apresentada no capítulo 6.

4 MINERAÇÃO NA WEB

Métodos tradicionalmente utilizados em recuperação de informação não são capazes de extrair o conhecimento implícito nos textos e não resolvem problemas como: a sobrecarga de informações, a qual ocorre quando um grande número de documentos é retornado pelos sistemas de recuperação de informação, sendo geralmente muitos destes documentos irrelevantes.

Vários métodos oriundos da área de Mineração de dados (*Data Mining*) vêem sedo utilizados para descoberta e extração de informações úteis da Web. A esta nova área de pesquisa é dado o nome de Mineração da Web (*Web Mining*), como definido em (KOSALA, R.; BLOCKELL, H., 2000): Mineração da Web é "a área de pesquisas, relacionada à mineração de dados, que pode ser definida como sendo o uso das técnicas automáticas de descoberta e extração de informação de documentos e serviços Web".

Os mesmos autores consideram que: Mineração da Web refere-se a todo o processo de descoberta de informação ou conhecimento potencialmente útil e não conhecido previamente, podendo ser visto como uma extensão da área de Descoberta de conhecimento em base de dados (*Knowledge Discovery in Databases*) aplicado a dados extraídos da Web. Porém, segundo Kroeze, J. H.; Matthee, M. C. e Bothma, T. J. D (2003): "Mineração de dados é um passo no processo de descoberta de informação de dados". Assim, neste trabalho, Mineração da Web será considerado como o equivalente a mineração de dados aplicada a Web.

Antes de prosseguir, convêm ser feito o esclarecimento sobre a correlação entre as áreas de Recuperação de Informação (RI), Extração de Informação (EI) e Mineração da Web, pois como estas compartilham técnicas em comum, existe certa confusão sobre a área de atuação de cada uma destas. Assim, como descrito em (KOSALA, R.; BLOCKELL, H., 2000) são feitas as seguintes distinções:

a) **Mineração da Web e Recuperação de Informação:** Mineração do conteúdo dos Documentos da Web (*Web Content Mining*) é vista por alguns como Recuperação Inteligente de Informação na Web, porém o principal foco de RI é a indexação e recuperação de documentos e as únicas tarefas de RI que

podem ser consideradas como de Mineração da Web são a categorização e agrupamento. Segundo este ponto de vista, Mineração da Web pode ser entendida como parte do processo de Recuperação de Informação;

b) **Mineração da Web e Extração de Informação:** Extração de informação tem como principal objetivo extrair informações úteis de documentos e converte-las para que possam ser utilizadas pelos métodos de mineração de dados, enquanto Recuperação de Informação seleciona documentos relevantes. Geralmente EI trabalha em um grau de granularidade mais fino do que Recuperação de Informação, sendo que sistemas de RI podem prover algumas funções de sistemas de EI. Tarefas de EI, como a de sumarização de documentos, podem utilizar técnicas de aprendizado de máquina ou Mineração de dados. Assim, Mineração da Web pode ser compreendida como parte do processo de EI.

Os mesmos autores também sugerem que o processo de Mineração da Web seja decomposto nas seguintes subtarefas:

a) Busca de recursos: tarefa de recuperar os documentos da Web;
b) Seleção e Pré-processamento da Informação;
c) Generalização: descoberta automática de padrões em um sítio ou entre sítios.

Como definido por Kosala, R. e Blockeel, H. (2000), Mineração na Web pode ser categorizada segundo três áreas de interesse: Mineração do Conteúdo da Web (*Web Content Mining*), Mineração da Estrutura da Web (*Web Structure Mining*) e Mineração do Uso da Web (*Web Usage Mining*), que são definidas como:

a) **Mineração do Conteúdo da Web:** É a descoberta de informação do conteúdo textual dos documentos da Web, utilizando técnicas de Mineração de texto, o que inclui tarefas como: categorização, sumarização e agrupamento;
b) **Mineração da Estrutura da Web:** É a descoberta de padrões na estrutura de *hyperlinks* da Web, sendo útil para gerar informações sobre similaridade,

relevância e correlação entre páginas, o que permite a descoberta de comunidades da Web;

c) **Mineração do Uso da Web:** Utiliza os registros de servidores Web para procurar por padrões de navegação dos usuários.

Este trabalho tem como principal foco de interesse as áreas de Mineração do conteúdo e da estrutura da Web. Portanto nas próximas seções será dada ênfase as tarefas de mineração relacionadas. As principais tarefas de mineração identificadas são:

- Sumarização;
- Classificação;
- Categorização;
- Agrupamento;
- Relevância de páginas na Web.

Para realizar estas tarefas, freqüentemente são utilizados métodos que envolvem aprendizado de máquina, que podem ser classificados como métodos de aprendizado supervisionado e não supervisionados, sendo:

- Métodos de aprendizado supervisionado: são os métodos nos quais o aprendizado é realizado em uma etapa de treinamento, utilizando-se exemplos ou modelos;
- Métodos de aprendizado não supervisionados: são os métodos nos quais o aprendizado é não supervisionado, não existindo uma etapa de treinamento, nem exemplos ou modelos.

Os métodos que utilizam aprendizado supervisionado necessitam de um tempo extra para treinamento e este treinamento só é possível se existir um conjunto de exemplos para o treinamento. Estes fatores devem ser considerados no custo e tempo gasto na implantação deste tipo de método.

4.1 Tarefas de Mineração na Web

Nas próximas seções, serão descritas as principais tarefas de Mineração da Web, será dada ênfase às tarefas de categorização, agrupamento e definição de relevância, as quais são de interesse para este trabalho.

4.1.1 Sumarização

A sumarização é a técnica de identificação automática de palavras e frases que melhor descrevem um texto, permitindo gerar um sumário sobre o texto, que deve fornecer uma visão geral sobre o conteúdo do documento (Rino, L.H.M.; Pardo, T.A.S, 2003). Este método também pode ser aplicado após um método de agrupamento para análise do centróide do grupo, que geralmente é utilizado para representar o grupo, pois o centróide corresponde ao conjunto de palavras estatisticamente mais importantes do grupo.

4.1.2 Classificação e Categorização

A classificação é a técnica que identifica a classe a que um documento pertence, geralmente, analisando-se o seu conteúdo. Cada possível classe de documento deve ter sido previamente definida segundo um conjunto de atributos ou através de formulação matemática (RIJSBERGEN, C. V.; 1979). O termo categorização é utilizado para definir a técnica de classificação que identifica o assunto ou tema de um documento através da análise do texto.

Diversos métodos têm sido utilizados para a categorização de documentos, tais como (SEBASTIANI, F.; 2002):

- Classificador Rocchio;
- Classificador Naive Bayes;
- Árvores de Decisão;
- Redes Neurais Artificial;

- Máquinas de Vetores Suporte (*Support Vector Machines - SVM)*;
- kNN (*k - Nearest Neighbor*).

Testes realizados por Yang, Y. (1999) para doze classificadores diferentes, utilizando as versões 3 e 4 da coleção Reuters[12], apresentaram os seguintes resultados: os classificadores kNN, Redes Neural e LLSF (Linear Least Squares) apresentam os melhores resultados, seguidos pelo grupo de classificadores (CLASSI, SWAP-1, RIPPER, Árvores de Decisão e CHARADE) que apresentaram resultados próximos uns dos outros. O classificador Naive Bayes foi o que apresentou os piores resultados. Em (YANG, Y.; LIU, X., 1999) é feita uma reavaliação de alguns dos classificadores, que apresenta melhores resultados para os classificadores SVM e kNN.

A seguir serão descritos alguns dos métodos utilizados em categorização de textos, que são mais freqüentemente citados na literatura.

4.1.2.1 Classificador Rocchio

O algoritmo classificador Rocchio (Hull, D.A.[13], 1994 apud SEBASTIANI, F. 2002) é baseado no método de *relevance feedback* de Rocchio para o modelo de espaço vetorial (JOACHIMS, T., 1997). Existem diversas variações deste algoritmo, as quais diferem nos métodos de seleção dos termos, nos métodos de escolha dos peso das palavras e nos algoritmos de similaridade (SALTON, G.; BUCKLEY, C., 1988). A variação do algoritmo do classificador Rocchio apresentada aqui é uma das mais simples deste tipo de algoritmo para categorização (JOACHIMS, T., 1997).

---

[12] Coleção de textos utilizados para testes de algoritmos de categorização de textos. Os textos foram inicialmente coletados e classificados pelo Carnegie Group, Inc. e Reuters, Ltd.

[13] Hull, D.A. *Improving text retrieval for the routing problem using latent semantic indexing*. In Proceedings of SIGIR-94, 17th ACM international Conference on Research and Development in Information Retrieval (Zürich, Switzerland), p. 278 – 288, 1996.

Este algoritmo baseia-se na suposição de que documentos com vetores de termos similares possuem conteúdo similar (JOACHIMS, T., 1997) e cada documento é representado como um vetor de termos[14].

A idéia básica do algoritmo é construir um vetor protótipo para cada categoria, também chamado de centróide da categoria, utilizando um conjunto de documentos para treinamento (YANG, Y, 1999). Os documentos deste conjunto de treinamento são divididos em categorias predefinidas, cada uma representando um assunto.

O algoritmo do classificador Rocchio aprende como identificar uma categoria combinando os vetores dos documentos de uma categoria do conjunto de treino em um vetor protótipo para a categoria. Este procedimento é repetido para todas as categorias (JOACHIMS, T., 1997). Os vetores protótipos $\overrightarrow{C_i} = (w_1, w_2, w_3, ..., w_t)$ são gerados calculando-se a média dos vetores de todos os documentos exemplos para a categoria, como definido na Equação 11.

$$w_{ki} = \beta \cdot \sum_{d_j \in POS_i} \frac{w_{kj}}{|POS_i|} - \gamma \cdot \sum_{d_j \in NEG_i} \frac{w_{kj}}{|NEG_i|}$$

Equação 11- Vetor protótipo (Sebastiani.,2002).

Sendo:

W_{ki} – o peso do termo t_k no vetor protótipo i;

W_{kj} - o peso do termo t_k no documento d_j;

POS – o número de exemplos positivos;

NEG – o número de exemplos negativos;

β e γ - são parâmetros de controle. Se $\beta = 1$ e $\gamma = 0$, C_i é o centróide dos exemplos positivos.

O classificador Rocchio pode ser utilizado para classificar um novo documento comparando-se o vetor do documento com o vetor protótipo de todas as

[14] Veja a seção 2.1.3 para mais detalhes sobre o modelo de espaço vetorial.

categorias, utilizando a função de similaridade para o modelo de espaço vetorial[15]. O documento será associado à categoria que apresentar o maior valor de similaridade:

$$Cat(d) = \arg\max_{c \in C}(sim(d, c))$$

Equação 12- Categoria de um documento (JOACHIMS, T., 1997) .

Sendo:

Cat(d) – a categoria do documento d.

argmax – a categoria com maior valor de similaridade entre o documento d e os vetores protótipo.

O classificador Rocchio também é chamado de classificador TFIDF, pois utiliza tf x idf (freqüência do termo x inverso da freqüência nos documentos) como o peso dos termos no vetor do documento (JOACHIMS, T., 1997).

Uma fraqueza potencial deste modelo é assumir um centróide por categoria, o que faz com que o classificador Rocchio não tenha bons resultados para uma categoria que forme naturalmente grupos separados (YANG, Y, 1999).

4.1.2.2 Classificador Naive Bayes

O classificador Naive Bayes procura determinar a probabilidade de um documento pertencer a uma determinada categoria. Este método baseia-se na suposição de que as ocorrências de palavras em um documento são independes umas das outras, i.e., a probabilidade condicional de uma palavra dada uma categoria é presumido ser independente da probabilidade condicional de outra palavra para esta categoria (YANG, Y.; LIU, X. 1999). Esta suposição torna o processamento do classificador Naive Bayes mais eficiente, do que abordagens não-Naive Bayes (YANG, Y., 1999).

A idéia básica da abordagem Naive Bayes é utilizar a probabilidade da ocorrência das palavras em uma categoria para estimar a probabilidade de um

[15] Veja a seção 2.1.3.

documento pertencer à categoria, sendo utilizado um conjunto de documentos de treinamento para estimativa destas probabilidades (Yang, Y.; LIU, X., 1999).

O classificador Naive Bayes procura estimar a probabilidade $\Pr(C/d)$ de um documento *d* ser da categoria *C*. O documento *d* será associado à categoria para a qual $\Pr(C/d)$ é maior (JOACHIMS, T., 1997), como mostra a Equação 13.

$$Cat(d) = \arg\max_{c \in C}(\Pr(C/d))$$

Equação 13 - Classificador Neiva Bayes (JOACHIMS, T., 1997).

Sendo:

Cat(d) – a categoria do documento *d*;

argmax $\Pr(C/d)$ - a categoria com o maior valor de $\Pr(C/d)$.

4.1.2.3 Árvores de Decisão

Árvore de decisão é um método de aprendizado de máquina para indução automática de árvores de classificação baseada em dados de treinamento. Aplicada à categorização de textos, define a categoria de cada documento baseando-se na ocorrência da combinação de determinadas palavras no documento (YANG, Y. 1999).

Uma árvore de decisão como classificador de texto é uma árvore na qual os nós internos são rotulados pelos termos. O valor de cada ramo é definido pelo valor do peso do termo e as folhas definem a categoria, veja Figura 14. Como muitos classificadores utilizam um valor binário para os pesos dos termos no vetor de termos do documento, estes utilizam árvores binárias para a categorização dos documentos (SEBASTIANI, F. 2002).

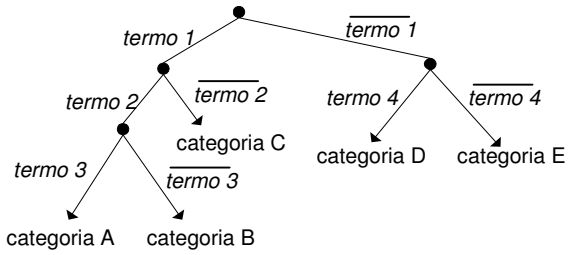

Figura 14 - Árvore de Decisão.

