Aplicação de Técnicas de Data Mining em Logs de Servidores Web

Ramon Chiara

Orientadora: ***Profa. Dra. Maria Carolina Monard***

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para a obtenção do título de Mestre em Ciências - Área de Ciências de Computação e Matemática Computacional.

USP - São Carlos
Março de 2003

Agradecimentos

Agradeço aos amigos do LABIC, por todo o apoio, em especial à Huei Diana Lee, à Cláudia Martins, ao Ronaldo Prati, ao Gustavo Batista e à Mariza Ferro. Um agradecimento super especial à Carolina Monard, pela amizade e pela orientação.

Agradeço muito aos meus pais, Jorge Luiz Chiara e Ana Christina P. Chiara, pelo amor e apoio incondicionais.

E, sobretudo, agradeço à Deus, por tudo!

Sumário

Lista de Figuras

Lista de Tabelas

Glossário

Bookmark É uma lista de *sites* que o usuário mantém em seu navegador e que considera interessantes.

Cache É um espaço separado para armazenar cópias temporárias de objetos de maneira que, se estes forem solicitados novamente, não seja necessário fazer uma nova transferência, podendo ser obtidos localmente.

Cookie É um registro colocado no computador do usuário, pelo navegador, em resposta a uma solicitação do servidor *Web*.

Host É qualquer computador na Internet com um nome de domínio.

Link É um referência a outro documento, em um arquivo HTML.

Login É um identificador de usuário para um sistema.

Metatag *Metatags* é uma *tag* especial de HTML utilizada para descrever palavras-chave ou títulos para *sites* de busca.

Proxy É um servidor que armazena os documentos frequentemente requisitados a fim de reduzir a carga em servidores *Web*.

Site Um *site* é um servidor *Web* que aparece para os usuários como uma entidade.

Spammer É um indivíduo responsável pela ampla distribuição de correspondência indesejável e não solicitada na Internet por meio do envio de uma mensagem a um grande número de destinatários.

Stop Words
São palavras que ocorrem com muita frequência em textos como, por exemplo, artigos e preposições.

Webmaster
É a pessoa que cuida da criação e/ou manutenção de um *site*.

AM Aprendizado de Máquina

CLF *Common Log Format*

Clicar Apertar o botão do *mouse* após colocar o cursor no ponto desejado.

Conhecimento de Fundo
Conhecimento prévio, ou anterior, sobre o domínio estudado.

DNS *Domains Name Service*— é um protocolo que relaciona nomes de domínios com os seus endereços. O DNS inverso serve para pesquisar o nome de domínio dado o endereço numérico.

DW *Data Warehouse*

ECLF *Extended Common Log Format*

Evento de Página
Eventos de página referem-se a páginas transferidas do servidor para o navegador, independente de qualquer conteúdo auxiliar como, por exemplo, as imagens que uma página contém.

Flash É uma tecnologia que permite o desenvolvimento de mini-aplicativos e animações para serem colocados dentro de páginas HTML.

Formulário Um formuário é um documento HTML que contém instruções para que o usuário entre com informações. Em um *site* de busca, por exemplo, existe uma página responsável por obter as palavras de busca que o usuário deseja encontrar. Essa página é um formulário.

FTP *File Transfer Protocol*— protocolo utilizado para a transferência de arquivos entre computadores.

Gesto do usuário
Entende-se por gesto a digitação de um endereço no navegador ou a efetuação um clique em um *link* de uma página, por parte do usuário.

HTML *HyperText Markup Language*— é uma linguagem de marcação de textos utilizada para definir as características de apresentação de documentos da *Web*.

IA Inteligência Artificial

ICMC Instituto de Ciências Matemáticas e de Computação

KDD *Knowledge Discovery in Databases*

Navegador Navegador é um programa que se comunica com servidores *Web* e exibe o conteúdo requisitado (texto, imagem, áudio, vídeo).

OLAP *Online Analytic Processing*— as ferramentas OLAP têm como objetivos o suporte e a simplificação da análise interativa dos dados. OLAP é essencialmente um processo dedutivo enquanto que o KDD é um processo indutivo.

OLTP *On-Line Transaction Processing*— a descrição original para sistemas associados à entrada confiável de dados em um banco de dados.

Ontologia É uma espécie de categorização.

Perguntas Feitas com Frequência
Frequently Asked Questions (FAQ) — muitos *sites* mantêm uma lista desse tipo.

PLI Programação Lógica Indutiva

Porta Uma porta em um servidor é a maneira utilizada para se identificar com qual programa deseja-se comunicar. Assim, um servidor *Web*, normalmente, é identificado pela porta 80, um servidor de FTP é, normalmente, identificado pela porta 21, e assim por diante.

Requisição HTTP
Uma requisição HTTP é qualquer pedido que o navegador faz para um servidor *Web*.

Sequência de cliques
É o conjunto de ações realizadas por um usuário em um navegador. A sequência de cliques existe na forma de *logs* dos servidores *Web*, nos quais cada registro está associado a uma única ação.

Sessão Sessão é a coleção de ações realizadas por um visitante de um *site* enquanto ele navega por ele, sem sair desse *site*.

Subdomínio Um domínio da Internet é um intervalo específico de endereços da Internet atribuído a um único usuário, o qual pode ser uma empresa. Um subdomínio é um conjunto contíguo de endereços que fazem parte de algum domínio.

URL *Universal Resource Locator*— é o endereço de um objeto específico na *Web*.

USP Universidade de São Paulo

WCM *Web Content Mining*

WM *Web Mining*

WSM *Web Structure Mining*

WUM *Web Usage Mining*

WWW *World Wide Web*

Resumo

Com o advento da Internet, as empresas puderam mostrar-se para o mundo. A possibilidade de colocar um negócio na *World Wide Web* (WWW) criou um novo tipo de dado que as empresas podem utilizar para melhorar ainda mais seu conhecimento sobre o mercado: a sequência de cliques que um usuário efetua em um *site*. Esse dado pode ser armazenado em uma espécie de *Data Warehouse* para ser analisado com técnicas de descoberta de conhecimento em bases de dados. Assim, há a necessidade de se realizar pesquisas para mostrar como retirar conhecimento a partir dessas sequências de cliques. Neste trabalho são discutidas e analisadas algumas das técnicas utilizadas para atingir esse objetivo. É proposta uma ferramenta onde os dados dessas sequências de cliques são mapeadas para o formato atributo-valor utilizado pelo Sistema Discover, um sistema sendo desenvolvindo em nosso Laboratório para o planejamento e execução de experimentos relacionados aos algoritmos de aprendizado utilizados durante a fase de Mineração de Dados do processo de descoberta de conhecimento em bases de dados. Ainda, é proposta a utilização do sistema de Programação Lógica Indutiva chamado Progol para extrair conhecimento relacional das sessões de sequências de cliques que caracterizam a interação de usuários com as páginas visitadas no *site*. Experimentos iniciais com a utilização de uma sequência de cliques real foram realizados usando Progol e algumas das facilidades já implementadas pelo Sistema Discover.

Abstract

There is a continuous growth in the size and use of the World Wide Web. This creates difficulties in the design of the web sites to suit a variety of different users as well as in the navigation through very large web structures of pages and links. Data recording the behaviour of web users interaction with a site is stored in web server logs. In a medium size site these logs can amount several megabytes per day. As the log data is collected in a raw format, it can be analyzed by using automated tools. Understanding users navigation preferences plays an important role in the process of customizing and adapting the site's interface for the users. This work focuses on techniques to study the user behaviour when navigating within a web site, using the information stored in web server logs. We proposed a computational system where log data is mapped into a standard attribute-value format used by the DISCOVER project, a major research system under development in our Laboratory for planning and execution of experiments related to the use of learning systems during the Data Mining phase of the Knowledge Discovery in Databases process. Moreover, we propose the use of the Inductive Logic Programming learning system Progol to extract relational knowledge from the set of user navigation sessions which characterize the interaction with the web pages visited. Initial experiments with a real log were conducted using Progol and some of the facilities already implemented in the DISCOVER System.

Capítulo 1

Introdução

A Internet, desde a sua criação, tem crescido constantemente. Uma vasta quantidade de serviços apareceu contribuindo para sua expansão. Serviços de correio eletrônico, listas de discussão, troca de arquivos (FTP[1]) e a *World Wide Web* (WWW) são alguns exemplos. Além de crescer em tamanho, o seu conteúdo também tem acompanhado esse crescimento.

Com essa massa crescente de dados disponível ao público, surgiram alguns problemas:

- As pessoas recorrem aos *sites*[2] de busca quando precisam encontrar alguma informação específica na *Web*. A recuperação de documentos relevantes está entre os maiores problemas que surgiram com essa nova tecnologia.

- A enorme quantidade de dados disponível na Internet a torna uma área fértil para a pesquisa em Mineração de Dados. Mas a aplicação dessas técnicas em dados da *Web* não é trivial. Isso decorre, em parte, do problema citado no item anterior. Para a aplicação de técnicas de Mineração de Dados, é necessário que se tenha uma coleção de dados disponível. Entretanto, o problema é conseguir dados relevantes para se extrair deles conhecimento potencialmente útil.

- As pessoas que acessam a Internet têm preferências diferentes. Moldar o conteúdo e a forma de apresentá-lo de maneira a agradar os usuários é um grande desafio. A idéia é tentar fazer um site atingir seus objetivos. Tem-se, então, o problema da personalização da informação.

- Com o crescente interesse no comércio eletrônico (*e-commerce*), descobrir quem são as pessoas que visitam um *site* tornou-se uma necessidade para as empresas. Enquanto que no caso anterior o problema consiste em descobrir as preferências dos

[1] *File Transfer Protocol* — protocolo utilizado para a transferência de arquivos entre computadores.
[2] Um *site* é um servidor *Web* que aparece para os usuários como uma entidade.

usuários, neste caso o problema consiste em descobrir quem está acessando o site de forma a, por exemplo, fazer um *marketing* adequado.

