Pós-Graduação em Ciência da Computação

"Uma Proposta para a Atualização da Base de Dados em Engenhos de Busca utilizando Classificadores"

Por

Luciano de Andrade Barbosa

Dissertação de Mestrado

Universidade Federal de Pernambuco

posgraduacao@cin.ufpe.br

www.cin.ufpe.br/~posgraduacao

Recife, Março/2003

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUCIANO DE ANDRADE BARBOSA

"Uma Proposta para a Atualização da Base de Dados em Engenhos de Busca utilizando Classificadores"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: ANA CAROLINA SALGADO
CO-ORIENTADOR: FRANCISCO DE ASSIS TENÓRIO DE CARVALHO

Agradecimentos

Ao Centro de Informática, por ter me disponibilizado parte dos recursos necessários para a conclusão deste trabalho,

À professora Ana Carolina Salgado, por sua orientação e apoio para a conclusão deste projeto,

Ao professor Francisco Tenório, por sua orientação e sugestões para este trabalho,

À empresa Mobile, pela disponibilização da infra-estrutura necessária para execução dos experimentos apresentados neste trabalho,

A minha família e amigos, pelo apoio constante, especialmente aos amigos Thiago Santos, Alexandre Damasceno, Vander Alves, Oscar Miranda, Fernando Trinta, Ivan Gesteira, Franklin Ramalho e Rodrigo Teixeira pelas observações sugeridas à dissertação,

Aos membros da banca examinadora:

Jacques Robin

Jorge Henrique Cabral Fernandes

Ana Carolina Salgado

Resumo

Os Sistemas de Recuperação de Informação (SRI) têm por tarefa básica processar uma consulta feita por um usuário, e, a partir dela, recuperar um conjunto de documentos armazenados em uma coleção de dados, que são relevantes às necessidades de informação deste usuário. As bibliotecas estão entre as primeiras instituições a adotarem um SRI para facilitar o acesso às suas informações, através do uso de consultas bibliográficas. Uma das características deste tipo de sistema em bibliotecas é a manipulação controlada dos dados, ou seja, há um controle, por exemplo, de quem são os autores dos documentos existentes, em que momento estes documentos foram inseridos, modificados ou retirados da coleção. Ao contrário dos dados utilizados por um sistema para consultas bibliográficas, a *Web* é uma coleção não controlada de documentos, ou seja, a todo instante, estão sendo inseridos e modificados documentos por diferentes pessoas, em locais distintos. Além dessa alta dinamicidade, a *Web* possui um número muito grande de documentos, que vem aumentando exponencialmente. No entanto, para que um SRI possa disponibilizar informações sobre o conteúdo da *Web* de forma eficiente, é preciso que ele saiba lidar com esses problemas apresentados. Para isso surgiu um novo tipo de SRI, os engenhos de busca. Eles têm como um de seus grandes desafios manter as informações de sua base de dados atualizadas, principalmente, porque eles são executados sobre recursos limitados (banda passante, memória e processamento). Neste sentido, este trabalho propõe uma solução para o problema da atualização da base de dados de um engenho de busca, focando na utilização racional dos recursos utilizados por ele para a realização desta tarefa. Esta solução baseia-se no uso de uma política não-uniforme, na qual elementos mais dinâmicos são atualizados mais freqüentemente do que os menos dinâmicos. Para a utilização desta política, são utilizadas técnicas de Aprendizagem de Máquina e estatística para identificar os grupos de páginas de dinamicidade semelhantes. Um protótipo contendo esta solução é desenvolvido a fim de validar seu desempenho em um

ambiente real e um estudo de caso é apresentado visando mostrar a maior eficiência desta solução em relação a uma abordagem tradicional de atualização.

Palavras-chave: engenho de busca, base de índices, classificadores, desperdício de recursos.

Abstract

Information retrieval systems (IRS) have as main task processing a query on behalf of a user and, based on such query, retrieving a set of stored documents in data collection, which are relevant to the users' information necessities. The libraries were the first institutions to adopt an IRS to facilitate the access to its information through the use of bibliographical queries. One of the features of this kind of systems is the controlled manipulation of data, in other words, there is a control of, for instance, who are the authors of the existent documents, when these documents were inserted, modified or removed from the collection. Unlike the data from a library system, the Web is an uncontrolled collection of documents, in other words, at every moment, documents are inserted and modified for different persons in dissimilar places. Besides that high dynamism, the Web has a great number of documents, that is growing exponentially. However, for an IRS to release information about the Web's content efficiently, it is necessary that it knows to how tackle these problems. For this reason, a new kind of IRS has emerged, the search engines. One of their biggest challengers is to keep up-to-date the information within their databases mainly because they are run under limited resources (bandwith, memory and processing). Therefore, this work proposes a solution to update the database of a search engine, focusing on the unwasteful utilization of the resources required to accomplish this task. This solution is based on the use of a non-uniform policy, where the more dynamic elements are updated more frequently than the less dynamic ones. In order to use this policy, Machine Learning and statistics techniques are used to identify the groups of pages of similar dynamism. A prototype with this solution is developed in order to validate its performance in real environment and a case study is presented to show how higher is the efficiency of this solution in relation to the traditional approach.

Key words: search engine, index base, classifiers, waste of resources.

Índice

Índice de Figuras

Índice de Equações

Índice de Tabelas

Capítulo 1 Introdução

A Internet é um meio de comunicação recente. Sua origem data do fim da década de 60 quando foi criada a ARPANET, uma rede de computadores que conectava algumas universidades americanas.

Desde então, ela vem tornando-se cada vez mais popular. No Brasil, por exemplo, o número de internautas vem crescendo constantemente. Segundo o Instituto Ibope E-ratings[1], estima-se que existam 7,8 milhões de internautas (dados de julho de 2002) e esse número vem aumentado a cada mês.

Uma das conseqüências da Internet ter se tornado popular é o aumento das informações contidas nela. De acordo com [28] e [29], o tamanho da *Web* dobrou em menos de dois anos, e essa taxa de crescimento está projetada para continuar nos próximos dois anos. Assim a *Web*, criou um novo desafio para a computação: encontrar informações relevantes nessa grande rede de dados.

Além do seu tamanho, a *Web* apresenta outra característica interessante: a dinamicidade das informações nela publicadas. Num estudo apresentado em [11], quase 40% das páginas da *Web* monitoradas foram modificadas num período menor ou igual a 1 semana.

Para poder lidar com essas características da *Web*, tamanho e dinamicidade, surgiram os engenhos de busca, com o intuito de ajudar as pessoas a encontrarem informações relevantes publicadas nela. Dessa maneira, os engenhos de busca

[1] Página *Web* do Instituto Ibope E-Ratings: http://www.ibope.com.br/eratings/ogrupo/empresa/eratings

desempenham um papel fundamental para facilitar o acesso a essas informações. Algumas estatísticas retiradas do *site* Search Engine Watch[2] confirmam este fato:

- Engenhos de busca geram de 7% a 8% do tráfego dos *sites* da *Web*;
- Nove em cada dez usuários da *Web* visitam um engenho de busca cada mês. Eles também o revisitam freqüentemente, cerca de 5 vezes por mês;
- Os engenhos de busca são os recursos mais utilizados pelos americanos quando procuram por respostas, usado 32% das vezes, mais que qualquer outra opção;
- A característica mais popular de um portal é a busca, usada em 49% das visitas;
- Os engenhos de busca lideram a forma que os usuários no Reino Unido localizam *site*s na *Web*. 81% disseram que os engenhos de busca os ajudaram a encontrar os *sites*.

Apesar de sua importância na recuperação de informações da *Web*, segundo [28], em 1999, os engenhos de busca juntos tinham indexado somente 42% de toda a *Web*. Isso ocorre porque os engenhos de busca são executados sobre recursos limitados, como banda passante, memória e processador.

Levantando-se em conta essa limitação de recursos e as características da *Web* mencionadas anteriormente (tamanho, crescimento e dinamicidade), faz-se necessário a utilização desses recursos (banda passante, memória e processador) da forma mais eficiente possível, evitando-se o seu desperdício. Para o processo de atualização da base de dados do engenho de busca, esse desperdício se torna bastante crítico pois, ao visitar-se páginas que não foram modificadas, estar-se-á desperdiçando tanto recursos do engenho de busca, como dos *site*s que hospedam essas páginas.

[2] O Search Engine Watch (http://www.searchenginewatch.com) é um *site* que provê informações sobre busca na *Web*, análise de engenhos de busca comerciais e ajuda aos donos de *site*s a melhorarem a acessibilidade de seus *site*s pelos engenhos de busca.

Tendo em vista a necessidade de se utilizar os recursos dos engenhos de busca da forma mais racional possível, este trabalho se propõe a elaborar e construir uma proposta que realize a atualização das informações contidas no engenho de busca de forma eficiente, evitando o desperdício de recursos para a realização desta tarefa.

Além do problema de recursos, que é algo interno ao funcionamento do engenho de busca, o resultado de uma boa atualização da base de dados implica uma melhor qualidade dos resultados apresentados aos usuários no momento da consulta. Em [28], é mostrado que uma página pertencente à base de dados dos engenhos de busca pode levar vários meses (média de 186 dias) para ser atualizada por esses, e que, em média, 5.3% das páginas retornadas por consultas feitas a engenhos de busca não mais existem. Isso demonstra que as técnicas utilizadas por esses engenhos para atualizarem suas bases não estão provendo uma qualidade boa aos resultados apresentados pelos mesmos, pois trazem nos seus resultados páginas desatualizadas ou, talvez, não mais existentes.

Uma outra motivação para a elaboração deste trabalho foi a construção de um componente de *software*, que implementa a proposta de solução. Este componente seria utilizado no engenho de busca Radix [37], pois a atualização da base deste engenho estava sendo realizada de forma ineficiente, o que levava a uma qualidade baixa das respostas e a um grande desperdício de recursos.

Além de engenhos de busca, essa proposta pode ser usada também por servidores *cache proxy* e por sistemas que desejam guardar retratos da *Web*, como os projetos *WebArchive*[3] e *Internet Archive*[4]. Portanto, de forma mais genérica, essa proposta ou parte dela pode ser utilizada por qualquer sistema que deseja ter uma cópia local da *Web* e manter essa cópia atualizada com a fonte dos dados.

1.1 Abordagem Utilizada

Como apresentado anteriormente, o principal objetivo desta dissertação é apresentar uma solução para o processo de atualização de base de dados em engenhos de busca,

[3] Projeto *WebArchive* - http://webarchive.cs.ucla.edu

[4] Internet Archive - http://www.archive.org

que seja mais eficiente do que as abordagens já existentes, de acordo com as medidas apresentadas na Seção 2.5.2. Para isso, vão se utilizar técnicas de Aprendizagem de Máquina e estatísticas, para a construção de classificadores, que neste trabalho em particular, são funções que inferem a freqüência de modificação da página baseado nas suas características ou no seu histórico de mudanças, para poder classificá-las em classes de alteração.

Para alcançar esse objetivo, foram realizados os seguintes passos:

1. Investigação do comportamento das páginas *Web*, a fim de que se possa inferir o seu comportamento a partir de suas características. Para isso foi realizado um monitoramento do conjunto das páginas mais acessadas da *Web* brasileira durante 100 dias. Obtendo, no final desse período, para cada página, seus atributos e seu histórico de alteração durante o período de 100 dias;
2. Construção do classificador que infere a freqüência de modificação da página baseado nas suas características. A partir dos dados obtidos no passo 1, foram realizadas as seguintes etapas para a construção do classificador: (1) definição dos grupos de alteração de páginas: utilizando os dados de histórico, para cada página, gerou-se um estimador relacionado com sua taxa de modificação, e sobre esses valores gerados para todas as páginas foi realizada a tarefa de agrupamento (mais detalhes sobre agrupamento ver Seção 3.3.2) a fim de que se definissem os grupos de alteração de páginas; (2) seleção dos atributos (ver Seção 3.3.3): nesta etapa foram selecionadas as características realmente relevantes para a construção do classificador, através de técnicas de seleção de atributos; e (3) execução de algoritmos de classificação (detalhes na Seção 3.3.4): uma vez definidos os grupos de modificação das páginas e os seus atributos relevantes, foi realizada a inferência entre esses atributos e esses grupos utilizando-se algoritmos de classificação, criando-se dessa maneira o classificador baseado nas características da página;
3. Construção dos classificadores baseado no histórico de mudança das páginas;
4. Utilizando-se os classificadores construídos nos passos anteriores, é realizada uma simulação utilizando-se parte dos dados do monitoramento (passo 1) a fim

de se comparar algumas configurações dos classificadores para que fosse escolhida qual a melhor delas a ser utilizada no protótipo que contém a proposta de solução;

5. Implementação do protótipo contendo a solução proposta;
6. Realização de um experimento para verificar o comportamento do protótipo utilizando a solução proposta em uma situação real de atualização de base de dados de engenhos de busca, comparando-a com uma proposta tradicional de atualização;

Assim, no final deste trabalho, foi construído um componente de *software* que contém a proposta de atualização, sendo este utilizado em um sistema real.

A linguagem de modelagem utilizada neste trabalho foi UML [36], para apresentar interações entre componentes, funcionamento de sistemas e composição de módulos. Essa linguagem foi adotada, por ela ter se tornado um padrão na área de modelagem tanto na academia quanto no meio comercial. No Apêndice A, é apresentado um pequeno resumo sobre as notações dos diagramas utilizados neste trabalho.

1.2 Contribuições Esperadas

No decorrer das etapas para a construção da proposta de solução, pretende-se obter algumas contribuições, são elas:

- Estatísticas mostrando como se comportam em termos de freqüência de modificação as páginas da *Web* brasileira.
- O classificador baseado nas características da página, que pode ser utilizado por servidores *proxy cache* para a atualização de seus dados locais ou por engenhos de busca no momento inicial de sua atualização;
- O classificador baseado no histórico de alteração das páginas, para ser utilizado por sistemas que desejam manter cópias locais de páginas Web e que as visitam em taxas regulares;

- O componente de *software* implementado em Java, contendo os dois classificadores citados anteriormente, que pode ser utilizado por sistemas que desejam atualizar páginas Web;
- Dados experimentais apresentando o comportamento dessa abordagem e mostrando a sua eficiência em relação a outras existentes.

1.3 Estrutura da Dissertação

Neste trabalho, é apresentada a elaboração e construção de uma proposta de solução para atualizar de forma eficiente as informações presentes no engenho de busca. Essa proposta é, então, implementada e embutida em um sistema real (*SiteSearch*) para que possa ser utilizada para melhorar o processo de atualização deste sistema. São mostrados, neste trabalho, os passos necessários para se chegar nesse objetivo. Assim, além dessa introdução, esta dissertação está organizada da seguinte forma:

Capítulo 2: Engenhos de Busca e a Atualização de seus Dados. É apresentada uma introdução sobre a área de Recuperação de Informação, e, mais particularmente, sobre os elementos principais que compõem os engenhos de busca e as interações entre eles. Alguns conceitos relativos ao processo de atualização das informações do engenho de busca também são mostrados, pois eles serão necessários em discussões nos capítulos seguintes. Por fim, são levantados alguns dos principais desafios em se manterem atualizadas as informações de um engenho de busca.

Capítulo 3: Técnicas de Aprendizagem de Máquina e Estatística e Ferramenta Utilizadas. São introduzidas, neste capítulo, as principais técnicas de Aprendizagem de Máquina e estatísticas utilizadas ao longo deste trabalho, como também a ferramenta utilizada para a construção de parte da proposta de solução.

Capítulo 4: Propostas para Atualização da Base de Índices. São discutidos os pontos negativos e positivos das principais abordagens existentes para a atualização da base de índices, que é onde estão armazenadas as informações contidas no engenho de busca.

Capítulo 5: Construindo os Classificadores para a Atualização da Base de Índices. Baseado na análise das outras abordagens realizada no Capítulo 4, inicialmente, são apresentadas as características básicas da abordagem proposta neste trabalho. De forma geral, essa abordagem tenta prever o comportamento das páginas *Web* em relação à sua taxa de modificação. É necessário, então, saber como essas páginas se comportam. Para isso, foi realizado o monitoramento de cerca de 90.000 páginas da *Web* brasileira, que é relatado neste capítulo. A partir dos dados obtidos nesse processo, apresenta-se um comportamento geral da *Web* brasileira e como foi o processo de implementação dos componentes que tentam prever o comportamento das páginas, que são a base da proposta. Por fim, com parte dos dados do monitoramento, são mostrados os resultados de simulações do comportamento desses componentes e de outras abordagens existentes.

Capítulo 6: Protótipo. É descrita a composição e o funcionamento do protótipo que utiliza a proposta de solução apresentada no Capítulo 5. É apresentado, também, um estudo de caso sobre o funcionamento desse protótipo em um ambiente real, e seu desempenho é comparado com uma abordagem tradicional de atualização de base de índices.

Capítulo 7: Conclusões. Mostram-se as principais conclusões obtidas com este trabalho, suas contribuições, suas limitações e quais atividades poderão ser feitas no futuro para estender e complementar este trabalho.

Capítulo 2 Engenhos de Busca e a Atualização de seus Dados

2.1 Introdução

Os engenhos de busca desempenham um papel fundamental para os usuários que utilizam a *Web*, pois facilitam o acesso às informações publicadas nela. Entretanto, para que esta tarefa seja bem desempenhada é necessário que as informações disponibilizadas estejam atualizadas, pois, caso contrário, grande parte das consultas realizadas pelos usuários retornarão páginas que não mais contêm a informação de que o usuário deseja, ou páginas que nem mais existem.

Neste capítulo, será apresentado o que são engenhos de busca e como eles estão constituídos, e alguns conceitos que contextualizam o problema de atualização das informações em um engenho de busca. O capítulo está organizado da seguinte maneira: na Seção 2.2 será apresentada uma introdução sobre a área de Recuperação de Informação; na Seção 2.3, descreve-se como um engenho de busca está constituído; na Seção 2.4 são mostrados alguns conceitos relativos à atualização das informações de um engenho de busca e quais os principais problemas deste processo de atualização; e, por fim, na Seção 2.5, é apresentada uma conclusão sobre este capítulo.

2.2 Recuperação de Informação

A área de Recuperação de Informação lida com a representação, armazenamento, organização e acesso a itens de informação. A representação e organização destes itens devem prover ao usuário fácil acesso à informação na qual ele está interessado. Para isso, o usuário precisa traduzir sua necessidade de informação em uma consulta para que possa ser processada pelo Sistema de Recuperação de Informação (SRI) [2], ou seja, um SRI tem por tarefa básica processar uma consulta feita por um usuário, e, a partir dela, recuperar um conjunto de documentos armazenados em uma coleção de dados, que são relevantes às necessidades de informação deste usuário.

As bibliotecas estão entre as primeiras instituições a adotarem um SRI para facilitar o acesso às suas informações, através do uso de consultas bibliográficas. Uma das características deste tipo de sistema é que a manipulação dos dados realizada nele é bem controlada, ou seja, há um controle, por exemplo, de quem são os autores dos documentos existentes, em que momento estes documentos foram inseridos, modificados ou retirados da coleção.

Ao contrário dos dados utilizados por um sistema para consultas bibliográficas, a *Web* é uma coleção não controlada de documentos, ou seja, a todo instante, estão sendo inseridos e modificados documentos por diferentes pessoas, em locais distintos, sendo o conteúdo e formato destes os mais variados possíveis. Além dessa alta dinamicidade, a *Web* possui um número muito grande de documentos. No início de 2001, estimava-se que ela possuía cerca 30 bilhões de páginas [5]. No entanto, para que um SRI possa disponibilizar informações sobre o conteúdo da *Web* de forma eficiente, é preciso que ele saiba lidar com esses problemas (dinamicidade e tamanho) apresentados. Para isso surgiu um novo tipo de SRI, os mecanismos de busca, ou engenhos de busca, ou simplesmente engenho.

[5] http://www.clienthelpdesk.com/statistics_research/web_statistics.html

2.3 Módulos do Engenho de Busca

As principais tarefas de um engenho de busca são: (1) extrair e armazenar informações da *Web*, e (2) disponibilizá-las para que os usuários possam realizar consultas sobre elas. Cada uma destas tarefas é desempenhada por um módulo do engenho. Dessa maneira, pode-se dividir um mecanismo de busca em dois módulos principais[6]: o **Módulo de Indexação,** que realiza a extração e o armazenamento dos documentos, e o **Módulo de Busca**, que disponibiliza informações sobre estes documentos, através de consultas. Estes dois módulos realizam operações sobre a **base de índices (BI)** ou simplesmente índice, que contém uma representação das páginas coletadas e outras informações contidas nela, como URL, título, *links* para os quais ela aponta etc. Neste trabalho, a representação das páginas mais essas informações contidas nelas, será uma entidade chamada de *documento*.

O Módulo de Busca procura pelas informações desejadas pelo usuário na BI. Na busca por informações essencialmente textuais, por exemplo, normalmente a consulta é representada por uma expressão contendo um conjunto de palavras-chave. Neste contexto, o processo de busca consiste em: (i) determinar quais dos documentos presentes à BI contêm as palavras-chave utilizadas na consulta; (ii) ordenar através de um algoritmo de ranqueamento esses documentos; e (iii) apresentar o resultado da consulta ordenada para o usuário (Figura 1).

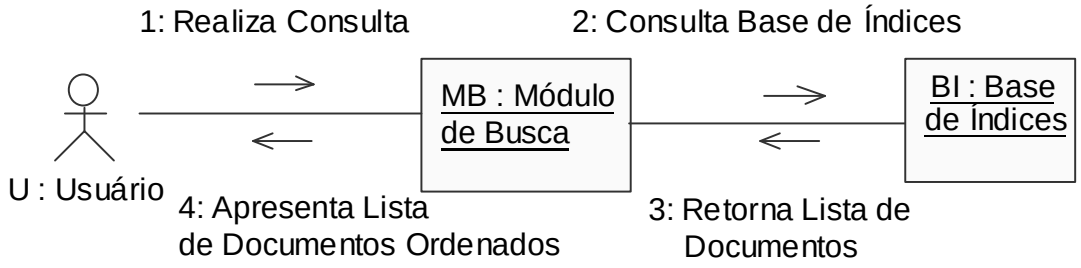


Figura 1 - Diagrama de Colaboração demonstrando uma busca por informações.

[6] Existem trabalhos ([1] e [2]) que particionam o engenho de busca em mais módulos, mas por simplicidade para este trabalho foram agrupados esses vários módulos em apenas dois.

O Módulo de Indexação coleta páginas da *Web*, cria os documentos a partir das informações contidas nas páginas e, então, armazena-os na BI. Este é o módulo, portanto, que realiza modificações sobre a base de índices (Figura 2). Dado que é sobre este módulo do engenho de busca que este estudo é realizado, serão apresentados mais detalhes sobre a sua composição e funcionamento.

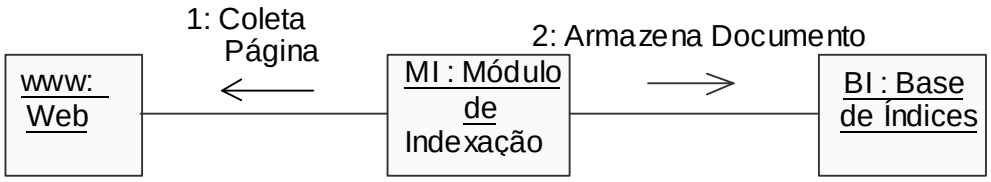


Figura 2- Diagrama de Colaboração demonstrando a indexação.

2.4 Módulo de Indexação

Como o Módulo de Indexação é o responsável por criar e manter a base de índices, é dele a tarefa de aumentar o número de documentos armazenados nela (aumento da cobertura), e de manter esses documentos atualizados (manutenção da atualidade). Para aumentar a cobertura é necessário que o Módulo de Indexação armazene novos documentos no índice e, para manter a atualidade, é necessário que sejam atualizadas os documentos já existentes na BI. Logo, o grande compromisso para o Módulo de Indexação é manter atualizado um grande número de documentos na BI.

Este módulo também é responsável por delimitar o conjunto de páginas *Web* que se deseja armazenar no índice, o que é chamado de escopo. O escopo, por exemplo, pode ser páginas pertencentes a um domínio, como **.br** (identifica páginas da *Internet* brasileira), ou páginas de um único *site* ou conjunto de *sites*, ou ainda toda a *WWW*.

É sobre esse escopo de páginas que serão desempenhadas as tarefas de coleta e extração das informações relevantes das páginas, criando os documentos, que serão inseridos na base de índices. Baseado em [1], para realizar cada uma dessas tarefas o Módulo de Indexação possui os seguintes componentes (Figura 3):

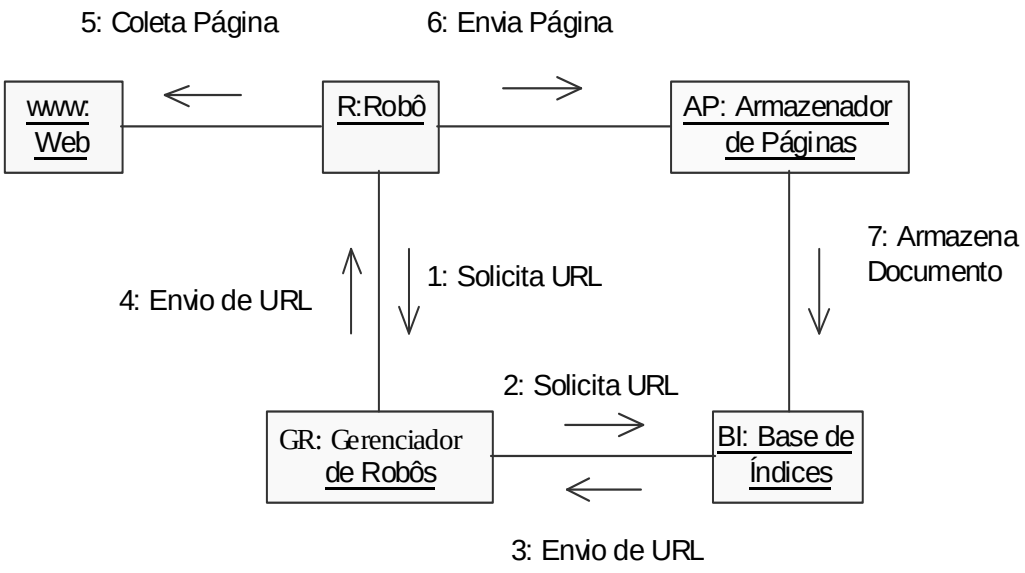


Figura 3 - Componentes do Módulo de Indexação e suas interações.

2.4.1 Robô

Um Robô é um programa que, a partir de uma *URL* (*Unified Resource Location*), realiza uma solicitação *HTTP* (*HiperText Transfer Protocol*) da página representada por essa *URL* ao servidor *Web* onde ela está hospedada. Deste servidor, ele recebe como resposta o conteúdo da página, ou um código de erro em caso da ocorrência de erro. O Robô pode ser chamado também de *crawler* ou *spider*.

2.4.2 Armazenador de Páginas

O Armazenador de Páginas recebe as páginas vindas do Robô, cria os documentos com as informações contidas na página e os armazena na base de índices. Para realizar esta tarefa, ele possui os seguintes componentes:

- *Parser*: a partir da página recebida pelo Robô, ele cria o documento a ser armazenado no índice. Isso pode ser feito da seguinte forma: inicialmente, o texto da página é quebrada em palavras. Após isso, as palavras do documento são comparadas contra uma *stoplist* - uma lista de palavras que

possuem pouco conteúdo semântico, como preposições, pronomes e advérbios. Somente são aceitas as palavras que não estiverem nessa *stoplist*. Terminada essa filtragem de palavras, são dados pesos às palavras desse documento, por exemplo, o peso de uma palavra num documento pode ser a sua freqüência neste [19]. No fim deste processo, temos um vetor de palavras que representam o documento, exceto aquelas que estavam na *stoplist*, cada palavra com o seu peso (Apêndice B)[7]. Além de obter o vetor de palavras que representa a página, o *Parser* também extrai: os *links* para os quais ela aponta, para serem, posteriormente, inseridos na BI e visitados pelo Robô; e algumas informações textuais da página para a sua apresentação no momento da consulta, como título e sua *URL* por exemplo. Todas essas informações estão representadas na entidade documento;

- Gerador de Índice: este componente recebe do *Parser* um documento e o insere nas estruturas de dados que representam a BI. É ele, portanto, o único componente que modifica a base de índices, inserindo ou atualizando dados. Em uma BI que, por exemplo, utiliza arquivos invertidos (Apêndice B), o Gerador de Índice irá criar esses arquivos com as informações dos documentos e inseri-los no índice.