A construção da árvore de decisão é feita através de um processo iterativo composto por três etapas (RICH E.; KNIGHT, K., 1993):

a) Seleção de um subconjunto dos documentos para treinamento;
b) Construção da árvore de decisão, utilizando o subconjunto de documentos para treinamento;
c) Teste de eficiência da árvore em classificar os outros documentos do conjunto de treinamento.

Um passo importante do algoritmo é a escolha dos termos para a partição dos ramos, que geralmente é feita em acordo com um critério de galho de informação ou entropia (SEBASTIANI, F. 2002). Existem vários algoritmos para a construção de árvores de decisão como o ID3 de Quinlan (1983).

O desempenho prático na predição de categorias por árvores de decisão algumas vezes não é bom como os resultados obtidos para o conjunto de documentos de treinamento. Este fenômeno ocorre freqüentemente, quando a árvore de decisão torna-se específica para a categorização do conjunto de treinamento (APTE, C.; DAMERAU, F.; WEISS, S.M., 1998). Muitos dos métodos de geração de árvores de decisão incluem um método de "poda" da árvore para evitar que a árvore torne-se específica para o conjunto de treino (SEBASTIANI, F. 2002).

4.1.2.4 Redes Neurais Artificial

Como mostra a Figura 15, uma rede neural artificial (RNA) como classificador de textos é uma rede composta por (SEBASTIANI, F., 2002):

- Unidades de entrada, representando os termos dos textos;
- Unidades de saída, representando as categorias;
- Unidades intermediárias, com os pesos das conexões representando relações de dependência entre termos e categorias.

Para classificar um documento, os pesos dos termos são “carregados” nas unidades de entrada e a ativação destas unidades é propagada através da rede. Os valores das unidades de saída representam à categoria do texto.

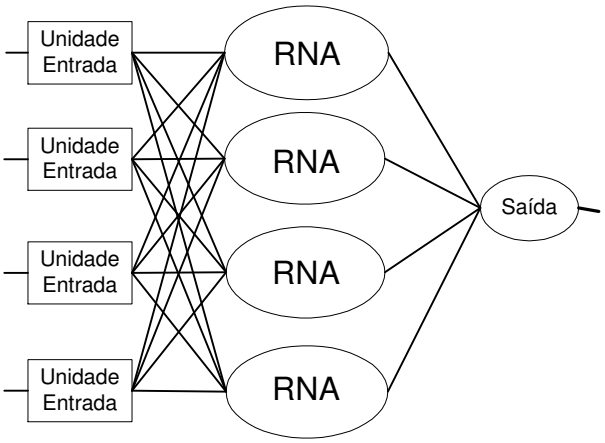

Figura 15 - Rede Neural Artificial.

O treinamento de uma RNA é tipicamente realizado utilizando-se um algoritmo de retropropagação que, no caso de um erro de classificação de um dos documentos do conjunto de documentos de treinamento, retropropaga o erro pela RNA através da alteração dos valores dos pesos das conexões, procurando eliminar ou minimizar o erro (SEBASTIANI, F., 2002).

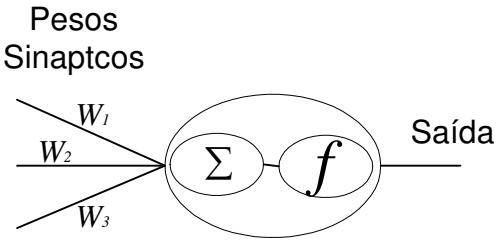

Figura 16 - Neurônio Artificial.

O tipo mais simples de RNA é o perceptrom, o qual modela um neurônio como a soma ponderada de suas entradas, com uma função de ativação, como mostra a Figura 16. A função de ativação utilizada em um perceptrom é função degrau que tem como saída o valor 1, caso a soma ponderada dos valores de entrada for maior que um valor predefinido, e 0 caso o valor for menor (RICH E.; KNIGHT, K., 1993). Existem outras formas de RNA, que utilizam diferentes funções de ativação, algoritmos de treinamento e estrutura.

Ng et. al. (1997) utilizaram perceptrom para classificação de textos e Wiener, E., Pedersen, J. O. e Weigend, A. S. (1995) utilizaram redes neurais artificiais com uma ou mais camadas e uma funções de ativação não linear para classificação. Em (YANG, Y.; LIU, X., 1999) e (YANG, Y, 1999) são apresentados os resultados e comparações entre classificadores RNA e outros classificadores. Yang, Y. e Liu, X., (1999) ressaltam que o custo de treinamento de RNA é geralmente superior ao custo de treinamento de outros métodos.

4.1.2.5 Máquinas de Vetores Suporte (Support Vector Machines)

Máquina de Vetores Suporte (SVM) é um método de aprendizado de máquina supervisionado, introduzido por Vapnik[16] (1995 apud YANG, Y.; LIU, X., 1999). O método é definido para um espaço de vetores, no qual o problema é encontrar uma superfície de decisão que seja a melhor para separar os dados de duas classes distintas. Para poder definir a melhor superfície de separação, é introduzido o

[16] Vapnic, V. The Nature of Statistical Learning Theory. Springer, New York, 1995.

conceito de *margem* entre duas classes (YANG, Y.; LIU, X., 1999). A Figura 17 ilustra a idéia.

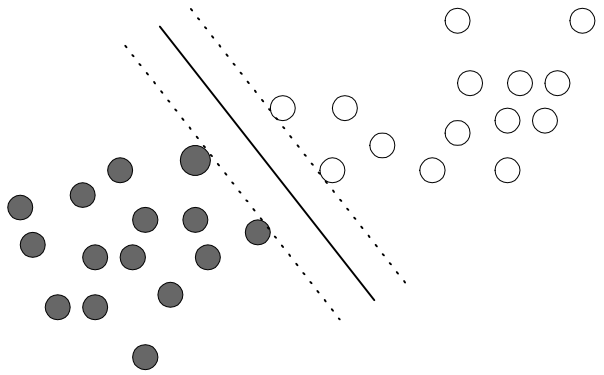

Figura 17 - Superfície de decisão com margem máxima (YANG, Y.; LIU, X., 1999)

As linhas pontilhadas mostram quando a superfície de decisão pode mover-se sem causar um erro de classificação e a distância entre as linhas pontilhadas é definida como a margem. O objetivo do SVM é encontrar a superfície de decisão com a máxima margem entre os dados no conjunto de treinamento. A superfície de decisão para um espaço linearmente separável é um hiperplano que pode ser escrita de acordo com Yang, Y. e Liu, X. (1999) como:

$$\vec{w}.\vec{x} - b = 0$$

Equação 14 - Superficie de decisão (Yang, Y. e Liu, X., 1999).

Sendo:

$\vec{x}$ - é um elemento qualquer a ser classificado;

$\vec{w}$ – é o vetor de pesos dos elementos do vetor suporte;

b - é a constante apreendida a partir do conjunto de treinamento.

De acordo com Yang, Y. e Liu, X. (1999), o problema do SVM é encontrar $\vec{w}$ e b que satisfaçam as seguintes restrições:

$$\vec{w}.\vec{x_i} - b \geq +1, \text{ para } y_i = +1$$

$$\vec{w}.\vec{x_i} - b \geq -1, \text{ para } y_i = -1$$

Sendo:

$y_i \in \{\pm 1\}$ a classificação de $\vec{x}$.

Uma propriedade interessante do SVM é que a superfície de decisão depende somente dos dados que estão a uma distância *1/w* do plano de decisão. Estes pontos são chamados de vetor suporte, os quais são efetivamente os únicos elementos efetivos no conjunto de treinamento, ou seja, se os outros pontos forem retirados, o algoritmo de aprendizado produzirá a mesma função de classificação (YANG, Y.; LIU, X., 1999).

Um SVM também pode ser utilizado com classes linearmente não separáveis através de funções de *kernel* que realiza a transformação dos elementos de espaço de entrada para outro, denominado espaço de características. As Funções de *kernel* geralmente são: lineares, gaussianas, redes neurais artificiais e funções polinomiais (KINTO, E. A.; DEL-MORAL-HERNANDEZ, E., 2005).

A aplicação de SVM para categorização de textos foi introduzida por Joachims. T. (1998). Outros exemplos do uso de SVM são os trabalhos de YANG, Y. e LIU, X. (1999), que realizaram uma comparação entre vários métodos de categorização de texto, e Kinto, E. A. e Del-Moral-Hernandez, E., (2005) para classificação de textos reduzidos em páginas na Internet.

4.1.2.6 Vizinhos Mais Próximos (k Nearest Neighbor)

Dado um novo documento a ser classificado, o método dos Vizinhos Mais Próximos calcula a similaridade deste documento para cada documento do conjunto de treinamento. A categoria dos *k* documentos mais similares são utilizadas para predizer a categoria do novo documento. Os valores de similaridade do novo documento em relação aos *k* vizinhos mais próximos, ou seja, documentos mais similares são utilizados como pesos para suas categorias e a soma dos pesos das categorias é utilizada como pontuação da categoria (YANG. Y., 1999).

4.1.3 Agrupamento

A técnica de agrupamento procura identificar a correlação entre documentos de texto, sendo um método de classificação não supervisionado que procura identificar padrões entre os documentos e agrupá-los de acordo com sua similaridade. Com a popularização da Web, além dos métodos de agrupamento por similaridade do texto, também foram propostos vários métodos de agrupamento que analisam a estrutura de *hyperlinks* entre as páginas na Web.

Os agrupamentos podem ser organizados de duas formas (JAIN, A. K.; MURTY, M. N.; FLYNN, P. J., 1999):

a) **Grupos isolados ou partições:** Neste tipo de agrupamento, os grupos são totalmente separados, não sendo definidas relações entre os grupos.
b) **Estrutura hierárquica:** Neste tipo de agrupamento, os grupos são organizados em uma estrutura hierárquica que define uma relação entre os grupos, na qual grupos mais específicos são derivados de um grupo mais geral. Neste tipo de agrupamento, o algoritmo de agrupamento é aplicado recursivamente até formar uma estrutura de categorias na qual as categorias mais abrangentes englobam a mais específicas.

O agrupamento hierárquico possui a desvantagem de consumir um tempo maior de processamento, pois o algoritmo de agrupamento deve ser aplicado aos grupos já identificados até se obter o nível de granularidade desejada.

A técnica de agrupamento pode auxiliar no processo de recuperação de informação, permitindo que um grupo de páginas similares seja recuperado quando uma página do grupo é considerada relevante para uma consulta do usuário e pode também ser aplicada antes do processo de classificação para auxiliar no processo de identificação de classes. A categorização de uma página pode ser aplicada aos centróides após o agrupamento das páginas, reduzindo o tempo de processamento para categorização.

4.1.3.1 Comunidades Web

Técnicas de agrupamento (*clustering)*, ou seja, métodos de divisão de uma população heterogênea em subpopulações que são de alguma forma mais coesas (KLEINBERG, J. M, 1999), tem sido desenvolvidos especificamente para Web, sendo muito destes métodos baseados na teoria de grafos, como a teoria de fluxo em redes e *small world*.

Comunidades Web são agrupamentos de páginas Web ou coleções de páginas inter-relacionadas por seus *hyperlinks*, na qual cada página membro tem mais *hyperlinks* para outros membros da comunidade do que para fora da comunidade, as quais podem representar um conjunto de páginas de um mesmo contexto ou assunto (FLAKE G. W.; GILES C. L.; COETZEE F. M., 2002). Métodos para descoberta de comunidades através dos *hyperlinks* são baseados em hipóteses como:

- Páginas em uma mesma comunidade possuem tópico relacionado;
- *Hyperlinks* entre páginas de sítios diferentes são mais relevantes para definir uma comunidade do que *hyperlinks* entre páginas de um mesmo sítio (FLAKE G. W.; GILES C. L.; COETZEE F. M., 2002).

O trabalho de análise de hyperlinks possui dificuldades próprias, como já citadas anteriormente. Muitos *hyperlinks* são criados por motivos diversos e não estão relacionados com o tópico da página Web, como *hyperlinks* para propaganda paga e *hyperlinks* criados apenas para navegação, e muitas empresas, devido à competição comercial, podem querer evitar *hyperlinks* diretos para outras empresas, assim evitando o endosso destas.

Uma alternativa à análise de *hyperlinks* é a análise do conteúdo, utilizando métodos para definir a similaridade entre textos ou conjuntos conceituais *fuzzy* como propostos por Tomiyama, T. et al. (2003). Porém, estes métodos também apresentam limitações principalmente associadas à ambigüidade da linguagem, à necessidade de definições feitas manualmente por pessoas e ao maior tempo de processamento necessário.

A seguir são apresentados alguns dos métodos propostos para a descoberta de agrupamentos, procurando abranger as várias metodologias adotas. Inicialmente são

apresentados métodos baseados em análise de *hyperlinks* que são específicos para hipertextos e em seguida métodos baseados na análise do texto.

4.1.3.2 Métodos de agrupamento baseados na análise da estrutura de hyperlinks

4.1.3.2.1 Identificação de comunidades com HITS

Gibson, D.; Kleinberg, J. e Raghavan, P. (1998) propõem uma abordagem baseada no algoritmo *HITS*[17], na qual comunidades podem ser vistas como contendo um conjunto central de páginas "autoridades" conectadas através de páginas "hub". A identificação destas comunidades seria feita utilizando-se o algoritmo HITS, pois, segundo os autores, um maior grau de ordenação da estrutura da Web pode ser identificado pelo algoritmo HITS, como encontrado em comunidades nas quais o número de páginas relacionadas e a densidade de *hyperlinks* é maior.

Imafuji, N. e Kitsuregawa, M. (2003) também propuseram o uso do algoritmo HITS para identificação de comunidade, porém este utiliza o HITS para definir o valor de capacidade de uma aresta para o algoritmo de máximo fluxo em grafos[18].