As técnicas de *Web Mining* e de Recuperação de Informação podem ser utilizadas para resolver parte desses problemas. A pesquisa em *Web Mining* é um ponto de encontro das áreas de Banco de Dados, Inteligência Artificial (especialmente Aprendizado de Máquina e Processamento de Língua Natural) e da própria Recuperação de Informação.

Na realidade, *Web Mining* pode ser considerada como uma instanciação de Mineração de Dados no processo de *Knowledge Discovery in Databases* (KDD[3]) sendo que, em *Web Mining*, os dados provêm da *Web*.

No processo geral de KDD, a etapa de extração de conhecimento é frequentemente realizada utilizando-se algoritmos de Aprendizado de Máquina. Esses algoritmos utilizam conjuntos de dados que devem ser relevantes e estar em um formato apropriado. Para a extração desses dados, são frequentemente utilizados os *Data Warehouse*, os quais não são imprescindíveis, mas a sua existência facilita o processo de KDD.

Uma das sub-áreas de *Web Mining* é a *Web Usage Mining*, na qual são estudados os arquivos de *log*[4] de servidores *Web*, entre outros tipos de arquivos de *log*. Assim como no processo geral de KDD, é preferível que esse arquivo de *log* encontre-se em um formato adequado para que as técnicas de *Web Usage Mining* sejam aplicadas. Isso pode ser alcançado com um *Data Webhouse* (Kimball and Merz 2000), conforme pode ser visto na Figura 1.1.

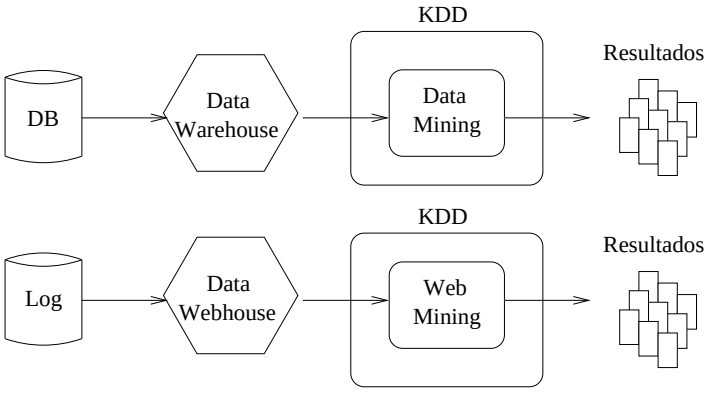


Figura 1.1: Paralelo entre *Data Warehouse* e *Data Webhouse*

Uma vez que os dados dos arquivos de *log* estejam em em *Data Webhouse*, eles têm o potencial de fornecer detalhes valiosos sobre cada gesto efetuado por um usuário de um

[3] *Knowledge Discovery in Databases*. A sigla em inglês é utilizada neste trabalho por ser amplamente difundida na comunidade.

[4] O *log* é um "livro de registros". Normalmente, os servidores *Web* mantêm um arquivo onde são registradas as requisições que são feitas à ele.

site da *Web*. Os *logs* são uma imensa fonte de dados comportamentais porque indivíduos interagem, por meio de seus navegadores, com os *sites* da *Web*. Além disso, estando esses dados em um formato apropriado, pode-se analisá-los, adaptá-los e combiná-los com outras fontes de dados, apoiando o processo de tomada de decisões.

O objetivo deste trabalho é ter um primeiro contato com a área de *Web Mining*, mais especificamente *Web Usage Mining*. Para isso, foi feito um estudo abrangente, desde o entendimento de como a *Web* funciona, passando pela pesquisa de ferramentas existentes para análise de *logs* até uma primeira tentativa de se aplicar algoritmos de Aprendizado de Máquina em *logs* de servidores *Web*.

Entender as interações entre um navegador e um servidor *Web* é essencial para compreender o significado dos dados dos *logs*. Quando navega-se na *Web*, muitas coisas estão acontecendo para que uma página requisitada seja transferida do servidor para a máquina do usuário e, finalmente, seja mostrada em seu navegador. Neste trabalho é feito um estudo sobre essa interação permitindo, também, entender as dificuldades que podem ser encontradas na mineração dos dados dos *logs*.

Além disso, neste trabalho é realizado um estudo sobre algumas ferramentas *freeware* existentes para análise de *logs*. Estas ferramentas permitem, entre outras utilidades, estruturar o *site* e sua infra-estrutura, fornecendo informações que darão o suporte necessário à tomada de certas decisões. Algumas experiências realizadas com essas ferramentas utilizando *logs* do servidor *Web* do ICMC — Instituto de Ciências Matemáticas e de Computação, bem como algumas alterações realizadas no código das ferramentas, também são descritas. Essa modificações somente foram possíveis devido ao fato de que as ferramentas analisadas são *open source*.

Neste trabalho, também foi realizada uma tentativa inicial de se aplicar algoritmos de Aprendizado de Máquina nos arquivo de *logs* de servidores *Web*. Para esse estudo, implementou-se uma ferramenta que tem como objetivo transformar os arquivos de *log* em um formato padrão. Essa ferramenta tem como base o Sistema Discover, que está sendo implementado no LABIC — Laboratório de Inteligência Computacional. A utilização do Sistema Discover como base para um *Data Webhouse* permite gerenciar a grande quantidade de dados que um arquivo de *log* armazena pois, após realizar a transformação do arquivo de *log* para o formato padrão do Sistema Discover, é possível iniciar o processo de extração de conhecimento desses *logs* utilizando as facilidades implementadas e a serem implementadas futuramente nesse sistema.

O trabalho está estruturado da seguinte forma:

No **Capítulo 2** é dada uma visão geral sobre *Web Mining* e suas sub-áreas (*Web Content Mining*, *Web Structure Mining* e *Web Usage Mining*). Antes, porém, é feita uma

contextualização da área, com uma breve explanação sobre *Data Warehouse*, *Knowledge Discovery in Databases* e Mineração de Dados.

No **Capítulo 3** são discutidos alguns dos motivos que levam as empresas quererem analisar os *logs* de seus servidores *Web*. Também é mostrado como essa análise pode ser feita. São discutidos, por exemplo, quais os elementos que podem ser analisados, dentro do contexto de *Web Usage Mining*.

No **Capítulo 4** são descritos alguns detalhes técnicos que devem ser levados em conta para a construção de um *Data Webhouse*. Em especial, é mostrado como são feitas as interações entre cliente e servidor no contexto da *Web*, bem como alguns problemas que podem surgir quando se tenta analisar os arquivos de *log*. Além disso, é mostrada uma possível arquitetura para o processamento desses arquivos.

No **Capítulo 5** é feito o estudo de duas ferramentas *freeware* e *open source* existentes. É mostrado que, pelo fato dessas ferramentas serem *open source*, é possível fazer modificações nessas ferramentas de modo a melhorar os resultados produzidos por elas. As alterações no código de uma das ferramentas é de nossa autoria. O estudo realizado envolve a utilização dessas ferramentas bem como a análise dos resultados produzidos pelas mesmas.

No **Capítulo 6** é descrita a ferramenta proposta para se fazer experimentos com os arquivos de *log*. Essa ferramenta está baseada no Sistema Discover, desenvolvido no LABIC e também descrito sucintamente nesse capítulo. O Sistema Discover facilita a manipulação de arquivos que contém conjuntos de dados para serem utilizados por algoritmos de Aprendizado de Máquina, bem como na análise dos resultados obtidos com a aplicação desses algoritmos.

No **Capítulo 7** são descritos os experimentos realizados utilizando a ferramenta proposta no Capítulo 6. Foram realizadas experiências com algoritmos de aprendizado que induzem conceitos proposicionais e relacionais. São mostrados os resultados obtidos nessas experiências, bem como uma análise dos mesmo.

No **Capítulo 8** encontram-se a conclusão do trabalho e uma relação de trabalhos futuros que possam vir a ser realizados, tomando como base os resultados obtidos neste trabalho.

No **Apêndice A** é apresentada uma visão geral sobre *clustering*, mostrando alguns algoritmos utilizados nessa técnica de Aprendizado de Máquina não supervisionado.

No **Apêndice B** é feita uma breve introdução às regras de associação.

No **Apêndice C**, a área de Recuperação de Informação é brevemente apresentada, mostrando-se, também, sua aplicação na Internet.

Capítulo 2

Do KDD ao *Web Mining*

2.1 Considerações Iniciais

Este capítulo apresenta uma visão geral do que é o *Data Warehouse*. Também mostra que o *Data Warehouse* pode ser utilizado para descobrir novos conhecimentos sobre os negócios de uma empresa, aplicando-se o processo de *Knowledge Discovery in Databases* (KDD) no mesmo. A etapa de Mineração de Dados dentro do KDD também pode ser considerado como um processo à parte. Também é feita uma apresentação do que é o *Web Mining*, como pode ser encaixado dentro do processo de KDD e para que aplicações ele pode ser útil.

2.2 *Data Warehouse*

A maioria das empresas possui uma enorme quantidade de informação em seus sistemas de banco de dados. Essas informações referem-se aos seus negócios, seus clientes e ao próprio mercado. O uso das mesmas é vital para as empresas. A sobrevivência dessas organizações depende do grau de inteligência com que utilizam essas informações para as estratégias de negócios e a possibilidade de descobrimento de novas oportunidades.

Muitas vezes, essas informações estão em diferentes bases de dados que, por sua vez, estão em diferentes departamentos que fazem parte da organização, de forma que o acesso aos dados para análise torna-se difícil. A solução encontrada pelas empresas foi o *Data Warehouse*.