2.4.3 Gerenciador do Robô

O Gerenciador do Robô desempenha um papel fundamental dentro do Módulo de Indexação. De forma simples, ele guia os passos do Robô através da *Web*. São decisões tomadas neste componente que: (1) delimitam o escopo das páginas a serem armazenadas na Base de Índice; (2) decidem quais novas *URLs* devem ser enviadas ao Robô; e (3) quais dos documentos existentes na BI devem ser atualizadas. De forma mais aprofundada, as principais tarefas desempenhadas por este componente são:

[7] Este modelo é apenas um exemplo de opção de representação que pode ser adotada.

- **Definir o escopo**: é definido nele qual o escopo das páginas que vai ser disponibilizado para a consulta do usuário;

- **Aumentar a cobertura do índice**: é preciso que sejam inseridas novas *URLs* na BI, para que seja aumentada sua cobertura. Um aumento de cobertura significa que o engenho aumenta a sua capacidade de disponibilizar informações sobre temas mais específicos, que não são citados em muitos documentos. Entretanto, na maioria dos casos, o Módulo de Indexação não consegue indexar todas as páginas pertencentes ao seu escopo, devido ao tamanho deste. Por exemplo, mesmo os maiores engenhos de busca que têm como escopo a *Web*, conseguem cobrir apenas uma pequena fração dela [28]. Por essa razão, é importante para o Módulo de Indexação indexar páginas mais importantes em primeiro lugar, para que essa porção da *Web* indexada seja a mais significativa possível[8];

- **Atualizar a BI:** para que a base de índices seja mantida atualizada, é necessário que este componente forneça ao Robô *URLs* para serem atualizadas. Para isso, ele deve decidir que páginas revisitar e com que freqüência. Uma forma de atualização, por exemplo, seria se uma certa página muda raramente, o Módulo de Indexação deve visitar essa página menos freqüentemente, para visitar com mais freqüência páginas que mudam mais [1]. Uma deficiência nesta tarefa implica uma base de índices desatualizada e, como conseqüência disso, respostas com páginas não mais existentes ou que já não mais possuem as informações procuradas pelos usuários.

Então, uma política que poderia ser utilizada pelo Gerenciador do Robô seria: (1) selecionar novas *URLs* a serem indexadas, dando-se prioridade àquelas mais

[8] Existem várias métricas que medem a importância de uma página *Web* como PageRank [32], HITS [23] e GHHITS [10], entretanto não serão apresentadas informações mais detalhadas sobre este assunto, por ele não se tratar do tema principal deste trabalho.

importantes; e (2) para a atualização as páginas já indexadas, separá-las em grupos de freqüência de mudanças e, dentro de cada grupo, dar prioridade às que são mais importantes. É sobre esse processo de atualização de páginas que é realizado este trabalho.

2.4.4 Base de Índices[9]

Na base de índices estão armazenadas os documentos, que representam as *URLs* que são utilizadas pelo Gerenciador do Robô para guiar os "passos" do Robô, e as informações textuais utilizadas para a apresentação dos resultados das consultas para os usuários, como título da página, a URL que a representa etc.

Uma estrutura de dados bastante utilizada para o armazenamento de documentos em SRIs são os arquivos invertidos, ou variantes deles [2], que consistem num conjunto ordenado de palavras, cada uma tendo ligações para os documentos que as contêm (ver Apêndice B).

2.5 Atualização da Base de Índices

Uma vez apresentado o que é um engenho de busca e como ele é constituído, nesta seção, vai-se apresentar, com mais detalhes, o processo de atualização da BI realizado pelo Módulo de Indexação, pois este é o tema principal deste trabalho. Na Seção 2.5.1, serão apresentadas políticas utilizadas para o processo de atualização, na Seção 2.5.2, vai-se introduzir algumas métricas que medem a eficiência desse processo e, por fim, na Seção 2.5.3, são mostradas algumas limitações que dificultam um processo de atualização eficiente.

2.5.1 Políticas de Atualização

Para realizar a atualização da BI, podem ser utilizados dois tipos de políticas [12]:

[9] Embora este componente não seja parte do Módulo de Indexação, é interessante apresentá-lo aqui para uma melhor compreensão do funcionamento deste módulo.

- *Política Uniforme de Visitação*: todos os elementos do índice são sincronizados à mesma taxa, não se preocupando com que freqüência eles mudam. Portanto, um dado elemento após ser visitado, somente será revisitado após ter sido realizada a visitação de todos os outros elementos do índice.

- *Política Não-Uniforme de Visitação*: os elementos do índice são sincronizados em diferentes taxas, onde os elementos com maior taxa de alteração são atualizados mais freqüentemente do que os com menor taxa.

A abordagem mais adotada na prática, segundo [39], é a política uniforme de atualização, devido a sua simplicidade de implementação. A política não-uniforme, no entanto, é mais complexa, pois para poder implementá-la, necessita-se de uma estratégia para definir grupos de modificação de páginas, ou seja, agrupar páginas com taxa de modificação semelhante em um mesmo conjunto, para então poder atualizar cada grupo de acordo com sua taxa de modificação.

2.5.2 Medidas de Eficiência

A eficiência do processo de atualização da base de índices de um engenho de busca pode ser aferida através das seguintes medidas:

- Atualidade da BI [12]: seja $S = \{e_1, ..., e_N\}$ uma base de documentos local com N elementos. Idealmente, todos os N elementos deveriam estar atualizados. Entretanto, somente M (< N) elementos estarão atualizados em um determinado intervalo de tempo (atualizado significa que os valores armazenados são iguais aos valores encontrados fora da base de documentos). Define-se a atualidade de S em um tempo t dado por $F(S; t) = M / N$. Em outras palavras, a atualidade é a fração da base de documentos que está atualizada. Por exemplo, F (S; t) será igual a 1 (um) se todos os elementos da base estiverem atualizados, e F (S; t) será igual a 0 (zero) se todos os elementos da base estiverem desatualizados.

É interessante ressaltar a dificuldade de medição exata da atualidade. Na prática, estimam-se valores dado uma amostra de como as informações são alteradas no mundo real. Para definir o valor exato da atualidade, precisa-se que instantaneamente comparar todos os elementos da base de índices com as páginas correspondentes a eles na Web, que é difícil quando se mantém uma grande quantidade de dados armazenados no índice;

- Idade da BI [12]: a idade de um elemento de uma base e_i em um tempo t é:

$A(e_i; t) = 0$, se e_i está atualizado no tempo *t* ou

$A(e_i; t) = t - (\text{tempo de modificação de } e_i)$

Em outras palavras, a idade de um elemento é o intervalo de tempo durante o qual ele se encontra desatualizado no índice. A idade de uma base de dados *S* é dada por:

$$A(S;t) = \frac{1}{N}\sum_{i=1}^{N} A(e_i;t)$$

Equação 1 - Fórmula para calcular a idade médias dos elementos da base de índice.

Portanto, a idade de uma base de dados *S* é definida como a média das idades de cada elemento da base.

- Taxa de mudança: uma mudança indica que a página visitada pelo processo de atualização foi modificada desde a sua última visita a ela. A proporção entre as páginas visitadas que foram realmente modificadas e o total de páginas visitadas é a taxa de mudança do processo de atualização. Os recursos sobre os quais o engenho e os servidores *Web* funcionam são limitados (banda, memória e processador). Quando o mecanismo de busca visita uma página que não foi modificada desde a última vez que ele a visitou estarão sendo desperdiçados recursos tanto dele como dos servidores *Web* visitados. Para o engenho, estes recursos poderiam ter sido melhor gastos, por exemplo, para visitar aquelas páginas que realmente foram alteradas e, para os servidores *Web*, para melhor atender seus usuários [7].

Portanto, quanto maior a taxa de mudança, menos o engenho de busca estará desperdiçando recursos tanto dele como dos servidores *Web* visitados.

As medidas de atualidade e idade do índice são complementares. Na primeira, tem-se uma visão geral do estado de atualização da BI, independentemente de quanto tempo estes elementos estão desatualizados, que é justamente o que a idade da BI mede. Se, por exemplo, para uma BI com muitos elementos sua atualidade for alta, provavelmente, a idade média de seus elementos será pequena, pois como existem poucos elementos desatualizados, a soma dos intervalos de tempo que eles estão desatualizados, será pequena em relação ao número total de elementos na BI.

Essas duas medidas demonstram o estado da base de índices, enquanto que a taxa de mudança se preocupa em como o processo de atualização utiliza os recursos que tem disponíveis. Entretanto, uma taxa de mudança alta no processo de atualização irá refletir numa base de índices com atualidade maior e menor idade, do que se esse processo possuísse uma taxa de mudança baixa.

Portanto, uma abordagem adequada para a atualização da base de índices de um engenho de busca é aquela que mantém a sua atualidade, a idade média das páginas pequena e consome menos recursos para realizar esta tarefa, ou seja, que possua uma alta taxa de mudança.

2.5.3 Limitações na Atualização da BI

Existem limites que impedem que se faça um processo de atualização satisfatório na BI e, conseqüentemente, mantenha-se alta a atualidade. São eles:

- Velocidade dos processos de atualização: quanto mais rápida for a velocidade dos processos de atualização, menor será o tempo para que a base de índices seja totalmente atualizada. Há, no entanto, limitadores para que essa velocidade seja alta;

- Limite ético: se os Robôs fizerem constantes *downloads* de páginas dos *site*s, realizando dessa forma uma série de requisições HTTP sobre o mesmo

servidor *Web*, por um período longo, com pequeno intervalo entre elas, isso irá sobrecarregá-lo. Esta não é uma prática aceitável, já que estão sendo utilizados seus recursos [25];

- Limite Operacional: o engenho de busca é executado sobre recursos limitados de banda, hardware e armazenamento. Deve-se, portanto, evitar que eles sejam sobrecarregados durante o processo de atualização;

- Tamanho e crescimento do escopo: tomando a *Web* como exemplo de escopo, vários estudos têm tentado estimar o seu tamanho. Embora eles publiquem números com algumas diferenças, a maioria deles concorda que estejam disponíveis mais de um bilhão de páginas. Dado que o tamanho médio de uma página *Web* é de cerca de 5 a 10 Kbytes, somente a quantidade de dados textuais é de dezenas de *terabytes* [1]. Se for levado em conta também que, de acordo com o *site Search Engine Watch*, os principais mecanismos de busca coletam de 3 a 10 milhões de páginas ao dia, no melhor caso se levariam mais de 100 dias para atualizar totalmente o índice de um engenho que indexa toda a *Web*. Considerando que a *Web* possui um crescimento exponencial, de acordo com [28] seu tamanho dobrou em menos de dois anos, manter atualizada uma base de índices cada vez maior, é uma tarefa bastante difícil;

- Taxa de atualização das páginas *Web*: segundo [11] cerca de 23% das páginas *Web* mudam diariamente. No domínio **.com** 40% das páginas mudam diariamente, e a idade de metade das páginas é cerca de 10 dias, ou seja, em 10 dias metade dessas páginas não mais existem na *Web*. Esta grande dinamicidade da *Web* é um problema para a atualização da BI, pois quanto maior for a taxa de modificação das páginas, mais vezes elas terão que ser visitadas para manter a atualidade da BI alta. É importante ressaltar que neste trabalho, por simplificação, não foi analisado o aspecto qualitativo dessas mudanças, ou seja, não se verificou se as mudanças ocorridas na

página eram relevantes para as consultas realizadas sobre a versão precedente desta página.

2.6 Recapitulação

Neste capítulo, foram apresentados os principais módulos que compõem um engenho de busca, com ênfase no módulo responsável pelo processo de indexação, o Módulo de Indexação.

Dentro deste módulo existe o componente que implementa a política de escolha de páginas a serem inseridas e atualizadas, que é chamado de Gerenciador do Robô. Esta política pode ser de dois tipos: (1) uniforme, que atualiza todos os elementos da BI à mesma taxa; e (2) não-uniforme, que atualiza os elementos em taxas diferentes de acordo com a freqüência que esses elementos mudam. Nos próximos capítulos, estas políticas são apresentadas com maior profundidade, mostrando-se com elas são utilizadas por algumas abordagens.

Para se poderem realizar comparações entre políticas de atualização de base de índices, existem as seguintes medidas: (1) atualidade do índice: indica quanto uma BI está atualizada; (2) idade do índice: mostra quanto tempo os elementos da BI estão desatualizados; e (3) taxa de mudança: indica quanto de recurso é desperdiçado ao realizar o processo de atualização das páginas.

Entretanto, para realizar a atualização existem algumas limitações como velocidade do processo de atualização, tamanho e crescimento da *Web* e a taxa de atualização das páginas.

Em resumo, as informações apresentadas neste capítulo são importantes para uma melhor compreensão de como funciona um engenho, como ele realiza o processo de atualização da base de índices e quais as dificuldades encontradas para ele desempenhar essas tarefas.

.

Capítulo 3 Técnicas de Aprendizagem de Máquina e Estatística e Ferramenta Utilizadas

3.1 Introdução

Como foi mencionado anteriormente, este trabalho utiliza técnicas de Aprendizagem de Máquina e estatística para a criação de classificadores, que são sistemas que classificam dados de entrada em um número de possíveis categorias (maiores detalhes Seção 3.3.1). Esses classificadores ajudarão no processo de atualização de base de índices de engenhos de busca. Para a utilização dessas técnicas, é necessário, ou implementá-las, ou possuir alguma ferramenta que já as implemente, por isso, será mostrado, inicialmente, neste capítulo, as principais características da ferramenta que implementa algumas delas (Seção 3.2). Em seguida, serão apresentados alguns conceitos e tarefas de Aprendizagem de Máquina, como também, introduzidas alguns algoritmos de classificação (Seção 3.3). Posteriormente, é apresentado um estimador estatístico utilizado para classificar páginas de acordo com seu histórico de mudança (Seção 3.4) usado nesta dissertação. E, por fim, serão apresentadas algumas medidas para a avaliação dos classificadores gerados por essas técnicas (Seção 3.5).

3.2 Ferramenta WEKA

A WEKA[10] é uma ferramenta que contém uma coleção de métodos de Aprendizagem de Máquina, cuja sigla significa *Waikato Environment for Knowledge Analysis*. Ela foi desenvolvida utilizando a linguagem Java, na Universidade de *Waikato*, na Nova Zelândia.

As principais características desta ferramenta são:

- *Formato de dados único*: caso se queira executar diferentes algoritmos de aprendizagem de máquina sobre um conjunto de dados, não se precisa ter nenhum esforço adicional de implementação para isso, devido ao formato dos dados único;

- *Código fonte aberto*: WEKA tem o seu código fonte aberto, permitindo, dessa maneira, que ela possa ser estendida e suas bibliotecas possam ser acessadas por qualquer programa;

- *Enfoque prático*: por exemplo, na geração de classificadores em árvores de decisão, que é uma técnica que será apresentada posteriormente (Seção 3.3.4.1), a WEKA gera o código fonte contendo o classificador, que pode ser levado para ser utilizado por um sistema que necessite deste classificador.

3.3 Aprendizagem de Máquina

Nesta seção, inicialmente, serão expostos alguns conceitos (Seção 3.3.1) da área de Aprendizagem de Máquina. Após isso, será dada uma pequena introdução sobre as tarefas de agrupamento (Seção 3.3.2), utilizada para criação de grupos contendo elementos semelhantes, e de seleção de atributos (Seção 3.3.3), que seleciona as características, ou atributos, mais importantes a serem utilizadas na etapa de geração do

[10] Weka - http://www.cs.waikato.ac.nz/~ml

classificador. Por fim, na Seção 3.3.4, são expostos alguns algoritmos de classificação que são utilizados para a criação dos classificadores.

3.3.1 Conceitos

Os principais conceitos de Aprendizagem de Máquina utilizados neste trabalho são:

- *Inferência indutiva*: a tarefa de uma inferência indutiva pura resume-se a dada uma coleção de exemplos de aprendizagem de uma função *f* a ser aprendida (um conjunto de resultados de uma função *f*), retornar uma função *h* que se aproxima de *f*. Essa função *h* é chamada de hipótese. Então, pode-se definir aprendizagem indutiva como o aprendizado de uma função a partir de exemplos de seus resultados [38]. É essa função h que irá classificar os novos exemplos e, portanto, o classificador será formado por ela.

- *Treinamento e teste*: o processo de inferência indutiva pode ser realizado da seguinte maneira: os dados são separados em duas partes, um conjunto de treinamento e um de teste. O algoritmo de classificação é executado sobre o conjunto de treinamento, construindo assim o classificador. Este classificador é utilizado para classificar as instâncias do conjunto de teste. A fração das instâncias do conjunto de teste corretamente classificadas sobre o total é a **taxa de acerto** sobre o conjunto de teste deste classificador. Na Seção 3.5, são dados mais detalhes sobre essa medida. O **erro de teste** é 1 subtraído da taxa de acerto e o **erro de re-substituição** 1 subtraído a taxa de acerto sobre o conjunto de treinamento;

- *Tempo de classificação*: o tempo dispendido pelo classificador para classificar uma nova instância é o tempo de classificação. Neste trabalho, será chamado **tempo para teste**, a soma dos tempos de classificação para todos os elementos do conjunto de teste. A partir desse tempo para teste, pode-se comparar o tempo de classificação das técnicas de Aprendizagem de

Máquina, dado que o conjunto de teste seja o mesmo e possua o mesmo número de elementos;

- *Overfitting*: é o fenômeno que acontece quando o classificador gerado se adapta muito bem ao conjunto de treinamento, ou seja, certas particularidades específicas do conjunto de treinamento são incluídas no classificador. Conseqüentemente, quando essas particularidades não refletem o comportamento geral dos dados, a taxa de acerto no conjunto de teste se torna baixa;

- *Validação cruzada*: é um método padrão para avaliar a taxa de acerto de uma técnica de aprendizagem em um dado conjunto de dados. Ele funciona da seguinte maneira: o conjunto de treinamento é dividido aleatoriamente em k amostras, o classificador começa a ser criado a partir do conjunto de treinamento sem uma das amostras **i**. É determinada, então, a taxa de acerto do classificador construído sobre essa amostra **i**. Esse procedimento se repete k vezes até que, no fim, cada amostra tenha sido utilizada exatamente uma vez para teste. A taxa de acerto sobre o conjunto de teste é calculada como a média das taxas de acerto sobre todas as amostras. Neste trabalho, vai-se utilizar esse método com o número de amostras igual a 10, que o estudo [43] cita como sendo um valor adequado;

- Aprendizagem supervisionada e não supervisionada: no aprendizado supervisionado, dado um conjunto de exemplos pré-classificados, aprende-se uma descrição geral que encapsula a informação contida nesses exemplos e que pode ser usada para prever casos futuros. No aprendizado não supervisionado, dada uma coleção de dados não classificados, eles são agrupados por regularidades.

3.3.2 Agrupamento

Agrupamento é uma tarefa que agrupa os dados em classes tal que os objetos dentro de cada grupo têm alta similaridade em comparação com os membros de seu grupo e alta dissimilaridade com os elementos de outros grupos. Em aprendizado de máquina, agrupamento é um exemplo de aprendizado não supervisionado, pois a sua entrada não é um conjunto de grupos pré-definidos, ao contrário, ele tenta encontrar grupos dentro dos dados. De forma geral, essa tarefa pode ser utilizada para:

- Descobrir a distribuição geral de regularidades nos dados: para a identificação, por exemplo, de regiões densas e esparsas, descobrindo-se assim a distribuição geral dos padrões e correlações interessantes entre os atributos dos dados;
- Passo de pré-processamento: pode ser utilizado como um passo de pré-processamento para outros algoritmos de Aprendizagem de Máquina operarem nas classes detectadas pelo *clustering*. É desta maneira que ele será utilizado neste trabalho.

Uma tarefa típica de *clustering* envolve as seguintes etapas:

1. Representação do padrão: refere-se ao número de classes, ao número de padrões disponíveis e o número, tipo e escala das características disponíveis para a tarefa de agrupamento. Opcionalmente, pode-se utilizar tarefas de seleção de atributos (ver Seção 3.3.3) para se obter um conjunto mais apropriado de atributos;
2. Métrica de proximidade: medida por uma função de distância definida por pares de padrões. Uma variedade de métricas de medida estão em uso em várias comunidades. Uma distância simples, como a Euclideana pode ser usada para refletir dissimilaridade entre dois padrões, enquanto outras medidas de similaridade podem ser usadas para caracterizar a similaridade conceitual entre dois padrões;
3. Agrupamento: os principais agrupamentos são o hierárquico e o de partição. Os algoritmos de agrupamento hierárquico produzem uma série de partições

aninhadas baseado em um critério para agrupar ou dividir grupos baseado na similaridade. O agrupamento de partição identifica a particionado os dados em grupos separados um dos outros utilizando as métricas de proximidade;

4. Abstração dos dados: é a etapa na qual se extrai uma simples e compacta representação dos dados, ou seja, qual o significado de cada grupo gerado pela etapa de agrupamento;
5. Análise da validade dos grupos gerados: a partir de critérios definidos avaliam-se os grupos gerados. Um exemplo de critério seria: os grupos gerados só seriam válidos se todos eles possuíssem uma proporção significativa de elementos em relação ao total de elementos.

Neste trabalho, vai-se utilizar a tarefa agrupamento para agrupar as páginas com taxa de modificação semelhante, definindo desta maneira grupos ou classes de página. Para isso, poder-se-iam ter utilizado classes pré-definidas, mas isso talvez não representasse uma divisão natural dos dados de acordo com o seu comportamento. Portanto, para este fim, foi utilizada a técnica de agrupamento para definir as classes de forma mais natural.

De acordo com o problema que se está tratando, escolhe-se qual desses métodos de agrupamento utilizar. Para este trabalho em particular, como se deseja criar conjuntos disjuntos de páginas, utilizou-se o método de partição, através de algoritmo *k-means* [42], por ele ser um dos métodos de particionamento mais conhecidos, por ele estar implementado na ferramenta que está sendo utilizada neste trabalho, a WEKA e por causa da sua simplicidade a sua execução é bastante rápida. Existem outros algoritmos implementados pelo WEKA para realizar o agrupamento de partição: *Cobweb* e *EM*. O *Cobweb* não foi utilizado por tratar atributos nominais e o *EM* devido à sua execução ser mais demorada do que a do *k-means*.

3.3.3 Seleção de Atributos

Teoricamente, quanto mais atributos ou características existirem nas instâncias a serem utilizadas para a construção de um classificador, mais preciso ele será. Entretanto, a

experiência prática com os algoritmos de aprendizado de máquina tem mostrado que isso nem sempre acontece.

A seleção de atributos é o processo de identificar e remover características tanto irrelevantes como redundantes. Isto reduz a dimensionalidade dos dados, permitindo que os algoritmos de classificação sejam executados de forma mais rápida, e evita a ocorrência de *overfitting*.

As técnicas de seleção de atributos são classificadas de acordo com a natureza da métrica utilizada para avaliar a relevância de um atributo. Dessa maneira, dividem-se essas técnicas da seguinte forma [43]:

- *Wrapper*: avalia os atributos através do uso da taxa de acerto provido por um algoritmo de aprendizagem. Para isso, ele utiliza esse algoritmo sobre cada subconjunto dos atributos e escolhe aquele sub-conjunto com maior taxa de acerto;
- *Filter*: observa as características estatísticas dos dados para avaliar os atributos e executa independentemente de um algoritmo de aprendizagem. Um método simples de *filter* é selecionar somente os atributos que dividam as instâncias do conjunto de treinamento de uma forma que separa todas elas, ou seja, se tiver um atributo, por exemplo, que seja igual para todas as instâncias ele deve ser retirado.

A técnica de *Wrapper* geralmente dá melhores resultados que a de *Filter*, mas em contrapartida o custo computacional da primeira técnica é muito mais alto, pois, para realizar a seleção dos atributos, a técnica de *Wrapper* executa o algoritmo de aprendizagem para cada subconjunto de atributos considerados durante a busca do melhor conjunto de atributos [21].

Como se deseja selecionar um subconjunto dos atributos, uma busca exaustiva no espaço de subconjuntos de atributos pode ser proibitiva, portanto deve-se ter alguma heurística para a busca nesse espaço, são elas:

- Eliminação *backward*, na qual inicialmente são utilizados todos os atributos e vão se retirando os atributos até que nenhuma das alternativas melhore a

precisão, ou seja, não aumente a taxa de acerto em relação ao melhor sub-conjunto;

- A seleção *forward* de forma contrária, parte de um conjunto com 1 atributo e vai adicionando atributos.

Segundo [43], em termos gerais, a eliminação *backward* produz maiores conjuntos de atributos e melhor precisão na classificação do que a seleção *forward*. A razão para isto é que a taxa de acerto do classificador é apenas uma estimativa. Assim, uma estimativa otimista do classificador sobre um sub-conjunto de atributos poderá causar uma parada prematura. Na eliminação *backward*, isso acarretará na seleção de um sub-conjunto com muitos atributos e na seleção *forward* com atributos não suficientes.

Neste trabalho, o número de características envolvidas para a geração do classificador não é grande. Entretanto, além dos objetivos citados anteriormente para a seleção de atributos, deseja-se verificar quais dos atributos são realmente relevantes para a classificação, dado que eles foram escolhidos baseados em algumas citações literárias ou suposições.

Como o número de atributos a ser considerado não é muito grande e como, geralmente, a técnica de *Wrapper* com eliminação *backward* consegue melhores resultados, resolveu-se utilizá-la para a seleção dos atributos.

3.3.4 Algoritmos de Classificação

Algoritmos de classificação são exemplos de aprendizado supervisionado. Até onde se sabe, não há na literatura nenhum trabalho na área de atualização de base de índices de engenhos de busca que utilizam algoritmos de classificação. Além disso, existe uma série de algoritmos de classificação que poderiam ser utilizados. Na ferramenta WEKA existem os seguintes algoritmos de classificação [43]: ZeroR[11], OneR[12], Bayesiano

[11] O algoritmo ZeroR classifica um exemplo na classe que for maioria no conjunto de treinamento se a classe for categórica e na classe média, se ela for numérica.

Ingênuo, DecisionTable, K-vizinhos mais Próximos, Árvore de decisão (J48), Support Vector Machine e Redes Neurais. Os algoritmos ZeroR, OneR e DecisionTable foram desconsiderados por ser serem bastante rudimentares [43].

Houve a tentativa de se utilizar as técnicas de Support Vector Machine e Redes Neurais, que são algoritmos bastante populares. O problema na execução dessas duas técnicas, sobre os dados do experimento que será apresentado na Seção 5.5, era que após um período bastante longo os processos eram abortados e não se conseguiu obter nenhum resultado.

Por fim, foram escolhidas as técnicas de Árvore de Decisão, Bayesiano Ingênuo e o K-vizinhos mais Próximos para serem utilizadas nos experimentos relacionados à construção de classificadores. As razões para a escolha delas foi: bastante utilizadas na prática, facilmente configuráveis e suas execuções terem sido realizadas com sucesso sobre o conjunto de dados. Essas técnicas serão melhor explicadas nas próximas seções.

Para facilitar a compreensão dos algoritmo e poder diferenciar o método utilizado por cada um, vão-se utilizar exemplos, que possuem como conjunto de treinamento as 14 instâncias apresentada na Tabela 1. Esse conjunto de treinamento será utilizado para a construção de um classificador indicando se é provável que um consumidor compre um determinado produto ou não, baseado em algumas de suas características.

Tabela 1 - Exemplo de conjunto de treinamento

idade	renda	estudante	avaliação_credito	classe: comprar_computador
<=30	alta	nao	regular	nao
<=30	alta	nao	excelente	nao
31..40	alta	nao	regular	sim
>40	media	nao	regular	sim
>40	baixa	sim	regular	sim
>40	baixa	sim	excelente	nao
31..40	baixa	sim	excelente	sim
<=30	media	nao	regular	nao
<=30	baixa	sim	regular	sim
>40	media	sim	regular	sim

[12] O algoritmo OneR gera uma árvore de decisão de um nível, que é expresso na forma de um conjunto de regras baseado em apenas um único atributo.