4.1.3.2.2 Small World

Adamic, L. (1999) propõe uma abordagem baseada no conceito de *small world*, no qual um subconjunto de vértices de um grafo forma um *small world* se à distância entre os vértices deste conjunto for menor que a distância média entre dois vértices do grafo. Assim documentos próximos fariam parte de uma mesma comunidade, sendo a distância entre documentos igual ao número de saltos necessários para se alcançar um documento a partir de outro documento. Watz, D. J.

---

[17] Veja seção 4.1.4.4.1 para descrição do algoritmo HITS.

[18] Veja o item 4.1.3.2.3. para mais detalhes.

e Strogatz, S. H. (1998) definiram as seguintes propriedades de um grafo *small world*:

a) O coeficiente de agrupamento *C* é normalmente maior do que o coeficiente de agrupamento de um grafo gerado aleatoriamente e com o mesmo número de vértices e a mesma média de arestas por vértice;
b) E o caminho característico médio *L* é normalmente menor do que em um grafo gerado aleatoriamente correspondente.

4.1.3.2.3 Máximo Fluxo e Mínimo Corte

O problema de fluxo em redes é um problema básico na teoria de grafos e otimização combinatória. Muitos algoritmos e estruturas de dados foram desenvolvidos para resolver o problema de fluxo em redes e vários problemas não relacionados podem ser tratados como sendo um problema de fluxo em rede. O problema de máximo fluxo consiste em determinar o fluxo máximo suportado pela rede e pode ser definido como: seja ***G = (V,E)*** um grafo direcionado com dois vértices distintos, ***s*** (fonte) com número de arestas de entrada igual a zero e ***t*** (sumidouro) com número de arestas de saída igual a zero. A aresta ***e*** em ***E*** tem um peso associado ***c(e)***, chamado capacidade de ***e***. A capacidade define o fluxo máximo que a aresta suporta. Este tipo de grafo é chamado de rede, veja Figura 18, e a função ***f*** define o fluxo nas arestas da rede e deve satisfazer as condições (MANBER, U., 1989):

a) $0 \leq f(e) \leq c(e)$, ou seja, o fluxo não pode exceder a capacidade da aresta.
b) Para todo $v \in V - \{s,t\}, \sum f(u,v) = \sum f(v,w)$, ou seja, o fluxo total de entrada de um vértice é igual ao fluxo total de saída deste vértice, exceto para os vértices ***s*** e ***t***.

Estas duas condições implicam que o fluxo total de saída de ***s*** é igual ao fluxo total entrando em ***t*** (MANBER, U., 1989). O teorema de máximo fluxo – mínimo corte prova que o máximo fluxo é idêntico ao corte mínimo, que interrompe o fluxo.

Portanto, se for conhecido o máximo fluxo entre dois pontos, também é conhecido quais as conexões que devem ser removidas para desconectar os mesmo dois pontos, ou seja, o conjunto de corte (FLAKE G. W.; GILES C. L.; COETZEE F. M., 2002).

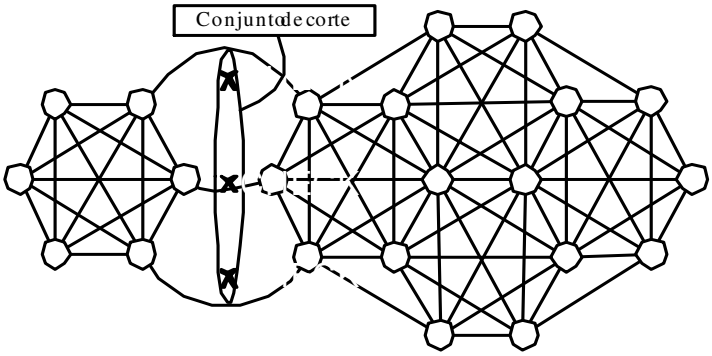

Figura 18 - Conjunto de corte (FLAKE G. W.; GILES C. L.; COETZEE F. M., 2002)

Flake G. W.; Giles C. L. e Coetzee F. M., (2002) propuseram o uso do conceito de conjunto mínimo de corte para separar os conjuntos de páginas que formam uma comunidade Web, sendo que os *hyperlinks* representam as arestas da rede, assim permitindo a identificação das comunidades Web. Imafuji, N. e Kitsuregawa, M. (2003) também propuseram o uso do algoritmo de máximo fluxo em grafos para extrair comunidades Web, porém utilizando o HITS como valor de capacidade de uma aresta, diferentemente de Flake et al (2002) que utiliza um valor fixo e igual para a capacidade de todos *hyperlinks*.

4.1.3.2.4 Grafo bipartido

Reddy, P. K. e Kitsuregawa, M. (2001) propuseram a utilização do conceito de grafo bipartido para descoberta de comunidades Web, sendo um grafo bipartido BG(T,I) um grafo direcionado, cujo conjunto de vértices pode ser dividido em dois conjuntos não vazios T e I, e toda aresta de BG conecta um vértice de T a um vértice de I.

a) **Grafo bipartido denso**

Como definido em Reddy, P. K. e Kitsuregawa, M. (2001), um grafo bipartido denso DBG(T,I,p,q) é um grafo BG(T,I), veja Figura 19, sendo:

- Cada vértice de T estabelece uma conexão com ao menos p ($1 \leq p \leq$ ic) vértices de I;
- E ao menos q ($1 \leq q \leq$ tc) vértices de T tem uma aresta com um vértice de I.

Sendo:

tc e *ic* o número de vértices dos grafos.

b) **Grafo bipartido completo**

Um grafo bipartido completo CGB(T,I,p,q) é um grafo DGB(T,I,p,q) (Reddy, P. K. e Kitsuregawa, M.; 2001), sendo p = ic e q = tc.

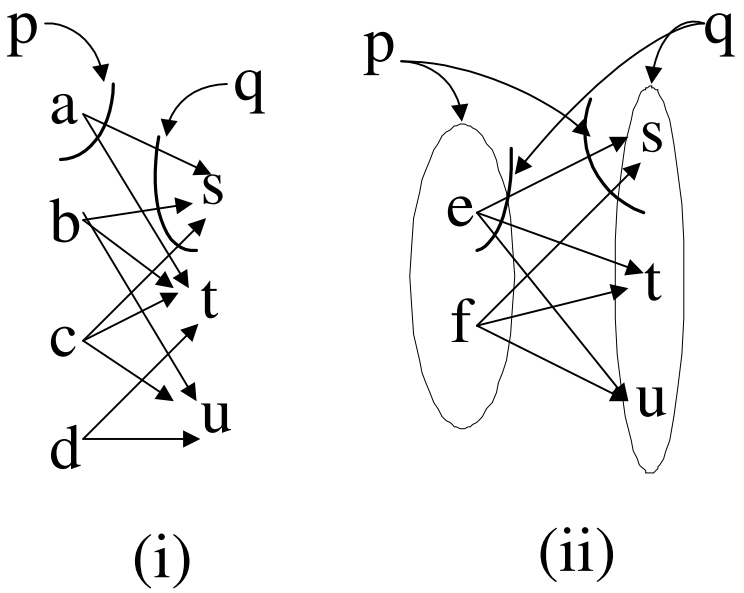

Figura 19 - Grafos:(i) DBG(T,I,p,q) e (ii) CBG(T,I,p,q)(Reddy, P.K.; Kitsuregawa, M. 2001).

Uma comunidade C(i,j) é igual a T, se existir um DBG(T,I,p,q) no conjunto de vértices com $p \geq p_t$ e $q \geq q_t$, sendo p_t e q_t valores de limiar de q e p.

4.1.3.3 Métodos de agrupamento baseados na análise do texto

4.1.3.3.1 Algoritmo de agrupamento k-means

O algoritmo de agrupamento *k-means* é um algoritmo de partição, isto é, divide uma coleção de documento em vários grupos, sendo muito popular por ser de fácil implementação e possuir complexidade de tempo *O(n)*, sendo *n* é o número de documentos (JAIN, A. K.; MURTY, M. N.; FYNN, P. J., 1999).

Considerando-se o modelo de espaço vetorial, o algoritmo *k-means* tem como dados de entrada o número desejado de grupos (***k***) e os documentos a serem divididos em grupos. A partir disto, os seguintes procedimentos são executados (CHAKRABARTI, S., 2000):

a) São selecionados ***k*** documentos para iniciar dos vetores dos ***k*** centróides de cada grupo;
b) Cada um dos documentos é associado ao centróide, com o qual tenha o maior grau de similaridade;
c) As coordenadas dos vetores de cada um dos centróides é recalculada, utilizando-se todos os membros do grupo;
d) Os passos *b* e *c* são repetidos até o critério de convergência seja atingido.

Freqüentemente, a função de erro quadrático é utilizada como função de convergência (JAIN, A. K.; MURTY, M. N.; FYNN, P. J., 1999), como definido na Equação 15.

$$e^2(x,l) = \sum_{j=1}^{k} \sum_{i=1}^{n_j} \left\| x_i^{(j)} - c_j \right\|^2$$

Equação 15 - Erro quadrático médio (JAIN, A. K.; MURTY, M. N.; FYNN, P. J., 1999).

Sendo:

$x_i^{(j)}$ - o i^{th} elemento do j^{th} grupo;

c^j - o centróide do j^{th} grupo.

Os principais problemas com este algoritmo são:

- A sua sensibilidade na seleção dos centróides iniciais, pois a função de convergência pode convergir para um mínimo local (JAIN, A. K.; MURTY, M. N.; FYNN, P. J., 1999);
- Alta dimensão dos vetores no calculo da similaridade (CHAKRABARTI, S., 2000);
- Métodos que utilizam o peso dos termos para calcular a similaridade dos documentos devem recalcular os valores dos pesos, quando novos documentos são incluídos na coleção.

Varias alterações a versões original do algoritmo *k-means* foram propostas, com diferentes critérios de seleção inicial para os centróides, diferentes funções de convergência e métodos que podem dividir um grupo ou unir dois grupos caso um valor de seja ultrapassado, para obtenção de um conjunto inicial melhor de grupos (JAIN, A. K.; MURTY, M. N.; FYNN, P. J., 1999).

4.1.3.3.2 Conjuntos Conceituais Fuzzy

Em (TOMIYAMA, T. et al, 2003) é apresentado um método que permite *minerar* a Web para conceitualmente classificar e agrupar páginas, baseado em formulações lingüísticas predefinidas e regras definidas por especialistas ou baseadas em um conjunto de páginas conhecidas. De acordo com a teoria de representação do significado pelo uso proposta por Wittgenstein, os vários significados de uma palavra podem ser representados por outras palavras. Assim, podem-se associar graus de ativação, que mostram o grau de compatibilidade entre termos (TOMIYAMA, T. et al, 2003).

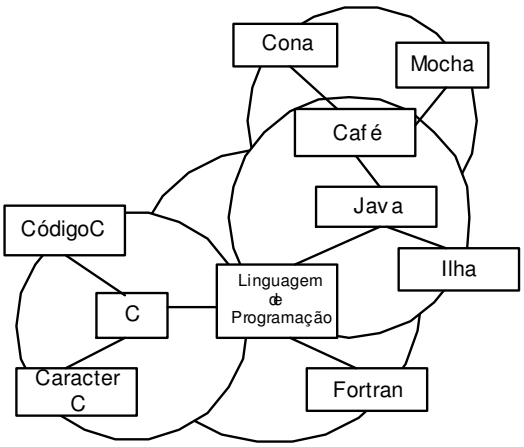

Figura 20 - Exemplo de rede RBF (TOMIYAMA, T. et al., 2003).

Estes conjuntos conceituais fuzzy são criados a partir de uma rede RBF (*Radial Basis Function Network*). A Figura 20 mostra um exemplo de RBF para o termo *java* e alguns conceitos relacionados. A Figura 21 mostra a estrutura de uma rede RBF.

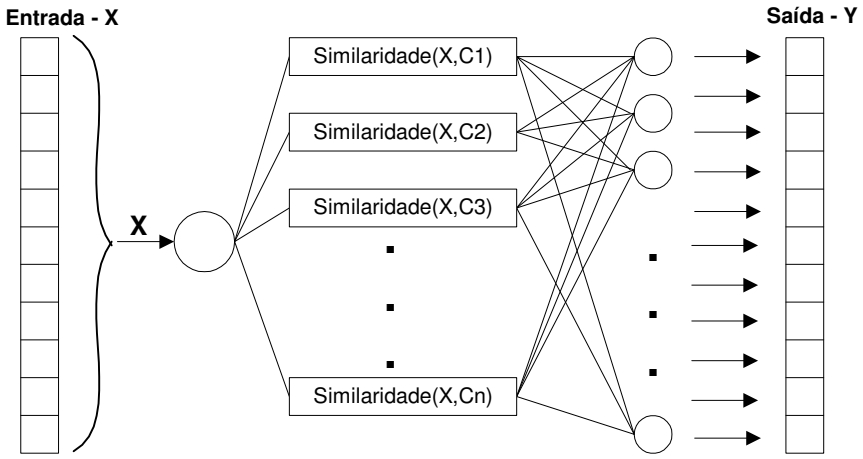

Figura 21 - Estrutura de um rede RBF (TOMIYAMA, T. et al., 2003).

Os conjuntos conceituais fuzzy expandem os termos selecionados dos documentos incluindo termos relacionado, o que deve posteriormente permitir um classificação mais precisa dos documentos por outro método, como por exemplo, um SVM.

4.1.3.3.3 Suffix Tree Clustering (STC)

Suffix Tree Clustering (STC) é um algoritmo para agrupamento de páginas, que não representa os documentos com um conjunto de palavras, mas sim, como uma seqüência de caracteres. STC utiliza árvores de sufixos para identificar conjunto de documentos que compartilham frases e utiliza esta informação para identificar agrupamentos de documentos.

Zamir, O. e Etizioni, O. (1998) ressaltam como vantagens deste algoritmo:

- Possuir complexidade temporal O(n), sendo n o número de documentos;
- A construção da árvore de sufixo ser incremental.