O *Data Warehouse* é um processo utilizado para armazenar os dados estratégicos em um único lugar, passando, antes, por um processo de extração, crítica, transformação e

padronização dos dados.

A etapa de extração refere-se ao processo em que os dados são extraídos das diversas bases operacionais e são armazenados em uma base única porém não definitiva. Isso porque, agora, os dados devem passar por uma fase de crítica, na qual são analisados por pessoas que entendem esses dados, de maneira que somente informações relevantes sejam armazenadas no *Data Warehouse*. Quando, finalmente, as informações relevantes foram selecionadas, elas passam por um processo de transformação para que qualquer incoerência seja eliminada (ex: em uma base X_1, o telefone de um cliente é armazenado como uma cadeia de caractere e, em outra base X_2, como um número) e, no final, haja uma padronização na forma de armazenamento dessas informações. Isso torna a base homogênea para o acesso das ferramentas que serão utilizadas para a análise dos dados, tais como ferramentas OLAP[1]. Na Figura 2.1 tem-se uma visão global do que é o *Data Warehouse*.

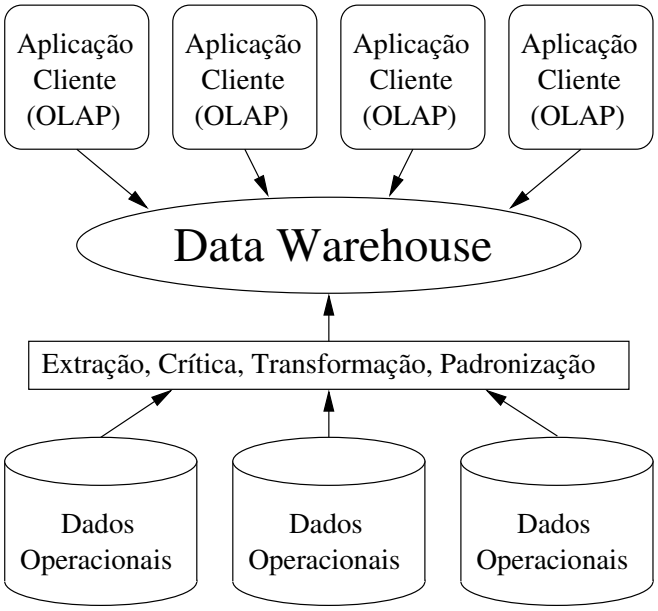


Figura 2.1: Visão geral do *Data Warehouse*

De uma forma geral, as empresas implementam um ambiente de *Data Warehouse* porque suas informações não podem ser analisadas adequadamente na forma em que estão armazenadas, *i.e.* em diversas bases operacionais diferentes. O *Data Warehouse* guarda essas informações de forma organizada. Porém, o suporte à tomada de decisões deve avaliar o que está por trás dos dados pois, no meio deles, escondem-se informações de importância estratégica para as organizações. É nesse momento que o processo de KDD entra em cena.

[1] *Online Analytic Processing*— as ferramentas OLAP têm como objetivos o suporte e a simplificação da análise interativa dos dados. De acordo com (Corporation 1999), OLAP é essencialmente um processo dedutivo enquanto que o KDD é um processo indutivo.

2.3 *Knowledge Discovery in Databases*

De acordo com Fayyad (Fayyad, Piatetsky-Shapiro, and Smyth 1996)

> *Knowledge Discovery in Databases é o processo não-trivial de identificar padrões válidos, novos, potencialmente úteis e, por fim, compreensíveis, em dados.*

Ele é não-trivial porque não se refere à cálculos diretos como, por exemplo, calcular a média de idade dos clientes. É um processo de descoberta de novas correlações através da mineração de grandes quantidades de dados armazenados.

O termo minerar talvez seja a causa da confusão entre KDD e Mineração de Dados. A indústria costuma utilizar esses termos como sinônimos. No entanto, Mineração de Dados é um dos passos no processo de KDD, o qual envolve vários passos, mostrados na Figura 2.2 e descritos a seguir.

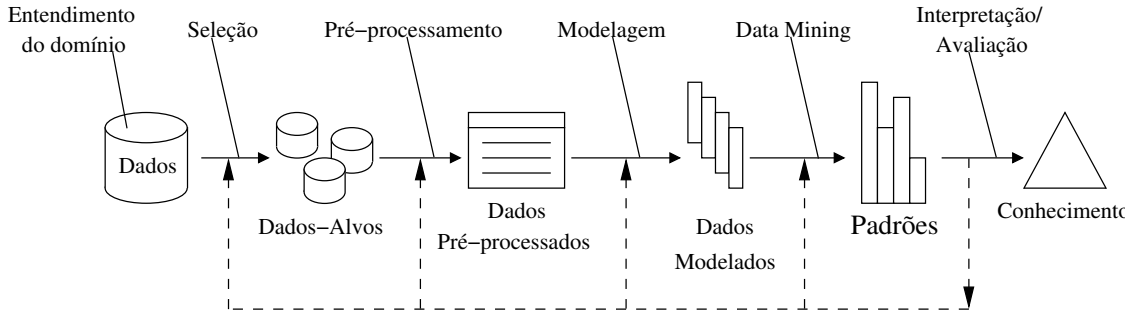


Figura 2.2: Visão geral do processo de KDD

Entendimento do domínio: o KDD não gera resultados sozinho. O entendimento do domínio a ser tratado é muito importante, assim como identificar os objetivos do KDD do ponto de vista da pessoa a realizá-lo.

Seleção dos dados: refere-se à seleção de um subconjunto dos dados disponíveis pois, normalmente, os bancos de dados possuem terabytes de informações.

Pré-processamento dos dados: nessa etapa é feita, entre outros, a limpeza dos dados. Isso sugere a remoção de informações interferentes (inconsistências e duplicações) e a decisão de como manipular as informações que estão faltando.

Modelagem dos dados: aqui é feita a escolha das variáveis relevantes para a representação dos dados, dependendo dos objetivos do KDD.

Mineração de Dados: por ser considerado um dos passos cruciais e mais complexos do KDD, a Mineração de Dados também pode ser considerada como um processo que, por sua vez, pode ser dividido em várias sub-etapas. Basicamente, os algoritmos a serem utilizados devem ser escolhidos de acordo com o problema que está sendo atacado (dados categóricos × dados reais; modelos descritivos × modelos preditivos). Normalmente existem vários métodos para um mesmo objetivo de KDD e a fase de Mineração de Dados inclui a aplicação de diversas técnicas assim como a avaliação e a comparação dos resultados obtidos (Rezende, Pugliesi, Melanda, and Paula 2003).

Interpretação dos resultados: os resultados da etapa anterior são analisados. Eles devem fazer sentido e, caso contrário, deve-se voltar à alguma das etapas anteriores.

Validação dos resultados: as regras obtidas como resultado da Mineração de Dados devem ser validadas com novos dados. Os padrões descobertos devem ser válidos em dados novos com um certo grau de certeza.

Deve ser observado que o *Data Warehouse* não é um requisito para o KDD. Contudo, os passos anteriores à Mineração de Dados são facilitados caso os dados já estejam em um *Data Warehouse*.

Outro ponto importante é que o resultado final de um processo de KDD é conhecimento. Este deve contribuir para um melhor entendimento do problema sendo tratado e, assim, ajudar na busca de soluções eficazes.

2.4 Mineração de Dados

Segundo Fayyad (Fayyad, Piatetsky-Shapiro, and Smyth 1996)

> *Mineração de Dados é um passo no processo de KDD que consiste na aplicação de análises de dados e algoritmos que, sob limitações de eficiência computacionais aceitáveis, produz uma relação particular de padrões a partir dos dados.*

O passo de Mineração de Dados pode ser visto como um subprocesso dentro do KDD. Ele consiste em uma preparação mais refinada dos dados provenientes das etapas anteriores, na aplicação de algoritmos apropriados e na pré-avaliação dos resultados (Horst and Monard 2000).

A preparação dos dados consiste em:

- Redução, por amostragem, do número de instâncias.

- Redução do número de atributos, que pode ser feita por um especialista ou por métodos apropriados de seleção de atributos (Lee 2000), (Blum and Langley 1997) e (John, Kohavi, and Pfleger 1994).
- Aprendizado construtivo, que consiste na construção de novos atributos a partir dos atributos originais (Lee and Monard 2000).
- Limpeza dos dados, com o tratamento de dados ausentes, padronização, discretização de atributos contínuos, tratamento de ruído, etc. (Batista 2003b).

Já os algoritmos para aquisição de conhecimento, *i.e.* algoritmos de Aprendizado de Máquina, podem ser divididos em (Langley 1996), (Mitchell 1997), (Baranauskas and Monard 2000), (Rezende 2003):

- Aprendizado Supervisionado (Predição). Esses algoritmos produzem um modelo preditivo com base em atributos-metas (Weiss and Indurkhya 1998). Podem ser:

  Algoritmos de Classificação: nos quais o atributo-meta possui valores discretos.

  Algoritmos de Regressão: nos quais o atributo-meta possui valores contínuos.

- Aprendizado Não-Supervisionado (Descrição). Utilizados para gerar uma descrição dos dados. São sub-divididos em:

  ***Clustering*:** tem como objetivo agrupar, em classes, vários objetos que possuem alto grau de similaridade (Apêndice A).

  Sumarização: compreende uma descrição compacta para um subconjunto dos dados.

  Regras de Associação: é a descoberta de associações ou conexões entre objetos (Apêndice B).

  Caracterização: consiste na abstração de um conjunto de dados que caracterizam uma classe.

  Discriminação: é a descoberta de propriedades que distinguem a classe objetivo das outras.