<=30	media	sim	excelente	sim
31..40	media	nao	excelente	sim
31..40	alta	sim	regular	sim
>40	media	nao	excelente	nao

3.3.4.1 Árvore de Decisão

Uma árvore de indução [34] é um gráfico de fluxo numa estrutura de árvore, no qual cada nó interno denota um teste em um atributo, cada ramo representa um resultado do teste e os nós-folha representam as classes. Para classificar uma nova observação, os valores dos atributos são testados contra a árvore de indução. Um caminho é traçado da raiz até uma folha que identifica em qual classe se encontra a amostra. Um exemplo de uma árvore é mostrado na Figura 4, ele é resultado a execução do algoritmo de árvore de decisão sobre o conjunto de treinamento apresentado na Tabela 1.

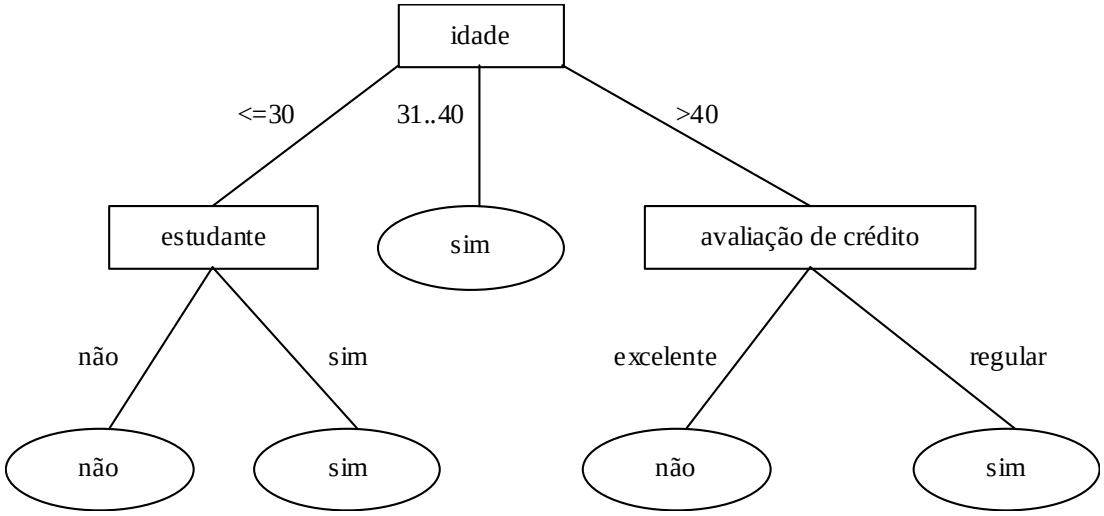


Figura 4 - Árvore de decisão indicando se é provável que um consumidor compre um determinado produto ou não.

Então, baseado nesta árvore um novo exemplo X, cujos atributos são: idade = "<=30", renda = "media", estudante="sim", avaliacao_credito="regular" é classificado como comprar_computador = "sim".

O algoritmo básico para a construção de uma árvore de decisão é o ID3 [34]. Ele pode ser resumido da seguinte forma:

1. A árvore inicia com o atributo que sozinho melhor classifica os dados do conjunto de treinamento, segundo um teste estatístico;

2. São criados ramos para cada possível valor desse atributo e as dados do conjunto de treinamento são separados por esses ramos;

O processo então é repetido usando o conjunto de treinamento associado com cada ramo para selecionar o melhor atributo para testar neste ponto da árvore;

Esse processo é parado quando: (i) todas os exemplos para um dado nó pertença à mesma classe; e (ii) não há mais atributos para particionar as amostras.

Para definir o melhor atributo a ser utilizado como nó, usa-se uma medida estatística chamada de ganho de informação, que mede quão bem um dado atributo separa os exemplos de treinamento de acordo com suas classes. Mais detalhes sobre essa medida ver em [22].

Então, para a construção da árvore mostrada na Figura 4, a partir do conjunto de treinamento da Tabela 1, inicialemente, o ganho de informação do atributo idade foi o maior e por isso ele foi colocado como nó-raiz (passo 1). É realizado, então, o passo 2, e sobre os subcojuntos das amostras, é verificado qual o melhor atributo, excetuando-se o atributo idade.

Um conceito importante dentro de árvore de decisão é o de poda (*pruning*, em inglês). Quando uma árvore de decisão é construída, muitos dos ramos poderão refletir anomalias existentes nos dados de treinamento e isso pode acarretar a existência de *overfitting* (3.3.1). Os métodos de poda tentam tratar este problema. Tais métodos usam medidas estatísticas para remover os ramos menos confiáveis, resultando numa classificação mais rápida e numa melhora da precisão na fase de teste [22].

Há duas abordagens para a poda: (1) *prepruning*: o processo de construção da árvore é parado prematuramente; e (2) *postpruning*: remove ramos de uma árvore totalmente construída, sendo que esta nova árvore apresenta melhores resultados de classificação do que a original.

A versão do algoritmo de árvore de decisão que implementa esse conceito de poda é o C4.5 [33]. Neste trabalho, utilizou-se uma extensão deste algoritmo, o J48, que é implementado pela ferramenta WEKA. O algoritmo C4.5 suporta também atributos numéricos e, como será mostrado, é esse o tipo dos atributos das instâncias utilizadas neste trabalho.

3.3.4.2 Bayesiano Ingênuo

O algoritmo bayesiano ingênuo [27] é baseado no teorema de *Bayes*. De forma genérica, esse algoritmo, baseado no conjunto de treinamento, calcula a probabilidade de uma dada instância pertencer a uma determinada classe, sendo que a classe com maior probabilidade é aquela a qual a instância é atribuída. O método bayesiano ingênuo assume que o efeito do valor de um atributo a uma dada classe é independente dos valores de outros atributos, ou seja, para ele todos os atributos são independentes entre si e igualmente relevantes. Esta suposição chama-se independência condicional, e é feita para simplificar o modelo, por isso o classificador é chamado de ingênuo [22].

Então, esse classificador funciona da seguinte forma:

1. Cada exemplo é apresentado como um vetor de características n-dimensional, $X = (x_1,x_2,...,x_n)$, ilustrando medidas feitas sobre os n atributos, respectivamente $A_1, A_2, ..., A_n$.
2. Suponha que existam m classes, $C_1,C_2,...,C_m$. Dado um exemplo não conhecido X, o classificador irá atribuir X a uma determinada classe, a que possuir a maior probabilidade condicionada a X. Em outras palavras, o classificador bayesiano ingênuo atribui um exemplo desconhecido X a uma classe C_i se e somente se

$P(C_i|X) > P(C_j|X)$ para $1 \leq j \leq m, j \neq i$

Então, maximiza-se $P(C_i|X)$. Pelo teorema de Bayes,

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}$$

Equação 2 - Cálculo da probabilidade de uma classe C_i condicionado a um exemplo X.

3. Como P(X) é constante para todas as classes, somente $P(X|C_i)P(C_i)$ precisa ser maximizado. Note que a probabilidade da classe anterior pode ser estimada por $P(C_i) = s_i /s$, onde s_i é o número de exemplos de treinamento da classe C_i, e s é o número total de exemplos de treinamento;
4. Como P(X) é constante para todas as classes, somente $P(X|C_i)P(C_i)$ precisa ser maximizado. Note que a probabilidade da classe anterior pode ser

estimada por $P(C_i) = s_i / s$, onde s_i é o número de exemplos de treinamento da classe C_i, e s é o número total de exemplos de treinamento;

5. Com conjunto de dados com muitos atributos, seria computacionalmente custoso calcular $P(X|C_i)$. Para reduzir essa computação, a suposiução ingênua da independência condicional é feita. Assume-se que os atributos são condicionalmente independentes um dos outros, dado o valor de uma classe do conjunto de entrada. Então,

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

Equação 3 - Cálculo da probabilidade de um exemplo X condicionado a uma classe Ci.

As probabilidades $P(x_1|C_i)$, $P(x_2|Ci)$,...,$P(x_n|C_i)$ podem ser estimadas a partir do conjunto de treinamento, como será mostrado posteriormente;

6. Para classificar um exemplo não conhecido X, $P(X|C_i)P(C_i)$ é avaliado para cada classe C_i. O exemplo X é então atribuído à classe C_i, para a qual $P(X|C_i)P(C_i)$ é máximo.

Utilizando como exemplo o conjunto de treinamento apresentado na Tabela 1. Vai-se mostrar como é atribuída uma classe a um novo exemplo X, cujos atributos são: idade = "<=30", renda = "media", estudante="sim", avaliacao_credito="regular".

Precisa-se maximizar $P(X|C_i)P(C_i)$, para i = 1,2. $P(C_i)$, a probabilidade para cada classe (as classes são comprar_computador="sim" ou comprar_computador="nao"). Inicialemente, calculam-se os valores de $P(C_1)$ e $P(C_2)$ baseado no conjunto de treinamento:

P(comprar_computador="sim") = 9/14 = 0,643

P(comprar_computador="nao") = 5/14 = 0.357

Para computar $P(X|C_i)$, para i = 1,2, computam-se as seguintes probabilidades condicionais:

P(idade="<30" | comprar_computador="sim") = 2/9 = 0.222

P(idade="<30"| comprar_computador="nao") = 3/5 = 0.6

P(renda="media"| comprar_computador="sim") = 4/9 = 0.444

P(renda="media"| comprar_computador="nao") = 2/5 = 0.4

P(estudante="sim"| comprar_computador="sim") = 6/9 = 0.667

P(estudante="sim"| comprar_computador="nao") = 1/5 = 0.2

P(avaliacao_credito="regular"| comprar_computador="sim") = 6/9 = 0.667

P(avaliacao_credito="regular"| comprar_computador="nao") = 2/5 = 0.4

Usando as probabilidades acima, obtêm-se:

P(X| comprar_computador="sim") = 0.222 X 0.444 X 0.667 X 0.667 = 0.044

P(X| comprar_computador="nao") = 0.6 X 0.4 X 0.2 X 0.4 = 0.019

P(X| comprar_computador="sim")P(comprar_computador="sim") = 0.044 X 0.643 = 0.028

P(X| comprar_computador="nao")P(comprar_computador="nao") = 0.019 X 0.357 = 0.007

Portanto, o classificador bayesiano atribui a classe comprar_computador="sim" para o exemplo X.

3.3.4.3 K-vizinhos mais Próximos

A técnica k-vizinhos mais próximos é baseada no aprendizado por analogia [22]. As instâncias de treinamento são descritas por atributos numéricos n-dimensionais. Cada instância representa um ponto num espaço n-dimensional, no qual cada dimensão é um atributo. Dessa maneira, todas as instâncias do conjunto de treinamento são plotadas neste espaço. Quando um novo exemplo é apresentado, o classificador procura neste espaço os k exemplos mais próximos do apresentado. O critério de proximidade entre os exemplos é definida pela distância euclidiana. Este novo exemplo é classificado para a classe mais comum dentre os k vizinhos mais próximos dele. Quando k=1, o novo exemplo é atribuído à classe dos exemplos do conjunto de treinamento mais próximo.

Uma restrição para o uso deste tipo de classificador é o tempo para classificar novos exemplos, dado que toda a computação é realizada neste momento. Isso pode ser uma restrição bastante forte para o uso dessa técnica, pois o classificador é executado em um sistema *on-line*. Outra restrição é que ele assume que todos os atributos possuem a mesma relevância assim como o bayesiano ingênuo [43].

Para a utilização dessa técnica com o conjunto de treinamento apresentado na Tabela 1, é necessário transformar os atributos nominais em numéricos. Assim, uma transformação simples seria: idade 1,2,3 para "<=30","31..40",">40"; renda 1,2,3 para "baixa","media","alta"; estudante 1,2 para "nao","sim"; avaliacao_credito 0,1 para "regular", "excelente"; e comprar_computador 0,1 para "nao","sim".

Então, para o novo exemplo X, cujos atributos são: idade = 1, renda = 2, estudante = 2, avaliacao_credito = 0. Calculando-se a distância desse exemplo em relação aos exemplos do conjunto de treinamento, para k = 1, o exemplo é classificado como comprar_computador="sim", pois o exemplo mais próximo de X é idade = 1, renda = 2, estudante = 2, avaliacao_credito = 1.

3.4 Estimador Estatístico

As técnicas mencionadas anteriormente neste capítulo dizem respeito à área de Aprendizagem de Máquina que utilizam as características das instâncias para poder atribuí-las a uma determinada classe.

Entretanto, para o problema específico deste trabalho, atualização de páginas *Web*, pode-se aprender como uma página se comporta e, a partir daí, inferir a que classe de atualização ela pertence. O estimador bayesiano [13] é um dos métodos que pode ser utilizado para este fim. Ele categoriza as páginas baseado no método de inferência bayesiano. De forma genérica, este método assume que as páginas possuem certa probabilidade de pertencerem a um número definido de classes, sendo a classe com maior probabilidade aquela a qual a página pertence. À medida que se vai visitando a página, essas probabilidades vão sendo modificadas. Se, por exemplo, para o caso onde as classes representam grupos de alteração de páginas, ela se modifica na freqüência de sua classe, a probabilidade dela pertencer à classe aumenta, caso contrário, diminui para esta classe e aumenta para aquela com a freqüência que mais de adequa a ela.

Para melhor ilustrar o seu funcionamento, é apresentado o seguinte cenário, retirado de [13]. Quer-se classificar uma página p_1 em duas classes, em páginas que mudam toda semana (C_W) e em páginas que mudam todo mês (C_M). Inicialmente, por

não se conhecer o comportamento de p_1, assume-se que a probabilidade dela pertencer a cada classe é igual a 50%. Após 5 dias, essa página é acessada e percebe-se que ela foi modificada (ocorreu o evento E). É preciso, então, alterar as probabilidades dela pertencer a cada classe. Como ela mudou em menos de uma semana, a probabilidade de pertencer à classe C_W, vai tornar-se maior do que C_M. Caso contrário, se ela não tiver sido modificada, ocorre o inverso, a probabilidade de C_M aumenta e a de C_W diminui.

Para o cálculo dessas probabilidades é utilizado o teorema de Bayes que, para a classe C_W deste exemplo está representado na Equação 4. Uma equação similar a esta pode ser construída para a classe C_M.

$$P\{p_1 \in C_W) \mid E\} = \frac{P\{(p_1 \in C_W) \wedge E\}}{P\{E\}} = \frac{P\{(p_1 \in C_W) \wedge E\}}{P\{E \wedge (p_1 \in C_W)\} + P\{E \wedge (p_1 \in C_M)\}}$$
$$= \frac{P\{E \mid (p_1 \in C_W)\} P\{p_1 \in C_W\}}{P\{E \mid (p_1 \in C_W)\} P\{p_1 \in C_W\} + P\{E \mid (p_1 \in C_M)\} P\{p_1 \in C_M\}}$$

Equação 4 - Fórmula para o cálculo da probabilidade condicional para a classe de páginas que mudam toda semana.

A probabilidade[13] de uma página ser alterada após um tempo t dado que ela pertença a uma determinada classe C_N com uma taxa de alteração igual a j (para C_W, j = 7 dias e para C_M, j = 30 dias) está apresentada na Equação 5:

$$P\{E \mid (p_1 \in C_N)\} = 1 - e^{-t/j}$$

Equação 5 - Probabilidade de uma página, pertencente à classe C_N, ser alterada após um tempo t.

No exemplo anterior, a página foi modificada após 5 dias, t = 5, então, a partir da Equação 5, são calculadas as probabilidades mostradas abaixo:

$P\{E \mid (p_1 \in C_W)\} = 1 - e^{-5/7}$;

$P\{E \mid (p_1 \in C_M)\} = 1 - e^{-5/30}$;

$P\{p_1 \in C_W\} = P\{p_1 \in C_M\} = 0.5$;

A Equação 6 calcula a probabilidade de p_1 pertencer a C_W dado a condição dela ter sido modificada no período menor ou igual a 1 semana (7 dias) e a Equação 7 a

[13] Em [13], não são apresentados detalhes sobre qual modelo a Equação 5 é proveniente.

probabilidade de p_1 pertencer a C_M dado a condição dela ter sido modificada no período menor ou igual a 1 mês (30 dias). Utilizando estas equações foram obtidos os seguintes resultados:

$$P\{p_1 \in C_W \mid E\} = \frac{(1-e^{-5/7})0{,}5}{(1-e^{-5/7})0{,}5 + (1-e^{-5/30})0.5} \approx 0{,}77$$

Equação 6 - Cálculo para encontrar o valor da probabilidade da página pertencer à classe Cw.

$$P\{p_1 \in C_M \mid E\} = \frac{(1-e^{-5/30})0{,}5}{(1-e^{-5/7})0{,}5 + (1-e^{-5/30})0.5} \approx 0{,}23$$

Equação 7 - Cálculo para encontrar o valor da probabilidade da página pertencer à classe Cm.

Isto é, p_1 pertence à classe C_W com probabilidade de 0,77 e p_1 pertence a C_M com probabilidade de 0,23. Como era esperado, a probabilidade de p_1 pertencer a C_W tornou-se maior do que C_M.

Uma informação final sobre esse método é que ele não foi considerado como bayesiano ingênuo, pois para [22], o classificador bayesiano precisa de um conjunto de dados de entrada com n características para, a partir desse conjunto de entrada, serem calculadas as probabilidades condicionais. Neste método aqui descrito, essas probabilidades são calculadas de forma diferente, embora também seja utilizado o teorema de *Bayes*.

3.5 Medidas para a Avaliação de Classificadores

Neste trabalho, vai-se realizar a criação de alguns classificadores. Para poder avaliá-los e compará-los, faz-se necessário a utilização de algumas medidas. Durante esta seção serão apresentados algumas delas.

Uma ferramenta útil para analisar o resultado de classificadores é a **matriz de confusão** [24], que é uma matriz que contém informação sobre a classificação real dos elementos e a realizada pelo classificador. O número de elementos classificados

corretamente está através da diagonal principal. A Tabela 2 mostra um exemplo de uma matriz de confusão para duas classes.

Tabela 2 - Exemplo de matriz de confusão para duas classes.

		classificado	
		positivo	negativo
verdadeiro	positivo	verdadeiro positivo	falso negativo
	negativo	falso positivo	verdadeiro negativo

Os componentes dessa matriz são:

- Verdadeiros positivos (VP) são os elementos que foram classificados como positivos e são realmente positivos;
- Falsos negativos (FN) são aqueles classificados como negativos mas são positivos;
- Falsos positivos (FP) são os que foram classificados positivos mas são negativos;
- Verdadeiros negativos (VN) são os elementos classificados como negativos e realmente são negativos.

A partir dos dados da matriz de confusão, pode-se calcular uma série de valores, são eles:

- Taxa de acerto (*accuracy*) [24]: reflete a corretude geral do classificador. É a proporção dos elementos classificados corretamente sobre o total. É determinado usando a Equação 8:

$$Acerto = \frac{VP + VN}{VP + FP + VN + FN}$$

Equação 8 - Fórmula para o cálculo do acerto.

- Sensibilidade ou cobertura [35]: indica a proporção de instâncias positivas classificadas corretamente A Equação 9 mostra como calcular a sensibilidade e especificidade, respectivamente.

$$Sensibilidade = \frac{VP}{VP + FN}$$

Equação 9 - Fórmula para o cálculo da sensibilidade.

- Especificidade [24]: mostra a proporção de instâncias negativas classificadas corretamente. A maneira de calculá-la está apresentado na Equação 10.

$$Especificidade = \frac{VN}{VN + FP}$$

Equação 10 - Fórmula para o cálculo da especificidade.

- Valor de predição positivo ou precisão [35]: indica a proporção de elementos classificados como positivos que são realmente positivos. A Equação 11 demonstra como calcular esse valor.

$$\Pr ecisão = \frac{VP}{VP + FP}$$

Equação 11 - Fórmula para o cálculo da precisão.

- Valor de predição negativo [24]: indica a proporção de elementos classificados como negativos que são realmente negativos. Esse valor é calculado utilizando a Equação 12.

$$Valor\_de\_predição\_negativo = \frac{VN}{VN + FN}$$

Equação 12 - Fórmula para o cálculo do valor de predição negativo.

- *F-measuse* [35]: relaciona a cobertura com a precisão em uma única métrica, calculando sua média harmônica. Para calculá-la utiliza-se a Equação 13.

$$F - Measure = \frac{2VP}{2VP + FP + FN}$$

Equação 13 - Fórmula para o cálculo do *F-measure*.

Portanto, através dessas medidas apresentadas pode-se realizar a comparação entre classificadores, no qual o melhor classificador é aquele que possui os valores dessas medidas mais próximos de 1.

3.6 Recapitulação

O principal objetivo deste capítulo foi apresentar uma visão geral das técnicas de Aprendizagem de Máquina e estatística utilizadas nesta dissertação e da ferramenta WEKA. Para isso, inicialmente foi dada uma introdução de alguns conceitos de Aprendizagem de Máquina importantes. Posteriormente, foram introduzidas as principais técnicas de Aprendizagem de Máquina necessárias para a construção da proposta de solução, que utilizam o conceito de inferência indutiva. Foi dada também uma apresentação sobre o estimador bayesiano que classifica elementos baseado no seu comportamento através do tempo. Por fim, para que os classificadores gerados por essas técnicas pudessem ser comparados, mostraram-se as principais medidas para realizar esse tipo de comparação.

Capítulo 4 Propostas para Atualização da Base de Índices

4.1 Introdução

Existe pouca informação sobre como o problema de atualização de base de índices é tratado pelos engenhos de busca comerciais, devido à competitividade que existe no mercado. O que há são trabalhos publicados por centros de pesquisa e/ou universidades.

Baseado nesses estudos, serão apresentadas, neste capítulo, as principais abordagens propostas para o processo de atualização. As primeiras abordagens a serem discutidas são as que utilizam a política uniforme, também chamada de tradicional (Seção 4.2). Na Seção 4.3, são apresentadas abordagens que tentam atacar os problemas encontrados na tradicional. Por fim, na Seção 4.4, são feitas as considerações finais sobre o capítulo.

4.2 Abordagem Tradicional

Como mencionado na Seção 2.5.1, a abordagem mais utilizada é a política uniforme de atualização. Apesar de sua simplicidade de implementação, esta abordagem apresenta algumas deficiências, são elas:

- Escalabilidade: um dos objetivos de um engenho de busca é ter uma boa cobertura de páginas em sua BI. A conseqüência disso é que, dependendo do

escopo adotado, haverá um número muito grande de páginas para ele atualizar. Para um escopo que engloba toda a *Web*, com mais de um bilhão de páginas e crescimento exponencial, levar-se-á muito tempo para poder atualizar todo este conteúdo, prejudicando, assim, a atualidade e a idade do índice.

- Desperdício de recursos: na política uniforme visitam-se todas as páginas da BI à mesma taxa. Entretanto, se estas páginas são modificadas em diferentes freqüências, esta abordagem não se mostra eficiente, pois muitas das páginas utilizadas na revisitação não haviam sido modificadas, causando o desperdício de recursos. Segundo [11], é justamente este o comportamento das páginas *Web*, ou seja, são modificadas em diferentes taxas.

Alguns trabalhos propõem o uso da política uniforme para a atualização da BI. Em [12], é estudado como atualizar a base de índices para melhorar a sua atualidade. Para isto, é apresentado um modelo formal para o problema de atualização. Baseado neste modelo, são estudadas várias políticas de atualização e propõe-se que a política ótima para o engenho de busca é a uniforme. Nesse trabalho, é considerado somente o quanto a base está atualizada, não levando em conta o desperdício de recursos que ocorre com a utilização desta política.

Em [8], estuda-se com que velocidade um engenho de busca deve reindexar a *Web* para permanecer atualizado, baseado na distribuição das alterações das páginas. Para isto, foi feito um estudo a fim de conhecer a distribuição das mudanças de um documento na *Web*. Foram observadas 100.000 páginas ao dia, por mais de 7 meses, indicando como e quando elas foram modificadas. Os dados indicaram que o tempo entre modificações de uma página *Web* típica pode ser modelado por uma distribuição de *Poisson* [40], que é parametrizada pela taxa de mudanças da página.

A partir desta constatação, criou-se um modelo para permitir estimar taxas de reindexação para manter a base de índices de um mecanismo de busca atualizada utilizando uma política uniforme. Este modelo se baseia na probabilidade que uma página escolhida aleatoriamente no índice esteja atualizada por um dado período.

Usando este modelo, para que 95% das páginas pertencentes à BI de um engenho de busca, que possui um escopo de cerca de 800 milhões de documentos, permaneçam atualizadas no período de 1 semana, com o uso da política uniforme, é necessário que o engenho vis*ite* pelo menos 45 milhões de páginas ao dia, o que requer uma largura de banda de 50 *megabits*/segundo[14]. Como essa abordagem utiliza a política uniforme ocorre o mesmo problema de desperdício de recursos.

4.3 Abordagens não Tradicionais

As abordagens não tradicionais tentam atacar os problemas encontrados na abordagem tradicional apresentada na Seção 4.2. Para isso, essas abordagens identificam com que freqüência os elementos da base de índices são modificados, para que estes possam ser atualizados de uma forma mais eficiente do que a realizada pela abordagem tradicional.

Dentre as abordagens não tradicionais, existem aquelas que o engenho de busca obtém a informação de quais elementos forma modificados, de entidades externas como servidores *proxy* ou produtores de documentos (servidores *Web*). Neste trabalho, esta é chamada de **abordagem passiva**, pois a identificação dos elementos modificados não é feita pelo engenho de busca. Já na **abordagem ativa**, o engenho não recebe nenhum auxílio de entidades externas para a atualização do índice. Ele tenta prever o comportamento das páginas *Web* em relação à modificação de seu conteúdo, a fim de melhorar a atualidade da base.

4.3.1 Abordagem Passiva

Nesta abordagem, as entidades externas auxiliam o engenho de busca na identificação dos elementos modificado ([6], [7], [20], [39]). Isto é realizado de duas maneiras:

- Envio das páginas modificadas: o produtor dos documentos ou alguma entidade externa como servidores *proxy*, envia as páginas modificadas para os mecanismos de busca.

[14] Para este cálculo, utilizou-se o tamanho médio de uma página igual a 12 Kbytes.

- Arquivo contendo a informação das páginas alteradas: em um local previamente determinado no servidor *Web*, é disponibilizada a lista de quais documentos foram alterados e a data da alteração de cada um deles, para que o engenho, de posse destas informações, possa recuperar somente as páginas modificadas.

Para cada uma delas serão apresentados mais detalhes de seu funcionamento.

4.3.1.1 Envio das Páginas Alteradas

O sistema Harvest [6] usa uma arquitetura distribuída para obter e distribuir dados (Figura 5), que possui os seguintes componentes:

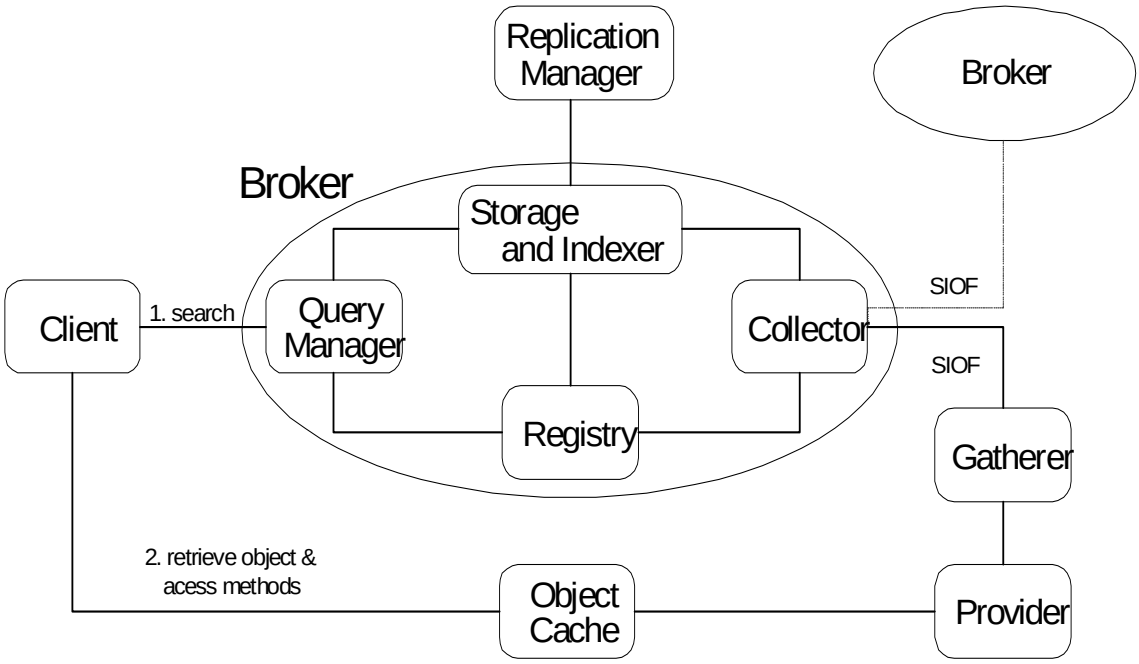


Figura 5 - Arquitetura do Harvest.