4.1.4 Métodos de ordenação por relevância em mecanismos de busca

Nesta seção são apresentados os conceitos básicos referentes à classificação por ordem de relevância de páginas na Web e a descrição dos principais métodos utilizados.

4.1.4.1 Valor de relevância

O valor de relevância de um documento tem por finalidade melhorar a qualidade dos resultados apresentados para uma consulta, assim evitando o problema do excesso de documentos irrelevantes apresentados para o usuário, o que dificulta a localização pelo usuário de documentos realmente relevantes a sua pesquisa. Assim, após o usuário realizar uma consulta, o mecanismo de busca deve retornar uma lista de resultados ordenados segundo a relevância dos documentos em relação à consulta feita pelo usuário. Como definido em Risvik, K. M.; Aasheim, Y. e Lidal, M. (2003), cada documento apresentado na lista possui um valor de relevância R*d*, sendo *d* identifica o documento. Este valor de relevância pode ser dividido em duas partes: um componente estático, Rs_d, que é independente da consulta, e um componente

dinâmico, $Rd_{d,q}$, que depende da consulta *q*. O valor total de relevância para uma consulta, $R_{d,q}$, é representado pela formula:

$$R_{d,q} = Rs_d + \alpha Rd_{d,q}$$

Equação 16 - Valor de relevância (RISVIK, K. M.; AASHEIM, Y.; LIDAL, M. 2003)

A constante α é utilizada para balancear o peso entre o valor de relevância estático e dinâmico. Para o calculo do valor de relevância são considerados vários atributos. Para o valor de relevância estático são geralmente consideradas somente as citações ao documento, o que indica sua autoridade.

Para o valor de relevância dinâmico são considerados:

- A freqüência do termo da consulta no documento;
- A presença dos termos da consulta nas partes mais relevantes do documento, como título, meta dados, cabeçalho e texto ancora.

Sendo a maioria dos serviços de busca na Internet comerciais, são poucas as publicações sobre como estes calculam o valor de relevância dos documentos. Porém, sabe-se pela literatura que outros fatores são levados em consideração. Por exemplo, o serviço de busca Google também considera:

- As informações de localização e proximidade dos termos;
- A Representação visual dos termos, como: tamanho da fonte, texto em negrito;
- E o texto âncora.

O texto ancora é tratado de forma especial pelo Google, pois este associa o texto ancora à página que o contém e à página referenciada pelo *hyperlink* do texto âncora (PAGE, L.; BRIAN, S., 1998). As vantagens apontadas na utilização desta estratégia são:

- O texto âncora provê uma descrição mais precisa sobre a página do que a própria página (PAGE, L.; BRIAN, S., 1998);
- O texto âncora pode ser utilizado para indexar documentos como: imagens, programas e base de dados, que não poderiam ser indexados por um mecanismo de busca em texto (PAGE, L.; BRIAN, S., 1998);
- O texto âncora possibilita retornar páginas que não foram recuperadas pelo sistema de coleta.

A utilização do texto âncora para indexação pelo mecanismo de busca pode trazer alguns problemas como:

- Retornar para o usuário páginas cuja existência não foi verificada pelo sistema de coleta (PAGE, L.; BRIAN, S., 1998);
- Número de âncoras muito grande para serem processadas pelo *indexador* (PAGE, L.; BRIAN, S., 1998).

Um conceito amplamente utilizado para definir a relevância de um documento é o conceito de autoridade, que será descrito na seção 4.1.4.3.

4.1.4.2 Calculo do valor de relevância

Três métodos têm sido utilizados para definir a relevância de um documento e melhorar as respostas para o usuário (BHARAT, K.; MIHAILA, G. A., 2002):

a) Classificação feita por pessoas: Uma pessoa manualmente associa um conjunto de categorias e palavras chaves a um subconjunto de documentos. Esta técnica tem sido utilizada por companhias como Yahoo e Mining Company. Os problemas com este método são:
    1. É lento e só pode ser aplicado a um número pequeno de páginas;
    2. Freqüentemente, as associações de palavras e categorias definidas por pessoas são inadequadas ou incompletas.

b) Classificação baseada em informações de uso: Alguns serviços, como DirectHit, coletam informações nas consultas que os usuários submetem ao mecanismo de busca, como quais as páginas que estes visualizam e o tempo gasto com cada página. Porém, este método apresenta o problema de classificar as páginas baseando-se em um conjunto de consultas potencialmente pequeno;
c) Classificação baseada em conectividade: Este método envolve a análise dos *hyperlinks* entre as páginas e assume que as páginas de um determinado tópico referenciam-se. Os problemas com este método são:
    1. Métodos independentes da consulta, como o PageRank (Page, L. et al, 1998), não distinguem entre páginas que são autoridade em geral e páginas que são "autoridade" para um tópico específico;
    2. Métodos que primeiro definem um subgrafo específico da Web para consulta são aplicáveis apenas a tópicos populares, pois é muito dispendioso calcular o subgrafo para todos os possíveis tópicos.

4.1.4.3 Fonte de autoridade

Kleinberg (1999) propôs o uso da estrutura de *hyperlinks* da Web para descobrir as páginas com maior autoridade em um dado tópico, dando origem ao algoritmo HITS (*Hypertext-Induced Topic Selection*). A estrutura da rede de *hyperlinks* pode ser uma rica fonte de informações sobre o conteúdo das páginas, desde que haja meios efetivos de compreendê-la (KLEINBERG, J. M, 1999). Os *hyperlinks* contêm um considerável montante de julgamento humano e este tipo de julgamento é o tipo de julgamento necessário para formular a noção de Autoridade (KLEINBERG, J. M, 1999). Um documento é considerado como uma autoridade em um assunto, quando um grande número de documentos faz referência a este documento. Porém, este não deve apenas ter uma grande popularidade, ou seja, ter um grande número de referências para este; também deve ter uma sobreposição do conjunto de documentos que o cita com os conjuntos de documentos que citam outras páginas sobre o mesmo tópico (KLEINBERG, J. M, 1999).

Assim podem-se identificar dois tipos de páginas:

a) **Páginas populares:** Estas são páginas com um grande número de referências, porém estas páginas não contêm informações relevantes para um tópico específico e geralmente não são possuem um tópico relacionado com o tópico da página que a referencia. Exemplos de páginas populares são as páginas de serviços na Web, como os mecanismos de busca Google e AltaVista, que freqüentemente são referenciadas, porém estas páginas não são relevantes para um tópico específico;
b) **Páginas relevantes:** Estas são páginas com um tópico específico e consideradas relevantes por pessoas interessadas no mesmo tópico, as quais incluem *hyperlinks* em suas páginas para a esta página.

Os documentos que contém um grande número de citações para documentos "autoridade" em um tópico são chamados "hubs", e permitem diferenciar entre páginas relevantes e páginas com uma grande popularidade, a Figura 22 procura ilustrar a idéia (KLEINBERG, J. M, 1999).

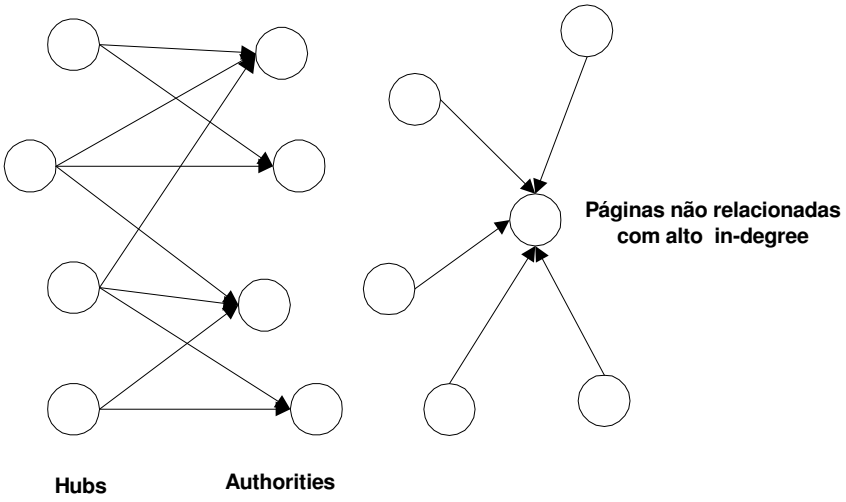

Figura 22 - Hubs e Authorities (KLEINBERG, J. M, 1999)

Documentos "hubs" e "autoridade" apresentam o que pode ser chamado de reforço mutuo de relacionamento, ou seja, um bom documento "hub" é um documento que cita bons documentos "autoridade"; e um bom documento

"autoridade" é um documento citado por muitos documentos "hub" (KLEINBERG, J. M, 1999).

A vantagem de se utilizar o conceito de documentos autoridade e hubs reside no fato de que muitas páginas não são suficientemente autodescritivas e muitas das páginas autoridades para uma consulta não contem necessariamente o texto utilizado em uma consulta (KLEINBERG, J. M, 1999).

Existem muitos problemas em se utilizar *hyperlinks* para identificar uma página autoridade (KLEINBERG, J. M, 1999):

- Muitos *hyperlinks* são criados apenas para propósito de navegação;
- Muitos *hyperlinks* são destinados à propaganda;
- Outro problema é diferenciar relevância e popularidade.

4.1.4.4 Métodos de cálculo do valor de relevância baseados em conectividade

Nesta seção são descritos os principais métodos utilizados para definir o valor de relevância de páginas na Web baseados na análise dos *hyperlinks* entre as páginas.

4.1.4.4.1 HITS

O algoritmo HITS (*Hyperlink-Induced Topic Search*), introduzido por Kleinberg J. M. (199) e utilizado pelo mecanismo de busca Teoma, calcula o valor de relevância de uma página executando um processo iterativo de propagação de pesos para criar uma estimativa numérica dos pesos de *autoridade* e *hub* de uma página (CHAKRABARTI, S. et al., 1999). Este processo é definido em Gibson, D.; Kleinberg, J. e Raghavan, P. (1998) como:

a) A partir de uma consulta do usuário feita a um serviço de busca na Web, o HITS monta um conjunto raiz *S* de páginas; tipicamente, com mais de 200 páginas;
b) Este conjunto é expandido, gerando-se um conjunto base *T* pelo acréscimo de todas as páginas referenciadas pelas páginas do conjunto *S* e que referenciam alguma página do conjunto *S*;
c) Para cada página *p* do conjunto *T* é associado um valor *h(p)*, que representa o *valor de hub* da página, e um valor *a(p)*, que representa o *valor de autoridade* da página. Inicialmente $h(p) = 1$ e $a(p) = 1$. Definindo $p \rightarrow q$ como: "página *p* tem um *hyperlink* para página q". HITS atualiza os valores de *h(p)* e *a(p)* como:

$$a(p) = \sum_{q \rightarrow p} h(q)$$

$$h(p) = \sum_{p \rightarrow q} a(q)$$

Equação 17 - HITS – autoridade e hub (Kleinberg J. M.; 199).

d) A atualização dos valores de *hub* e *autoridade* é repetida para todas as páginas e os valores são normalizados após cada interação, que é repetida até esta convergir para um conjunto estável de valores.

4.1.4.4.2 PageRank

O mecanismo de busca Google, como apresentado em (PAGE, L. et al., 2001), procura tirar vantagem da estrutura dos *hyperlinks* para definir um valor de relevância dos documentos, chamado *PageRank* (PAGE, L.; BRIAN, S., 1998), o qual segundo a definição anterior faz parte do valor de relevância estático. O método de calculo de citações em literatura acadêmica tem sido feito contando-se as referências feitas para um determinado documento. O calculo do *PageRank* estende esta idéia, porém não contando as citações igualmente para todas as páginas e

normaliza o número de citações nas páginas. O *PageRank* simula o comportamento de um "surfista" navegando aleatoriamente pela Web, seguindo os *hyperlinks* selecionados aleatoriamente; o *PageRank* fornece a probabilidade de um "surfista", navegando aleatoriamente, encontrar uma página a partir de uma página selecionada aleatoriamente (GRECO, G.; ZUMPANO, E., 2002). O *PageRank* é definido por Page, L. e Brian, S., (1998). da seguinte forma:

"Nós assumimos que a página A tem T1 ... Tn páginas que a cita. O parâmetro d é um fator de amortecimento, o qual deve estar entre 0 e 1. Nós utilizamos *d* igual a 0.85. C(A) é definido como o número de citações da página para outras páginas"

A formula para o calculo do PageRank é:

$$PR(A) = (1-d) + d(\sum_{i=1}^{n} PR(Ti)/C(Ti))$$

Equação 18 – PageRank [PB98]

Note que o *PageRank* forma uma distribuição probabilística das páginas na Web. Então a soma de todos os *PageRank* é igual a 1 (PAGE, L.; BRIAN, S., 1998).

4.1.4.5 Problemas com a definição do valor de relevância

Freqüentemente os criadores de páginas para Web procuram aumentar o valor de relevância de suas páginas artificialmente, o que dificulta a tarefa de calcular o valor de relevância, principalmente por mecanismos que o fazem automaticamente sem a interferência humana. Como exemplos, têm-se:

- Freqüentemente palavras são adicionadas no documento para aumentar a sua relevância;
- *Hyperlinks* são incluídos em documentos apenas para aumentar o valor de relevância de outros documentos;

- Algumas páginas são criadas propositadamente para enganar os mecanismos de busca, sendo estas popularmente conhecidas como páginas *spam*. Uma das técnicas de *spam* deliberadamente retorna alguma página popular para o sistema de coleta do mecanismo de busca ao invés da página real, tentando desviar o a trafego de outras páginas (BHARAT, K.; MIHAILA, G. A., 2002);
- Quando algoritmos tradicionais, baseados na análise do conteúdo, são utilizados para classificar documentos para consultas populares, eles não podem distinguir entre páginas "autoridade" e "não-autoridade", isto é, falham em detectar páginas *spam* (BHARAT, K.; MIHAILA, G. A., 2002);
- A utilização de listas de palavras comuns e algoritmos de derivação de palavras podem alterar significativamente a classificação de um texto (RILOFF, E., 1995).