  Evolução: é a detecção de regularidades na evolução do dado ao longo do tempo.

  Análise Sequencial: visa identificar padrões de comportamento nos dados.

Um ponto a ser observado é que algumas técnicas são melhores para determinados problemas e domínios de conhecimento que outras. Portanto, não há um método universal de Mineração de Dados. A escolha de um algoritmo particular para um determinado problema deve ser analisado empiricamente.

2.5 *Web Mining*

Como já mencionado, o crescimento da Internet e, principalmente da WWW, gerou uma enorme quantidade de documentos que estão disponíveis para os usuários. Essa enorme massa de dados é uma nova fonte de pesquisa para a área de descobrimento de conhecimento. Em contraste com a Recuperação de Informação da Internet, o descobrimento de conhecimento na Internet tem como objetivo a extração de conhecimento implícito contido nesses documentos.

De acordo com (Kosala and Blockeel 2000)

> *Web Mining é o uso de técnicas de Mineração de Dados para descobrir e extrair automaticamente informações a partir de documentos e serviços da Web.*

Portanto, *Web Mining* refere-se ao processo completo de se descobrir informação ou conhecimento útil, previamente desconhecido, a partir de dados da *Web*. Ele cobre, implicitamente, o processo padrão do KDD. Assim, pode-se ver *Web Mining* como uma extensão do KDD aplicado à dados da *Web*.

Dessa forma, *Web Mining* pode ser decomposto nas seguintes etapas:

Recuperação de documentos: é o processo de recuperação de dados da *Web*. Para isso, as técnicas de Recuperação de Informação podem ser utilizadas (Apêndice C).

Pré-processamento: refere-se a qualquer transformação nos dados originais. Como exemplo disso temos a retirada de *stop words*[2]. Outro tipo de pré-processamento seria a transformação dos documentos para a forma de lógica de primeira ordem. Por exemplo, em (Craven, DiPasquo, Freitag, McCallum, Mitchell, Nigam, and Slattery 1998), algoritmos de Programação Lógica Indutiva (PLI) são utilizados para esse fim.

Generalização: nessa etapa, algoritmos de Aprendizado de Máquina e técnicas de Mineração de Dados são aplicados.

Análise: essa é a etapa na qual os resultados obtidos na etapa anterior são validados. Como a *Web* é um meio interativo, a fase de análise é muito importante.

Web Mining também pode ser visto como parte do Processo de Recuperação de Informação, pois ele pode ajudar na indexação, na busca e na colocação (*ranking*) dos

[2] *Stop words* são palavras que ocorrem com muita frequência em textos como, por exemplo, artigos e preposições.

documentos. Por exemplo, *clustering* (técnica frequentemente utilizada em Mineração de Dados) pode ser utilizada para indexar documentos semelhantes. Contudo, nem todos os métodos de indexação utilizam técnicas de Mineração de Dados, como é o caso de Índices Invertidos (no Apêndice C são discutidos a indexação por *clustering* e os Índices Invertidos).

Web Mining pode ser dividido em três sub-áreas[3]:

1. *Web Content Mining*
2. *Web Structure Mining*
3. *Web Usage Mining*

descritas brevemente a seguir.

2.5.1 *Web Content Mining*

Muito do conhecimento na *Web* está dentro dos documentos, ou seja, no seu conteúdo. O processo de descoberta de informações úteis a partir desse conteúdo é chamado de *Web Content Mining*. Os principais usos de *Web Content Mining* são:

Sumarização: utilizando a pouca estruturação que o HTML[4] oferece, é possível sumarizar o conteúdo das páginas da *Web*. Uma aplicação bem interessante é a recuperação dos preços de produtos nos sites de compras. Em (Deogun, Sever, and Raghavan 1998), é citado o ShopBot, que é um agente de *Web Mining* especializado em catálogos eletrônicos. Ele utiliza uma descrição dos domínios e dos *sites* como conhecimento de fundo[5] para comparar atributos (ex. preço).

Categorização: algoritmos de Aprendizado de Máquina podem ser aplicados ao conteúdo das páginas de forma a permitir que o computador classifique essas páginas de acordo com uma ontologia[6]. Em (Slattery and Craven 1998), (Craven, DiPasquo, Freitag, McCallum, Mitchell, Nigam, and Slattery 1998) e (Craven, Slattery, and Nigam 1998), é mostrada a aplicação de alguns algoritmos de Aprendizado de Máquina em páginas de universidades. O sistema aprende a classificar uma página como sendo a de um estudante, de um projeto de pesquisa, de um curso, etc.

[3] Os nomes dessas sub-áreas foram mantidos em inglês porque essa é a forma normalmente utilizada pela comunidade.

[4] *HyperText Markup Language*— é uma linguagem de marcação de textos utilizada para definir as características de apresentação de documentos da *Web*.

[5] Conhecimento prévio, ou anterior, sobre o domínio estudado.

[6] Ontologia é uma espécie de categorização.

Descoberta de conhecimento: a enorme coleção de textos disponíveis na *Web* mostra-se uma valiosa massa de dados para a descoberta de novos conhecimentos. Em (Loh, Wives, and Oliveira 2000) é mostrado um experimento de extração de conhecimento relacionado ao que a imprensa estava dizendo sobre o prefeito de uma grande cidade do Brasil. Alguns relacionamentos com tráfico de drogas, empréstimos e educação foram encontrados e analisados.

Pode-se notar que o uso de *Web Content Mining* pode ajudar no Processo de Recuperação de Informação. Sumarização pode ser utilizada para extrair informações relevantes dos documentos para indexação enquanto que categorização pode ser utilizada nos *sites* de catálogos (Apêndice C).

2.5.2 *Web Structure Mining*

Graças a interconexão entre documentos, a WWW pode revelar mais informações do que simplesmente as relacionadas ao conteúdo dos documentos. Por exemplo, muitos *links*[7] apontando para um documento indicam sua popularidade, enquanto muitos *links* saindo de um documento indicam uma riqueza de tópicos cobertos pelo mesmo. O processo que tenta descobrir o modelo que está por trás dessa estrutura de *links* é chamado de *Web Structure Mining*.

A idéia é que os *links* codificam uma considerável quantidade do julgamento humano. Mais especificamente, a criação de um *link* numa página indica que seu autor conferiu autoridade para a página sendo referenciada por esse *link*. Páginas em que chegam muitos *links* são chamadas de *autoridades*. *Hubs*, ao contrário, são as páginas que centralizam essas autoridades. Em (Slattery and Mitchell 2000) são descritos algoritmos que encontram *hubs* e autoridades.

Percebe-se que *hubs* e autoridades mantêm uma relação de reforço mútuo: bons *hubs* apontam para boas autoridades, e uma boa autoridade é aquela que é apontada por um bom *hub*. Na Figura 2.3 é ilustrada essa relação.

Os possíveis usos para *Web Structure Mining* são:

Colocação (*ranking*): *Web Structure Mining* pode ajudar no Processo de Recuperação de Informação, mais precisamente na fase de colocação. Verificando que um documento é uma autoridade, ele é favorecido na colocação.

[7] *Link* é um referência a outro documento em um arquivo HTML.

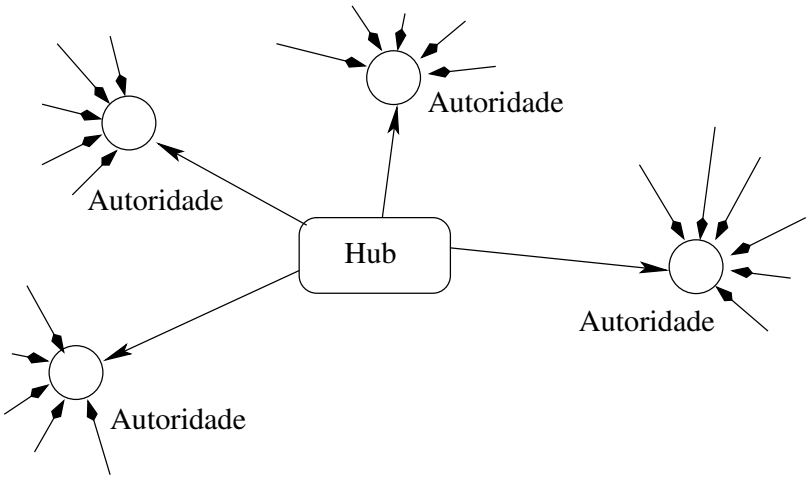


Figura 2.3: *Hubs* e Autoridades

Fluxo de informação: descobrindo a estrutura que os *links* formam, pode-se estudar como o fluxo de informações afeta o projeto de um *site*, fornecendo dicas de como melhorá-lo.

2.5.3 *Web Usage Mining*

Cada servidor *Web* guarda, localmente, uma coleção de registros bem estruturados: os *logs* de acesso. Os servidores *Web* guardam essas informações sobre a interação dos usuários cada vez que é feito um acesso ao *site*. *Web Usage Mining* utiliza-se desses dados para descobrir informações sobre os usuários da *Web*, tais como seus comportamentos e seus interesses.

Como a informação dos *logs* é bem estruturada, pode-se aplicar técnicas típicas de Mineração de Dados sobre esses dados. Além disso, pode-se fazer uso do conhecimento do domínio, que pode ser o assunto que o *site* trata e/ou a sua topologia.

Os possíveis usos para *Web Usage Mining* são:

Personalização: a descoberta do perfil do usuário pode ser útil na personalização da interface, ou do conteúdo, de forma a ajudar o *site* a atingir seus objetivos.

***Marketing*:** saber quem frequenta um determinado *site* pode ser de grande valia para *marketing*. Fu (Fu, Sandhu, and Shih 1999a) dá um exemplo: se alguns usuários passam muito tempo olhando páginas de "mobílias para bebês" e "brinquedos para bebês" então é provável que eles sejam futuros pais. Isso pode sugerir rearranjos no *site* de forma a tornar esses assuntos interconectados.