- *Provider*: componente residente no servidor *Web* do *site* que varre os documentos deste *site*, periodicamente, e disponibiliza todo o conteúdo destes documentos através de uma *cache*, colocando-os em um único arquivo. Estas informações podem ser recuperadas de uma só vez, ao invés de página por página, como é feito na forma tradicional pelos engenhos de busca, diminuindo, substancialmente, o tráfego de rede;

- Gatherer: sistema que coleta informações do provider. Extrai informações relevantes dos documentos, como resumo, título etc, antes de passá-lo para um Indexer remoto. Além disso, permite ao Broker recuperar somente aqueles documentos que tenha mudado desde um tempo especificado e transmite a resposta de forma compactada;

- *Broker*: provê uma interface de consulta sobre a informação indexada pelos *gatherers*. Ele recolhe informação de um ou mais *gatherers* ou outros *brokers*, e atualiza seus índices incrementalmente. O *broker* é o componente de distribuição da arquitetura, responsável pela distribuição da informação contida no *gatherer* e também é capaz de se comunicar com outros *brokers*, possibilitando a construção de uma complexa árvore hierárquica de sistemas de indexação. Ele é composto pelos componentes: *Registry* e *Storage Manager*, *Query Manager* e *Collector*;

- *Registry* e *Storage Manager*: mantêm uma lista dos documentos existententes no Broker. Para cada objeto, o Registry guarda um identificador único e um tempo de vida. O Storage Manager armazena uma coleção de documentos no disco, sendo cada documento um arquivo no sistema de arquivos local;

- *Collector*: periodicamente requisita atualizações de cada Gatherer em um determinado Broker. O Gatherer responde com uma lista de documentos a serem criados, removidos ou atualizados desde a última vez que foram armazenados no Broker. O Collector trata a resposta e a repassa para o Registry processar;
- *Query Manager*: exporta objetos para a rede. Ele aceita uma consulta, traduz esta para uma representação intermediária e repassa-a para um engenho de busca. O engenho de busca responde com uma lista de identificadores. É construída uma descrição dos documentos a partir dos identificadores. Ele, então, retorna uma lista de descrição de objetos para o cliente;

- *Replication Manager*: para diminuir a carga na rede, o Replication Manager mantém várias cópias idênticas de *Brokers.* Ele permite que um número de *Brokers* seja executado sobre os mesmos dados para que a carga tanto de rede como de processador seja dividida entre as várias máquinas onde são executados os *Brokers*;
- *Object Cache*: uma hierarquia de *cache* de objetos, para responder às demandas de acesso de rede e informação dos servidores.

Para que essa abordagem seja utilizada é necessário que em cada servidor *Web* exista um *Provider*, que indexa as páginas novas e modificadas do *site*, ou seja, requer que todo servidor *Web* possua um componente para comunicar mudanças, o que torna difícil a sua utilização em larga escala, devido a não existência de um padrão para esses produtores na prática. Ele é utilizado, entretanto, em aplicações para conjuntos menores da *Web*, por exemplo, pela *CIA*, *NASA*, e Academia Nacional de Ciências dos Estados Unidos que utilizam este *software*, que é de domínio público. Existe também uma versão comercial do *Harvest*, o *Netscape`s Catalog Server* [6].

Seguindo a mesma linha de Harvest, é proposta em [20] uma abordagem na qual os próprios servidores *Web* monitoram as mudanças que acontecem nos seus arquivos e as propagam para os engenhos de busca. É proposto um algoritmo que usa tanto a atualidade quanto a popularidade das páginas, dado que o servidor *Web* sabe quais são as suas páginas mais populares e quando elas são modificadas. Para evitar uma sobrecarga sobre os engenhos e sobre a conexão de rede dos servidores *Web*, para o caso, por exemplo, de milhões de servidores *Web* enviarem informações aos EBs de forma ininterrupta, é proposto um módulo chamado de *middleman*. Esse componente é responsável por um conjunto de servidores *Web* e envia as atualizações para os EB com uma determinada periodicidade. A maior limitação dessa proposta, assim como no Harvest, é que ela também não é usada na prática, dado que não existe um padrão que obrigue aos servidores Web implementarem essa política.

Uma abordagem um pouco diferente das duas anteriores é proposta por [39]. Nela, o envio das páginas que foram modificadas é realizado por um servidor de *cache proxy*, próximo ao mecanismo de busca. Neste caso, a atualidade da BI depende da

política de atualização do servidor de *cache proxy*. Esta idéia tem duas vantagens: (1) usa as tecnologias de *Web caching* disponíveis com pequeno custo de implementação e (2) através do seu uso, documentos com grande quantidade de acessos teriam cópias locais mais atualizadas, o que vai de acordo com os resultados apresentados em [14], o qual, resumidamente, mostra que páginas mais freqüentemente acessadas são mais jovens que aquelas menos acessadas. Esta abordagem é chamada de *Cache* Parasita.

A implementação corrente é específica para o servidor de *proxy cache* Squid[15]. Ela pode ser facilmente utilizada com os servidores já em operação. Um redirecionador de *URLs* é instalado no servidor de *proxy*. Este redirecionador é um pequeno *script* (facilmente incorporado na instalação do Squid) que intercepta cada requisição do usuário para o servidor *proxy* e envia a *URL* para o engenho. A *URL* é checada para determinar se ela é nova ou foi coletada num período maior que um tempo pré-estabelecido, para que possa ser indexada. Em caso positivo, um coletor especial, o *proxy collector*, requisita ao servidor *proxy* o documento que a *URL* representa. É importante ressaltar que esta proposta, embora totalmente implementada, ainda não está operacional em servidores *proxy* comerciais, porque não está definido como padrão para servidores *proxy*, é justamente este o principal fator limitante para sua utilização na prática.

4.3.1.2 Manutenção de Arquivo contendo a Informação das Páginas Alteradas

Em [7], propõe-se que o servidor *Web* mantenha um arquivo contendo uma lista de *URLs* e suas respectivas datas de última modificação. Assim, inicialmente, o Robô deve baixar o arquivo contendo estas *URLs*, identificar quais foram modificadas desde a sua última visita, e requisitar somente as páginas daquelas que foram modificadas, evitando assim o desperdício de recursos do servidor *Web*.

Outras vantagens do uso desta proposta são: o arquivo contendo as *URLs* pode também ajudar os robôs a alcançarem páginas dinamicamente geradas por esse *site*; este

[15] Squid home page - http://squid.nlanr.net/Squid

arquivo, que terá que ser coletado pelo Robô, levará a um consumo menor de banda do que se o Robô visitasse as páginas que não mudaram; mídias específicas serão indexadas sem a necessidade de se indexar outros tipos de mídias, pois poderão ser filtrados os *links* com a mídia desejada.

É mostrado, pelos experimentos realizados, que essa proposta permite uma maior atualidade da base com um menor consumo de banda, em relação à indexação tradicional. Ao contrário das propostas mostradas anteriormente, que demandam esforço extra dos servidores *Web*, para essa proposta, é necessário apenas que o servidor *Web* disponibilize uma lista das *URLs* e suas datas de modificação. Entretanto, como essa política não é especificada com padrão para os servidores *Web*, essa abordagem não é utilizada amplamente.

4.3.2 Abordagem Ativa

Na abordagem ativa, o mecanismo de busca não recebe nenhuma informação de entidades externas sobre a modificação de páginas. Ele tenta inferir com que freqüência as páginas se modificam e implementa políticas para sincronizar estas páginas modificadas com a sua base de índices. Estas abordagens, portanto, propõem a atualização da base de índices utilizando uma política não uniforme (Seção 2.5.1).

No modelo apresentado por [15], não é feita nenhuma suposição *a priori* de como as páginas mudam. Entretanto, à medida que vai se visitando as páginas, seu histórico vai sendo guardado para que se possa supor o seu comportamento a partir desse histórico. De forma mais detalhada, neste modelo, cada vez que uma página é visitada, é guardado se ela mudou ou não, desde a última visitação do engenho. Esta informação é utilizada para colocar a página em um dos grupos de freqüência de modificação semelhantes. Para este agrupamento de páginas é utilizado um modelo matemático de equações não-lineares. Este modelo é adaptativo, pois, quanto mais vezes uma página for visitada, mais confiáveis serão os dados.

Os problemas dessa proposta são:

- Complexidade: tem-se que lidar com problemas não-linerares para encontrar o grupo ao qual a página pertence. Esse algoritmo após um certo número de visitas a uma página pode levar um tempo infinito para ser executado, por isso os autores sugerem que, de vez em quando, alguns contadores desse algoritmo devam ser zerados e comecem do zero;

- Não supõe inicialmente o comportamento das páginas: são necessários mais ciclos de indexação para agrupar as páginas em seus devidos grupos, do que se, inicialmente, houvesse uma indicação a qual grupo a página pertence. Isso leva a um maior desperdício de recursos.

Uma outra abordagem [11] também tenta aprender como as páginas se modificam com o passar do tempo, sem, inicialmente, fazer nenhuma suposição sobre o comportamento delas.

Para poder estimar com que freqüência uma determinada página muda, existe um Módulo de Atualização que guarda o *checksum*[16] da página anteriormente e o compara com o *checksum* da página atual. Desta comparação, o módulo pode dizer se a página mudou ou não. Para tentar prever a freqüência de alteração, o Módulo de Atualização possui heurísticas, chamadas de "estimadores".

Um tipo de estimador é baseado no processo de *Poisson*. Para implementá-lo tem-se que guardar quantas vezes o engenho detectou mudanças para uma página nos últimos, digamos, 6 meses. Esse valor é usado para obter um intervalo de confiança para a freqüência de mudança dessas páginas. O outro estimador é o apresentado na Seção 3.4, baseado no processo de inferência *bayesiano*.

Ao contrário de [15], a heurística usada para aprender em que classe está uma página é menos complexa, pois os estimadores utilizam equações simples para a descoberta do comportamento de uma página. Entretanto, ela apresenta também o mesmo problema da abordagem anterior, não faz nenhuma suposição inicial de como as páginas mudam.

[16] Checksum é um valor inteiro gerado por algum algoritmo que, de forma simplificada, é uma função que mapeia um vetor de caracteres a um número inteiro.

4.4 Conclusão

Foram apresentados, neste capítulo, os principais estudos para melhorar a atualidade de uma base de índices de um engenho de busca.

Na abordagem tradicional, todo o índice é atualizado à mesma freqüência. Apesar da simplicidade de sua implementação, seu uso acarreta um desperdício de recursos computacionais por parte do engenho. Por esta razão, ela não é considerada uma boa proposta para o problema de atualização de índice.

As abordagens não tradicionais tentam justamente atacar este problema de desperdício de recursos, para tanto propõem que sejam visitadas as páginas que realmente mudaram desde a última visita do engenho.

Em uma dessas abordagens, a abordagem passiva, existe um agente externo ao mecanismo de busca que identifica quais elementos foram modificados. Estes agentes ou notificam ao engenho estes elementos, ou disponibilizam essa informação para que o engenho recupere os documentos modificados.

A principal desvantagem da abordagem passiva é a dependência do engenho de busca de agentes externos para a identificação dos elementos que se modificam. Como não há nenhum padrão, tanto para os produtores de documentos, como para servidores de *cache proxy*, esta abordagem na prática se torna difícil de ser implementada. Em [7] tenta-se diminuir o esforço por parte desses agentes ao disponibilizarem este tipo de informação, mas ainda assim, persiste a dependência.

A abordagem ativa, ao contrário, não utiliza agentes externos, ela tenta prever como as páginas se modificam, justamente, por não haver esta dependência de um agente externo ela pode ser implementada na prática. Entretanto, justamente por não utilizarem esses agentes, não sabem exatamente quais documentos foram alterados, sendo esta imprecisão, portanto, a sua desvantagem.

Os principais trabalhos dentro desta abordagem apresentam modelos que tentam prever o comportamento das páginas *Web*. Para isso não assume, inicialmente, qualquer comportamento dessas páginas. Somente com o passar do tempo é que vai-se aprendendo como elas se comportam, podendo, dessa forma, atualizar mais aquelas que

se modificam mais, e menos, as que se modificam menos. Pelo fato de não assumirem *a priori* o comportamento das páginas, pode-se levar muito tempo para que as páginas sejam colocadas em suas classes de alteração corretas, sendo esta, portanto, sua principal deficiência.

Pelo exposto neste capítulo, podemos concluir:

1) A abordagem tradicional desperdiça bastante recursos;
2) A abordagem passiva, ao contrário, desperdiça menos recursos, mas é difícil de ser utilizada na prática;
3) A abordagem ativa, embora não seja tão eficiente quanto a passiva, pode ser implementada na prática, pois não necessita de nenhum agente externo ao engenho para implementar sua política de atualização;
4) O ponto fraco das propostas que utilizam a abordagem ativa, é não fazer nenhuma suposição inicial sobre o comportamento das páginas, aprendendo como elas se comportam à medida que elas são visitadas. Este aprendizado pode ser demorado, o que acarreta num maior desperdício, pois para aprender o comportamento das páginas é necessário visitá-las e, durante este processo, estarão sendo visitadas páginas que não foram alteradas.

Baseado na análise destas abordagens, vai ser proposta uma solução para o problema de atualização de base de índices que utilize a abordagem ativa, mas atacando o seu ponto fraco que é a não suposição inicial de como as páginas se comportam. Como isso será feito, será melhor exposto no próximo capítulo.

Capítulo 5 Construindo os Classificadores para a Atualização da Base de Índices

5.1 Introdução

No capítulo anterior foram apresentadas propostas para a atualização da base de índices de um engenho de busca. A fim de atacar as principais deficiências destas abordagens, será apresentada, neste capítulo, uma proposta que resolva, de forma eficiente, o problema de atualização de base de índices.

Vai ser apresentado inicialmente neste capítulo (Seção 5.2), como a proposta está composta. Basicamente, ela se baseia na construção de dois classificadores para agrupar as páginas em suas corretas classes de alteração: um baseado nas características das páginas e outro no histórico de mudanças delas. Assim, na Seção 5.3, é mostrado como foi realizado o monitoramento do comportamento das páginas da *Web* brasileira, para a obtenção dos dados utilizados nos processos de construção dos classificadores. Esses dados são os valores de alguns atributos das páginas e o histórico delas. A partir desses dados, foram geradas algumas estatísticas sobre o comportamento da *Web* brasileira, como mostrado na Seção 5.4.

O procedimento utilizado para a elaboração e construção da proposta de solução é apresentado nas Seções 5.5, 5.6 e 5.7 (ver Figura 6). Na Seção 5.5, é mostrada a construção do classificador baseado nos atributos das páginas. Para isso, inicialmente, a partir dos dados de histórico das páginas, é realizado o agrupamento das páginas que

possuem taxas de modificação semelhantes, criando, dessa forma, classes de páginas. De posse dessas classes geradas e dos valores dos atributos das páginas, é realizada a etapa de seleção de atributos para eliminar aqueles atributos irrelevantes para a construção deste classificador. Por fim, utilizando como entrada os valores dos atributos selecionados e as classes geradas nas etapas anteriores, é construído o classificador baseado nos atributos da página, utilizando algumas técnicas de Aprendizagem de Máquina. Na Seção 5.6, são apresentadas as técnicas utilizadas para a construção do classificador baseado no histórico de mudanças. Na seção 5.7, os classificadores iniciais (classificador baseado nas características da página e aleatório) são compostos com os baseados no histórico de mudanças (bayesiano e heurístico) através de uma simulação sobre dados de histórico, para que fosse escolhida aquela configuração que apresentasse o melhor desempenho a ser utilizada na proposta de solução. Por fim, na Seção 5.8, conclui-se o que foi exposto neste capítulo.

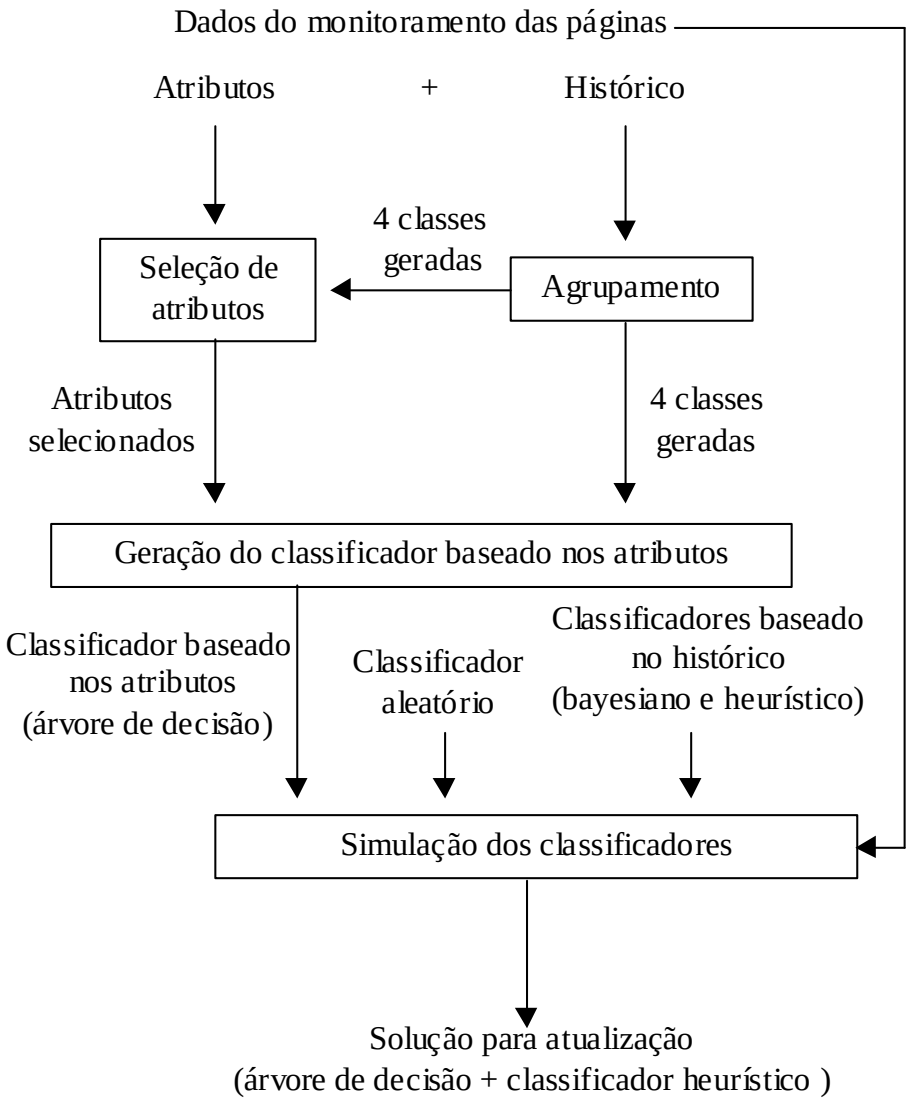


Figura 6 - Fluxograma das atividades para a construção da proposta de atualização.

5.2 Características da Proposta

A proposta apresentada neste trabalho se baseia em alguns pontos positivos das abordagens apresentadas no Capítulo 4, como também procura solucionar alguns problemas existentes nessas abordagens. Para isso, ela possui as seguintes características:

- Abordagem ativa: como apresentado no Capítulo 3, apesar da abordagem passiva ser mais eficiente, ela é difícil de ser utilizada na prática, como conseqüência disto, foi escolhida a abordagem ativa. Com isso o engenho de busca pode implementar sua política de atualização da base de índices, ficando independente de agentes externos ao engenho para isso;

- Política não-uniforme: pelos problemas mostrados na Seção 4.2 em relação à política uniforme, o método proposto utiliza a política não-uniforme para a atualização dos elementos da base;

- Identificação do comportamento de cada página: para o uso da política não-uniforme é necessária uma estratégia para definir quais os grupos de páginas de atualização. Além de utilizar um histórico do comportamento de cada página, neste trabalho, também, é utilizado um classificador de páginas baseado em suas características para que, na primeira vez que se colete a página, já se identifique a que classe de modificação ela pertence. Este classificador será chamado, neste trabalho, de classificador inicial. Após esta classificação inicial, utiliza-se o classificador baseado no histórico de alteração das páginas para classificá-las com o passar do tempo.

A utilização deste classificador no início do processo de atualização vem atacar o problema encontrado nas abordagens que utilizam apenas o histórico de mudanças das páginas para identificar o seu comportamento. Essas abordagens, no início do processo de atualização, colocam as páginas de forma aleatória nos grupos de atualização, ou seja, não assumem o comportamento das páginas em termos de sua freqüência de modificação. Isto pode acarretar na perda de tempo para que as páginas sejam colocadas em suas classes de alteração corretas, desperdiçando, dessa forma, recursos durante este processo. Um outro problema dessa abordagem é, por exemplo, para páginas que possuem um pequeno período de vida. Neste caso, poderá não haver tempo suficiente para que elas possam ser atualizadas de forma eficiente.

Além da utilização em engenhos de busca, esse classificador pode ser utilizado por servidores *proxy* para tentarem prever com que taxa as páginas são modificadas e conseqüentemente atualizar sua *cache*.

Para a criação deste classificador serão utilizadas técnicas de Aprendizagem de Máquina e, até onde se sabe, não há na literatura nenhum trabalho que tenha criado um classificador como este para tentar prever o comportamento das páginas *Web*.

No decorrer deste capítulo, será apresentado com que dados foi criado este classificador, quais as características (atributos) das páginas foram utilizadas para a sua construção e quais as técnicas de Aprendizagem de Máquina utilizadas para esse fim.

5.3 Monitoramento da *Web*

O monitoramento das páginas Web teve como objetivo obter as informações necessárias para a construção dos classificadores. Para isso, são necessárias as seguintes informações: (1) as características das páginas e (2) o comportamento delas com o passar do tempo (histórico de mudanças). Para o classificador inicial são utilizadas as características das páginas para, a partir delas, poder-se inferir o comportamento das mesmas de acordo com grupos de alteração criados baseados nas informações de histórico. Para o classificador baseado no histórico, a partir do comportamento das páginas, são escolhidos os melhores parâmetros utilizados por este classificador para que ele obtenha um bom desempenho. Mais detalhes sobre estes classificadores serão apresentados nas próximas seções.

Os dados utilizados neste trabalho tiveram como escopo a *Web* brasileira, por existirem poucos estudos sobre o seu comportamento [17], e por este projeto, na época de sua elaboração, ter sido financiado pela Mobile, empresa que provia tecnologia para o *site* Radix [37], que é uma máquina de busca para a *Web* brasileira.

Inicialmente, tentou-se utilizar as informações contidas no *Internet Archive*[17], que é uma organização que "congela" certas parcelas da *Web* em determinados períodos para que sejam feitos estudos. Entretanto, seu uso foi inviável, pois esses dados não

[17] Internet Archive - http://www.archive.org

possuíam informação sobre algumas das características das páginas, necessárias para a realização deste experimento. Por exemplo, a profundidade em relação à raiz do domínio, que representa a distância em *hiperlinks* de uma determinada à raiz do domínio. O conceito de domínio para este trabalho é, por exemplo, para o *site* www.hpg.com.br, todas as páginas pertencentes a *.hpg.com.br. Foi preciso, portanto, construir um sistema para que se obtivesse estas informações.

A função deste sistema é obter as informações de histórico de mudanças de páginas por um certo período e as características dessas páginas[18]. Para isso é necessário que um conjunto preestabelecido de páginas seja monitorado em relação à sua modificação, com uma determinada freqüência por um certo período de tempo. A realização deste monitoramento será apresentada nas próximas seções.

5.3.1 O Experimento

Como não é praticamente possível fazer um estudo sobre toda as páginas da *Web* brasileira, que segundo [42], já possuía cerca de 18 milhões de páginas em 2000, procurou-se um conjunto de páginas que representasse o que fosse mais importante para o usuário da *Web* brasileira, ou seja, aqueles domínios mais acessados pelos internautas brasileiros.

Existe uma série de institutos de pesquisa na Internet brasileira que aferem audiência. Através de um deles, o Ibope eRatings.com, foram obtidos os *sites* mais acessados da *Web* brasileira em maio de 2001, que segundo o próprio instituto, os acessos a esses *site*s representaram cerca de 92% do total de acessos à *Web* brasileira neste período.

A partir das páginas iniciais destes *sites* realizou-se uma busca em largura até uma profundidade 9, considerando a raiz com profundidade igual a um. A largura da busca foi limitada em cada domínio em 500 páginas. A limitação da largura e da profundidade foi porque não se queria sobrecarregar os *site*s que estavam sendo

[18] Mais detalhes da constituição e funcionamento deste sistema está exposto no Apêncie C.

utilizados pois seria obtido um número muito grande de *URLs* desses *sites*. Por fim, obtivemos 84.699 páginas pertencentes aos domínios de cada *site*.

Durante a realização da busca em profundidade foram obtidas as características das páginas utilizadas para a construção do classificador inicial, são elas:

- Número de *links*: números de *links* para os quais a página aponta;
- Número de *e-mails*: número de *links* para *e-mails* existentes na página;
- Existência do campo de cabeçalho *HTTP last-modified*: este cabeçalho indica a data de última modificação da página. Os servidores proxy cache, para verificar se uma página mudou ou não, desde a última vez que esta foi colocada na *cache*, recuperam essa informação do servidor Web no qual a página está hospedada. A sua não existência pode significar que a página é tão dinâmica que não vale a pena colocá-la em *cache*;
- Tamanho do arquivo: tamanho em *bytes* da página sem os marcadores *html*;
- Profundidade da página no domínio: profundidade a partir da raiz através de uma busca em profundidade pela estrutura de *links* do domínio;
- Número de imagens: número de imagens referenciadas na página;
- Nível da *URL* da página em relação à *URL* raiz do servidor *Web*: o nível é contado através da estrutura de diretório dos servidores *Web*. O diretório raiz do servidor é o nível 1, os seus sub-diretórios nível 2 e assim sucessivamente. Por exemplo, a *URL* http//www.cin.ufpe.br possui nível 1 e http://www.cin.ufpe.br/~lab nível 2.

Estas características foram escolhidas devido a alguns fatores:

1. *Facilidade para obtenção*: a maioria dessas características está contida no conteúdo da página HTML que representa o documento ou na *URL* que a representa. A única exceção é a profundidade da página no domínio, obtida durante a execução da busca em profundidade. Existem outras características

da página que poderiam ter sido utilizadas (*backlinks*, PagaRank [32], HITS [23] e GHHITS [10]), mas que são difíceis de serem conseguidas. Por exemplo, para a obtenção do número de *backlinks* de uma página, número de *links* que apontam para a página, seria necessário um procedimento a mais no sistema de monitoramento, que contabilizasse este valor para cada página;

2. *Citação na literatura*: alguma delas já foram citadas na literatura ([3], [8],[14]) como características que estão correlacionadas com a freqüência de modificação de uma página. No trabalho apresentado por [3], uma de suas conclusões é, para o caso de *sites* de notícias, que as páginas com menor profundidade em relação à raiz do *site* possuem maior taxa de alteração. Em [8], afirma-se, por exemplo: (1) uma página onde não há a existência do cabeçalho *HTTP last-modified* possui o dobro de probabilidade de ser modificada do que uma que possui este cabeçalho; (2) páginas mais novas são freqüentemente maiores e possuem mais imagens. Nos resultados apresentados por [14], as páginas mais modificadas são as que possuem mais imagens.

Uma vez obtido o conjunto de 84.699 *URLs* e suas características, foi agendado para que, diariamente, fossem visitadas todas as *URLs* deste conjunto, a fim de que se obtivesse o histórico de alteração para cada página. A duração deste experimento foi de 100 dias. Diariamente todo o conjunto era visitado num período médio de 5 horas. A visita às páginas foi realizada durante a madrugada, pois, normalmente, este é o período com menor carga de usuários sobre os *site*s.

Ao final dos 100 dias, tinham sido coletadas para cada página suas características, obtidas na busca em largura, e uma lista de datas de modificação, representado o histórico de alteração das páginas. A partir destes dados, foram retiradas algumas estatísticas sobre o comportamento da *Web* brasileira e foram gerados os classificadores que compõem a proposta de solução.