Sem o auxilio de informações do texto, algoritmos que exploram a estrutura de *hyperlinks,* como PageRank e HITS, têm um sucesso limitado na identificação de coleções de páginas relacionadas (FLAKE G. W.; GILES C. L.; COETZEE F. M., 2002).

5 MÉTODOS DE CONTEXTUALIZAÇÃO

Neste capítulo, são analisadas as possíveis formas para composição de métodos para contextualização em um mecanismo de busca, quais os custos envolvidos na implementação e descritas alternativas de implementação.

5.1 Classificação dos métodos

Os métodos para contextualização podem ser classificados segundo a forma de seleção das páginas:

a) **Filtro de seleção:** Seleciona uma página de um conjunto de páginas, baseando-se em um critério de seleção. Por exemplo: selecionar apenas páginas com domínio .br;
b) **Filtro de descarte:** Filtra as páginas baseando-se em um critério de descarte. Por exemplo: descartar todas as páginas do domínio .br;
c) **Ponderação:** Todas as páginas são selecionadas para análise posterior, porém estas são ordenas segundo um critério de pontuação. Por exemplo: a relevância da página;
d) **Agrupamento:** As páginas podem ser selecionadas individualmente ou em grupo, como todas as páginas de um domínio, diretório ou categoria.

5.2 Métodos de contextualização

Nas próximas seções são apresentados os métodos identificados para contextualização de páginas. Muitos destes métodos são originários da área de recuperação de informação e mineração na Web, e já foram descritos em seções anteriores. Devido à grande variedade de métodos que podem ser aplicados para contextualização, estes foram agrupados segundo a sua função básica, ou seja,

métodos de categorização, agrupamento, etc. Assim, os métodos identificados para contextualização de páginas Web são:

a) Consulta por termos;
b) Contextualização pela URL;
c) Contextualização por referência direta.
d) Contextualização por agrupamento por referência direta;
e) Contextualização por agrupamento das páginas, analisando a estrutura de *hyperlinks*;
f) Contextualização pelo agrupamento das páginas, utilizando similaridade do texto;
g) Contextualização pela categorização da página;

5.2.1 Consulta por termos

Este método para contextualização é a forma básica de pesquisa fornecida pelos mecanismos de busca na Internet e consiste na busca de páginas com um ou mais termos. Podem ser utilizadas várias formas de consulta como: consulta booleana, consulta por proximidade, etc. A consulta por termos pode ser aplicada como um método inicial de busca para seleção de páginas para outros métodos como os métodos de agrupamento.

5.2.2 Programação por Análise de URL

Com a análise de URLs pretende-se definir métodos que permitam identificar informações contidas nas URLs, as quais possam auxiliar na programação de contextualização das páginas.

5.2.2.1 Uniform Resource Locator (URL)

A localização e acesso a recursos na Internet é feita através de uma seqüência de caracteres chamada URL, que tem sua sintaxe e semântica definida pela RFC1738 e possui o seguinte esquema geral:

<esquema>:<esquema especifico>

Figura 23 - Esquema geral de uma URL (RFC1738).

Sendo:

esquema - um protocolo como: ftp, http, gopher, etc.;

esquema específico - a parte específica do esquema e depende do esquema escolhido.

A sintaxe comum para o esquema específico (*Comam Internet Specific Scheme Syntax*) para o protocolo HTTP é definida como:

http://<host_domain>:<port>/<path>?<search_part>

Figura 24 - Sintaxe do esquema para http (RFC1738).

Sendo:

h**ost_domain:** o nome DNS do *host*;

port: o número de uma porta TCP/UDP;

path: o caminho para um arquivo;

search_part: parâmetros extras dependentes do protocolo.

O host_domain pode ser definido segundo o esquema:

hostname[.DN]$^{1 - N.}$GTLD.ccTLD

Figura 25 - Sintaxe para o esquema do nome de um computador (RFC1738)

Sendo:

hostname: Nome de um computador ou equipamento de rede;

DN: *Domain Name* (Nome de domínio) - pode existir um ou mais nomes de domínio;

GTLD: *Generic Top-Level Domains* (Nome de Domínio Genérico de primeiro nível);

ccTLD: *Country-Code Top-Level Domains* (Sigla do Nome do País). Domínios Internet registrados nos EUA não são obrigados a utilizar a sigla do país.

A *Internet Corporation for Assigned Names and Numbers* (ICANN) é a organização internacional responsável pela alocação do espaço de endereçamento IP, pelos identificadores de protocolo, pelos nomes de domínios genéricos (gTLD) e siglas de identificação de países (ccTLD). Estas definições podem ser utilizadas para a análise do domínio para contextualização.

No Brasil, o Núcleo de Informação e Coordenação do Ponto BR (NIC.BRa), a entidade executora do Comitê Gestor da Internet no Brasil e responsável pela definição dos nomes de domínios no Brasil, faz uma pequena distinção entre a forma de nomeação para os domínios. Assim, têm-se duas formas de nomeação, que são (NIC.BRb):

a) Forma Geral.

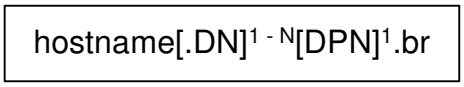

Figura 26 - Forma general (NIC.BRb) .

Sendo:

hostname: Nome de um computador ou equipamento de rede;

DPN: Domínios de Primeiro Nível.

b) Forma reservada: Para os domínios do CGI – Brasil e Universidades com nome de domínio não genérico.

hostname[.DN]$^{1 - N}$.br

Figura 27 - Forma Reservada (NIC.BRb).

Sendo:

hostname: Nome de um computador ou equipamento de rede;

DN: Nome de Domínio.

5.2.2.2 Contextualização pela URL

Segundo os esquemas de uma URL, apresentados na seção 5.2.2.1, é definido o seguinte esquema genérico para uma URL, que divide a URL em quatro partes para análise individual de cada uma destas:

http://<host_domain>:<port>/<path>?<search_part>

Figura 28 - Esquema genérico para análise.

Nas próximas seções são analisadas as possíveis formas de programação de contextualização das páginas pela análise dos termos de sua URL.

5.2.2.2.1 Análise do domínio

Análise pelo domínio pode ser realizada por:

- Código de país;
- Domínio genérico;
- Termos específicos.

a) Código do País

Nesta forma de contextualização pode-se selecionar ou filtrar páginas pela origem. Útil em caso como:

- *Clipping* de notícia para uma empresa com atividades restritas a um único país, no qual seriam selecionadas apenas páginas de um país específico;
- Casos de desrespeito à legislação local, que não são aplicáveis para domínios hospedados em outras localidades.

b) Domínio Genérico

Nesta forma de contextualização pode-se selecionar ou filtrar páginas pela categoria geral de seu domínio. As páginas poderiam ser contextualizadas segundo:

- **A categoria definida para o domínio:** Como definido pelo NIC.br[19], as páginas podem ser contextualizadas segundo a categoria do domínio de primeiro nível. Útil em casos como:
    - *Clipping* de notícias, no qual apenas páginas em domínios genéricos de instituições de comunicação como jornais e TV seriam selecionados. Assim, os domínios destinados a instituições de comunicação (.INFO.BR), radio difusão (.TV.BR), e o domínio de empresas comerciais (.COM.BR) formariam o contexto para *clipping*;
    - Serviços de Pesquisa de Preço na Internet poderiam realizar um busca apenas no contexto de empresas comerciais, ou seja, páginas em domínios de empresas comerciais (.COM.BR);
    - Pesquisa de mercado. Uma empresa de instrumentos musicais poderia realizar uma pesquisa de mercado apenas entre músicos

[19] Veja a seção 5.2.2.1 para mais detalhes.

profissionais, utilizando o contexto músicos, que inclui o domínio de músicos profissionais (.MUS.BR).

- **O grupo de categorias de domínio genérico:** As páginas também poderiam ser contextualizadas segundo um grupo de domínios. Como exemplo, as categorias de domínios podem ser organizadas segundo as distinções de domínios. O NIC.br agrupa as categorias de domínios em:
    - Categorias para Instituições;
    - Categorias para Profissionais Liberais;
    - Categorias para Pessoas Físicas.

  Desta forma, poderia ser definido um contexto para profissionais liberais, que poderia ser utilizado em uma pesquisa de mercado.
- **Termos específicos no nome de domínio próprio:** Em muitos casos o domínio genérico não é suficiente para caracterizar o contexto de uma página. A busca por termos específicos no nome do domínio próprio e no nome do *host* pode aprimorar a seleção de páginas. Por exemplo:
    - Considerando-se a heurística de que o termo "blog" refere-se apenas a páginas de diários ou históricos e raramente a uma página de notícias. A presença do termo "blog" no nome do domínio ou do *host* poderia ser utilizado para reduzir a prioridade de análise de páginas em domínios COM.BR ou do próprio domínio no contexto para clipping;
    - O termo "news" ou "notícias" no domínio próprio pode ser utilizado como heurística para ordenação das páginas no contexto para clipping para análise posterior por outros métodos.

5.2.2.2.2 Análise do caminho de diretórios e parâmetros extras

A mesmas heurísticas e técnicas de procura por um termo específico, utilizadas para análise de domínio específico, podem ser aplicadas na análise do caminho de diretórios de uma página e parâmetros extras para a definição de contexto.

5.2.2.3 Métodos de análise de padrão em URL

Para realizar a análise de padrões em URLs deve-se utilizar um método de busca em seqüência de caracteres. Estes métodos de busca em seqüência de caracteres podem ser classificados como:

a) **Métodos para busca exata:** Compreende os métodos de busca por uma seqüência de caracteres na URL que seja exatamente igual uma outra seqüência. Neste grupo de métodos têm-se os algoritmos:
   - **Força bruta:** A busca pela seqüência de caracteres é feita seqüencialmente, sem a utilização de algum método de otimização do algoritmo de busca;
   - **KMP (Knuth, Morris e Pratt):** O algoritmo de Knuth, Morris e Pratt soluciona o problema de busca de seqüências de caracteres em tempo linear, isto é, o algoritmo possui complexidade O(n+m), sendo n e m o tamanho das seqüências de caracteres. O algoritmo obtém informações de como ***p*** se comporta quando confrontado com deslocamentos dele próprio através do pré-processamento da seqüência ***p***, a ser procurado em uma seqüência ***x*** de caracteres. Estas informações são utilizadas para se evitar verificações desnecessárias, o que torna o algoritmo mais eficiente (ALMEIDA, N. F.; TELLES, G. P.; MARTINEZ, F. H. V., 2005);

b) **Métodos para busca inexata:** Os métodos para busca inexata compreendem os algoritmos de busca em seqüências de caracteres que procuram por seqüências semelhantes, ou seja, toleram pequenas variações que esteja de acordo com um determinado padrão. Neste grupo de métodos têm-se os algoritmos:
   - **Distância de Hamming:** A distância de Hamming pode ser interpretada como o número de caracteres diferentes entre duas seqüências de caracteres (MANBER, U., 1989);

- **Distância de Levenshtein (edit distance):** A distância de Levenshtein (Levenshtein, V. I.[20], 1965 apud MANBER, U., 1989) é a medida de similaridade entre duas seqüências de caracteres, sendo o igual ao número de remoções, inserções ou substituições necessárias para transformar uma seqüência na outra. Utilizado em verificação ortográfica, reconhecimento de discurso, análise de DNA, detecção de plágio (MANBER, U., 1989);
- **Expressões Regulares:** Expressão regular define um padrão a ser usado para procura em uma seqüência de caracteres.

5.2.3 Contextualização por referência direta

Os *hyperlinks* das páginas Web formam uma grafo direcionado, isto é, permitem navegação em apenas um sentido, porém a navegação no sentido inverso poderia ser útil em várias situações como:

- Em um trabalho de *clipping* de notícias para uma determinada empresa, o conhecimento das páginas que fazem referência à empresa permitiria uma melhor seleção do conjunto inicial de páginas para análise.

Os mecanismos de busca na Internet geralmente não fornecem este tipo de serviço. Entretanto, este método pode ser implementado utilizando-se a base de dados de URLs do sistema de coleta de páginas.

Portanto, páginas que fazem referência direta a uma determinada página são entendidas como fazendo parte do contexto de referências para esta página.

5.2.4 Contextualização pelo agrupamento de páginas

A contextualização pelo agrupamento de páginas permite a seleção de páginas com algum tipo de relacionamento. Os métodos de descoberta de agrupamentos de páginas na Web por análise da estrutura de *hyperlinks*,

---

[20] Levenshtein, V. I. *Binary codes capable of correcting deletions, insertions, and reversals*, Doklady Akademii Nauk SSSR, 1965.

apresentados na seção 4.1.3.2, e métodos baseados na similaridade do texto, apresentados na seção 4.1.3.3, podem ser utilizados para definir o contexto de páginas pelo grupo do qual fazem parte. Estes métodos permitem realizar a seleção de páginas em grupos e definem duas formas distintas de contextualização:

a) Agrupamento de páginas pela análise da estrutura de hyperlinks.
b) Agrupamento de páginas pela similaridade do texto.

5.2.5 Agrupamento por referência direta

Este método é uma versão simplificada de agrupamento pela análise da estrutura de *hyperlinks*, que utiliza a idéia de contextualização por referência direta. Neste método, são agrupadas apenas as páginas diretamente relacionadas a uma página previamente selecionada, ou seja, páginas que permitem uma navegação direta até a página selecionada e páginas que podem ser acessadas através da navegação direta a partir da página selecionada inicialmente. A partir desta página selecionada inicialmente é realizada uma expansão limitada do conjunto, não realizando uma análise posterior como é proposto em outros métodos de agrupamento. Para limitar o tamanho do conjunto podem ser aplicados dois critérios:

- A distância máxima que uma página poderia ter a partir de página inicialmente selecionada, ou seja, o número máximo de saltos pelos *hyperlinks*;
- O tamanho máximo do agrupamento.