Proxies: descobrindo-se o padrão de acesso dos usuários, pode-se programar um servidor *proxy*[8] para efetuar o *download* das próximas páginas que o usuário provavelmente irá visitar, enquanto ele lê a primeira página. Um algoritmo que descobre esses padrões é descrito em (Aumann, Etzioni, Feldman, Perkwitz, and Shmiel 1998).

Eficiência: descobrir quais páginas não estão sendo acessadas pode sugerir futuros rearranjos no *site*. Mais que isso, pode-se descobrir qual o padrão de acesso dos usuários que compram produtos no *site* e quais são apenas visitantes. Com isso, pode-se reorganizar o *site* de forma a transformar os visitantes em compradores potenciais. É justamente isso que Spiliopoulos (Spiliopoulou, Pohle, and Faulstich 1999) sugere.

Recuperação de Informação: uma outra fonte de dados para *Web Usage Mining* são os *logs* das máquinas de busca (Apêndice C), ou seja, quais as palavras que foram buscadas e quais os *sites* que o usuário achou relevantes para aquelas palavras. Beeferman (Beeferman and Berger 1999) sugere um algoritmo que melhora a eficiência das máquinas de busca baseado em seus *logs*.

2.6 Considerações Finais

As empresas possuem muitas informações em seus sistemas, mas algumas não sabem o que fazer com essas informações. O processo de KDD objetiva a extração de conhecimento dessas grandes quantidades de dados e ajudar no seu entendimento. Os métodos descritivos de Mineração de Dados podem ser o primeiro passo para realizar essa tarefa.

Neste capítulo viu-se que a utilização de algoritmos de Aprendizado de Máquina e técnicas de Mineração de Dados em dados da *Web* caracterizam o *Web Mining*. Apesar disso, o termo ainda não é bem definido e a distinção entre *Web Content Mining*, *Web Structure Mining* e *Web Usage Mining* é muito tênue.

A utilização de *Web Mining* é muito variada e vai desde a descoberta de novos conhecimentos nos dados da *Web* até a melhoria na Recuperação de Informação. Além disso, existe a possibilidade de se aumentar a eficiência dos *sites* com a definição do comportamento dos usuários, por meio da análise dos *logs* de acesso.

[8] *Proxy* é um servidor que armazena os documentos frequentemente requisitados a fim de reduzir a carga em servidores *Web*.

Capítulo 3

Data Webhouse

3.1 Considerações Iniciais

A *Web* é uma imensa fonte de dados comportamentais porque indivíduos interagem, por meio de seus navegadores[1], com *sites* da *Web*. Embora esses dados comportamentais, chamados de *sequência de cliques*[2], normalmente estejam em um estado bruto e não tenha a forma adequada, eles têm o potencial de fornecer detalhes valiosos sobre cada gesto[3] efetuado por um usuário de um *site* da *Web*. Isso fez com que as empresas se tornassem mais interessadas em aprender sobre os usuários de seus *sites* com a finalidade de fornecer informações úteis a eles.

De acordo com Kimball (Kimball and Merz 2000), o *Data Webhouse*, tratado neste capítulo, é a instanciação *Web* do *Data Warehouse*. Dessa forma, ele armazena dados de sequência de cliques e outros dados comportamentais da *Web* que guiam a compreensão do comportamento do cliente.

3.2 Motivação para o *Data Webhouse*

Os servidores *Web* fornece uma nova fonte de dados chamada de sequência de cliques. Essa sequência de cliques é um *log* de cada gesto feito por um visitante de um *site* da *Web*. A sequência de cliques é, potencialmente, um registro muito melhor do comportamento

[1]Navegador é um programa que se comunica com servidores *Web* e exibe o conteúdo requisitado (texto, imagem, áudio, vídeo).

[2]Sequência de cliques é o conjunto de ações realizadas por um usuário em um navegador. A sequência de cliques existe na forma de *logs* dos servidores *Web*, nos quais cada registro está associado a uma única ação.

[3]Entende-se por gesto a digitação de um endereço no navegador ou a efetuação um clique em um *link* de uma página, por parte do usuário.

do usuário do que outras fontes de dados mais tradicionais. Por exemplo, os famosos dados de registros de chamadas telefônicas das companhias de telecomunicações somente mostram que uma pessoa A fez uma chamada para uma pessoa B e quantos minutos elas ficaram "conectadas". Não há qualquer maneira de saber sobre o que falaram ou se elas ficaram satisfeitas. Mesmo as fontes de dados de processamento de transações on-line (OLTP[4]) omitem muitas informações interessantes. Uma fonte de dados OLTP em um ambiente varejista normalmente registra somente a última etapa da venda. Não se sabe o que conduziu a essa venda.

A sequência de cliques, por outro lado, é uma série cronológica de ações que podem ser agrupadas em sessões[5]. A trajetória das ações que termina em uma compra ou em outro comportamento pode ser analisada e entendida. Pode-se analisar como um usuário chegou até um *site*, qual era o seu propósito e qual foi a qualidade de sua experiência. Pode-se saber quanto tempo um usuário levou para localizar algo no *site* e é possível induzir, analisando o seu comportamento, se teve satisfação ou descontentamento com o *site*.

3.3 Analisando o Comportamento

Para analisar o comportamento do usuário, não basta colocar cada gesto do usuário em um banco de dados. Uma simples sequência de cliques não fornece uma descrição útil de comportamento e poderá levar a conclusões precipitadas. Uma descrição mais útil de comportamento é o propósito. Mas mesmo se o propósito é conhecido, não se pode apenas agregar à sequência de cliques um pequeno número de descrições de propósito e jogar fora os detalhes. O propósito comportamental tem muitas interpretações possíveis. Por exemplo, o propósito de longo prazo de um conjunto de eventos de página[6] pode ser "comprar um produto", mas o propósito de curto prazo pode ser "obter a descrição de um produto".

Uma possível lista de tipos de comportamentos que pode ser inferida da página que está sendo visualizada é a seguinte:

Pesquisar: localizar um produto, serviço ou fonte de informações específicas.

Recolher informações: recolhimento de características de um produto ou de informa-

[4] *On-Line Transaction Processing* — a descrição original para sistemas associados à entrada confiável de dados em um banco de dados.

[5] Sessão é a coleção de ações realizadas por um visitante de um *site* enquanto ele navega por ele, sem sair desse *site*.

[6] Eventos de página referem-se a páginas transferidas do servidor para o navegador, independente de qualquer conteúdo auxiliar como, por exemplo, as imagens que uma página contém.

ções gerais, comparar produtos e preços, leitura de perguntas feitas com frequência[7] ou de suporte.

Educação: utilizar manuais, livros *on-line* e artigos. Leitura de notícias, relatórios técnicos ou outro tipo de informação.

Monitoração: localizar remessas expressas, verificar bolsa de valores, rastreamento da situação de um pedido.

Fazer *download*: buscar imagens, programas e outros tipos de arquivos.

Comprar: selecionar e comprar produtos. Dar um lance (em *sites* de leilão).

Entrada acidental: clicar em um *link* errado, erros de URL[8], *links* quebrados.

No entanto, pode-se inferir informações mais significativas de comportamento se forem identificados:

- Evento bem sucedido de compra × evento de compra incompleto, mal sucedido ou cancelado.
- Localização das informações procuradas × não localização das informações.
- Saída de um usuário de um *site*.
- Exibição incompleta de informações, mas o usuário permaneceu no *site*, ou seja, o usuário não esperou uma página ser completamente carregada para ir para outra página do mesmo *site*.
- Exibição incompleta de informações e o usuário saiu do *site*, ou seja, o usuário cansou de esperar pelo carregamento de uma página e decidiu sair do *site*.

O valor real da identificação do comportamento está em aprimorar a qualidade da interação que o usuário tem com a empresa que mantém o *site*. Aprimorar a interação se traduz diretamente na lealdade de cliente, rendas e lucros aumentados.

[7] *Frequently Asked Questions* (FAQ) — muitos *sites* mantêm uma lista desse tipo.
[8] *Universal Resource Locator* — é o endereço de um objeto específico na *Web*.

3.3.1 Monitorando as Ações de um Usuário

Em um ambiente de varejo baseado na *Web*, tem-se a oportunidade de seguir o cliente durante toda sua viagem de compras. Pode-se medir o que ele olha, quanto tempo olha, o que seleciona e o que rejeita. É como se fosse possível olhar por sobre o ombro do cliente enquanto ele faz suas compras. O mais interessante é que se pode modificar a loja enquanto o cliente "anda" por ela. Por exemplo, pode-se reorganizar as "gôndolas virtuais" para mostrar itens que provavelmente um cliente compraria (Apêndice B).

Em uma loja real, o cliente é cativo. Ele permance lá até que saia, com ou sem compras. Mas ele sempre passará pelo caixa do supermercado. Na *Web*, entretanto, um cliente está a um clique de *mouse* de deixar o *site*. Na Figura 3.1 é mostrada essa diferença.