5.4 Estatísticas sobre a Dinamicidade da *Web* Brasileira

É apresentada, nesta seção, uma visão geral de como a *Web* brasileira se comporta em relação à sua modificação, ou seja, quanto tempo leva uma página para mudar, segundo os dados obtidos. Para medir esse tempo usou-se o intervalo médio de mudança, como utilizado também por [11], que é calculado da seguinte forma: se uma página mudou 10 vezes no período de 100 dias, então o intervalo médio de mudança dessa página é de 10 dias (100/10). Uma observação importante sobre essa forma de medição é que, considerando que uma página foi visitada apenas uma vez ao dia e que esta pode mudar mais de uma vez neste período, pode-se obter um valor subestimado para o intervalo médio de mudança para ela.

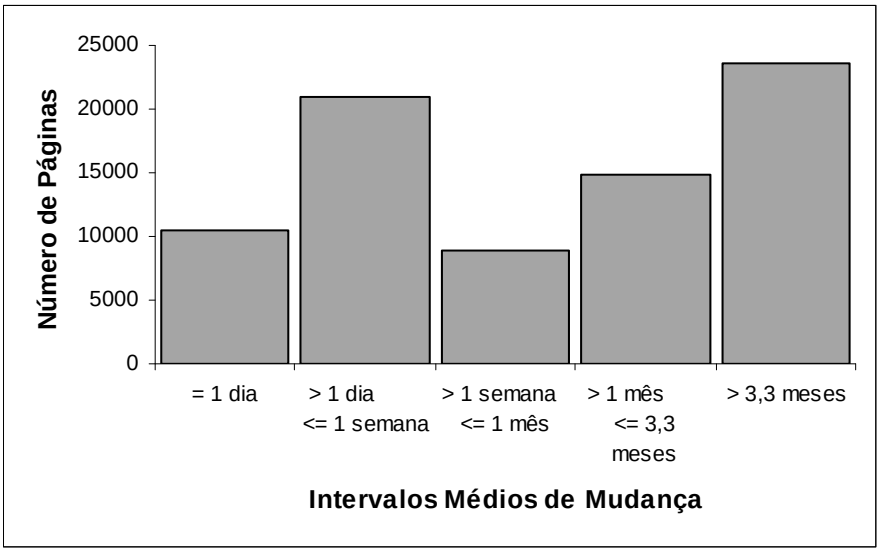


Figura 7 - Total de páginas modificadas por intervalo médio de mudança.

Na Figura 7, estão apresentadas as porcentagens das páginas dentro de cada intervalo médio de mudança para todo conjunto de páginas coletadas. De acordo com os dados, aproximadamente 40% do conjunto de páginas observadas da *Web* brasileira muda em um período menor ou igual a uma semana e aproximadamente 30% num período maior que 3,3 meses, que são páginas que não foram modificadas durante todo o período do experimento. Esses números vão de acordo com [17], pois uma de suas

conclusões foi que, na Web brasileira, "muitos documentos são pouco modificados, muitos são modificados freqüentemente e poucos estão nos níveis intermediários".

Dentro desses mesmos intervalos médio de mudança obtiveram-se outros resultados para as páginas no domínio **.com** (Figura 8), como também para os outros domínios que não são **.com** (.org, .gov, entre outros), chamado, aqui neste trabalho, de **outros** (Figura 9). Percebe-se por esses números que páginas pertencentes ao domínio **.com**, são mais modificadas do que as não pertencentes a ele, pois mais de 50% do domínio .**com** são modificadas em menos de 1 mês, enquanto que em **outros** a maioria está no intervalo de 1 a 3,3 meses. Isso é devido ao caráter comercial das páginas pertencentes ao domínio .**com**, que precisam estar sempre atualizadas para atrair os usuários**.** Um resultado semelhante foi encontrado por [11] para toda a *Web*.

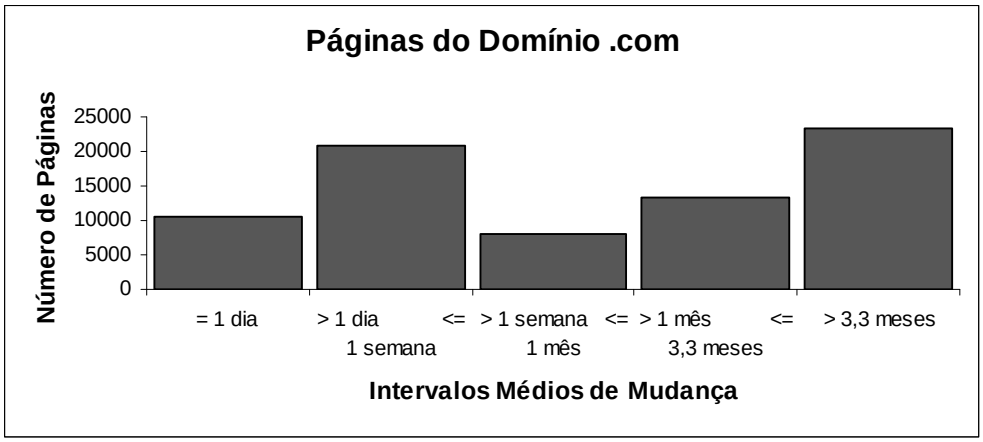


Figura 8 - Total de páginas do domínio .com modificadas por intervalo médio de mudança.

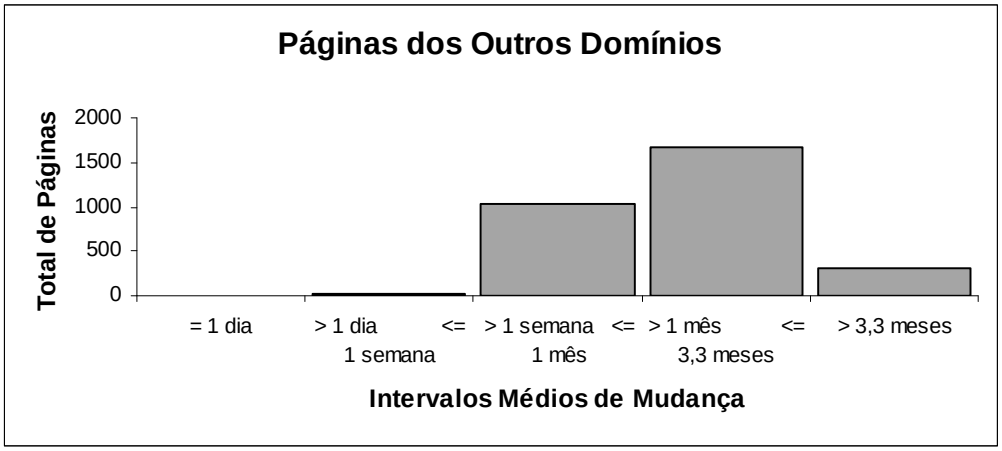


Figura 9 - Total de páginas dos outros domínios modificadas por intervalo médio de mudança.

Esses números apresentados são relativos às páginas mais acessadas da Web brasileira, e como apresentado por Douglis et. al. [14], páginas que são muito acessadas são páginas muito modificadas. Portanto, não se pode assumir que todas as páginas da Web brasileira possuam esse comportamento e sim as mais acessadas pelos usuários.

5.5 Classificador baseado nas Características das Páginas

Um dos objetivos que se pretende alcançar com este estudo, é inferir o comportamento das páginas *Web*, com relação à sua freqüência de modificação, a partir das características das páginas, citadas na seção 5.3.1. Com a realização do monitoramento da *Web*, obteve-se a massa de dados para que se pudesse construir este classificador.

Esses dados contêm as informações obtidas de cada uma das 84.699 páginas monitoradas, que são as suas características e um histórico de mudanças (uma lista de tamanho 100 de datas de modificação das páginas em cada dia de visita do sistema indicando se a página foi modificada ou não nesse dia). Deste total de *URLs* foram utilizados dois terços (56.466) para a construção do classificador e, o um terço restante, para um experimento que compara o classificador gerado com algumas abordagens já existentes (Seção 5.7).

As etapas para a construção do classificador são as seguintes:

1. *Clustering*: a partir dos dados obtidos, foram gerados grupos de páginas que possuem taxa de alteração semelhantes;
2. Preparação da entrada: de posse destes grupos, foi criado o arquivo contendo as páginas pertencentes aos grupos e suas características;
3. Seleção de atributos: dentre as várias características das páginas foram removidas aquelas redundantes ou irrelevantes para a construção do classificador;
4. Construção do classificador: a partir do arquivo gerado na preparação de dados, foi construído o classificador baseado nas características da página, excetuando-se aquelas características removidas na etapa de seleção de atributos.

Cada etapa será melhor detalhada nas próximas subseções.

5.5.1 Agrupamento

A primeira tarefa a ser realizada é agrupar as páginas com freqüências de modificação semelhantes. Para que fossem, então, encontradas essas classes de alteração das páginas, foi necessário para cada página estimar com que freqüência ela mudava baseado no seu histórico de mudança. Isto foi realizado através do uso de um estimador, que será descrito posteriormente. A partir desta estimativa representada por um valor, realizou-se o *clustering* desses dados.

Em Cho e Molina [13], é feito um estudo sobre qual o melhor estimador em relação à forma que é realizado o monitoramento dos elementos que se deseja estimar o comportamento. O monitoramento da *Web* brasileira apresentado anteriormente foi realizado da seguinte maneira:

- Ativo: foram monitoradas ativamente as páginas, podendo-se configurar com que taxa elas seriam visitadas;

- Intervalo de acesso regular: as páginas foram visitadas para verificar o seu estado num intervalo regular de 1 dia;

- História de mudança incompleta: como as páginas foram visitadas em uma janela de tempo, muitas delas já existiam antes de começar o monitoramento, como também muitas continuaram existindo depois do término do monitoramento. Além disso, não se sabe se uma página mudou entre os acessos a ela.

Baseado nestas características de monitoramento, o estudo [13] comprova que o melhor estimador relacionado com a freqüência de alteração das páginas está apresentado na Equação 14, onde n é o número de visitas feitas à página e X o número de modificações da página:

$$-\ln(n - X + 0.5 / n + 0.5)$$

Equação 14 - Estimador para a freqüência de alteração de páginas.

Então, para cada uma das páginas, de um total de 56.466, do conjunto de entrada, foi realizado o seguinte procedimento: (1) a partir do seu histórico de alteração, foram calculados os valores n e X (ver Equação 14) e, (2) utilizando-se a Equação 14, calculou-se o valor do estimador relacionado com a freqüência de alteração. Os valores calculados para todo o conjunto de páginas serviram, então, como entrada para o algoritmo de *clustering*.

De posse da entrada para o algoritmo, utilizou-se o algoritmo *k-means* gerando 2, 3, 4, 5 e 6 *clusters*, para que se decidisse qual dessas configurações é a mais interessante segundo alguns critérios. Para a escolha de qual configuração de *clusters*, deve-se ter em mente que o objetivo final desta tarefa é escolher grupos que serão usados por engenhos de busca para a atualização da sua base de índices. Portanto, foram definidos os seguintes critérios para a escolha:

- *Priorizar páginas mais modificadas*: deseja-se dar prioridade para atualizar as páginas que mais se modificam. Como apresentado nos resultados do monitoramento da *Web* brasileira, existe uma maior proporção de páginas do domínio **.com** nas classes de páginas que mais se modificam, e são justamente essas páginas as mais acessadas pelo usuário da *Web*. Segundo dados do

Ibope E-Ratings, 90% dos *site*s mais acessados da *Web* brasileira são do domínio **.com**;

- *Evitar grupos com poucos elementos*: é importante que cada grupo gerado pelo algoritmo de *clustering* possua uma proporção significativa sobre o total, dado que, como citado anteriormente, a existência de um grupo significa o uso de recursos por parte do engenho de busca, o que não valeria a pena se este grupo não possuísse uma parcela razoável do total de páginas.

Nas tabelas abaixo estão representados, para cada configuração de *cluster*, os números de elementos de cada grupo e a freqüência relativa média dos elementos do grupo, que, por exemplo, quando ela é igual a 2, significa que, em média, as páginas deste grupo mudam a cada 2 dias.

Tabela 3 - Números para 2 *clusters*.

Grupos	Número de elementos	Freqüência relativa média em dias
1	11382	1,96
2	45084	87,94

Tabela 4 - Números para 3 *clusters*.

Grupos	Número de elementos	Freqüência relativa média em dias
1	11260	1,9
2	6316	31,27
3	38889	96,89

Tabela 5 - Números para 4 *clusters*.

Grupos	Número de elementos	Freqüência relativa média em dias
1	6085	1
2	5148	3,11
3	6259	31,81
4	38844	96,94

Tabela 6 - Números para 5 *clusters*.

Grupos	Número de elementos	Freqüência relativa média em dias
1	6215	1

2	5136	3,09
3	5609	29,90
4	4093	68,71
5	35386	99,33

Tabela 7 - Números para 6 *clusters*.

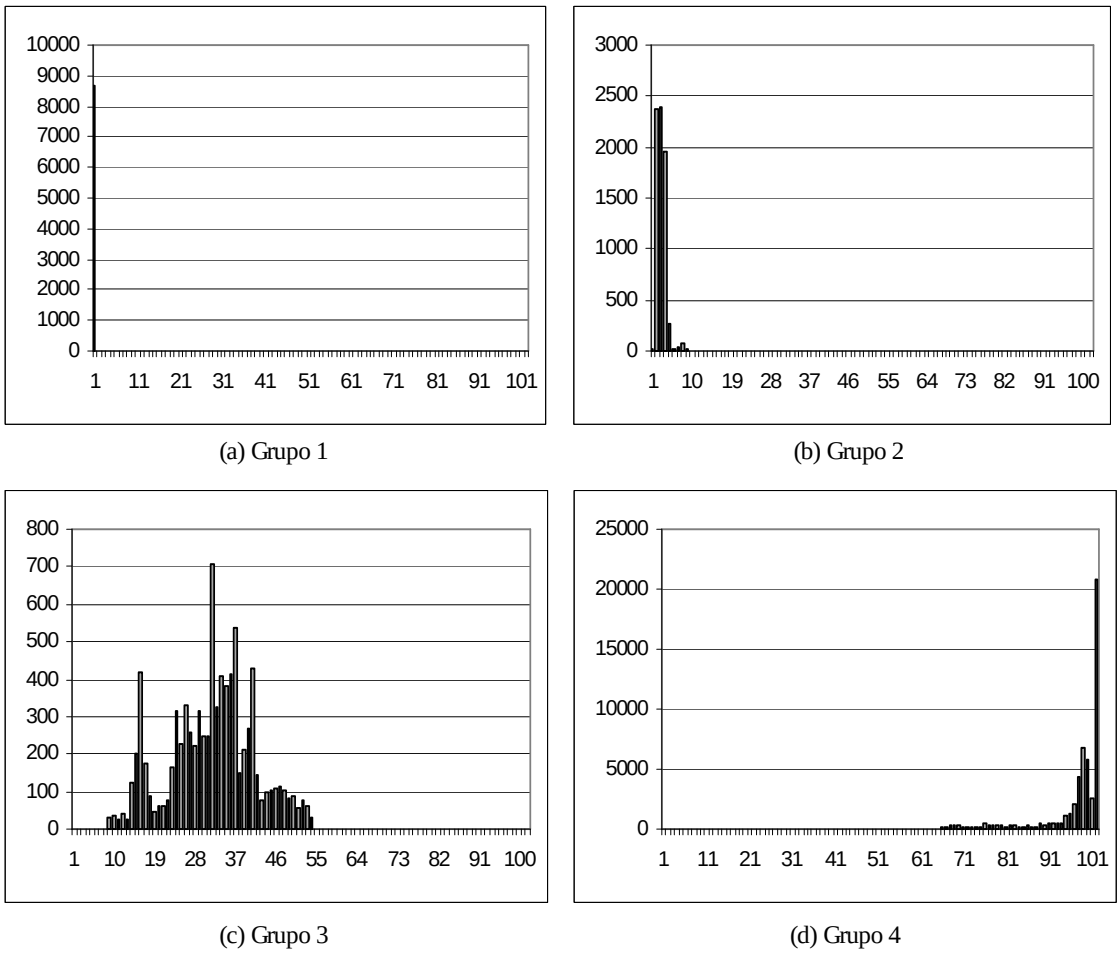
Grupos	Número de elementos	Freqüência relativa média em dias
1	6085	1
2	5050	3,02
3	1111	15,44
4	4976	33,96
5	4053	71,9
6	35056	99,48

Pelos números apresentados e baseado nos critérios citados anteriormente, escolheu-se a configuração com 4 classes, pelos seguintes motivos:

- Em relação às configurações com 2 e 3 classes: o grupo de 4 classes beneficia a atualização de páginas mais dinâmicas já que existem os grupos para páginas que mudam a cada 1 dia e 3,11 dias, enquanto que as configurações de 2 e 3 classes unem estes grupos em um só, com a freqüência relativa de cerca de 1,9;
- Em relação à configuração com 5 classes: só os grupos de páginas menos dinâmicas iriam beneficiar-se com esta configuração;
- Em relação à configuração com 6 classes: um dos grupos é pequeno em relação ao total de páginas. O grupo 3 na configuração de 6 classes possui uma porcentagem de apenas 1,9% do total de elementos.

A distribuição da freqüência de alteração dos elementos da configuração com 4 grupos está apresentada na Figura 10. Por ela pode-se perceber a existência de cada grupo bem definido: o Grupo 1, Figura 10(a), com somente páginas que mudam 1 vez ao dia; o Grupo 2, Figura 10(b), com páginas que mudam entre 2 a 9 vezes ao dia; o Grupo 3, Figura 10(c), com as que são alteradas entre 10 e 55 dias; e, Grupo 4, Figura 10(d), as páginas que se modificam mais que 55 dias.

Baseado nesta configuração de 4 classes é que serão construídos os classificadores utilizados no processo de atualização da base de índices do mecanismo de busca.



(a) Grupo 1 (b) Grupo 2

(c) Grupo 3 (d) Grupo 4

Figura 10 - Distribuição das freqüências em cada um dos 4 *clusters*.

5.5.2 Preparação da Entrada

Baseado nessas 4 classes geradas através da técnica de *clustering*, foram construídos os dados de entrada para ser utilizado nos próximos passos de construção dos classificadores. Para isso, precisou-se realizar algumas tarefas sobre os dados de entrada. Inicialmente, realizou-se o mapeamento dos elementos pertencentes a cada

classe criada no processo de *clustering*, com suas características ou atributos. Após isso, foi colocado o mesmo número de elementos para cada um dos 4 conjuntos, tomando-se como base o número de elementos do menor conjunto (cerca de 5.000) criou-se, então, um conjunto de entrada com 20.000 elementos, 5.000 de cada classe. Não foi necessária a criação de dois arquivos um para treinamento e outro para teste, dado que a ferramenta utilizada, WEKA, já separa os conjuntos automaticamente.

5.5.3 Seleção de Atributos

Nesta etapa, deseja-se selecionar quais os atributos relevantes para o processo de construção do classificador. Existem inúmeras técnicas de aprendizagem de máquina. Para a seleção do melhor sub-conjunto dos atributos foram utilizadas a árvore de indução, o classificador bayesiano ingênuo e o k-vizinhos mais próximos (ver Seção 3.3.4) por serem algoritmos bastante utilizados na prática e por estarem disponíveis na ferramenta WEKA.

Após a execução da seleção de atributos utilizando os três algoritmos de aprendizagem citados anteriormente, em todos os casos, o melhor sub-conjunto de atributos escolhido pela técnica de seleção de atributos foi o formado pelos seguintes atributos: número de *links*, número de *e-mails*, número de imagens, tamanho da página, existência ou não do cabeçalho *last-modified* e o nível da *URL*.

O atributo de profundidade no domínio foi eliminado do conjunto de atributos para a classificação. A inclusão inicial deste atributo foi baseada em um estudo realizado [3] que mostrava que páginas próximas à raiz de um *site* são mais modificadas do que aquelas mais distantes. Entretanto, naquele trabalho foi utilizado o conceito de profundidade em relação ao *site*[19] e não em relação ao domínio, sendo esta provavelmente a causa deste atributo não se mostrar relevante para a construção do classificador.

[19] Profundidade em relação ao site signfica por exemplo que somente serão consideradas páginas pertencentes ao site. Por exemplo, para o site http://www.jb.com.br, só serão aceitas todas as páginas pertencentes a http://www.jb.com.br/*.

5.5.4 Geração do Classificador

Esta é a fase final para a construção do classificador baseado nas características das páginas. Para a realização desta tarefa, utilizou-se como entrada o arquivo obtido da etapa de preparação da entrada, retirando-se o atributo de profundidade do servidor, pois este foi considerado irrelevante pela etapa anterior. Então, a partir do melhor sub-conjunto de atributos escolhido pela tarefa de seleção de atributos, gerou-se o classificador de páginas de acordo com uma das 4 classes de modificação mencionadas anteriormente. Foram utilizados então os algoritmos apresentados anteriormente na Seção 3.3.4 para serem gerados classificadores baseados no conjunto de treinamento.

Para a escolha do melhor classificador que será gerado por cada uma dessas técnicas, foram estipulados os seguintes critérios: (1) alta taxa de acerto, (2) pequeno tempo para classificação das páginas e (3) facilidade de uso em um sistema real, ou seja, que possa ser facilmente embutido em um sistema.

5.5.4.1 Árvore de Indução

Na Seção 3.3.4, é apresentada uma pequena introdução sobre a técnica de árvore de indução ou decisão. Neste estudo, foi utilizado o algoritmo de árvore de indução J48, por ser ele implementado pela WEKA e por ele suportar atributos numéricos, pois a maioria das características da página estudada neste trabalho é do tipo numérica. O único atributo que não é realmente numérico é a existência ou não do cabeçalho *last-modified*, mas este foi transformado em um atributo binário.

Uma das vantagens do uso de árvores de decisão é a interpretabilidade do resultado da classificação. Para que isso ocorra, é necessário que a árvore gerada após a fase de treinamento não possua um número muito grande de nós. As árvores geradas neste trabalho foram muito grandes, tornando muito difícil a sua compreensão. Para tentar minimizar isso, foi sugerido por um dos autores da ferramenta WEKA, o aumento do valor de poda na construção da árvore. Entretanto, isto tornou o processo de construção do classificador mais demorado, e não reduziu consideravelmente o tamanho da árvore, não melhorando assim a sua interpretabilidade.

O grande número de valores possíveis para cada atributo foi a causa de terem sido geradas árvores tão grandes, uma outra solução para melhorar a interpretabilidade dessas árvores seria discretizar os valores desses atributos em faixas. No entanto, isso não foi realizado, porque ao ser feita essa discretização haveria perda de informação e como a interpretabilidade das árvores não é o principal objetivo deste trabalho e sim a sua acurácia, decidiu-se, então, não realizar esse procedimento.

Os resultados do erro de classificação do conjunto de treinamento, utilizando *cross-validation* (10 *folds*) com os diferentes parâmetros para a poda estão apresentados na Tabela 8.

Tabela 8 – Valores relativos ao uso de poda na árvore de indução.

	Erro de re-substituição	**Erro de teste**	**Tamanho da árvore**	**Número de folhas**	**Tempo para teste**
Sem poda	6,7	11,9	1395	698	2.41 s
postpruning	7,1	10,7	741	371	1.63 s

O WEKA só implementa o método de poda *postpruning*, pois esse é mais eficiente que o *prepruning*. Como se percebe pela Tabela 8, a melhor configuração da árvore é a com a poda, pois o erro de teste e o tempo para testar as instâncias foram menores. A partir desta árvore, foi gerada, então, através do WEKA, uma classe Java [30] contendo o código fonte do classificador obtido com a técnica de árvore de indução para que possa ser utilizado por um engenho de busca na classificação das páginas, caso seja esta a técnica escolhida para este fim.

5.5.4.2 Bayesiano Ingênuo

O algoritmo bayesiano ingênuo foi apresentada na Seção 3.3.4. Os resultados obtidos utilizando esta técnica sobre os dados de entrada foram: erro de re-substituição igual a 40,3%, erro de teste 40,5% e tempo para teste 120,5 segundos. Por estes números percebe-se que este tipo de classificador não apresentou um bom resultado. Isto pode ter sido causado por haver dependências entre os atributos e/ou alguns estarem mais correlacionados com as classes do que outros.

5.5.4.3 K-vizinhos mais próximos

A técnica de k-vizinhos mais próximos foi mais uma utilizada para a geração do classificador. Na Seção 3.3.4, foram apresentadas algumas de suas características. As configurações utilizadas e o valores de erro correspondentes a cada uma delas estão apresentadas na Tabela 9.

Tabela 9 - Configurações utilizadas e respectivos valores de erro para o classificador k-vizinhos mais próximos.

	Erro de re-substituição	Erro de teste	Tempo para teste
k=1	1,37	11,27	6335,49 s
k=2	6,2	11,88	4393,15 s

Pelos valores apresentados, com o valor de k igual a 1, obteve-se uma melhor precisão do que k igual a 2, mas um tempo para teste maior. Mesmo com esta pequeno erro de teste, o tempo para testar novas instâncias foi alto se comparado com as técnicas já utilizadas. Por essa razão, o uso deste classificador deve levar em conta esse tempo, pois como este será utilizado em um sistema que classifica centenas de páginas (amostras) por segundo, a velocidade de classificação pode ser um fator limitante de seu uso.

5.5.5 Considerações sobre os Classificadores baseados nas Características das Páginas

Foram apresentados os resultados com o uso de três das principais técnicas de Aprendizagem de Máquina. A seleção da melhor técnica se baseia nos critérios mostrados na Seção 5.5.4:

- A técnica utilizada pelo classificador bayesiano ingênuo apresentou uma taxa de erro de teste bastante alta, sendo, portanto, a primeira a ser desconsiderada;

- O classificador k-vizinhos mais próximos apresentou uma pequena taxa de erro de teste, mas despende muito tempo para classificar uma nova página comparado com outras abordagens;

- A técnica de árvore de decisão, ao contrário, classifica as novas páginas num tempo bem menor. Além disso, sua taxa de erro de teste também foi a menor na configuração com *postpruning* e a além da ferramenta WEKA ter gerado uma classe Java, que é a linguagem utilizada pelo protótipo.

Portanto, o classificador de árvore de decisão foi o que melhor atendeu aos três critérios de escolha de melhor classificador, sendo este utilizado pelo protótipo para atualização da base de índices do engenho de busca.

5.6 Classificador baseado no Histórico das Páginas

Através dos procedimentos que utilizaram técnicas de Aprendizagem de Máquina, apresentados anteriormente, foi criado o classificador de páginas para que, quando o engenho de busca coletasse a página pela primeira vez, já pudesse classificá-la em algum grupo de modificação.

Entretanto, além dessa classificação inicial, é necessário que o engenho continue monitorando o comportamento dessas páginas, pois além desse classificador inicial poder ter cometido erros, é possível que as páginas mudem sua taxa de alteração com o passar do tempo. Com esse objetivo, foi apresentado na Seção 3.4, o estimador bayesiano, que se baseia no método de inferência bayesiano.

Um outro algoritmo para poder classificar a página com o passar do tempo é proposto neste trabalho, o **estimador heurístico**. Nele, para cada classe de alteração existe uma freqüência de visitação para as páginas pertencentes a ela. Após um número **n** de visitas, sendo **n** chamado de tamanho da janela de visitas, é verificado em quantas dessas visitas a página realmente mudou. Com esses dados é calculada a razão entre a quantidade de mudanças pelo tamanho da janela. Se o valor calculado for acima de um limiar máximo pré-estabelecido para essa classe, então a página é classificada na classe

de alteração com freqüência de modificação logo acima desta classe, se esta classe existir, caso o valor for menor que um limiar mínimo, ela é classificada na classe de modificação com menor freqüência, também se esta existir.

Na Figura 11, é apresentado um pseudo-código desse algoritmo, para um exemplo com 4 classes de alteração (DIA, SEMANA, MES, MAIORQUEMES). Por simplicidade assumiu-se, nesse algoritmo, que todas as classes possuíam o mesmo limiar máximo e o mesmo limiar mínimo.

Este método baseia-se no estimador [13], chamado de **estimador de freqüência intuitivo**. A diferença deste método em relação ao estimador de freqüência intuitivo é que, no estimador intuitivo, o valor das mudanças dividido pelas visitas classifica a página em uma das classes determinadas de acordo com limiares pré-determinados. Já no método proposto neste trabalho, dado que uma página já pertença a uma classe, este valor é utilizado somente para avaliar se a página deve continuar na classe ou deslocar-se para uma classe com maior ou menor taxa de modificação ou com menor taxa.