Sendo uma versão simplificada de métodos de agrupamento, para que este seja válido o valor que limita o tamanho do conjunto deve ser baixo, pois através destes métodos pretende-se descobrir páginas estreitamente relacionadas e não definir páginas que possuam assunto similar como é proposto por métodos de agrupamento para localização de comunidades Web.

5.2.6 Contextualização pela categorização da página

A contextualização pela categorização da página permite selecionar páginas de um assunto específico. Os métodos de categorização utilizados em mineração de

conteúdo da Web, apresentados na seção 4.1.2, podem ser utilizados para definir esta forma de contextualização. Este método de contextualização pode ser utilizado em situações na qual se deseja encontrar páginas de um assunto específico em um conjunto de páginas de assuntos variados. Por exemplo, em um serviço de *clipping* de notícias, como apresentado anteriormente, um método de contextualização pela URL poderia identificar uma grande quantidade páginas de sítios de notícias, porém não seria capaz de identificar o assunto destas páginas. Porém, quando o assunto da página também é relevante, a contextualização pela categorização poderia ser aplica em seqüência ao método de contextualização pela URL.

5.3 Fatores a serem considerados para escolha de um método

A escolha de quais métodos utilizar e a seqüência de aplicação destes dependem de vários fatores que devem ser considerados, entre estes fatores:

a) Facilidade de uso: Um método utilizado não deveria necessitar de conhecimento especializado do usuário, como o conhecimento de métodos de treinamento utilizados em aprendizado de máquina, pois isto limitaria a aplicação deste;

b) **Necessidade de treinamento:** Métodos que necessitam de treinamento para gerar um classificador, como os métodos de categorização, apresentam dois fatores que dificultam a sua aplicação:

- Dificuldade para se definir as classes;
- Dificuldade para se definir os conjuntos de dados para treinamento;

c) Custo extra de armazenamento de dados: O espaço extra de armazenamento para novas estruturas de dados específicas para um método;

d) Custo computacional: Custo computacional necessário para pré-processamento dos dados, treinamento e aplicação do método;

e) **Tempo de espera:** O tempo de espera do usuário pela resposta também deve ser considerado;
f) **Aplicação:** A aplicação de um método esta diretamente ligada aos custos e a complexidade de aplicação. Alguns métodos podem ser utilizados inicialmente como filtro para seleção de páginas para outros métodos com maior custo computacional, o que reduziria o custo de aplicação de um método mais sofisticado.

Os vários métodos apresentados anteriormente para contextualização possuem características próprias, que podem definir a melhor forma de aplicação deste. Porém, existe uma grande variedade de métodos que podem ser utilizados em sua programação, portanto, não serão analisados detalhes específicos de implementação. Os métodos de contextualização serão analisados:

a) Consulta por termos;
b) Contextualização pela URL;
c) Contextualização por referência direta;
d) Contextualização por agrupamento por referência direta;
e) Contextualização por agrupamento das páginas, analisando a estrutura de *hyperlinks*;
f) Contextualização pelo agrupamento das páginas, utilizando similaridade do texto;
g) Contextualização pela categorização da página;

5.3.1 Treinamento

Métodos que necessitam de uma fase de treinamento, como os métodos de categorização que utilizam técnicas de aprendizado de máquina, apresentam maior grau de dificuldade, tanto para implementação quanto para utilização. Os principais fatores que dificultam a implantação são:

a) **Dificuldade em definir as classes:** Inicialmente não são conhecidas as classes ou categorias que o classificador deve ser capaz de reconhecer. Como as possíveis classes não são conhecidas inicialmente e mesmo que fosse seria provavelmente impossível gerar um classificador capaz de lidar com todas as classes possíveis. Assim, a utilização de classificadores binários pode ser uma alternativa viável para utilização em métodos de classificação, que simplificaria a fase de treinamento e reduziria o tempo necessário para aplicação destes.

b) **Dificuldade em definir os conjuntos de treinamento:** A seleção do conjunto de treinamento é um fator de grande influência na eficiência dos classificadores, pois os critérios de relevância, utilizados para definir selecionar as páginas do conjunto de treinamento, são subjetivos e depende da análise feita por pessoas. Como utilizam apenas um conjunto de treinamento, classificadores binários são mais fáceis de serem implantados.

A Tabela 7 identifica quais métodos, que são úteis na contextualização de páginas, necessitam de uma fase de treinamento:

Tabela 7 - Método x Treinamento.

Método	**Necessita Treinamento**
Consulta por termos	Não
Contextualização pela URL	Não
Contextualização por referência direta	Não
Contextualização por agrupamento por referência direta	Não
Contextualização por agrupamento das páginas, analisando a estrutura de hyperlinks	Não
Contextualização pelo agrupamento das páginas, utilizando similaridade do texto.	Depende do método escolhido
Contextualização pela categorização da página.	Sim

5.3.2 Novas estruturas de dados

Métodos que necessitam de novas estruturas de dados, que não são geralmente utilizadas em mecanismos de busca, podem tornar inviável a sua aplicação prática devido ao espaço extra para o armazenamento destas estruturas. A Tabela 8 identifica quais métodos necessitam de novas estruturas de dados:

Tabela 8 - Método x Novas estruturas de dados.

Método	**Novas estruturas de dados e custo extra de armazenamento**
Consulta por termos	Não
Contextualização pela URL	Não
Contextualização por referência direta	Não
Contextualização por agrupamento por referência direta	Não
Contextualização por agrupamento das páginas, analisando a estrutura de *hyperlinks*	Não, Considerando que o mecanismo de busca sempre mantenha uma base de dados de *hyperlinks*.
Contextualização pelo agrupamento das páginas, utilizando similaridade do texto	Sim, entre 30% e 50% do espaço necessário para o armazenamento das páginas é necessário para os vetores de termos.
Contextualização pela categorização da página	Sim, entre 30% e 50% do espaço necessário para o armazenamento das páginas é necessário para o armazenamento dos vetores de termos.

Para reduzir o espaço de armazenamento de dados do mecanismo de busca podem-se remover outras estruturas de dados não utilizadas pelos métodos de contextualização e que não são vitais ao funcionamento do mecanismo de busca, porém isto poderia causar a perda de algumas funcionalidades. Por exemplo: o conteúdo das páginas poderia não ser mantido após a indexação, o que reduziria em muito o espaço de armazenamento necessário. Porém, isto implicaria na perda da

função de sumarização, que é geralmente fornecida pelos mecanismos de busca e útil para o usuário na seleção de páginas.

5.3.3 Custo computacional

O custo computacional dos métodos é um dos principais aspectos que devem ser considerados na definição dos métodos a serem adotados na implementação de um sistema. Métodos que necessitam de uma fase de treinamento geralmente necessitam um maior tempo para implantação, devido à fase de treinamento e testes, por isto devem ser aplicados com cautela. Métodos de implantação mais simples e com tempo menor de resposta ao usuário devem ser utilizados preferencialmente.

De forma geral, a complexidade computacional de um método deve ser definida para todas as etapas envolvidas. Para métodos que necessitem de uma etapa de treinamento, devem ser definida a complexidade computacional para as etapas de pré-processamento, treinamento e execução. Para métodos que não possuam uma etapa de treinamento, deve ser apenas definida a complexidade computacional para as etapas de pré-processamento e execução.

A análise da complexidade dos métodos que podem ser aplicados para a contextualização seria uma tarefa complexa e demorada, devido a grande quantidade de métodos existente. Portanto, devido ao tempo disponível para a realização deste trabalho ser restrito, e como a definição da complexidade dos muitos métodos envolvidos não ser o foco principal deste trabalho, a definição da complexidade destes métodos não será realizada.

6 PROPOSTA PARA PROGRAMAÇÃO DE CONTEXTUALIZAÇÃO

Como apresentado anteriormente, a utilização de um único método geralmente não é suficiente para realizar satisfatoriamente busca na Internet para serviços especializados, Uma possível alternativa é composição de métodos.

Este capítulo trata a respeito da programação de contextualização, discute formas de aplicação dos métodos, define e analisa formas de implementação destes em no mecanismo de busca Nutch.

6.1.1 Seqüência de aplicação dos métodos.

É importante definir-se uma seqüência de aplicação de métodos para melhor uso dos recursos computacionais, minimizando o tempo de execução e o espaço para armazenamento de dados.

Os métodos devem ser aplicados em seqüência de complexidade, servindo como filtro para o método seguinte, a Figura 29 ilustra a idéia.

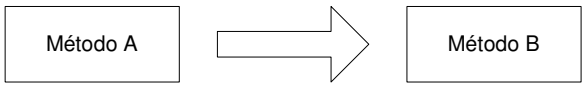

Figura 29 - Método Utilizado como Filtro.

A aplicação de um método como filtro para outro método tem como objetivo reduzir o número de páginas a serem analisadas por outros métodos mais sofisticados, porém de maior custo computacional.

A seqüência de aplicação dos métodos não deve ser fixa e sim programada pelo usuário para o contexto, pois em alguns casos, um método pode eliminar incorretamente uma página do conjunto de páginas relevantes.

6.1.2 Forma de Aplicação

Na composição de um método para contextualização, a seqüência de aplicação dos métodos deve ser considerada cuidadosamente, pois isto pode causar um grande impacto na eficiência do sistema. Métodos com aplicação mais difícil ou com tempo de execução maior devem ser apenas aplicados quando realmente necessário e, preferencialmente, no final do processo. Como apresentado na seção 6.1.1, um método pode ser utilizado para a seleção de páginas para outro método, assim, reduzindo o número de páginas a serem analisadas por métodos mais sofisticados. A Tabela 9 apresenta uma seqüência recomendada para aplicação dos métodos:

Tabela 9 - Método x Forma de Aplicação.

N.º	**Método**	**Forma de Aplicação**	**Seqüência de aplicação**
1	Consulta por termos	Filtro para seleção ou descarte e ponderação	Inicial
2	Contextualização pela URL	Filtro para seleção ou descarte e ponderação	Inicial ou Intermediária
3	Contextualização por referência direta	Filtro para seleção	Inicial ou Intermediária
4	Contextualização por agrupamento por referência direta	Filtro para seleção de grupo	Intermediária
5	Contextualização por agrupamento das páginas, analisando a estrutura de *hyperlinks*	Filtro para seleção de grupo	Intermediária
6	Contextualização pelo agrupamento das páginas, utilizando similaridade do texto	Filtro de seleção. Cuidado devido ao custo.	Final

7	Contextualização pela categorização da página	Filtro de seleção e ponderação. Cuidado devido ao custo	Final

Deve-se ter cuidado na aplicação de um método como filtro para outro método, pois páginas relevantes podem ser descartadas por estes. Para minimizar isto, deve-se selecionar seus parâmetros com cuidado para não restringir excessivamente os resultados. Por exemplo: Quando utilizar o método de consulta simples por termos como filtro de seleção, deve-se utilizar um número maior de termos para tornar o consulta um pouco mais abrangente. Assim, o uso de sinônimos pode tornar um consulta mais abrangente.

6.2 Histórico de Contextos

O uso de histórico de classificações prévias de contextos em um mecanismo de busca na Web tem a função de armazenamento de resultados relevantes de um busca para uso futuro. O histórico poderia ser utilizado das seguintes formas:

a) Como um histórico temporário para armazenar os resultados intermediários obtidos através de um algum método de contextualização para uso posterior pelo usuário. A Figura 30 ilustra a idéia.

Figura 30 - Uso do histórico.

b) Como um histórico permanente para armazenar os resultados de um método de contextualização de maior custo computacional para uso futuro em métodos como categorização. A Figura 31 ilustra a idéia.

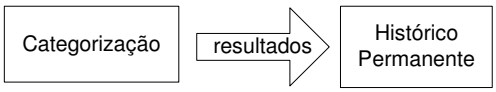

Figura 31 - Uso do histórico.

Como os métodos podem ser aplicados em diferentes seqüências, o histórico temporário também tem a função de padronizar a forma de acesso aos dados pelos métodos, o que simplifica a implementação. Assim, qualquer método utilizado para contextualização deve implementar uma interface para leitura do histórico e deve ser capaz de escrever os resultados no histórico.

6.3 Integração de métodos de contextualização no Nutch

Nas próximas seções serão analisadas as possíveis formas de integração de métodos de contextualização no Nutch.

6.3.1 Contextualização pela URL

Para integrar o método de contextualização pela URL são possíveis três abordagens:

a) Utilização de mecanismos de consulta já existentes do Nutch;
b) Criação de um tipo específico de consulta para o Nutch;
c) Realização de consulta diretamente na base de dados WebDb.

Como descrito na seção 5.2.2.2, a análise da URL para contextualização pode ser feita para o domínio, termos específicos do domínio próprio, no nome do *host*, no diretório e parâmetros extras.

Para executar-se a análise da URL por cada uma das abordagens citas anteriormente, tem-se:

a) Utilização de mecanismos de consulta já existentes do Nutch

No Nutch, uma URL é dividida em cada um dos seus termos e cada termo é indexado separadamente em um índice exclusivo. Os termos da URL são identificados pelo campo *url*. Porém, os campos da URL, como o domínio e nome de computador, não são diferenciados.

Assim, como o Nutch não diferencia os campos que compõem uma URL, não é possível realizar a análise de cada um dos componentes de uma URL separadamente do domínio.

Como o Nutch suporta apenas consultas booleana por palavras e frases, consultas com "*wildcards"* de substituição por um caractere (?) e substituição por múltiplos caracteres (*), e busca aproximada com distância de Levenshtein (MANBER, U., 1989), não é possível utilizar um método de busca aproximada em texto mais sofisticado, como expressões regulares.