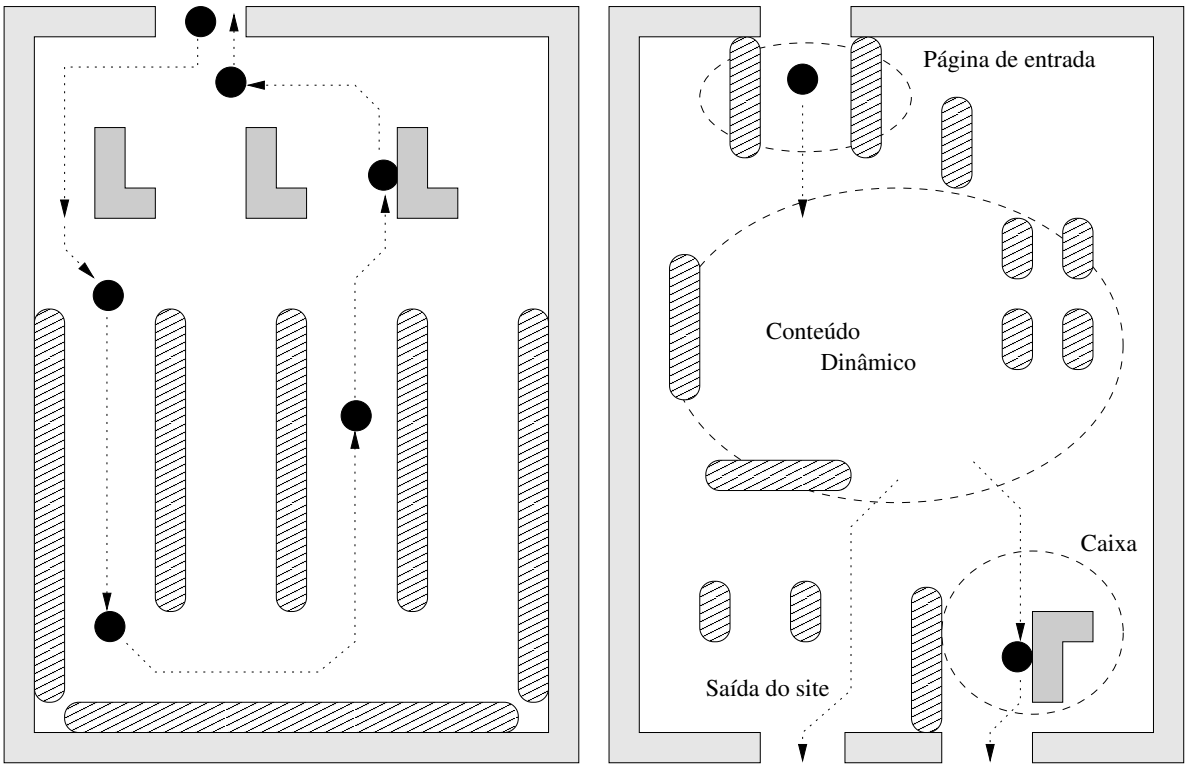


Figura 3.1: Uma loja real × uma loja virtual

Independentemente da motivação de um usuário para acessar a *Web*, o número de ações reais que ele pode realizar é limitado pela mecânica do programa de navegação e pelos protocolos da *Web*. Pode-se capturar muitas das ações dos usuários usando duas fontes de dados:

1. coletando informações que estão contidas nos protocolos de comunicação da *Web*, *i.e.* armazenadas nos *logs* dos servidores *Web*

2. instrumentando um *site* para capturar informações adicionais sobre as atividades de

um usuário uma vez que ele tenha entrado no *site* e estabelecido uma sessão.

É importante observar que os visitantes podem pular, a qualquer momento, de um lado para o outro entre as páginas e, portanto, podem facilmente se perder em um labirinto de páginas e *links* e não serem capazes de encontrar o caminho de volta para onde gostaría-se que eles mantivessem o seu foco. Como foi visto na Seção 2.5.3 na página 13, uma das possíveis utilizações de *Web Usage Mining* é justamente o de melhorar a eficácia de um *site*.

3.3.2 Elementos de Monitoração

A primeira etapa em uma compra de produto é o reconhecimento de alguma necessidade pelo cliente. Essa é a primeira oportunidade de monitorar as ações de um usuário. Se um produto contém uma URL que é única e utilizada em rótulos de embalagens (utilizando, por exemplo, um subdomínio[9]), pode-se identificar que um acesso foi realizado em virtude da leitura desse rótulo. Na Figura 3.2 é mostrado que a Sony pode saber, através dos subdomínios `dvd` e `discman`, se um acesso ao seu *site* foi feito porque o usuário tinha interesse em um DVD ou em um *Discman*, além de saber se o acesso foi feito a partir de um outro site, através do seu endereço normal (`www.sony.com`). Nesse caso, ela pode saber qual página tem um *link* para seu *site*, podendo explorar também essa informação (ver Seção 4.3 na página 32).

Uma vez que o cliente entra num *site*, ele irá tentar achar o que precisa. Cada *site* oferece uma maneira diferente para o cliente obter êxito nessa empreitada. Pode ser utilizado um catálogo ou um mecanismo de busca. De qualquer forma, pode-se monitorar se o cliente está conseguindo achar ou não o que ele precisa. Enquanto o cliente estiver explorando alternativas, pode-se capturar cada palavra de pesquisa sendo digitada, a partir do *log* do servidor da *Web*. Essa informação pode dizer muito sobre as motivações e desejos de um usuário, bem como se o *site* está adequadamente organizado.

Tendo achado o que quer, um cliente pode selecionar esse produto para compra. Pela ação de selecionar um item específico, ele não apenas indica o que escolheu, mas também os ítens que rejeitou. Pode-se adivinhar, a partir de sua seleção, se ele seria capaz de pagar mais por uma marca famosa ou se prefere economizar comprando uma marca desconhecida. As informações recolhidas podem ser utilizadas para planejar a venda de produtos semelhantes (Apêndice B).

[9]Um domínio da Internet é um intervalo específico de endereços da Internet atribuído a um único usuário, o qual pode ser uma empresa. Um subdomínio é um conjunto contíguo de endereços que fazem parte de algum domínio específico.

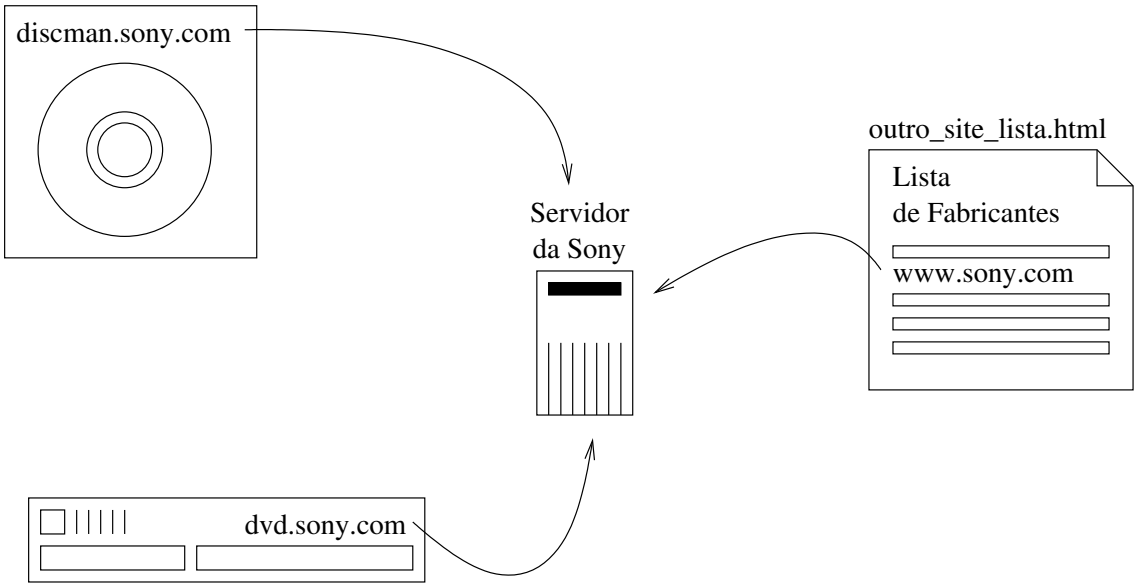


Figura 3.2: Monitorando a entrada do usuário através de um subdomínio.

Algumas informações como a origem de um usuário, ou como ele encontrou um *site* na *Web*, são de extrema importância para o departamento de *marketing* e para os *webmasters*[10], pois elas são um indicativo da eficácia de um *site*.

A lista a seguir mostra algumas das informações que se pode monitorar:

A origem do usuário: um usuário pode ter chegado a um *site* de várias formas. Se um *site* é bem conhecido, é provável que o usuário tenha ele como a página inicial de seu navegador. Isso acontece muito no caso de provedores de Internet. Infelizmente, não existe uma maneira de saber, a partir de um *log*, se o *site* está ou não configurado para ser a página inicial de um navegador.

Uma outra forma bem mais comum de um usuário chegar a um *site* é a partir de uma pesquisa em um *site* de busca como o Yahoo ou o Alta Vista. Nesse caso, pode-se saber, através do *log*, as palavras de consulta que foram digitadas pelo usuário. Essa informação pode ser utilizada para identificar palavras-chave específicas a serem incluídas em *metatags*[11] para *spiders* (Apêndice C).

Outra forma utilizada para chegar a um *site* é o *bookmark*[12] do navegador. Esse tipo de origem também não pode ser identificada no *log*. No entanto, seria de muita valia conhecer se a origem foi essa pois, normalmente, indica um alto nível de interesse do usuário no *site*.

[10] *Webmaster* é a pessoa que cuida da criação e/ou manutenção de um *site*.

[11] *Metatags* é uma *tag* especial de HTML utilizada para descrever palavras-chave ou títulos para *sites* de busca.

[12] *Bookmark* é uma lista de *sites* que o usuário mantém em seu navegador e que considera interessantes.

Por fim, um *site* pode ser alcançado como resultado de um clique em um *link* de outro *site*. Nesse caso, o *site* de referência quase sempre é identificado nos dados do *log*.