Para adotar o estimador de freqüência intuitivo deve-se ter um histórico grande do comportamento das páginas para que se possa prever a que classe de modificação elas pertencem. Entretanto, para adquirir esse histórico é necessário que se passe um longo tempo monitorando o comportamento das páginas para depois poder classificá-las. Essa pode ser uma restrição para seu uso prático, dado que o processo de atualização da base de índices irá iniciar somente quando as páginas forem classificadas, ou seja, após o término do monitoramento.

O método heurístico, ao contrário do estimador de freqüência intuitivo, parte das páginas já inicialmente classificadas por algum outro processo anterior a ele. A partir desta classificação inicial, o método heurístico é executado, classificando as páginas de acordo com o seu histórico de mudanças. Dessa maneira, não é necessário esperar por um determinado tempo antes de começar-se a classificar as páginas.

```
Entrada          razao: razão entre a quantidade de mudanças pelo tamanho da janela
                 classe: classe atual da página

Procedimento

int DIA = 0;          //inteiro que representa a classe de páginas que mudam todo dia

int SEMANA = 1;  // inteiro que representa a classe de páginas que mudam toda
                 //semana

int MES = 2;          // inteiro que representa a classe de páginas que mudam todo
                      //mês

int MAIOR_QUE_MES = 3;  // inteiro que representa a classe de páginas que
                        // mudam mais que um mês

float LIMIAR_SUPERIOR = 0.8; //limiar superior para o a razão entre a
                             //quantidade de mudanças pelo tamanho da janela

float LIMIAR_INFERIOR = 0.2; //limiar inferior para o a razão entre a
                             //quantidade de mudanças pelo tamanho da janela

se (razao > LIMIAR_SUPERIOR) { // se a razão for maior que o LIMIAR_SUPERIOR, a página
        se (classe > DIA)              // é classificada em uma classe de alteração com maior
                classe = classe -1;  // freqüência de alteração, caso ela já não pertença à classe de
}                                    // maior freqüência (DIA).

senao  se (razao < LIMIAR_INFERIOR) {  // se a razão for menor que o LIMIAR_INFERIOR,
        se (classe < MAIOR_QUE_MES)  // a página é classificada em uma classe de alteração
                classe = classe +1;          // com menor freqüência de alteração, caso ela já não
        }                                    // pertença à classe de menor freqüência
                                             // (MAIOR_QUE_MES).
```

Figura 11 - Exemplo de algoritmo para o classificador heurístico.

A partir desse estimador heurístico foi implementado um classificador na linguagem do protótipo, como será mostrado no Capítulo 6. Foi feito o mesmo para o o bayesiano. A seleção do melhor dos dois estimadores será apresentada na Seção 5.7.

5.7 Composição e Avaliação dos Classificadores

Nesta seção, são avaliados os métodos de classificação apresentados anteriormente e a composição entre eles – classificador inicial e classificador baseado em histórico[20]. A combinação que apresentar o melhor resultado será a utilizada na identificação do comportamento das páginas que é um dos itens que compõe a proposta de atualização (ver Seção 5.2).

A combinação de classificadores é uma abordagem que torna o resultado da classificação mais confiável. Existem vários métodos para isso [43], *bagging* e *boosting* são dois dos mais utilizados. Eles combinam a decisão de várias técnicas de classificação em um único resultado. Na composição proposta neste trabalho, o classificador baseado em uma determinada técnica atribui uma classificação inicial para um elemento, dando um resultado, e depois um outro classificador baseado numa técnica diferente classifica o mesmo elemento, baseando-se na classificação inicial dele, dando um outro resultado, igual ou não ao anterior. Portanto, as técnicas de combinação de classificadores, *bagging* e *boosting*, não apresentam o memso tipo de composição realizada neste trabalho.

Os dados utilizados para esta simulação foi o terço restante das páginas, com suas características e seus históricos, obtido através do monitoramento da *Web* brasileira (28.233 *URLs*).

Para o processo de classificação inicial, foi utilizado o classificador baseado nas características das páginas que utiliza a técnica de árvore de decisão, pelas suas vantagens apresentadas na Seção 5.5.5. Foi usado, também, um classificador aleatório, para que se pudesse realizar uma comparação entre este e o de árvore de indução. Um outro motivo para a utilização deste classificador aleatório nesta avaliação, é que as principais propostas que utilizam classificadores baseado no histórico das páginas ([11], [15]), não fazem nenhuma suposição *a priori* sobre o comportamento das páginas, ou

[20] É importante salientar que vão ser utilizados, nesta simulação, componentes que serão utilizados pelo protótipo, como os classificadores, ou seja, já se está trabalhando com os componentes na linguagem de programação utilizada pelo protótipo.

seja, usam um classificador aleatório para a classificação inicial. Assim, pode-se comparar a proposta de solução deste trabalho, que utiliza inicialmente o classificador baseado nas características das páginas para, a partir desta classificação, utilizar o classificador baseado no histórico de mudanças, com outras propostas que utilizam somente o histórico de mudanças. Serão comparados, também, os classificadores baseados no histórico das páginas apresentados anteriormente: bayesiano e heurístico. Aquele que apresentar o melhor resultado será o adotado na proposta.

Portanto, os principais objetivos desta avaliação são: (1) comparar o desempenho do classificador baseado nas características da página *versus* um classificador aleatório; (2) comparar os dois métodos de classificação baseados em histórico (bayesiano e heurístico); e (3) selecionar a melhor combinação de classificadores, baseado nos resultados obtidos dessas duas comparações.

Para se verificar a precisão, medida através da taxa de acerto de cada classificador, foram realizadas as seguintes etapas:

1. Classificação: como o classificador inicial gerado baseou-se nas 4 classes originadas pelo procedimento de *clustering*, então, durante a sua execução, todas as abordagens deveriam classificar as páginas em cada uma dessas 4 classes.
2. Verificação da taxa de acerto: como os *clusters* apresentados anteriormente foram gerados a partir de valores relativos de um estimador (Equação 14). Os elementos pertencentes a cada *cluster* possuem seus valores relativos ao estimador compreendidos em um determinado intervalo. Assim, os intervalos dos estimadores relacionados a cada um dos 4 grupos são: grupo 0 - [4,488636;∞); grupo 1 – [2,47009;4.488636); grupo 2 - [0,63827;2,47009) ; grupo 3 – [0;0,6382 7). Para uma página pertencer a uma determinada classe, ela deve possuir o valor de seu estimador dentro do intervalo de valores relativos dos estimadores desta classe. Então, para cada página, é gerado, baseado no histórico de 100 dias de monitoramento, o seu estimador relativo à freqüência (Equação 14) e verificado em qual dos intervalos esse valor está compreendido, determinando assim qual a real classe da página. Por exemplo,

para a classe de páginas que mudam num período maior que 100 dias, o intervalo do estimador relacionado à freqüência de alteração é [0;0,63827) e, portanto, para uma página pertencer a esta classe seu estimador deve encontrar-se neste intervalo. Por fim, para saber se a página foi corretamente classificada, verifica-se se a classe real é igual à determinada pela classificação.

Foi também necessária a definição do valor de alguns parâmetros para os classificadores baseados no histórico de mudanças:

- Período de visitação: como cada uma das 4 classes deve possuir o seu período de visitação, foram então definidos os seguintes valores: classe 0 = 1 dia; classe 1 = 3 dias; classe 2 = 30 dias; classe 3 = 96 dias, que se baseia na distribuição da modificação das páginas pertencentes a cada classe apresentada na Tabela 5. Estes parâmetros são utilizados por ambos os classificadores baseados no histórico;

- Tamanho das janelas: para o método heurístico, o tamanho da janela de visitas para cada classe foi: classe 0 =3; classe 1 = 2; classe 2 = 2; classe 3 = 1. Para um melhor entendimento do significado desses valores é dado o seguinte exemplo: o tamanho da janela da classe 1 é igual a 2, o que significa que após a segunda visita a uma página desta classe (cada visita demora 3 dias - período de visitação da classe 1), vai-se verificar se ela ainda pertence a esta classe ou deve migrar para uma outra, ou seja, depois de 6 dias. O critério para a definição do tamanho da janela para cada classe baseou-se, em primeiro lugar, no seu período de visitação. Por exemplo, para a classe 3, que possui um período de visitação de 96 dias, o tamanho da janela é igual a 1, o que significa que a página será classificada no período igual a 96 x 1. Se o tamanho da janela fosse maior, ele seria multiplicado por 96, o que resultaria num período muito longo, mesmo porque o histórico de dados utilizados neste experimento compreende 100 dias. Portanto, quanto maior o período de visitação, menor é o tamanho da janela. Após a definição

dos tamanhos das janelas obedecendo esse critério inicial, foi executada algumas vezes a simulação para que fossem alterados esses valores até se encontrar os tamanhos das janelas com os quais o classificador heurístico obtivesse o melhor resultado.

Uma vez apresentados quais os objetivos dessa avaliação, o processo de verificação de precisão dos classificadores e os parâmetros utilizados por esses classificadores, são apresentados, na Tabela 10, os resultados obtidos para os classificadores iniciais e, na Tabela 11, para cada uma das abordagens. Foram combinadas as duas abordagens para a classificação inicial com as duas baseadas em histórico, gerando 4 configurações.

Pelos números apresentados na Tabela 10, a primeira conclusão é que o classificador utilizando árvore de indução é melhor que um classificador aleatório, o que significa que utilizá-lo na classificação inicial das páginas é uma melhor abordagem do que simplesmente não assumir *a priori* algum comportamento das páginas, como fazem algumas propostas para atualização de base de índices ([11], [15]).

Tabela 10 - Taxa de acerto para os classificadores iniciais.

Técnica	Taxa de Acerto
Aleatório	24,78
Árvore de Indução	74,36

Tabela 11 - Taxa de acerto para cada combinação.

Combinação	Técnica	Taxa de acerto
1	Aleatório + Bayesiano	65,27
2	Aleatório + Heurístico	71,67
3	Árvore de Indução + Bayesiano	62,13
4	Árvore de Indução + Heurístico	85,05

Em todas as configurações apresentadas pelos valores da Tabela 11, o classificador heurístico se mostrou mais eficiente do que o bayesiano, principalmente naquela que usa o classificador de árvore de indução como o ponto de partida.

Como os classificadores bayesiano e heurístico se baseiam no histórico de mudança das páginas para sua classificação, resolveu-se verificar a evolução da

porcentagem de erro desses dois tipos de classificadores com o passar do tempo. Para isso, foi medido, a cada 10 dias de histórico, qual a porcentagem de erro dos classificadores, como pode ser visualizado pela Figura 12. O classificador inicial utilizado foi o aleatório.

Pela Figura 12, percebe-se que o classificador bayesiano, inicialmente, converge mais rápido que o heurístico, por exemplo, em 10 dias, ele já alcançou uma taxa de acerto de 52,59%, enquanto que o heurístico obteve 39,12%. Isso ocorre porque no classificador heurístico uma página que está, por exemplo, no grupo 0 e na verdade pertence ao grupo 3, terá que "passar" pelos grupos 1 e 2 até chegar ao grupo 3. Isso, no entanto, não acontece no classificador bayesiano, pois as páginas podem mudar para classes não adjacentes. Apesar disso, o classificador heurístico mostrou, após 30 dias, uma porcentagem de erro sempre menor do que o bayesiano. Como o tempo de convergência é um fator menos importante do que precisão, pois se pretende utilizar o classificador por um período indeterminado, decidiu-se utilizar o classificador heurístico para a proposta de atualização da base de índices.

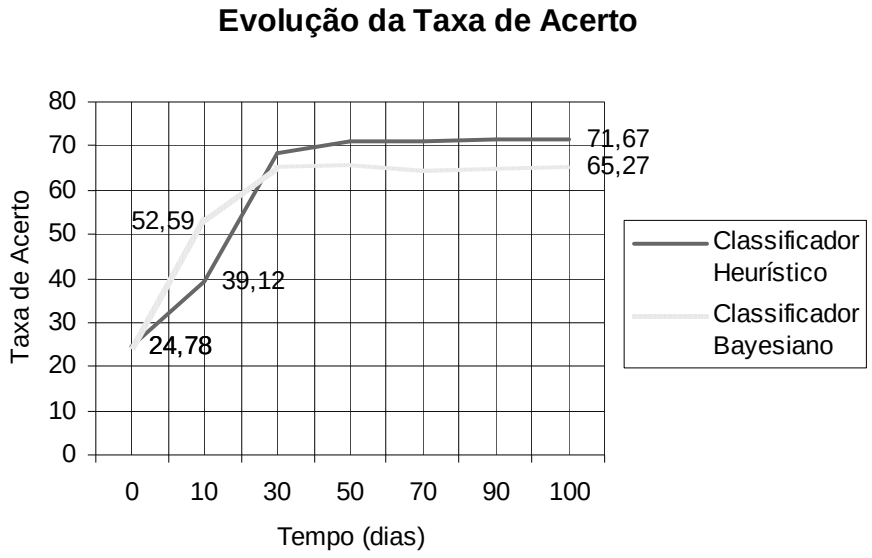


Figura 12 - Gráfico apresentado a evolução da porcentagem de erro dos classificadores.

Portanto, a configuração mais eficiente, ou seja, aquela que classificou melhor as páginas e conseqüentemente, desperdiçará menos recursos, é a configuração 4 da Tabela

11, na qual é utilizado como classificador inicial a árvore de indução e como classificador baseado em histórico, o heurístico. É com esta configuração, que será executado o protótipo.

5.8 Recapitulação

Foi apresentado, neste capítulo, todo o processo para a construção da proposta para a atualização de base de índices. Definiu-se, inicialmente, que seria utilizada uma abordagem ativa e uma política não-uniforme. Como para utilizar a política não-uniforme faz-se necessário que as páginas estejam agrupadas em suas classes de modificação, foram construídos dois classificadores de páginas: um para a etapa inicial, realizada na primeira vez que se indexa a página, na qual não se possui um histórico de alterações da página, e por isso, baseia-se nas características da página para classificá-la, e outro para a segunda etapa, com a classificação da página baseada no histórico de alteração das páginas, quando já se possui essa informação de histórico.

Para a construção do classificador da etapa inicial, foi necessária a implementação de um sistema para a obtenção dos dados utilizados no processo de inferência entre as características de uma página e sua classe de modificação. Esses dados foram obtidos após a execução do monitoramento diário de 84.699 *URLs* da *Web* brasileira, que durou 100 dias.

De posse desses dados, foram utilizadas algumas das principais técnicas de Aprendizagem de Máquina para a obtenção do melhor classificador baseado nas características das páginas. Foi escolhido, então, o classificador que utiliza a técnica de árvore de indução, por ser mais preciso (menor erro de teste), mais rápido na classificação das páginas e pela ferramenta utilizada gerar o código do classificador na linguagem utilizada para a construção do protótipo. Além de ser utilizado para a finalidade de atualização de um engenho de busca, esse classificador pode ser utilizado também por servidores de *cache proxy* para a política de atualização das páginas armazenadas.

Em relação ao classificador baseado no histórico das páginas, foi apresentado um novo método, o estimador heurístico, que de forma simplificada monitora o comportamento da página em sua determinada classe e após um certo número de visitas a ela, decide se esta deve deslocar-se para uma classe de alteração maior ou menor, ou permanecer na mesma classe.

Através de simulações sobre parte dos dados obtidos durante a etapa de monitoramento, mostrou-se que o classificador que utiliza a técnica de árvore de indução apresentou melhores resultados em relação a um classificador aleatório, e que o classificador heurístico melhor do que o bayesiano. Baseado nesses resultados, a proposta de solução é composta pela utilização inicial do classificador baseado em árvore de decisão e do classificador heurístico com o passar do tempo. Será apresentado no Capítulo 6 como esta proposta se comporta em um protótipo.

Capítulo 6 Protótipo

No capítulo anterior mostrou-se como foi construída a proposta para a atualização da base de índices dos engenhos de busca. Para que essa proposta possa ser implementada e implantada por um engenho de busca é necessário que ela seja colocada no sistema de atualização deste engenho. Neste capítulo, inicialmente, será apresentado como foi implementado esse sistema de atualização e como a proposta foi embutida nele (Seção 6.1). Por fim (Seção 6.2), serão mostrados os resultados sobre o uso deste protótipo numa situação real de atualização de base de índices, comparando o seu desempenho com a abordagem tradicional apresentada no Capítulo 3.

6.1 Desenvolvimento do Protótipo

Para a implementação desse protótipo foram utilizadas algumas tecnologias amplamente utilizadas pelo mercado. Serão apresentadas, na Seção 6.1.1, essas tecnologias. Na 6.1.2, são mostrados os componentes que constituem o protótipo e como eles se relacionam. Finalmente, na Seção 6.1.3, apresenta-se o componente responsável por prover a política de atualização do sistema, fornecendo o suporte à proposta de atualização apresentada neste trabalho .

6.1.1 Tecnologias Utilizadas

- Java [30]: Apesar de Java ser uma linguagem orientada a objetos bastante utilizada mundialmente e possuir um grande conjunto de bibliotecas que

facilitam a construção de aplicativos distribuídos e para a *Web*, o principal motivo deste sistema de atualização ter sido implementado em Java foi o fato de que ele é uma extensão de uma sistema de busca já existente conhecido como *SiteSearch* que é implementado em Java.;

- Objetos distribuídos: a tecnologia de objetos distribuídos nasceu da união de duas outras tecnologias: a orientação a objetos com a tradicional arquitetura cliente/servidor, a fim de unir as vantagens de cada uma para o desenvolvimento de aplicações distribuídas. Objetos distribuídos baseiam-se no conceito de *middleware.* O *middleware* é uma categoria de *software* que permite a conexão entre aplicações distribuídas buscando transparência nas comunicações, ou seja, abstraindo complexidades sobre protocolos de rede, sistemas operacionais ou linguagens de implementação entre as aplicações que trocam informações [41]. Na construção de aplicações distribuídas com objetos distribuídos, as aplicações são representadas por conjuntos de objetos, entidades que encapsulam e servem como unidades básicas de informação para estas aplicações. Tais objetos interagem entre si através da troca de mensagens, no estilo pedido/resposta, tradicional da arquitetura cliente/servidor, porém sem a necessidade de objetos estarem localizados em uma mesma aplicação, nem em uma única máquina, deixando as tarefas de localização dos objetos e tráfego de requisições para um *middleware*. A implementação de objetos distribuídos utilizada foi RMI[21]. RMI (*Remote Method Invocation*) é um dos mecanismos que a linguagem Java possui para permitir a chamada de procedimentos em objetos distribuídos. A escolha por RMI como a plataforma de objetos distribuídos decorreu por vários fatores: a abstração de objetos que Java possibilita; o conhecimento existente sobre essa tecnologia; e a sua dependência exclusiva da linguagem Java [16].

[21] *Home page* de RMI: http://*www*.java.sun.com/rmi

6.1.2 Arquitetura

O protótipo utilizando a proposta de solução foi desenvolvido a partir da arquitetura do sistema do engenho de busca do Radix, conhecido como *SiteSearch*, mais especificamente do seu Módulo de Indexação. Inicialmente, este engenho possuía a seguinte arquitetura entre os componentes do Módulo de Indexação (Figura 13):

- Servidor de *Links*: é responsável pelo fornecimento de *links* a serem processados pelo Robô. É neste servidor que é implementada a política de atualização de páginas existentes na base de índices. A única política utilizada é a uniforme, devido à sua facilidade de implementação;

- Robô: obtém os *links* no Servidor de *Links*, realiza o *download* do documento referenciado por esse link, e envia-o ao Armazenador de Documentos. Além disso, envia os *links* presentes no documento recuperado para o Armazenador de *Links*;

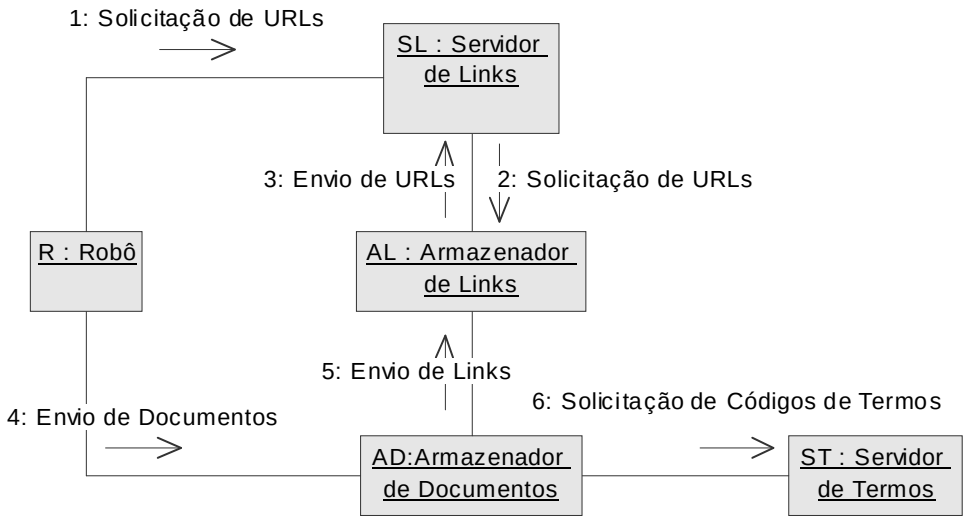


Figura 13 - Diagrama de colaboração do *Site*Search.

- Servidor de Termos: responsável pela manipulação dos termos existentes nos documentos. Realiza a conversação de termos para códigos, usados para

a construção das estruturas de arquivos invertidos (Apêndice B) utilizadas pela busca;

- Armazenador de Documentos: armazena os documentos processados pelo Robô na base de índices. É nele que são criados os arquivos invertidos das páginas. Para isso, ele se comunica com o Servidor de Termos para obter os códigos dos termos. Ele armazena também informações como título e resumo da página que são apresentados como resultados às consultas do usuário;

- Armazenador de *Links*: armazena as *URLs* processadas pelo Robô, para que depois possam ser utilizadas pelo Servidor de *Links*, enviando-as ao Robô.

A única alteração realizada neste sistema já existente, para a construção do protótipo, foi a retirada da política de atualização do Servidor de *Links* para um novo componente do sistema, que trataria exclusivamente desta atividade. Este componente foi chamado de Servidor de Atualização. Este servidor foi implementado para que pudesse ser utilizada qualquer política de atualização, tanto a uniforme quanto a não-uniforme.

Nessa nova arquitetura, a interação do Servidor de Atualização com os outros componentes é a seguinte:

1. Envio de páginas: quando o Robô processa uma página, ele a envia ao Servidor de Atualização para que ela possa ser classificada em algum grupo de atualização, caso se esteja utilizando uma política não-uniforme. No caso da uniforme haverá apenas um único grupo de atualização;
2. Seleção de *URLs*: para realizar a atualização de um determinado grupo, o Servidor de *Links* solicita ao Servidor de Atualização as *URLs* pertencentes a esse grupo para que aquele possa enviá-las ao Robô.

A nova arquitetura do sistema com este novo componente está apresentada na Figura 14. Esta nova arquitetura, portanto, tornou o sistema mais modular, como também, tornou possível a utilização de qualquer política de atualização para a BI.

Detalhes de como foi implementado o Servidor de Atualização serão apresentados na próxima seção.

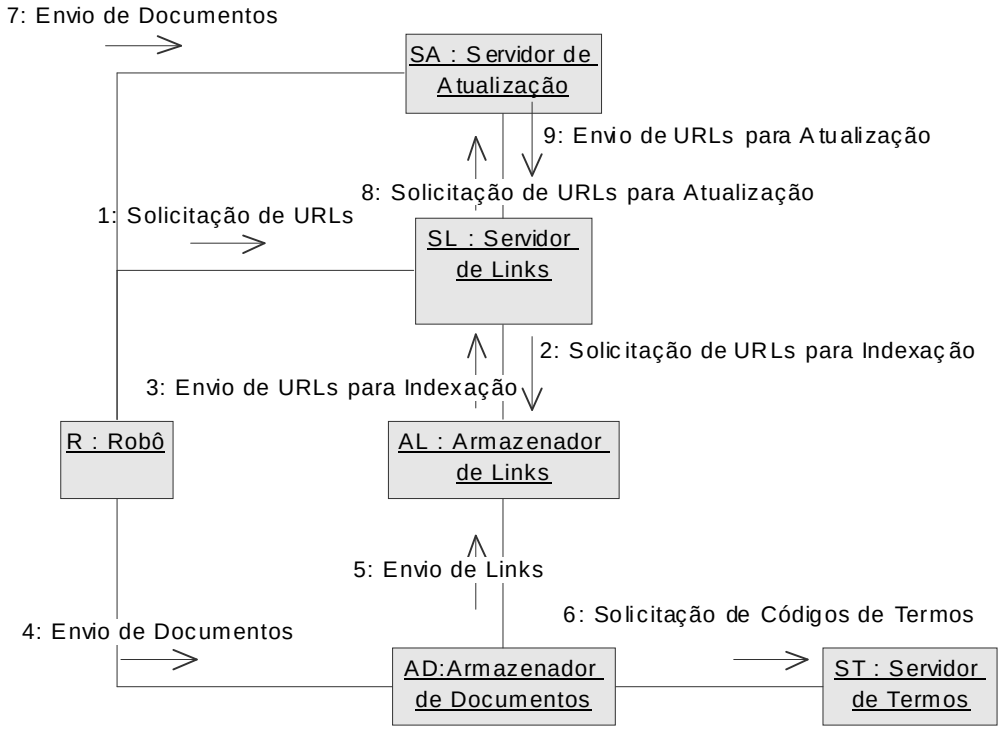


Figura 14 - Diagrama de colaboração da nova arquitetura do *Site*Search.

6.1.3 Servidor de Atualização

O Servidor de Atualização implementa as políticas de atualização utilizadas para a atualização da base de índices de um engenho de busca. Alguns detalhes estruturais e funcionais deste servidor são apresentados nesta seção.

6.1.3.1 Composição

Com o objetivo de implementar a proposta de solução, é necessário agrupar as páginas em suas respectivas classes de alteração. Esta tarefa é realizada pelas classes apresentadas no diagrama de classes da Figura 15 .

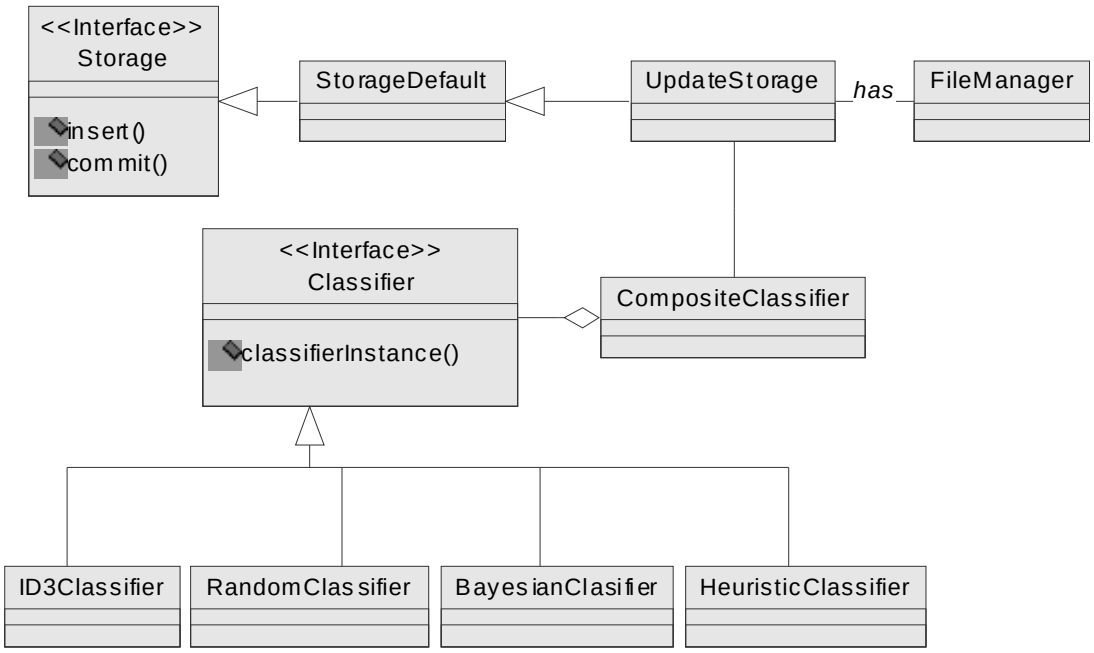


Figura 15 - Diagrama das principais classes do Servidor de Atualização.