Devido ao mecanismo de expansão das consultas, utilizado pelo Nucth, uma busca em URL seria expandida para uma busca no conteúdo e no texto ancora, o que aumentaria o tempo de processamento.

b) Criação de um tipo específico de consulta para o Nutch

A criação de um tipo específico de consulta permitiria contornar os problemas de expansão das consultas e também o uso de um método de busca aproximada em texto, como por exemplo, distância de Levenshtein (MANBER, U., 1989), o qual já suportado pelo Lucene. Porém, para busca por padrões em textos, utilizando expressões regulares, seria necessária a implementação destes métodos de busca no índice.

Para analisar separadamente cada um dos campos que compõe uma URL seria necessário criar um índice separado para cada componente da URL, o que aumentaria o espaço necessário em disco para armazenamento. Uma alternativa à criação de um índice para cada componente da URL seria o uso de *n-grams,* porém o uso desta alternativa implicaria em alterar o mecanismo de indexação para garantir que os termos dos domínios fossem indexados como um *n-gram* e no aumento no

tamanho do índice sem apresentar vantagens significativas em relação à primeira abordagem.

Uma alternativa é criar um tipo específico de consulta que realize a busca sem a diferenciação dos campos da URL e que posteriormente analise as URLs recuperadas. Este método tem como vantagem não necessitar de espaço extra de armazenamento, permitir a aplicação de métodos de busca inexata em seqüência de caracteres e análise posterior da URL, e permitir a aplicação de métodos mais sofisticados, porém de maior custo computacional.

c) Realização de consulta diretamente na base de dados WebDB

A realização de consulta diretamente na base de dados WebDB para análise de URLs permitiria o acesso fácil a todas as URLs e a aplicação dos métodos de busca inexata em seqüência de caracteres ou expressão regulares facilmente, porém implicaria na necessidade de se realizar a busca e análise seqüencial de todas as URLs da base de dados, o que pode consumir muito tempo de processamento desnecessariamente.

6.3.2 Contextualização por Agrupamento

Para contextualização por agrupamento existem as seguintes abordagens possíveis:

a) **Agrupamento por similaridade de documentos**

Métodos como k-means[21] e outros similares que utilizam o vetor de termos do documento geram dois problemas para implementação:

- Não são práticos para serem aplicados em um número muito grande de documentos, como é o caso com mecanismos de busca para Web, pois o tempo de processamento seria muito grande. Aplicação deste tipo de

[21] Veja seção 4.1.3.3.1.

método seria mais conveniente como um passo posterior no processo de contextualização, quando o número de documentos para análise é menor;

- Como este tipo de método não é usual em mecanismos de busca, estes métodos não possuem suporte e também não armazenam os vetores de termos dos documentos. Para programar um método deste tipo seria necessário incluir suporte para geração de vetores de termos no mecanismo de busca e espaço extra de armazenamento. O espaço extra para armazenamento dos vetores de termos dos documentos seria algo entre 30% e 50% do espaço necessário para o armazenamento dos documentos (SALTON, G. 1983), se os termos irrelevantes forem removidos.

b) **Agrupamento por análise de *hyperlinks***

A forma de agrupamento pode ser classificada como:

- ***Agrupamento por partição*:** Métodos que utilizam partição, como o método de partição por máximo fluxo/mínimo corte (FLAKE G. W.; GILES C. L.; COETZEE F. M., 2002), utilizam a técnica de dividir um grafo em grupos menores até atingir uma meta predefinida, como um número específico de grupos ou um tamanho de grupo. Este tipo de método possui alguns problemas como:
  - O consumo de tempo de processamento para ser aplicado na base de dados de um mecanismo de busca comercial;
  - O fato de ser desnecessário, pois não é desejado encontrar todos os grupos possíveis;
- ***Agrupamento por aglomeração*:** Os métodos que utilizam aglomeração partem de um número predefinido de grupos e adicionam a este as páginas mais relacionadas. Um método como este poderia ser aplicado para identificar as páginas relacionadas a uma página predeterminada através da análise dos *hyperlinks*.

Assim poderiam ser adotadas as seguintes formas de implementação:

a) Utilização de um método que agrupe as páginas, utilizando uma página selecionada inicialmente para gerar um conjunto de páginas e posteriormente aplicar um método de agrupamento neste conjunto. Este conjunto de páginas poderia ser gerado pela expansão de um conjunto inicial com a inclusão das páginas referenciadas por páginas deste conjunto e por páginas que referenciam alguma página deste conjunto;
b) Utilização de agrupamento por referência direta.

6.3.3 Contextualização pela Categoria

A categorização automática de páginas não é uma funcionalidade padrão dos mecanismos de busca na Web, portanto estes não fornecem suporte a vetores de termos que são geralmente necessários na categorização. Porém, a biblioteca de aplicação Lucene, utilizada pelo Nutch para indexação e recuperação de páginas, fornece suporte para gerar e manipular os vetores de termos dos textos.

Para um método que utiliza os vetores de termos seria necessário:

a) A implementação do suporte a vetores de termos no mecanismo de busca;
b) Espaço extra para armazenamento dos vetores de termos;

Alguns dos problemas encontrados na categorização de documentos em mecanismos de busca são:

a) A categorização de todas as páginas é muito custosa, pois exige um tempo extra para o processamento, treinamento e espaço extra para armazenamento;
b) Dificuldade para definir as categorias;
c) Geralmente, as categorias não são completamente conhecidas para uma aplicação de uso genérico.

Uma alternativa seria aplicar um método de categorização apenas em páginas previamente selecionadas por outro método, gerando os vetores de termos para as páginas selecionadas e descartando-os após as páginas terem sido categorizadas, o que reduziria o espaço para armazenamento.

Como geralmente não há interesse em classificar todas as páginas em todas as categorias possíveis, classificadores binários poderiam ser utilizados, o que permitiria definir se uma página pertence ou não a uma categoria de interesse, sem processamento desnecessário de categorias não relevantes no momento.

6.3.4 Histórico de Contextos

Para implementação do histórico podem ser adotadas duas abordagens possíveis:

a) **Armazenar os dados do histórico em um arquivo:** Esta abordagem seria mais apropriada para o histórico temporário e poderia ser implementada utilizando-se arquivos no formato XML, que teria como vantagens:
   - Não provocar impacto significativo no resto do sistema, como alterar as bases de dados do Nutch;
   - Fácil implementação;
   - Fácil alteração.

b) **Armazenar os dados em uma base de dados:** Esta abordagem seria mais apropriada para a implementação do histórico permanente, podendo-se criar tabelas extras para a categoria e agrupamento associados a uma página.

A implantação de métodos de contextualização em mecanismos de busca deve ser analisada cuidadosamente, devido aos custos envolvidos, tanto em recursos materiais como espaço para armazenamento de dados, quanto em recursos humanos como pessoal especializado para realizar tarefas como treinamento de um método.

6.4 Implementação

Nesta seção são descritos os métodos implementos no Nutch, testes realizados e resultados. Os seguintes métodos foram implementados no mecanismo de busca Nutch:

- Contextualização pela URL.
- Contextualização por referência direta.
- Contextualização por agrupamento por referência direta.
- Contextualização por categorização, utilizando o classificador Rocchio.

A escolha do classificador Rocchio deve-se ao fato de ser fácil de implementar e integrar ao mecanismo de busca, treinamento fácil de ser realizado e obter bons resultados na categorização de documentos.

6.4.1 Ambiente de testes.

Na implementação e testes foram utilizados os seguintes componentes de hardware e software:

- Computador: Pentium 4 – 3GHz, 512MB de RAM, 40GB de disco.
- Nutch versão 0.72.
- Lucene versão 1.91.
- Coleção Reuters-21578 de documentos para treinamento e testes.

6.4.2 Testes realizados e resultados.

Foram realizados testes de categorização utilizando-se a coleção Reuters-21578, composta por 21578 artigos publicados pela agência Reuters. Nos testes

foram removidos os artigos não classificados, resultando uma coleção com 10788 artigos classificados em 90 categorias.

O valor médio de f1-measure obtido na categorização utilizando o classificador Rocchio foi de micro-avg f1-measure igual a 69,5%.

6.4.2.1 Análise dos resultados.

Comparado com resultados obtidos para o classificador Rocchio por terceiros como: micro-avg f1 de 66% (YANG, Y. 1999) e micro-avg f1 de 82% (MOSCHITTI, A. 2003), pôde-se notar uma grande diferença entre os resultados obtidos. Este fato pode ser explicado pelos seguintes fatores, como relatado por Yang, Y.; Liu, X (1999), Yang, Y. (1999) e Moscchitti, A. (2003):

- Uso de valores diferentes para os parâmetros β e γ;
- Uso de versões diferentes da coleção Reuters;
- Uso de diferentes esquemas de pesos dos termos;
- Uso de algoritmos diferentes para normalização morfológica;
- Remoção de termos irrelevantes utilizando listas de termos diferentes.

Apesar de f1-measure de 69,5% poder ser considerado baixo, o resultado de micro-avg f1-measure de 82% obtido por Moschitti, A. (2003) mostra que o classificador Rocchio pode obter resultados próximo ao classificador SVM (Support Vector Machines), considerado um dos melhores classificadores, que apresenta resultados de micro-avg f1-measure de 85% (MOSCHITTI, A. 2003), 85,9% (Liu, Y., Yang, Y., 1998), 86% (Joahims, T., 1998).

7 CONSIDERAÇÕES FINAIS.

A Internet é um dos principais meios de comunicação e tornou-se o principal repositório de conhecimento e cultura da humanidade.

Devido a sua popularidade, a Internet adquire cada vez mais importância para as empresas, que cada vez mais ampliam sua presença na Internet. Neste cenário, surgiram os mais variados tipos de problemas como:

- Desvio de tráfego ou clientela;
- Incremento de tráfego em buscas;
- Fraudes com objetivos financeiros, políticos ou outros;
- Aumento de credibilidade do sítio por falsas parcerias;
- Denegrir imagem da empresa, produtos ou marca.

Assim, uma nova gama de serviços na Internet surgiu para tentar solucionar estes problemas:

- Serviços anti-phishing.
- Serviços anti-plágio.
- Serviços de detecção de violação de propriedade intelectual como:
    - Uso indevido de marcas.
    - Uso indevido de logotipos.
    - Uso indevido de músicas.

Também, surgiram serviços de *clipping* de notícias para o gerenciamento da imagem da empresa, de suas marcas e produtos na Internet.

Estes novos serviços têm em comum a necessidade de mecanismo que permitam localizar páginas na Internet de forma rápida e precisa. Porém, estimativas indicam que o número total de páginas na Web seja superior a 11 bilhões. Isto cria uma grande dificuldade para encontrar a informação desejada na Internet.

Mecanismos de busca na Internet foram desenvolvidos para tentar solucionar este tipo de problema, utilizando técnicas de recuperação de páginas com determinados termos e ordenação por relevância. Porém, estes métodos não são capazes de solucionar o problema e geralmente retornam uma quantidade muito grande de páginas irrelevantes. Não sendo capazes de solucionam o problema enfrentados para a descoberta de uso indevido de obra autoral, uso indevido de

propriedade intelectual ou marca registrada de empresas, pois são voltados para definir qual página é mais relevante ou popular para uma consulta, e não em qual esta sendo cometido algum ato ilícito.

Esta nova classe de serviços depende de mecanismos de busca e necessitam de métodos mais sofisticados de busca, pois não estão interessados em páginas que possuam apenas um determinado termo, mas sim, estão interessados em páginas de um assunto determinado, localizadas em um determinado lugar, ou seja, estão interessados em páginas que preencham vários requisitos e de algum forma inseridas em um determinado contexto.

Vários métodos de mineração da Web foram propostos para tratar alguns destes problemas, porém isoladamente estes também não são capazes de solucionar estes problemas, pois tratam de um único aspecto do problema, como por exemplo, categorizar uma página. Assim, atividades, como as citadas anteriormente, necessitam de vários métodos sejam combinados para que, de forma automática, possam ser realizadas mais eficientemente e com maior precisão.

Baseando-se no conceito de contextualização, apresentado no início deste trabalho, que procura vincular páginas na Internet a sua categoria, sua origem, seu relacionamento com outras páginas, associado às necessidades da aplicação, é proposto à programação de contextualização que utiliza o encadeamento de métodos para criar métodos mais sofisticados necessário em vários serviços de Internet.

7.1 Conclusão.

A contextualização de páginas na Internet fornece um novo recurso para tratamento de páginas no ambiente da Internet. A programação de contextualização, utilizando o encadeamento de métodos, no qual métodos mais simples e rápidos são utilizados como filtros para seleção de páginas para posterior aplicação de métodos mais sofisticados, devem possibilitar que métodos mais sofisticados, que geralmente não são utilizados em mecanismos de busca, possam ser utilizados em mecanismos de busca na Internet de forma satisfatória.

Assim, a contextualização utilizada adequadamente pode minimizar o montante de informações a serem analisadas nas mais variadas atividades, pois o

contexto da página permite uma identificação mais precisa deste e deve reduzir a quantidade de páginas irrelevantes retornadas para o usuário.

Este trabalho apresentou alguns métodos que podem ser utilizados para a contextualização, como:

- Métodos de busca na Web através de mecanismos de busca.
    - Métodos para definição de relevância de páginas.
- Métodos de categorização:
    - Classificador Rocchio (Hull[22], 1994 apud SEBASTIANI, F. 2002)
    - Classificador Naive Bayes;
    - Árvores de Decisão;
    - Redes Neurais Artificiais (Ng et. al., 1997; WIENER, E., PEDERSEN, J. O. WEIGEND, A. S., 1995);
    - Máquinas de Vetores Suporte (Vapnik[23], 1995 apud YANG, Y.; LIU, X., 1999).;
    - kNN (k - Nearest Neighbor).
- Métodos de agrupamento:
    - Agrupamento utilizado HITS (GIBSON, D.; KLEINBERG, J. E RAGHAVAN, P., 1998).
    - Small World, proposto por Adamic, L. (1999).
    - Máximo Fluxo e Mínimo Corte, proposto por Flake G. W.; Giles C. L. E Coetzee F. M. (2002).
    - Grafos bipartido, proposto por Reddy, P. K. e Kitsuregawa, M., (2001).
    - Conjuntos conceituais fuzzy, proposto por Tomiyama, T. et al., (2003).