Identificação de sessão: uma das informações cruciais para se montar um *Data Webhouse* é a identificação da sessão do usuário. O protocolo básico para a WWW, o *Hypertext Transfer Protocol* (HTTP) (Group 1999a), não oferece informações de estado, isto é, ele não guarda qualquer tipo de informação sobre o estado da conexão entre o navegador e o servidor, não permitindo, assim, a identificação da sessão. A identificação da sessão pode ser estabelecida de outras formas, discutidas a seguir:

- Em muitos casos, as ações individuais abrangendo uma sessão podem ser consolidadas por meio da intercalação de registros de *log* do mesmo *host*[13] e contíguas no tempo. Se o *log* tiver várias entradas com o mesmo *host* de origem em um curto período de tempo (por exemplo, uma hora), é razoável supor que os registros pertencem a uma mesma sessão. A heurística normalmente utilizada é a de que, se não houver nenhum registro do *host* em 30 minutos, então os próximos registros provenientes desse *host* formam uma nova sessão.
- Um outro método, mais satisfatório, é permitir que o servidor *Web* coloque um *cookie*[14] de nível de sessão no navegador do usuário.
- O protocolo HTTP possui uma camada de segurança básica (Group 1999b), na qual o usuário deve identificar-se com um *login*[15] e uma senha.
- A geração dinâmica de páginas pode tentar manter a identificação da sessão do usuário usando uma variável que é passada de página para página. O incoveniente desse método é que ele requer um controle perfeito sobre a geração das páginas.
- Por fim, pode-se colocar um *cookie* persistente na máquina do usuário que não é excluído pelo navegador quando a sessão termina.

Identificação do usuário: identificar um usuário específico é um dos maiores problemas com que se defronta um projetista de *site*, pois:

- Os usuários da *Web* desejam ser anônimos. Eles querem ter a liberdade de utilizar a *Web* sem se preocupar se alguém está os observando ou se seu endereço de correio eletrônico está sendo dado para algum *spammer*[16].

[13] *Host* é qualquer computador na Internet com um nome de domínio.

[14] *Cookie* é um registro colocado no computador do usuário, pelo navegador, em resposta a uma solicitação do servidor *Web*.

[15] *Login* é um identificador de usuário para um sistema.

[16] *Spammer* é um indivíduo responsável pela ampla distribuição de correspondência indesejável e não solicitada na Internet por meio do envio de uma mensagem a um grande número de destinatários.

- Caso seja solicitada a identidade do usuário, é possível que ele minta. Não se deve confiar nas informações de identidade fornecidas pelos usuários. É melhor sempre monitorar os usuários utilizando o identificador de sessão, mantendo o seu anonimato.
- Não é possível ter certeza sobre qual o membro de uma família que está visitando um *site*. Uma vez que vários membros de uma família podem utilizar o mesmo computador, não é possível saber quem está navegando no *site*, mesmo que um *cookie* seja usado, pois o mesmo apenas identifica o computador sendo utilizado e não a pessoa. Muitas vezes pensa-se que se sabe algo sobre um usuário quando na verdade se sabe sobre a família dele.
- Um usuário nem sempre está no mesmo computador. Ele pode estar utilizando o computador do escritório ou de sua casa para acessar o mesmo *site*. É difícil estabalecer uma identidade para esses usuários a menos que eles utilizem um *login* comum. Se eles fornecerem um *login* em casa e outro no escritório, não haverá como saber que eles são da mesma pessoa.

Ponto de entrada: há muitas formas de um usuário chegar a um *site*, conforme visto no item "A origem do usuário" desta lista. Tão importante quanto saber de onde o usuário veio é saber por qual página ele entrou no *site*. As informações de ponto de entrada são importantes para *marketing* e projeto pois toda página muito utilizada para entrada deve convidar o usuário a explorar o *site* por inteiro.

Permanência: a permanência é o período de tempo em que o usuário realmente tem uma página da *Web* visível no seu navegador. Ela será o tempo entre duas solicitações de arquivos HTML menos o tempo requerido para fazer a transferência do conteúdo auxiliar da primeira página (imagens, por exemplo). Se o tempo de permanência for curto, pode-se supor que o conteúdo da página é irrelevante para o usuário. Se for negativo, significa que o usuário interrompeu o carregamento da página antes que este fosse concluído. Se o tempo de permanência for muito longo, pode-se supor que o usuário abandonou o *site*. Mas também pode ser o caso do usuário estar revisitando uma página do *site* que esteja no *cache*[17] do navegador, como a ser comentado na Seção 4 na página 28. Se a página contém algum arquivo de audio ou vídeo o tempo de trasferência desse arquivo deve ser levado em conta, bem como o tempo que o usuário levará visualizando esse vídeo ou ouvindo esse áudio.

Consulta: as palavras que um usuário digita para fazer uma pesquisa em um *site*[18] podem

[17] *Cache* é um espaço separado para armazenar cópias temporárias de objetos de maneira que, se estes forem solicitados novamente, não seja necessário fazer uma nova transferência, podendo ser obtidos localmente.

[18] Nesse caso não é a pesquisa em um *site* de busca, mas sim a pesquisa no próprio *site*.

dizer muito sobre as preferências do usuário bem como sobre a usabilidade do *site*. Se um usuário chega às informações desejadas após uma ou duas consultas, então tem-se um sistema efetivo de indexação (veja Apêndice C).

Para monitorar essas pesquisas, precisa-se capturar as palavras-chave de pesquisa e o resultado, ou seja, se o usuário foi para uma página que ele pesquisou e permaneceu lá por um tempo razoável.

Navegação no *site*: a maneira pela qual um usuário navega por um *site* pode ser utilizada para ajustá-lo e otimizá-lo. Em um extremo, tem-se o usuário *hit-and-run* que sempre vai para a mesma página do *site*, sem explorar muito além. No outro extremo tem-se o usuário que faz um passeio aleatório e não-direcionado pelo *site* (*window-shopping*) somente para verificar qual o conteúdo. Na Figura 3.3 são ilustrados esses padrões de navegação.

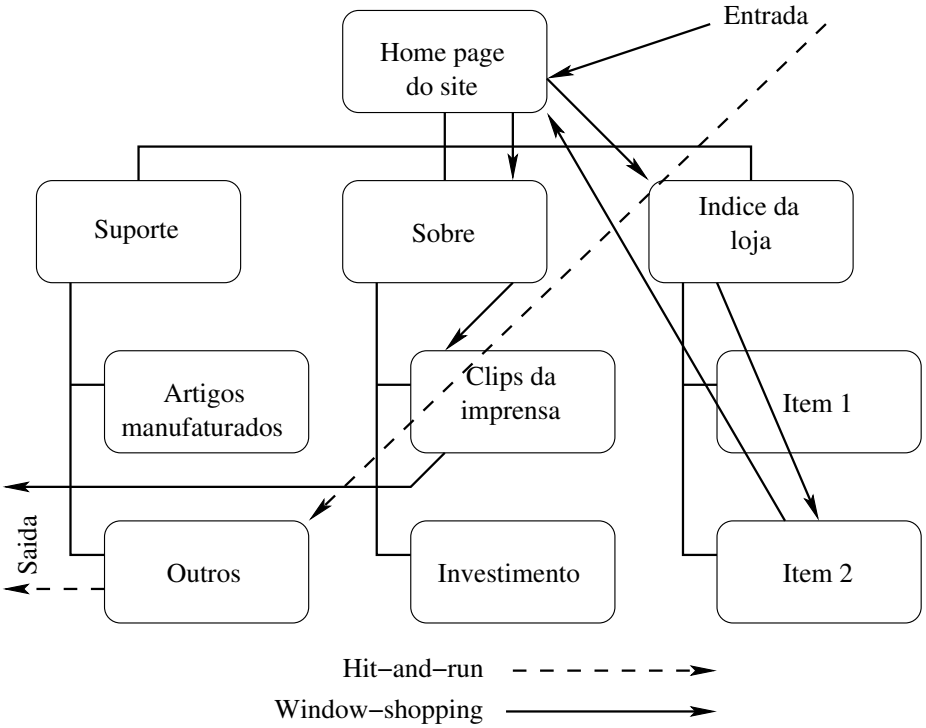


Figura 3.3: Os dois tipos de usuários

Ponto de saída: quando um usuário sai de um *site*, ele não deixa qualquer rastro porque o HTTP é um protocolo que não guarda informações de estado. Deve-se assumir que, se o usuário não fez qualquer requisição de conteúdo em, por exemplo, 30 minutos, ele foi embora. A última página solicitada é considerada o ponto de saída.

3.4 Apoio à Tomada de Decisões

Uma vez que o *Data Webhouse* foi construído, existem algumas decisões que se gostaria de tomar com base nos dados comportamentais. Para isso, é importante lembrar da seguinte progressão:

$$\text{dados} \rightarrow \text{informações} \rightarrow \text{conhecimentos} \rightarrow \text{decisões}$$

Os dados por si mesmos não apresentam grande utilidade. Entretanto, quando os dados são organizados coerentemente e pode-se perceber padrões, então tem-se algo mais útil: as informações. Mas mesmo assim, ver e descrever padrões não leva à ação. Quando pode-se identificar causa, efeito e correlação, então consegue-se refinar ainda mais as informações na forma de conhecimento útil. No entanto, não adianta ter conhecimento e não agir. O resultado final de qualquer processo de KDD deve ser uma decisão tomada como resultado do conhecimento adquirido.

A seguir é dada uma lista de possíveis conhecimentos que podem ser inferidos da sequência de cliques e que podem apoiar algumas decisões. Em alguns casos, os dados da sequência de cliques podem guiar as decisões por si próprios. Em outros, eles deverão ser combinados com outros conjuntos de dados, como por exemplo os dados de transações de vendas.