A interface *Classifier* define que todo classificador a ser implementado deve implementar o método *classifierInstance* que recebe uma instância a ser classificada e retorna um número natural, identificando a que classe esta instância pertence[22]. Todos os classificadores implementaram a interface *Classifier* (Figura 15). Assim, os classificadores implementados são:

- *ID3Classifier* - implementa o classificador baseado na técnica de árvore de indução apresentada na Seção 3.3.4;

- *RandomClassifier* - classifica as instâncias de forma aleatória nas n classes existentes;

- *BayesianClassifier* - a classificação é realizada de acordo com o histórico das páginas, utilizando o estimador bayesiano apresentado na Seção 3.4;

- *HeuristicClassifier* - também é baseado no histórico de alteração das páginas, mas utiliza o estimador heurístico, introduzido na Seção 5.6;

- *CompositeClassifier* - classificador composto por dois classificadores, um que classifica páginas novas e outro que classifica páginas já existentes na base de índices.

Existem ainda as classes:

- *FileManager*: tem como tarefa recuperar todas as páginas classificadas em uma dada classe;

- *UpdateStorage*: implementa as tarefas do Servidor de Atualização. Ela utiliza o *CompositeClassifier* para realizar a classificação das páginas. Quando instanciada, essa classe se torna um servidor, para que possa ser acessada pelo Robô e Servidor de *Links* (ver Figura 14).

6.1.3.2 Funcionamento

O Servidor de Atualização possui duas operações básicas:

- *insert* - recebe do Robô a página indexada (objeto do tipo *Page*) a ser classificada em categorias. São classificadas tanto páginas novas, quanto páginas já existentes na base de índices. A classe *CompositeClassifier* se encarrega de decidir se a página é nova ou não, enviando-a para o classificador responsável pela sua classificação, para isso, transforma o objeto do tipo *Page* em um do tipo objeto *Instance*. É apresentado, na Figura 16, o diagrama de seqüência para a operação com uma página nova e, na Figura 17, para uma página já existente;

[22] Por exemplo, para o estudo de caso que será apresentado posteriormente, o número 0 representava a classe das páginas que mudavam todo dia, o número 1 que mudavam de 3 em 3 dias, o número 2 que mudavam de 30 em 30 e o número 3 de 96 em 96 dias.

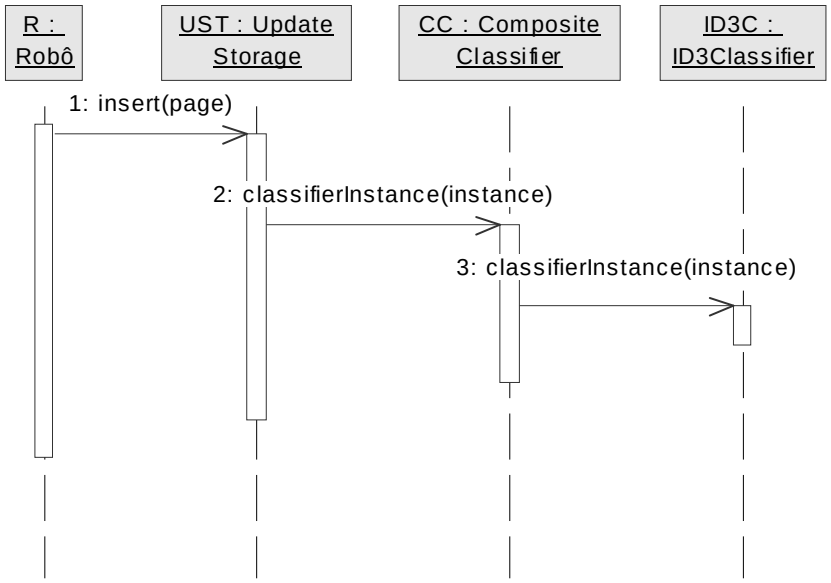


Figura 16 - Diagrama de seqüência para a operação de classificação de uma nova página.

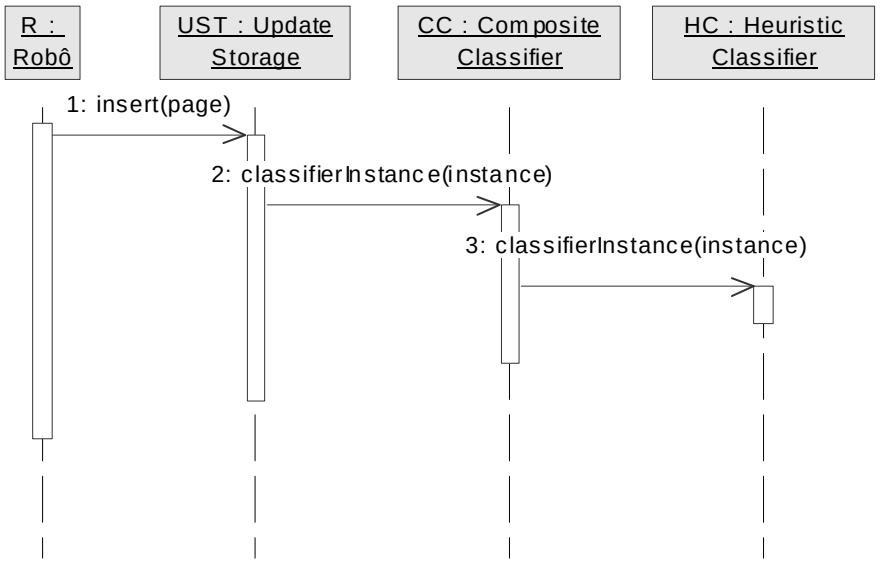


Figura 17 - Diagrama de seqüência para a operação de classificação de uma página já existente.

- *commit* - envia as *URLs* de determinada classe para o Servidor de *Links*. Para isso, são selecionados pelo *FileManage*r os códigos das páginas pertencentes à classe pedida pelo Servidor de *Links*. A partir desses códigos

recuperam-se as URLs que as representam e elas são enviadas então para o Servidor de Links, através do método *AddResource*. Na Figura 18, é apresentado o diagrama de seqüência dessa operação.

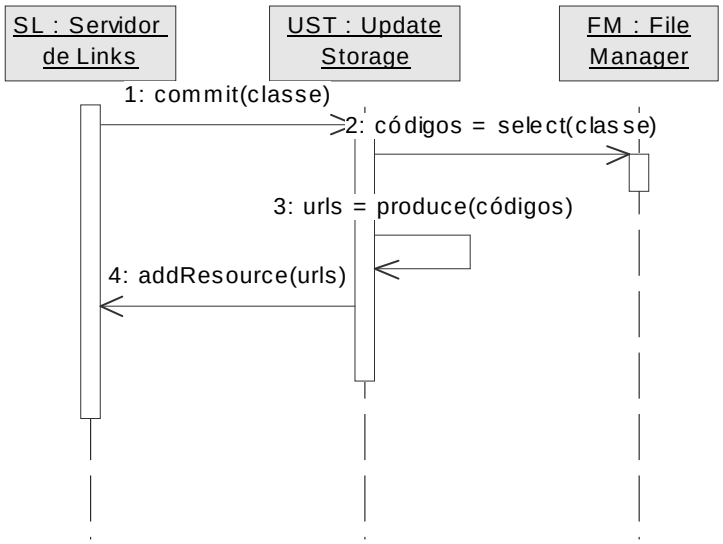


Figura 18 - Diagrama de seqüência do operação *commit*.

6.2 Estudo de Caso

Uma vez apresentado a composição e o funcionamento do sistema de atualização de base de índices, deseja-se nesta seção, apresentar algumas medidas obtidas na execução deste sistema referentes ao processo de atualização.

Foram configurados dois cenários para a execução do estudo de caso. No primeiro, o sistema foi configurado para utilizar a política uniforme de atualização (Seção 2.5.1) e, no segundo, para utilizar a proposta de atualização sugerida neste trabalho. A escolha desses dois cenários foi devido a se querer comparar a eficiência da versão anterior do sistema *SiteSearch*, que utilizava a política uniforme, *versus* a da nova versão com as alterações apresentadas anteriormente.

Os recursos utilizados para a execução deste estudo de caso, como máquinas e rede, estavam sendo compartilhados por outros processos, pois ele foi executado na infra-estrutura da empresa Mobile. Por esta razão, decidiu-se executá-lo diariamente,

numa janela de tempo de 4 horas, das 0h às 4h, que é o horário de menor utilização das máquinas, e limitou-se o período máximo para a execução do estudo de caso de 1 mês.

Para ambos os cenários, o número total de *URLs* utilizadas para o estudo de caso foi 300.000. Desse total, cerca de 90.000 eram visitadas diariamente. Estas *URLs* utilizadas foram originadas de uma busca em largura a partir das páginas iniciais de 1.000 servidores, obtidos de forma aleatória, pertencentes à base de índices do Radix.

Para realizar uma comparação entre as abordagens, foi utilizada a taxa de mudança, ou seja, verificou-se qual a porcentagem do total de *URLs* visitadas haviam sido realmente modificadas.

Não foi utilizada a medida da atualidade nos dois cenários, pois a atualização da base de índices, em ambos, foi iniciada um dia após ela ter sido populada. Isso significa que os valores da atualidade das bases de índices, nos dois cenários, estavam bem próximos no início do processo de atualização. Se esse valor estivesse diferente, provavelmente a taxa de mudança do que estivesse atualizando a base de índices mais "antiga" seria maior. Portanto, como os valores da atualidade dos índices estão parecidos e eles visitam a mesma quantidade de páginas por dia, o cenário que apresentar maior taxa de mudança terá a base de índices mais atualizada.

6.2.1 Uso da Política Uniforme

A política uniforme caracteriza-se por atualizar todas as *URLs* da base de índices à mesma taxa, o que significa, que há apenas um grupo a ser atualizado. Portanto, durante o processo de classificação inicial das *URLs* elas foram classificadas como pertencentes à mesma classe. Após isso, executou-se durante 15 dias o processo de atualização das páginas de acordo com a política uniforme. É apresentado, na Figura 19, a evolução da taxa de mudança nesse período.

Por ela, percebe-se que houve uma variação na taxa de mudança ao longo do tempo. Pode-se considerar esse um resultado normal, pois em certos dias o processo de atualização pode visitar uma proporção de páginas mais dinâmicas do que em outros dias. O valor médio da taxa de mudança durante esses 15 dias foi de 19,9%.

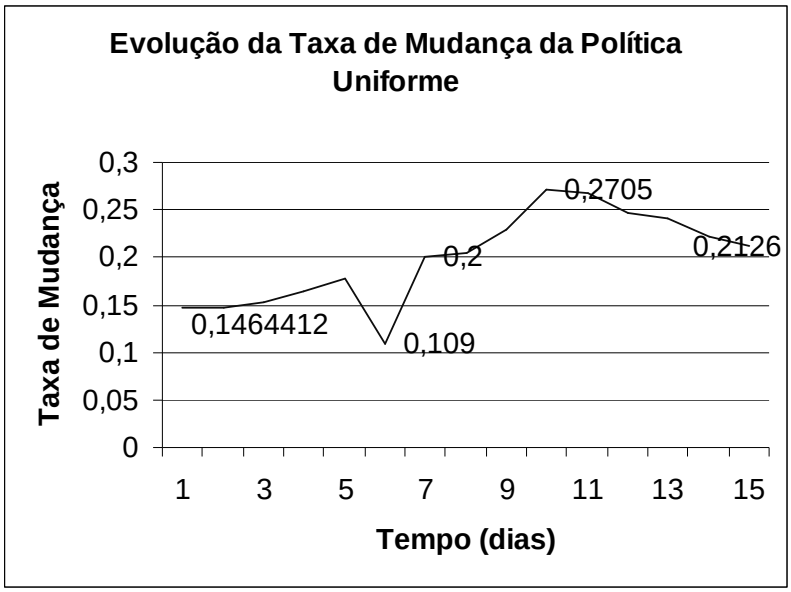


Figura 19 - Evolução da taxa de mudança utilizando-se a política uniforme.

6.2.2 Uso da Proposta de Atualização

O principal objetivo na execução desse cenário é, além de realizar a comparação com a política uniforme, verificar o comportamento da proposta de atualização apresentada neste trabalho. Como mencionado anteriormente, essa proposta caracteriza-se por possuir um classificador inicial, que utiliza a técnica de árvore de indução, e um classificador heurístico, baseado no histórico de alteração das páginas, para classificar as páginas com o passar do tempo.

6.2.2.1 Classificador Inicial

Inicialmente, vai-se fazer uma análise sobre o comportamento do classificador inicial. Como foi citado anteriormente, o número total de *URLs* a serem classificadas nos 4 grupos de alteração[23] apresentados na Seção 5.5.1, era de cerca de 300.000. Elas foram, então, classificadas nesses grupos. Após essa classificação, o número de elementos nesses 4 grupos estão distribuídos como apresentado na Tabela 12.

[23] Grupo 0 representa páginas que mudam todo dia, grupo 1, páginas que mudam a cada 3 dias, grupo 2, a cada 30 dias e grupo 3, páginas que mudam num período maior que 3 meses.

Tabela 12 - Distribuição das páginas pelos grupos após a classificação inicial.

	total de páginas	porcentagem do total
grupo 0	32.875	11%
grupo 1	12.377	4%
grupo 2	17.732	6%
grupo 3	231.080	79%

Ao ser executado o processo de atualização sobre esses grupos, cujas as páginas foram classificadas pelo classificador inicial, obteve-se, no seu primeiro dia, uma taxa de acerto de visitação de 29,6%, como pode ser verificado na Figura 20. Por se achar este valor pequeno, resolveu-se fazer um estudo sobre o comportamento do classificador inicial.

A primeira avaliação que se deseja fazer é verificar se esse classificador inicial possui poder de classificação, ou seja, se o seu desempenho é melhor do que um classificador aleatório. Para isso, foram realizados experimentos em paralelo à execução do processo de atualização com o objetivo de comparar o classificador inicial com o aleatório. Com esse objetivo, foi realizada a classificação das mesmas páginas (cerca de 300.000) utilizadas pelo classificador inicial baseado em árvore de indução, utilizando-se o classificador aleatório.

De posse desses dois conjuntos de páginas classificadas (árvore de indução e aleatório), deseja-se identificar a distribuição das páginas que mudam todo dia nos grupos de alteração em cada uma dessas configurações. Foram escolhidas páginas que mudam todo dia para esse estudo, devido ao seu processo de monitoramento ser mais rápido do que ao de páginas pertencentes aos outros grupos, pois, por exemplo, para páginas que mudam todo mês, seria necessário um monitoramento de meses para se obter algum resultado.

Dessa maneira, foram monitoradas, durante 10 dias, amostras contendo 30%[24] das páginas de cada um dos 4 grupos de cada conjunto de páginas classificadas (árvore de indução e aleatório), para verificar como estava a distribuição de páginas que mudam

todo dia nos seus grupos de alteração. A distribuição dessas páginas para o classificador inicial está na Tabela 13 e para o classificador aleatório na Tabela 14.

Tabela 13 - Distribuição das páginas modificadas diariamente para o classificador aleatório.

	Total de páginas monitoradas	Páginas que mudam diariamente
grupo 0	25.670	2.042
grupo 1	2.382	223
grupo 2	2.237	291
grupo 3	51.152	4.045

Tabela 14 - Distribuição das páginas modificadas diariamente para o classificador inicial (árvore de indução).

	Total de páginas monitoradas	Páginas que mudam diariamente
grupo 0	25.670	6.161
grupo 1	2.382	173
grupo 2	2.237	12
grupo 3	51.152	255

A partir dos valores de distribuição das páginas, foram calculadas a matriz de confusão para cada configuração de classificação para poder compará-las. Como só foram monitorados os elementos que mudam todo dia, só se pode identificar esses elementos e aqueles que mudam num período maior que 1 dia. Foi feita, então, a matriz de confusão com duas classes: *igual a 1 dia* (= 1 dia), que representa as páginas que mudam todo dia, e *maior que 1 dia* (> 1 dia), para páginas que mudam com freqüência maior que 1 dia. Na

Tabela 15, está apresentada a matriz para o classificador aleatório e, na Tabela 16, para o classificador de árvore de indução.

[24] Foram monitoradas 30% das páginas pois essa era a capacidade máxima diária de monitoramento, ou seja, o sistema só conseguia visitar cerca de 90.000 páginas por dia dentro da janela de tempo de execução, como mencionado anteriormente.

Tabela 15 - Matriz de confusão para o classificador aleatório.

		classificado	
		= 1 dia	> 1 dia
verdadeiro	= 1 dia	2.042	4.559
	> 1 dia	23.628	51.212

Tabela 16 - Matriz de confusão para o classificador inicial (árvore de indução).

		classificado	
		= 1 dia	> 1 dia
verdadeiro	= 1 dia	6.161	440
	> 1 dia	19.509	55.331

Com os dados obtidos nas matrizes de confusão foram calculados alguns valores que medem a eficiência de um classificador (Seção 3.5), que estão apresentados na Tabela 17.

Tabela 17 - Medidas de qualidade dos classificadores.

	Aleatório	**Árvore de indução**
Acerto	0,654	0,755
Cobertura	0,309	0,933
Especificidade	0,684	0,739
Precisão	0,079	0,24
Valor de predição negativo	0,918	0,992
F-measure	0,125	0,381

Pelos números apresentados, pode-se concluir:

1. O valor da precisão do classificador que utiliza a técnica de árvore de indução é baixo, o que significa que ele classificou muitas páginas que não são da classe "= 1 dia" como sendo desta classe. Em compensação, seu valor de cobertura foi alto, pois 93,3% dos elementos que mudavam todo dia foram classificados na classe "= 1 dia". Mesmo assim o valor do *f-measure* do classificador foi baixo;
2. O classificador baseado em árvore de indução possui um poder de classificação melhor que um classificador aleatório, pois em todas as

medidas, ele obteve teve um melhor resultado do que o classificador aleatório, ou seja, valores mais próximos a 1.

Uma última observação sobre os números obtidos neste experimento, é em relação aos dados deste experimento comparados com os de outro experimento apresentados na Tabela 10. Naquele experimento, as páginas foram classificadas em 4 classes, enquanto que neste em apenas duas. Sendo essa, por exemplo, a causa da discrepância de valores entre a taxa de acerto do classificador aleatório nos dois experimentos.

6.2.2.2 Classificador Heurístico

Após o processo de classificação inicial, durante 12 dias, o protótipo foi executado utilizando o classificador heurístico baseado no histórico das páginas. Não foram atualizadas as páginas do grupo 3, por elas serem alteradas num período maior que 2 meses, que é superior à duração desse experimento.

Como apresentado anteriormente, o classificador heurístico possui os seguintes parâmetros: taxa de visitação de cada grupo, que identifica a periodicidade de visitação do engenho de busca para as páginas pertences a este grupo, e tamanho de janela, que representa após quantas visitações a uma determinada página pertencente a um grupo, será verificada se ela realmente pertence a essa classe ou não. Os valores desses parâmetros para a execução deste estudo de caso está apresentado na Tabela 18 e são os mesmos utilizados na Seção 5.7.

O processo de atualização foi executado diariamente. Para cada página do grupo 2 visitada diariamente, foram visitadas 8 do grupo 1 e 20 do grupo 0. Essa relação baseou-se na quantidade de páginas de cada grupo deve ser atualizada diariamente. Por exemplo, para o grupo 0, esse valor é igual a 32.875, para o grupo 1 igual a 4.165 e para o grupo 2, 591. Essa quantidade é calculada dividindo-se o número total de elementos do grupo (Tabela 12) pela taxa de visitação do grupo (Tabela 18).

Tabela 18 - Parâmetros utilizados para o classificador heurístico.

	Taxa de visitação	**Tamanho da janela**

grupo 0	1 dia	3
grupo 1	3 dias	2
grupo 2	30 dias	2
grupo 3	96 dias	1

A taxa de mudança média para o período de 12 dias foi de 46,2%. A sua evolução está apresentado no gráfico da Figura 20. Algumas informações podem ser extraídas deste gráfico:

- A taxa de mudança cresceu com o passar do tempo, iniciando com um valor inicial de 29,6% alcançando um valor final de 73%;

- De três em três dias a taxa de mudança realiza um salto. A causa disso é a atuação do classificador heurístico migrando páginas que estão no grupo 0 e que não pertencem a esse grupo, que pela Tabela 14 é um número grande de elementos. Isto significa que aquelas páginas que, por exemplo, não foram modificadas nesses 3 dias (tamanho da janela do grupo 0) migraram para o grupo 1. A seguir, serão apresentados dados mais detalhados sobre esse comportamento.

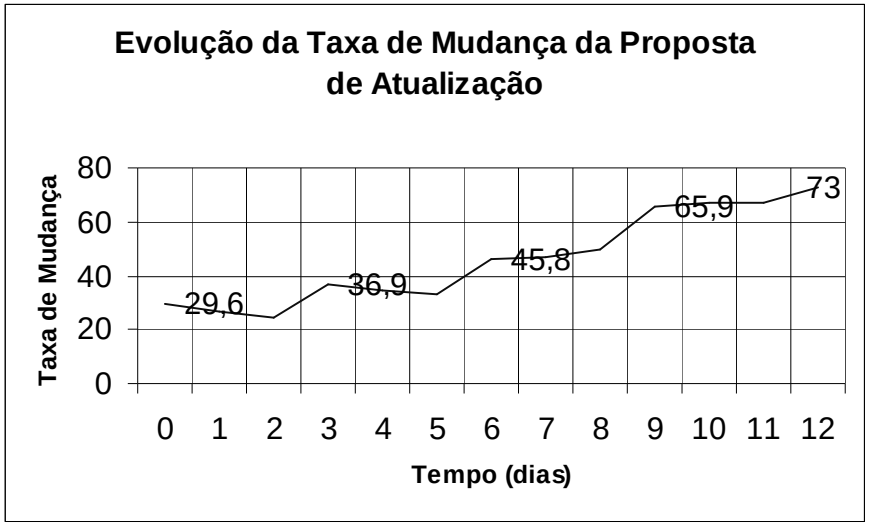


Figura 20 - Evolução da taxa de mudança do processo de atualização em relação ao tempo.

Uma apresentação mais detalhada da evolução da taxa de mudança de cada grupo está apresentada na Figura 21. Nela, pode-se perceber que as taxas de mudança dos

grupos 0 e 1, estão bem próximas no começo, mas com o passar do tempo a do grupo 0 cresce, de 32,3% a 75,7%, e a do grupo 1 diminui, saindo de 28,8% para 8%. A razão para o comportamento da taxa do grupo 0 é que, inicialmente, muitas páginas não pertencentes a este grupo estão nele, mas à medida que o classificador vai atuando, essas páginas vão sendo migradas para o grupo 1. A taxa é alta do grupo 1, no começo, porque esse grupo possui páginas muito dinâmicas, que vão migrando para o grupo 0, após 6 dias, e posteriormente, com o deslocamento dessas páginas e a migração das páginas menos dinâmicas vindas do grupo 0, a taxa de mudança desse grupo diminui, mesmo com a migração de páginas menos dinâmicas desse grupo para o grupo 2.

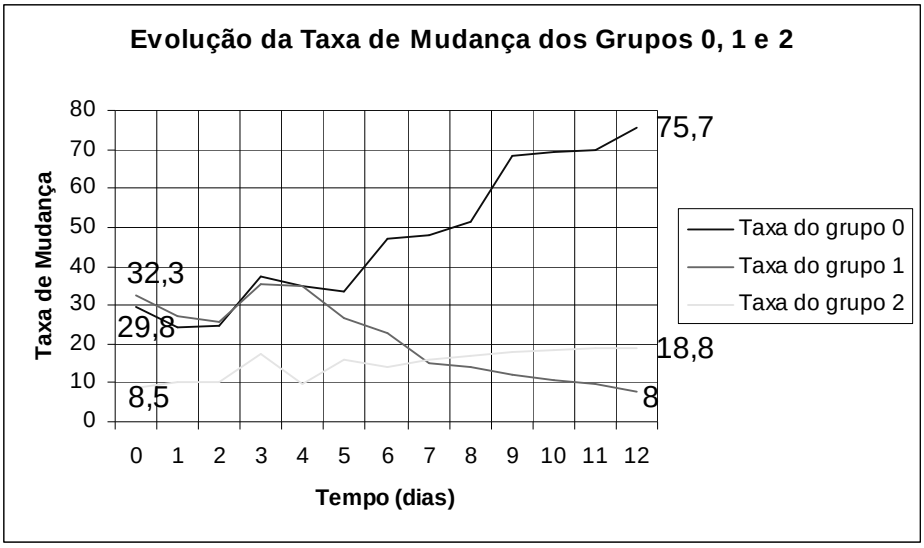


Figura 21 - Evolução da taxa de mudança dos grupos.

Quanto ao grupo 2, verifica-se que a sua taxa de mudança vai aumentado com o passar do tempo. Mas esse resultado não foi causado pela ação do classificador, pois como o tempo do experimento foi de 12 dias e, portanto, não houve tempo para que as páginas menos dinâmicas desse grupo migrassem para o grupo 3, e nem para as que vieram do grupo 1 pudessem entrar no seu processo de atualização, já que elas só serão atualizadas no início do próximo ciclo de atualização desse grupo que dura 30 dias (tamanho da janela). O aumento da taxa para esse grupo, portanto, pode ser considerada uma variação normal dentro do processo de atualização, da mesma forma que o demonstrado pela política uniforme na seção anterior.

Como não estava sendo atualizado o grupo 3, uma preocupação era a existência de elementos dinâmicos nesses grupos. Mas, pelo experimento apresentado anteriormente (6.2.2.1) sobre a distribuição de elementos do grupo 0 nos outros grupos, pode-se assumir que essa estratégia de não atualizar o grupo 3, não está prejudicando a atualidade da base de índices. O contrário seria verdade, ou seja, se houvesse uma grande quantidade de elementos dinâmicos nesse grupo.

6.2.3 Considerações

A política uniforme tem como principal qualidade a facilidade de implementação, sendo esta a razão dela ser bastante utilizada na prática. Entretanto pelos números apresentados neste estudo de caso, ela apresentou uma baixa taxa de mudança média ao longo do tempo se comparado com a proposta deste trabalho. Esse resultado já era de certa forma esperado, pois esta política é mais indicada para atualizar páginas que possuem taxas de alteração parecidas. Esse, entretanto, não é o comportamento das páginas da *Web* brasileira como mostrado na Seção 5.4, nem o de toda a *Web* pelos números apresentados por [11].

No aspecto geral, a proposta de atualização apresentou um crescimento de sua taxa de mudança. Isso demonstra que o classificador heurístico, com o passar do tempo, vai aprendendo o comportamento das páginas, colocando-as na classe de modificação que mais se adequa à taxa de alteração delas, como pôde ser visto pelo crescimento da taxa de mudança para o grupo 0. Entretanto, devido às restrições de tempo, a proposta de solução não pôde ser executada até o momento em que a classificação chegasse num ponto de convergência, ou seja, as páginas já tivessem sido migradas para suas classes de modificação corretas.

Outro ponto a ressaltar nesse estudo de caso é o comportamento do classificador inicial que utiliza a técnica de árvore de indução. Pelo experimento feito sobre as páginas que são modificadas diariamente, verificou-se que elas, em sua maioria, foram classificadas nos grupos mais dinâmicos. Entretanto, nesses grupos, havia também muitas páginas pouco dinâmicas, sendo essa a razão pela qual, para o grupo 0, por exemplo, a taxa de mudança fosse baixa inicialmente (32,3%). O classificador

heurístico atuou, então, nesses grupos migrando os elementos menos dinâmicos rapidamente para seus respectivos grupos, fazendo com que a taxa de mudança do grupo 0 aumentasse. Se, no entanto, as páginas mais dinâmicas tivessem sido classificadas em sua maioria nos grupos menos dinâmicos, elas demorariam muito a serem colocadas em seus grupos corretos, pois o intervalo de tempo para a visitação nesses grupos menos dinâmicos é maior. Por exemplo, se uma página que muda diariamente for classificada no grupo 3, ele irá demorar 96 (tempo para classificação no grupo 3) + 30 (tempo para classificação no grupo 2) + 6 (tempo para classificação no grupo 1) = 132 dias para migrar para o seu grupo correto.