O trabalho apresenta a questão de como tais métodos poderiam ser compostos para a finalidade de contextualização.

---

[22] Hull, D.A. *Improving text retrieval for the routing problem using latent semantic indexing.* In Proceedings of SIGIR-94, 17th ACM international Conference on Research and Development in Information Retrieval (Zürich, Switzerland), p. 278 – 288, 1996.

[23] Vapnic, V. The Nature of Statistical Learning Theory. Springer, New York, 1995.

7.2 Contribuições.

Este trabalho agrupa em um único documento diversos métodos que podem ser utilizados para programação de contextualização, cujas referências encontram-se espalhadas por diversas referências bibliográficas.

Além disso, apresenta o problema do encadeamento dos métodos para programação de contextualização, tópico que não é tratado na literatura.

7.3 Trabalhos futuros.

Este trabalho serve como base para outros trabalhos relacionados à contextualização de páginas na Internet. Trabalhos futuros incluem:

- Análise da composição destes métodos.
- Análise da complexidade computacional dos métodos de contextualização.
- Realização de estudo de caso para problemas específicos como *clipping* de notícias e detecção de violação de propriedade intelectual na Web.

Também, pode ser explorado o uso de alguns destes métodos em um metabuscador para Internet, procurando viabilizar o uso prático destes métodos fora de um ambiente de teste.

REFERÊNCIAS

ADAMIC, L. *The Small World Web,* Proceeding of ECDL´99, p. 443-452, 1999.

ALMEIDA, N. F.; TELLES, G. P.; MARTINEZ, F. H. V. *Algoritmos e heurísticas para comparações exata e aproximada de seqüências*. XXIV Jornadas de Atualização em Informática. São Leopoldo, v. 1, p. 1545-1587, 2005.

ALTAVISTA. Disponível em http://www.altavista.com. Acessado em maio de 2005.

APACHE LUCENE PROJECT. *Lucene 1.4 API*. Disponível em http://lucene.apache.org/java/docs/api/index.html. Acessado em out. de 2005.

APACHE LUCENE PROJECT; *Apache Lucene – Overview*. Disponível em http://lucene.apache.org/java/docs/. Acessado em out. de 2005.

APTE, C.; DAMERAU, F.; WEISS, S.M. *Text Mining with Decision Trees and Decision Rules.* Conference on Automated Learning and Discovery, Carnegie-Mellon University, 1998.

BHARAT, K.; MIHAILA, G. A. *When Experts Agree: Using Non-Affiliated Experts to Rank Popular Topics*, ACM Transaction on Information System, Vol. 20, Nº 1, p. 47-48, 2002.

CARDOSO, O. N. P.; *Recuperação de Informação*; INFOCOMP Journal of Computer Science, Depto Ciência da Computação, Universidade Federal de Lavras , Lavras, MG, Brazil, Vol. 2, N. 1, p.33-38, 2000.

CHAKRABARTI, S. et al. *Mining the Web's Link Struture*, IEEE Computer, 1999.

CHAKRABARTI, S. *Data mining for hypertext: a tutorial survey*, ACM SIGKDD Explorations Newsletter. Vol. 1 , Issue 2, p. 1-11, ACM Press, 2000.

CLIPEX. Disponível em http://www.clipex.com.br. Acessado em jun. de 2005.

CUTTING, D. *Jakarta Lucene 1.4 Index File Formats*. Disponível em http://wiki.apache.org/nutch/. Acessado em out. de 2005.

CYVEILANCE. Disponível em http://www.cyveilance.com. Acessado em jul. 2005.

DIEB. *Dicionário Interativo da Educação Brasileira*. Disponível em http://www.educabrasil.com.br/eb/dic/dicionario.asp.

FLAKE G. W.; GILES C. L.; COETZEE F. M. *Self-organization and identification of Web communities*. IEEE Computer Society Press, p. 66 - 71, 2002.

FERREIRA, A. B. H. *Novo Dicionário Aurélio da Língua Portuguesa*. Editora Positivo Livros S.A., 3ª ed., 2004.

GIBSON, D.; KLEINBERG, J.; RAGHAVAN, P. *Inferring Web Communities from Link Topology*, Proceedings of the 9th ACM Conference on Hypertext and Hypermedia, 1998.

GOOGLE. Disponível em http://www.google.com. Acessado em maio de 2005.

GRECO, G.; GRECO, C.; ZUMPANO, E. *A stochastic Approach for Modeling and Computing Web Communities*, IEEE Computer Society - Proceedings of the 3rd International Conference on Web Information Systems Engineering, p. 43 – 52, 2002.

GULLI, A.; SIGNORINI, A. *The indexable web is more than 11.5 billion pages,* International World Wide Web Conference, Special interest tracks and posters of the 14th international conference on World Wide Web, ACM, p. 902 – 903, 2005.

HTDIG. Disponível em http://www.htdig.org/. Acessado em ago. 2005.

HU, W. C. at al. *An Overview of World Wide Web Search Technologies*, Proceedings of the 5th World Multi-Conference on System, Cybernetics and Informatics – SCI2001, 2001.

IMAFUJI, N.; KITSUREGAWA, M. *Finding a Web community by maximum flow algorithm with HITS score based capacity*. Proceeding of the Eighth International Conference on Database Systems for Advanced Applications (DASFAA'03) IEEE, 2003.

INFO4. Disponível em http://www.info4.com.br; Acessado em jun. de 2005.

ITHENTICATE. Disponível em http://www.ithenticate.com. Acessado em jun. 2005.

JAIN, A. K.; MURTY, M. N.; FYNN, P. J. *Data Clustering: A Review.* ACM Computing Surveys, Vol. 31, Nº 3, p. 264 - 323, 1999.

JOACHIMS, T. *A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization*, Proc. of the 14th International Conference on Machine Learning ICML97, p. 143 -151, 1997.

JOACHIMS, T. *Text Categorization with Suport Vector Machines: learning with many relevant features*. In Proceedings of ECML-98, 10th European Conference on Machine Learning, p. 137 – 142, 1998.

KARTOO. Disponível em http://www.kartoo.com. Acessado em maio de 2005.

KHARE, R. et. al. *Nutch: A Flexible and Scalable Open-Source Web Search Engine*. CommerceNet Labs Technical Report, 2004.

KINTO, E. A.; DEL-MORAL-HERNANDEZ, E.; "Classificadores de Texto Reduzido Baseados em SVM"; Proceedings of the VII CBRN - Congresso Brasileiro de Redes Neurais, Natal, 2005.

KLEINBERG, J. M. *Authoritative Source in a Hyperlinked Environment*, Jornal of the ACM, Vol. 46, Nº 5, p. 604-632, 1999.

KOSALA, R.; BLOCKELL, H. *Web Mining Research: A Survey*; ACM SIGKDD Explorations, Vol. 2, Issue 1, 2000.

KROEZE, J. H.; MATTHEE, M. C.; BOTHMA, T. J. D. *Differentiating data and text mining terminology*, Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology SAICSIT '03, 2003.

LOH, S.; WIVES, L. K. e FRAINER, A. S. Uma abordagem para a Busca Contextual de Documentos na Internet; RITA v. 4, n. 2; 1997.

MANBER, U. *Introduction to algorithms: A creative approach,* Editora Addison-Wesley Publishing Company Inc. p. 238 – 243, 1989.

MARON, M. E.; KUHNS, J. L. *On Relevance, Probalistic Indexing and Information Retrieval*. Journal of the ACM (JACM), Vol. 7, Issue 3, p. 216 – 244, 1960.

MCBRYAN, O. A. *GENVL and WWWW: Tools for Taming the Web.* First International Conference on the World Wide Web. CERN, 1994. disponível em < http://www94.web.cern.ch/PapersWWW94/mcbryan.ps>, acessado em 12 dez 2004.

METACRAWLER. Disponível em http://www.metacrawler.com. Acessado em maio de 2005.

MUKHERJEA, S. e FOLEY, J. D. Showing the Context of Nodes in the World-Wide Web. In Human Factors in Computing Systems: Companion to the CHI '95 Conference. New York: ACM, 1995.

NG, H. T.; GOH, W. B.; LOW, K. L. *Feature selection, perception learning, and a usability case study for text categorization*. ACM SIGIR Forum , Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval SIGIR '97, Vol. 31, Issue I, 1997.

NIC.BRa - Núcleo de Informação e Coordenação do Ponto BR. Disponível em http://www.nic.br/dominios/tabela_b.htm. Acessado em 15 ago. 2005.

NIC.BRb - Núcleo de Informação e Coordenação do Ponto BR. *Nota de Esclarecimento sobre utilização do DPN .edu*. Disponível em http://www.nic.br/dominios/nota-edu.htm. Acessado em 15 ago. 2005.

MOSCHITTI, A. A study on optional parameter tuning for Rocchio Text Classifier. In proceedings of the 25th European Conference on Information Retrieval Research (ECIR), Pisa, Italy, 2003.

MNOGOSEARCH. Disponível em http://www.mnogosearch.org/. Acessado em ago. de 2005.

NUTCH. Disponível em http://www.nutch.org. Acessado em maio de 2005.

NUTCH 0.7 API, Disponível em http://lucene.apache.org/nutch/apidocs/index.html; Acessado em set. de 2005.

OPEN Directory Project. Disponível em http://www.dmoz.org. Acessado em maio de 2005.

PAGE, L. et al. The PageRank Citation Ranking: Bringing Order to the Web, 1998.

PAGE, L.; BRIAN, S. *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, In Proceedings of the Seventh International World Wide Web Conference, 1998.

PIROLLI, P.; PITKOW, J.; RAO, R. *Silk from a sow's ear: extracting usable structures from the Web*, Conference on Human Factors in Computing Systems, 1996. Disponível em http://www2.parc.com/istl/projects/uir/pubs/items/UIR-1996-18-Pitkow-CHI96-WWW.pdf. Acessado em 15 abr. 2005.

REDDY, P.K.; KITSUREGAWA, M. *An approach to relate the web communities through bipartite graphs*. Proceedings of the Second International Conference on Web Information Systems Engineering. IEEE, Vol. 1 p. 301 - 310, 2001.

RFC1738 - Uniform Resource Locators (URL). Disponível em http://www.ietf.org/rfc.html. Acessado em 15 ago de 2005.

RICH E.; KNIGHT, K. *Inteligência Artificial*. Ed. Makron Books, 1993.

RIJSBERGEN, C. V. *Information Retrieval*. 2ª ed., Butterworths, London, 1979.

RILOFF, E. *Little words can make a big difference for text classification*, Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, p. 130 – 136, 1995.

RINO, L.H.M.; PARDO, T.A.S. *A Sumarização Automática de Textos: Principais Características e Metodologias*. Anais do XXIII Congresso da Sociedade Brasileira de Computação, Vol. VIII. III Jornada de Minicursos de Inteligência Artificial (III MCIA), p. 203 - 245. 2003.

RISVIK, K. M.; AASHEIM, Y.; LIDAL, M. *Multi-tier Architecture for Web Search Engines*, Proceedings of the First Conference on Latin American Web Congress, p. 132, 2003.

RUTHVEN, I.; LALMAS, M. *A survey on the use of relevance feedback for information access systems.* Knowledge Engineering Review. Vol 18. Issue 2. pp 95-145. 2003.

SALTON, G. *Introduction to Modern Information Retrieval*. McGraw-Hill Computer Science Series. McGraw-Hill, New York. 1983.

SALTON, G.; BUCKLEY, C. *Term-Weighting approaches in automatic text retrieval*, Information Processing & Management, Vol. 24, No. 5, p. 513-523, 1988.

SALTON, G.; WONG, A.; YANG, C. S. *A Vector Space Model for Automatic Indexing,* Communications of the ACM, vol. 18, n. 11, p. 613 - 620, 1975.

SEBASTIANI, F. *Machine Learning in Automated Text Categorization*, ACM Computing Surveys, Vol. 34, No. 1, pp. 1 – 47, 2002.

SCHWARTZ, C. *Web Search Engines*, Journal of the American Society for Information Science, vol. 49, p. 973 – 982, 1998.

SPEECHBOT: A Search Engine for Sound. Disponível em http://www.hpl.hp.com/news/2003/apr_jun/SpeechBot.html. Acessado em maio de 2005.

SWISH-E. Disponível em http://swish-e.org/. Acessado em ago. de 2005.

TURNITIN. Disponível em http://www.turnitin.com. Acessado em jun. de 2005.

TOMIYAMA, T. et al. *Concept-Based Web Communities for Google Search Engine*, The IEEE International Conference on Fuzzy Systems, 2003.

WATTS, D. J.; STROGATZ, S. H. *Collective dynamics of "small-world" networks*, Nature 393, 1998.

WEBFERRET. Disponível em http://www.webferret.com. Acessado em maio de 2005.

WEBGLIMPSE. Disponível em http://webglimpse.net/. Acessado em ago. de 2005.

WIENER, E.; PEDERSEN, J. O.; WEIGEND, A. S. *A Neural Network Approach to Topic Spotting*. Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval, 1995.

YAHOO. Disponível em http://www.yahoo.com. Acessado em maio de 2005.

YANG, Y. *An Evaluation of Statistical Approaches to Text Categorization*, Information Retrieval, Publisher Kluwer Academic Publishers, Volume 1, Issue 1 - 2, p. 69 – 90, 1999.

YANG, Y.; LIU, X. *A re-examination of text categorization methods.* Proceedings of the 22nd annual International ACM SIGIR conference on research and development in information retrieval, 1999.

YATES, R. B.; RIBEIRO B. A. *Modern Information Retrieval*, ACM Press/Addison Wesley Longman Limited, 1999.

ZAMIR, O.; ETZIONI, O. *Web document clustering: A feasibility demonstration*. In Proc. ACM SIGIR'98, 1998.