O exposto no parágrafo anterior ilustra uma das características do classificador heurístico: a migração mais rápida das páginas pouco dinâmicas que estão em grupos mais dinâmicos para o seu grupo correto do que o inverso. A razão para esse comportamento é que na classificação nos grupos mais dinâmicos, o tempo para a classificação é menor do que nos menos dinâmicos. Por exemplo, de acordo com os parâmetros do estudo de caso, para o grupo 0, após 3 dias, uma página que não tenha sido modificada migra para o grupo 1, enquanto que, uma página que é modificada diariamente, que se encontra no grupo 3, só irá migrar para o grupo 2, após 96 dias.

Como consideração final, tem-se o fato de que com o uso da proposta de solução foram desperdiçados menos recursos do engenho de busca, do que com a utilização de uma proposta tradicional de atualização, já que se obteve uma taxa de mudança maior. Isso acarreta numa base de índices como atualidade maior e, conseqüentemente, uma melhor qualidade de resposta para os usuários que realizam buscas nesse engenho de busca.

Capítulo 7 Conclusões

7.1 Realizações

A Internet é um meio de comunicação que vem sendo disseminado pela sociedade e com isso ganhando uma importância cada vez maior. Como resultado disso, ela vem concentrando uma grande quantidade de informação. Atualizar essa informação tem sido um desafio para os engenhos de busca por causa do volume e dinamicidade de seu conteúdo.

O principal objetivo deste trabalho foi apresentar uma proposta para o problema de atualização da base de índices de engenhos de busca. Para melhor contextualizar o domínio desse problema, foram apresentados alguns conceitos importantes sobre a área de Recuperação de Informação e quais são os principais módulos de um engenho de busca. Como a proposta se baseia na utilização de técnicas de Aprendizagem de Máquina e estatística, foram introduzidos alguns conceitos e técnicas relativos a essas áreas.

Foram apresentados, também, os principais trabalhos na área de atualização de base de índices de mecanismos de busca, indicando suas qualidades e falhas. Baseado nesse levantamento, elaborou-se uma proposta de solução que atacou os principais problemas dessas abordagens, mas tentando utilizar o seus pontos fortes.

Essa proposta baseia-se na utilização da política não uniforme para atualizar as páginas de acordo com suas taxas de modificação. Para isso, é necessário classificar as

páginas em classes que possuem taxas de modificação semelhantes. Essa tarefa foi realizada por classificadores construídos a partir dos dados obtidos do monitoramento do comportamento das páginas *Web* em relação a sua taxa de modificação, utilizando-se técnicas de Aprendizagem de Máquina e estatísticas, como mencionado anteriormente.

Foram criados, então, dois tipos de classificadores: um primeiro baseado nas características das páginas, para ser utilizado na primeira vez que se obtém a página, quando não se conhece o comportamento da página, através de seu histórico de mudanças; e um segundo, baseado no comportamento do histórico da página, que utiliza estimadores estatísticos para realizar a classificação.

Para o classificador baseado nas características da página foi escolhido o que utiliza a técnica de árvore de indução, por apresentar menor erro de teste e maior velocidade de classificação em relação ao bayesiano ingênuo e k-vizinhos mais próximos. Para o classificador baseado em histórico, selecionou-se o classificador heurístico, que foi proposto por este trabalho, por apresentar maior taxa de acerto do que o bayesiano, baseado numa simulação realizada com os dois algoritmos.

Este trabalho não se limitou a apenas apresentar aspectos conceituais dessa proposta. Para demonstrar sua viabilidade em um ambiente real foi construído um protótipo. Com esse protótipo foi realizado um estudo de caso que mostrou seu bom desempenho frente à política uniforme, que por ser mais fácil de ser implementada é mais utilizada para atualização da base de índices. Nesse estudo de caso verificou-se que o classificador inicial não apresentou uma alta taxa de acerto, mas apresentou um poder de classificação melhor do que um classificador aleatório, pelos melhores valores apresentados nas medidas de avaliação de classificadores. Por fim, o classificador baseado no histórico de alteração das páginas, conseguiu obter uma taxa de mudança de 73%, um valor muito melhor do que o obtido pela política tradicional de atualização.

7.2 Contribuições

Existem alguns trabalhos [4], [44] que tratam da relação entre as áreas de Aprendizagem de Máquina e Recuperação de Informação, mas eles focam mais no problema relativo

ao melhoramento da qualidade da resposta do engenho de busca, em relação ao ranqueamento das páginas retornadas na resposta, a fim de que aquelas mais relevantes para o usuário apareçam melhores ranqueadas, utilizando técnicas de Aprendizagem de Máquina.

Neste trabalho, também, é feito o uso de técnicas de Aprendizagem de Máquina em aplicações de Recuperação de Informação, mas o objetivo de sua utilização é diferente do citado anteriormente. Algumas dessas técnicas (*clustering*, seleção de atributos e métodos de classificação) fizeram parte da construção de uma proposta de solução que tornou mais eficiente o processo de atualização da base de índices de um engenho de busca, desperdiçando menos recursos do que outras abordagens já existentes de atualização (economia em média de 50% em relação a uma política tradicional, durante a execução dos experimentos), e melhorando, conseqüentemente a qualidade da sua resposta para o usuário.

Essa proposta é composta por dois classificadores, que foram, inicialmente, propostos neste trabalho. O primeiro utiliza a técnica de árvore de indução para classificar as páginas de acordo com suas características e o segundo utiliza o estimador heurístico que atribui uma dada classe à página de acordo com o seu histórico de muidanças.

Os classificadores elaborados e construídos neste trabalho podem ser utilizados não somente por engenhos de busca, mas também por qualquer sistema que lide com atualização de informações de páginas *Web*. Um exemplo de uma aplicação que poderia fazer uso desses classificadores seria os servidores de *proxy cache* e sistemas congelam parte da *Web*, pois eles guardam cópias locais das páginas *Web* e precisam freqüentemente atualizá-las.

Apresentou-se também neste trabalho o comportamento geral das páginas da *Web* brasileira. Os resultados obtidos mostraram que 40% das páginas da *Web* brasileira muda em um período menor ou igual a uma semana.

Como resultado final deste trabalho, obteve-se um componente de software que implementa a proposta de solução, que foi embutido em um engenho de busca a fim de melhorar o processo de atualização de sua base de índices.

7.3 Trabalhos futuros

Neste trabalho, foram utilizadas técnicas de Aprendizagem de Máquina para ajudar no processo de atualização de base de índices. Como trabalho futuro, propõe-se para a construção do classificador inicial o uso de outras técnicas de Aprendizagem de Máquina como Redes Neurais [5] e *Suport* Vector *Machines* [9]. Neste trabalho, foram realizados alguns estudos preliminares sobre a viabilidade do uso dessas técnicas, mas percebeu-se que essas técnicas requerem um processo laborioso de definição de valores de parâmetros dos algoritmos utilizados e algumas modificações nos dados de entrada. Sugere-se, também, a realização de um experimento com o classificador baseado em árvore de indução em um servidor de *cache proxy* para verificar se há uma melhora na eficiência deste servidor, ou seja, se ele desperdiça menos recursos, evitando visitar páginas que não foram modificadas desde a última vez que ela foi colocada em *cache*.

Outro trabalho a ser realizado seria a execução do protótipo com a proposta de solução por um período de tempo maior do que o realizado aqui, como um ano, por exemplo, a fim de se verificar o seu comportamento quando a classificação das páginas tivesse chegado num estado de convergência, ou seja, quando as páginas estivessem em suas classes de modificação corretas.

Referências Bibliográficas

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

[2] Baeza-Yates,R. e Ribero-Neto,B. "Modern Information Retrieval". Addison Wesley, 1999.

[3] Barbosa, L. A.; Ramalho, F. S.; Salgado, A. C. "*Dynamic Indexing of Information in the Web: The Case of News Sites*". In proceedings of 14th IRMA International Conference, Filadelfia, 2003.

[4] Berger,A. “Statistical Machine Learning for Information Retrieval”. PhD thesis. School of Computer Science, Carnegie Mellon University, Pittsburgh, 2001.

[5] Bigus, J. P. "Data Mining with Neural Networks". MacGraw Hill, 1996.

[6] Bowman, C. M.; Danzig, P. B., Hardy, D. R.; Manber, U. e Schwartz, M. F. “Harvest: A Scalable, Customizable Discovery and Acces System”. Technical Report CU-CS-732-94, Departamento de Ciência da Computação, Universidade do Colorado – Boulder, 1994.

[7] Brandman,O.; Cho, J.; Garcia-Molina, H e Shivakumar,N. “Crawler-Friendly *Web* Servers”. In Proceedings of the Workshop on Performance and Architecture of *Web* Servers (PAWS), 2000.

[8] Brewington, B. E. e Cybenko, G. “How Dynamic is the *Web*?” In Proceedings of the 9th World-Wide *Web* Conference (WWW9) , 2000.

[9] Burges, C. J. C. "A Tutorial on Support Vector Machines for Pattern Recognition". Data Mining and Knowledge Discovery 2(2), 1998.

[10] Coelho, R. S. "SAAL - Um Sistema para Análise e Armazenamento de *Links* da Web". Dissertação de Mestrado, Centro de Informática, Universidade Federal de Pernambuco, Recife, 2002.

[11] Cho, J. e Garcia-Molina, H. “The Evolution of the *Web* and Implications for an Incremental Crawler”. In Proceedings of 26th International Conference on Very Large Databases (VLDB), 2000.

[12] Cho, J. e Garcia-Molina, H. “Synchronizing a Database to Improve Freshness”. In Proceedings of 2000 ACM International Conference on Management of Data (SIGMOD), 2000.

[13] Cho, J. e Garcia-Molina, H. “Estimating Frequency of Change”. Technical report, Department of Computer Science, Stanford University, 2001.

[14] Douglis,F. ;Feldman,A e Krishnamurthy, B. “Rate of Change and other Metrics: a Live Study of the World Wide *Web*”. In Proceedings of the USENIX Symposium on Internetworking Technologies and Systems, 1999.

[15] Edwards, J.; McC*URL*ey, K. e Tomlin, J. “An Adaptive Model for Optimizing Performance of na Incremental *Web* Crawler” In Proceedings of the 10th World-Wide *Web* Conference (WWW10), 2001.

[16] Fernandes, M. R. " Engenhos de Busca Distribuídos - Uma Abordagem Visando Escalabilidade para Crawling e Indexação". Dissertação de Mestrado, Centro de Informática, Universidade Federal de Pernambuco, Recife, dezembro de 2001.

[17] Fonseca, N. M.; Resende, R.e Pádua, C. "Aspectos Dinâmicos dos Documentos da Web Brasileira". XXVII Seminário Integrado de Software e Hardware. Curitiba, PR, 2000.

[18] Fowler, M. e Scott, K. "UML Distilled: a Brief Guide to the Standard Object Modeling Languange". Second Edition, Addison Wesley Longman, 2000.

[19] Frakes, A. e Baeza-Yates, R. "Information Retrieval". Prentice Hall, 1992.

[20] Gupta, V. e Campbell, R. “Internet Search Engine Freshness by *Web* Server Help”, In Proceedings of Symposium on Applications and the Internet (SAINT-2001), January 2001.

[21] Hall, M. A; Holmes, G. "Benchmarking Attribute Selection Techniques for Data Mining". Technical Report, Department of Computer Science, University of Waikato, 2000.

[22] Han, J. e Kamber, M. "Data Mining: Concepts e Techniques". , Morgan Kaufmann Publishers, 2001

[23] Kleinberg, J. "Authoritative Sources in a Hyperlinked Enviroment". Journal of the ACM, 46(5):604-632, November, 1999.

[24] Kohavi R., Provost F. “Glossary of Terms”. Editorial for the Special Issue on Applications of Machine Learning and the Knowledge Discovery Process. Volume 30, number 2/3, February/March, 1998.

[25] Koster, M. “Guidelines for Robot Writers”. http://info.*Web*crawler.com/mak/projects/robots/guidelines.html, 1993.

[26] Krause, M. e Tipton, H. F. “*Information Security Management Handbook*”, Quarta edição, Volume 1, Editora Auerbach.

[27] Langley,P.; W. Iba e Thompson, K."An Analysis of Bayesian Classifiers". In Proceedings of Tenth Conference on Artificial Inteligence, San Jose, 1992.

[28] Lawrence, S. e Giles, C. L. “Accessibility of information on the *Web*”. Nature, 400:107-109, 1999.

[29] Lawrence, S. e Giles, C. L. "*Searching the World Wide Web*". Science, 280(5360):98,1998.

[30] Lindholm,T. e Yellin,F. “The Java Virtual Machine Specification”. The Java Series. Addison Wesley, 1st. edition, September 1996.

[31] Michalski, R., Stepp, R. E., e Diday, E. "Automated Construction of Classifications: Conceptual Clustering Versus Numerical Taxonomy". IEEE Trans. Pattern Anal. Mach. Intell. PAMI-5, 5, 396-409, 1983.

[32] Page, L.; Brin, s.; Motwani, r. e Winograd, T. "The Pagerank Citation Ranking: Bringing Order to the *Web*". Technical Report, Computer Science Departament, Stanford University, 1998.

[33] Quinlan, J. R. "C4.5: Programs for Machine Learning". Morgan Kaufmann Publishers,1993.

[34] Quinlan, J. R. "Induction of Decision Trees". Machine Learning 1(1):81-106, 1986.

[35] Rijsbergen, V. C. J. “Information Retrieval”. 2nd edition, London, Butterworths, 1979.

[36] Rumbaugh, J.; Jacobson, J. e Booch, G. "The Unified Modeling Language Reference Manual". Addisno wesley Longman, 1st edition, 1999.

[37] Radix Home Page. http://www.radix.com.br

[38] Russel, S. e Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall. 1995.

[39] Silva, A. S.; Veloso, E. A.; Golgher, P. B.; Ribeiro-Neto, B.; Laender, A. H. F. e Ziviani, N. “CoB*Web* – A Crawler for the Brazilian *Web*”. In Proceedings of String Processing and Information Retrieval Symposium & International Workshop on Groupware, 1999.

[40] Taylor, H. M. e Karlin, S. "An Introduction to Stochastic Modeling". Academic Press, 3rd edition, 1998.

[41] Trinta, F. "Arquiteturas Distribuídas para Co-Autoria Cooperativa de Aulas na Internet". Dissertação de Mestrado, Centro de Informática, Universidade Federal de Pernambuco, Brasil, 2000.

[42] Veloso, E.; Moura, E.; Golgher, P.; Silva, A.; Almeida. R.; Laender, A.; Ribeiro-Neto e Ziviani, N. "Um Retrato da Web Brasileira". XXVII Seminário Integrado de Software e Hardware. Curitiba, PR, 2000.

[43] Witten, I. H. e Frank, E. "Data Mining - Pratical Machine Learning Tools and Techniques with Java Implementations", Morgan Kaufmann Publishers, 2000.

[44] Wong, S. K. M. e Ziarko, W. "A Machine Learning Approach to Information Retrieval". In proceedings of the 9th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Pisa, 1986.

Apêndice A - Notação dos Diagramas de UML

Diagrama de Classe

Um diagrama de classe descreve os tipos de objetos no sistema e os vários tipos de relacionamentos estáticos que existem entre eles. Há dois tipos principais de relacionamentos [18]:

- Associação: representam relacionamentos entre instâncias de classes. Por exemplo, quando um funcionário trabalha em uma empresa (Figura 22);

Figura 22 - Diagrama de classe contendo uma associação entre classes.

- Generalização: quando um objeto é um sub-tipo de outro. Por exemplo, uma enfermeira é um tipo de pessoa (Figura 23).

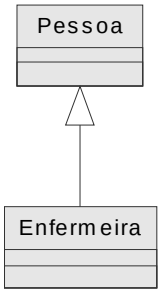


Figura 23 - Diagrama de classe contendo uma generalização.

Os diagramas de classe também mostram os atributos e as operações de uma classe e as restrições que se aplicam à forma que os objetos são conectados. Na Figura 24, é apresentado um exemplo, no qual a classe Funcionário possui o atributo *nome* e o método *trabalhar*.

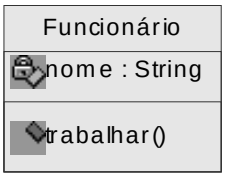


Figura 24 - Diagrama de classe apresentando atributo e método de uma classe.

Diagramas de Interação

Os diagramas de interação são modelos que descrevem como grupos de objetos colaboram em algum comportamento. O diagrama mostra objetos e mensagens que são passadas entre eles. Existem dois tipos de diagramas de interação [18]:

- *Diagrama de seqüência*: dentro de um diagrama de seqüência, um objeto é mostrado como uma caixa em cima de uma linha vertical hachurada. Esta linha vertical é chamada de linha da vida do objeto. Cada mensagem é representada por uma seta entre as linhas da vida de dois objetos. A ordem que estas mensagens acontecem é mostrada de cima para baixo no diagrama. Cada mensagem é rotulada no mínimo com o nome da mensagem (método) e numerada de acordo com a ordem que ela acontece. Há nesse diagrama a idéia de condição para se enviar uma tarefa. A mensagem somente é enviada se a condição dentro dos colchetes é verdadeira. Um exemplo de diagrama de seqüência é apresentado na Figura 25. Nele, é mostrada a interação entre os objetos G do tipo Gerente e D do tipo Desenvolvedor. Neste exemplo, o Gerente pede ao Desenvolvedor para realizar uma tarefa, caso ele a termine, ele deve avisar o seu fim ao Gerente.

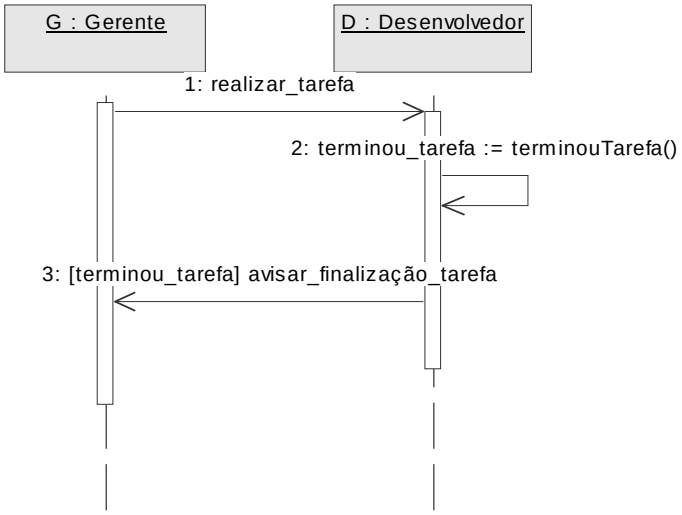


Figura 25 - Exemplo de diagrama de seqüência.

- *Diagrama de colaboração*: os objetos são apresentados como caixas. Como no diagrama de seqüência, as setas indicam as mensagens enviadas e cada mensagem possui um nome e um número que significa a ordem em que a mensagem acontece. Na Figura 26, é apresentado um diagrama de colaboração para o mesmo exemplo apresentado no diagrama de seqüência.

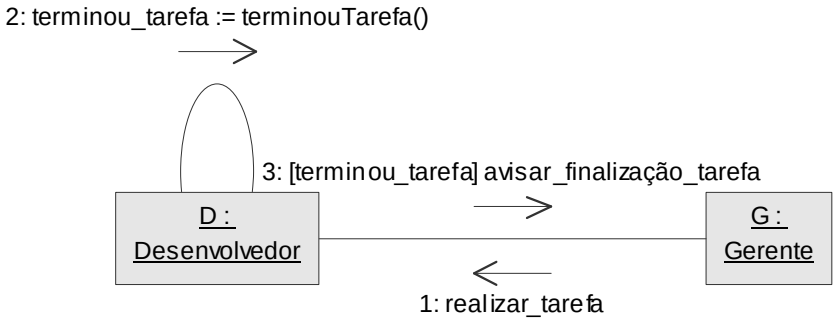


Figura 26 - Exemplo de diagrama de colaboração.

Apêndice B - Modelos de Recuperação de Informação

Vector Space Model

O procedimento no *Vector Space Model* pode ser dividido em três estágios. O primeiro estágio é a indexação do documento na qual os termos com maior conteúdo semântico são extraídos do texto do documento. O segundo estágio é dar pesos aos termos indexados para melhorar a recuperação de documentos relevantes pelo o usuário. O último estágio ranqueia o documento em relação à consulta de acordo com uma medida de similaridade.

- Indexação do Documento: é evidente que muitas palavras de um documento não descrevem seu conteúdo, como artigos e conjunções. Ao realizar a indexação, são retiradas do vetor de palavras, que representa o documento, as palavras não relevantes. São indexadas, assim, somente as palavras que o descrevem;

- Peso dos Termos: os pesos, nesse modelo, são inteiramente baseados em simples estatísticas dos termos. Há três principais fatores para dar pesos aos termos: o fator da freqüência do termo, o fator da freqüência na coleção e o fator de normalização do tamanho do documento. Uma forma usada para dar pesos aos termos dentro de um documento é usar a freqüência do termo no documento. A freqüência do termo é algo que descreve o conteúdo dos documentos e é geralmente utilizado como base do vetor de pesos de um documento. Há várias formas de colocar pesos para discriminar um

documento de outro. Uma delas, a freqüência inversa do termo na coleção, admite que a importância de um termo é inversamente proporcional ao número de vezes que ele aparece em toda a coleção. O terceiro fator, normalização do tamanho do documento, é realizado para evitar que documentos grandes tenham uma maior probabilidade de serem recuperados do que os menores;

- Coeficientes de Similaridade: a similaridade, nesse modelo, é determinada utilizando o coeficiente de similaridade baseado no produto escalar do vetor do documento e o vetor da consulta, no qual palavras iguais indicam similaridade.

Arquivo Invertido

Um arquivo invertido (ou índice invertido) é um mecanismo orientado à palavra para indexar uma coleção de texto a fim de aumentar a velocidade da tarefa de busca. A estrutura de arquivo invertido é composta de dois elementos: o vocabulário e as ocorrências. O vocabulário é o conjunto de todas as palavras diferentes do texto. Para cada palavra há uma lista de todas as posições no texto onde as palavras aparecem. O conjunto de todas as listas é chamado de ocorrências. Um exemplo de um arquivo invertido é mostrado na Figura 27. As palavras são convertidas para letras minúsculas e algumas não são indexadas, por não apresentarem um conteúdo semântico, como, por exemplo, os artigos e as preposições.

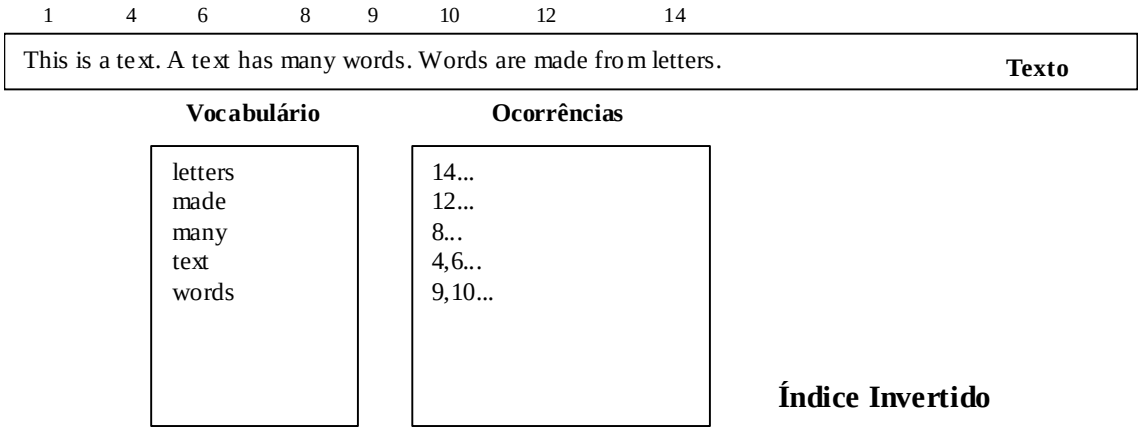


Figura 27 - Um exemplo de um texto e um arquivo invertido construído sobre ele.

Apêndice C - Sistema de Monitoramento

O principal objetivo do sistema de monitoramento é monitorar um conjunto definido de *URLs* tentando sobrecarregar o mínimo possível os servidores onde as *URLs* estão hospedadas, mas também realizando essa atividade num tempo razoável.

Módulos

Este sistema é composto por módulos, cada um desempenhando uma função específica dentro do sistema, são eles (Figura 28):

- Servidor de Códigos de *URLs* - é o servidor que disponibiliza informações dos códigos das *URLs* utilizadas.

- Servidor de *Links* - solicita os códigos das *URLs* ao Servidor de Códigos de *URLs* e as fornece junto com seus códigos para o Robô. Esse fornecimento obedece a um revezamento de servidores, ou seja, para evitar a sobrecarga sobre um servidor, uma página de um dado servidor só será revisitada depois que as páginas de todos os outros servidores tiverem sido visitadas.

- Robô - realiza o download das *URLs* e verifica se elas foram modificadas. Para esta verificação são retirados todos os marcadores *htmls* do conteúdo das páginas. Sobre este conteúdo, gera-se um valor inteiro, utilizando-se o algoritmo MD5 [26] que, de forma simplificada, é uma função que mapeia um vetor de caracteres a um número inteiro. A partir deste inteiro gerado, compara-se o seu valor do dia atual com o do dia anterior. Este algoritmo foi

escolhido devido à sua ampla utilização na área de segurança para a verificação de assinaturas digitais.

- Armazenador de Datas - para cada página, este módulo armazena as informações sobre o seu histórico de mudança. Assim, quando o Robô envia a informação sobre a modificação ou não da página desde a sua última visita, isto é armazenado em um *array* que corresponde ao histórico de alteração desta página em meio persistente.

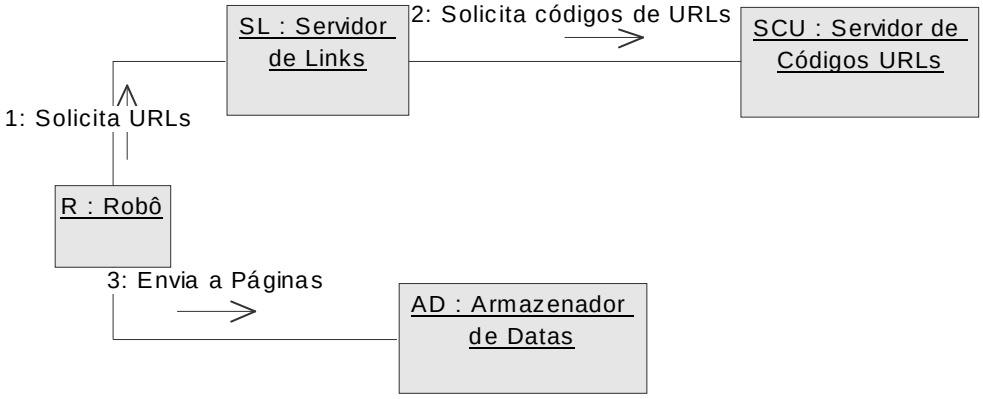


Figura 28 - Diagrama de colaboração do sistema de monitoramento.

Aspectos de Desempenho

A velocidade de armazenamento do sistema é um fator primordial para que ele funcione de forma eficiente, pois como se deseja visitar um número pré-estabelecido de *URLs* com uma determinada freqüência, é preciso que se possua uma velocidade compatível com essas duas variáveis. Portanto, durante a construção do software, houve uma preocupação com os aspectos relativos à velocidade de armazenamento das informações.

Para este sistema conseguiu-se uma velocidade de armazenamento de cerca de 500.000 páginas por dia. Essa velocidade pode ser configurada de acordo com a quantidade de *site*s que se deseja obter as informações. Caso esse número seja pequeno, essa velocidade tem que ser baixa, para não sobrecarregá-los, embora seja implementado no sistema o conceito de revezamento de servidores para minimizar isso.

Entretanto, ela não pode ser tão baixa a ponto que não se consiga recuperar as informações num tempo máximo desejado. Por exemplo, admita que se deseje usar o sistema para monitorar diariamente um conjunto de 100.000 *URLs*, se a velocidade estiver configurada para 50.000 páginas por dia, não será possível visitar todo o conjunto. Portanto, a velocidade deve ser configurada de acordo com o número de *URLs* que se quer observar, sendo que essa versão do sistema possui uma velocidade máxima de 500.000 páginas por dia[25].

[25] Máquina com 256Mb de RAM e 1 processador (Pentium 2 - 400MHz)