Personalização do *marketing*: caso seja identificado um cliente, pode-se personalizar a interação com o mesmo quando ele retorna ao *site*. Dessa forma, identifica-se:

- Clientes de alto lucro × clientes de baixo lucro. Nesse caso é necessário identificar realmente o usuário para que se possa ter acesso ao quê e quanto ele comprou no passado.
- Cliente novos × clientes que retornam.
- Clientes que ficam com o que compram × clientes que frequentemente retornam produtos.

Agrupamento dos clientes: se houver a possibilidade de identificar um retorno, pode-se agrupar os clientes (Apêndice A) de acordo com as seguintes medidas:

Recentidade[19]**:** quantos dias se passaram desde a última vez que o cliente visitou o *site*.

Frequência: quantas vezes o cliente visitou o *site*.

[19] *Recentity*

Intensidade: a soma total das compras do cliente ou alguma outra medida quantitativa que se deseja medir.

Agrupar os cliente de acordo com essas medidas é atraente porque pode-se utilizar os dados da sequência de cliques por si próprios para realizar toda a análise. As medidas de recentidade e frequência podem, com certeza, ser obtidas a partir de um *log*, mas a medida de intensidade pode ou não estar na sequência de cliques, dependendo se o *site* foi construído com essa funcionalidade.

Essa análise pode ser muito mais sofisticada se houver uma boa base demográfica dos clientes. Também, de acordo com (Murray and Durrell 1999), é possível inferir alguns dados demográficos a partir dos acessos dos clientes.

Uma análise interessante que pode ser feita é agrupar os cliente de acordo com a sua história e, em consequência, a probabilidade de responderem a certos tipos de promoções.

Pagamento de *banners*: muitas empresas colocam anúncios, na forma de *banners*, em outros *sites* e pagam uma quantia de dinheiro por isso. Como pode-se, através da sequência de cliques, verificar de onde veio o cliente, é possível verificar por quais *banners* os "melhores clientes" vêm até o *site*. Ou melhor, saber se esses anúncios estão surtindo efeito.

Clientes que abandonam o *site*: durante uma sessão, é possível identificar um cliente que não consegue localizar o que quer. A velocidade e a amplitude das requisições de páginas é um indicador. Se o cliente estiver clicando muito rapidamente é porque ele não está lendo a página. Outro indicador é saber como ele chegou ao *site* e, especialmente, quais foram as palavras de busca que foram dadas pelo cliente, no caso da origem ter sido um *site* de busca.

Aprimoramento da eficácia do *site*: o estilo e o conteúdo de um *site* são responsáveis pela comunicação efetiva com os clientes. Algumas decisões específicas seriam:

- A escolha dos estilos das páginas. Poderiam ser: uma grande quantidade de anúncios, complexas, aparência corporativa formal e limpa, paginas muito simples, etc.
- Verificar se escolhas comuns feitas por clientes são óbvias e rapidamente localizadas.
- Identificar quais as características de uma página, ou do *site*, fazem com que um usuário o abandone.

- Verificar se o conteúdo do texto do *site* e as *metatags* estão fornecendo a informação correta para os sistemas de indexação dos *sites* de busca.

Fornecimento de produtos: quase todos os produtos e serviços concebíveis podem ser descritos em um *site*. Qualquer produto que possa ser descrito, solicitado e monitorado é um candidato a ser fornecido por um *site*. Pode-se utilizar a sequência de cliques para verificar:

- Quais as descrições de produtos que estão sendo lidas, e por quanto tempo.
- Qual a correlação entre a descrição de um produto e um produto sendo comprado.
- Quantos cliques, a partir do início da sessão, são necessários para um cliente, novo ou antigo, solicitar algum produto.
- Quantas sessões de requisição de produto são abandonadas sem conclusão.
- Com que frequência um cliente visita o *site* para monitorar a situação de seu pedido.

Verificando a lucratividade: uma vez que um negócio na *Web* tem um forte foco no cliente, é desejável conhecer:

- Quais grupos de clientes são lucrativos.
- Quais grupos de produtos são lucrativos.
- Durante que período de tempo o negócio é lucrativo.
- Quais promoções são lucrativas.

No entanto, para conhecer cada um desses ítens, a sequência de cliques não é suficiente. É necessário, também, ter-se os dados de cada fonte de renda e de custo.

3.5 Considerações Finais

Foi visto, neste capítulo, quais as ações que podem ser capturadas de um visitante navegando por um *site*. Além disso, também foi mostrado quais as decisões que podem ser apoiadas por um *Data Webhouse* se uma perfeita representação dos dados da sequência de cliques estiver disponível.

As fontes de dados coletadas pelo enorme fluxo de sequências de cliques estão se tornando os maiores bancos de dados conhecidos, superando até mesmo os grandes e famosos exemplos das empresas de telecomunicações e seguros. Quando a *Web* é trazida para o

warehouse, essas massa enorme de dados é trazida junto, de forma que pode-se analisá-la, adaptá-la e combiná-la com outras fontes de dados existentes no *Data Warehouse*.

Capítulo 4

Construindo um *Data Webhouse*

4.1 Considerações Iniciais

A fonte de dados que alimenta o *Data Webhouse* consiste da sequência de cliques que um usuário efetua num *site*, ou seja, os registros de *log* produzidos pelo sevidor *Web* toda vez que uma requisição HTTP[1] é completada. Neste capítulo, o conteúdo da sequência de cliques é analisado em maiores detalhes.

4.2 Interação Cliente-Servidor

Entender as interações entre um navegador e um servidor *Web* é essencial para compreender a fonte e o significado dos dados da sequência de cliques. Como já mencionado, o navegador e o *site* da *Web* interagem um com o outro através da Internet utilizando o protocolo de comunicação HTTP.

Quando se navega na *Web*, muitas coisas estão acontecendo para que uma página requisitada seja transferida do servidor para a máquina do usuário e, finalmente, seja mostrada em seu navegador. Quando se digita, no navegador, um endereço de uma página que contenha apenas texto, os seguintes passos, ilustrados na Figura 4.1, são executados:

1. O navegador procura pelo servidor `www.nossosite.com` e conecta-se a ele utilizando a porta[2] *default* 80, caso nenhuma outra tenha sido indicada.

[1] Uma requisição HTTP é qualquer pedido que o navegador faz para um servidor *Web*.

[2] Uma porta em um servidor é a maneira utilizada para se identificar com qual programa deseja-se comunicar. Assim, um servidor *Web*, normalmente, é identificado pela porta 80, um servidor de FTP é, normalmente, identificado pela porta 21, e assim por diante.

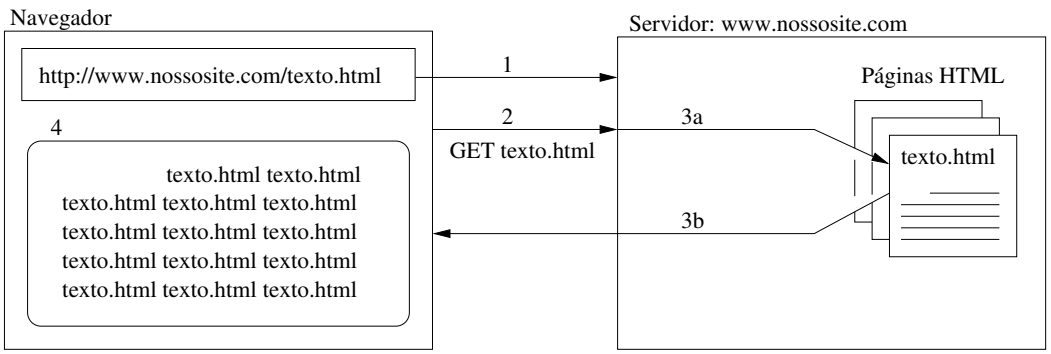


Figura 4.1: Transferência de um arquivo HTML sem figuras através do protocolo HTTP

2. Uma vez conectado, o navegador envia uma requisição para o servidor pedindo a página `texto.html`. A forma pela qual é feita essa requisição segue o protocolo HTTP, mencionado anteriormente.

3. (a) O servidor, de posse do nome do arquivo requisitado, procura-o em seu disco.
   (b) Uma vez achado, ele envia o conteúdo deste arquivo para o navegador. Deve-se observar que o protocolo HTTP ainda está em efeito. Ele define, por exemplo, o que o servidor deve enviar para o navegador caso o arquivo requisitado não exista em seu disco.

4. Neste momento, o arquivo requisitado já foi transferido e seu conteúdo encontra-se na memória do computador do usuário. O navegador lê esse conteúdo interpretando-o, fecha a conexão com o servidor e mostra o arquivo para o usuário. Observa-se que, normalmente, o navegador primeiro interpreta o arquivo para depois fechar a conexão. Isso acontece para que ele aproveite a conexão aberta para mandar outras requisições caso isso se faça necessário — caso de arquivos HTML que contenham figuras, por exemplo.

Esse cenário traduz o funcionamento básico da *Web*. No caso de um arquivo HTML que contenha texto e figuras, o funcionamento é basicamente o mesmo, com a diferença que, após o navegador ter lido e interpretado o arquivo HTML e detectado figuras, ele faz novas requisições independentes para o servidor, como mostrado na Figura 4.2. Uma vez que as figuras foram transferidas, elas podem ser mostradas nos seus devidos lugares, ou seja, no lugar do texto onde o arquivo HTML indicou que haviam figuras.

Resumindo, o protocolo HTTP é um protocolo do tipo *request/response*, ou seja, o navegador faz requisições e o servidor responde a elas. Além disso, o HTTP é, também, um protocolo *stateless*, isto é, ele não guarda nenhum tipo de informação sobre o estado da conexão entre o navegador e o servidor, de forma que, depois de uma requisição ser atendida, se o mesmo usuário fizer outra requisição o servidor não se lembrará que se trata do

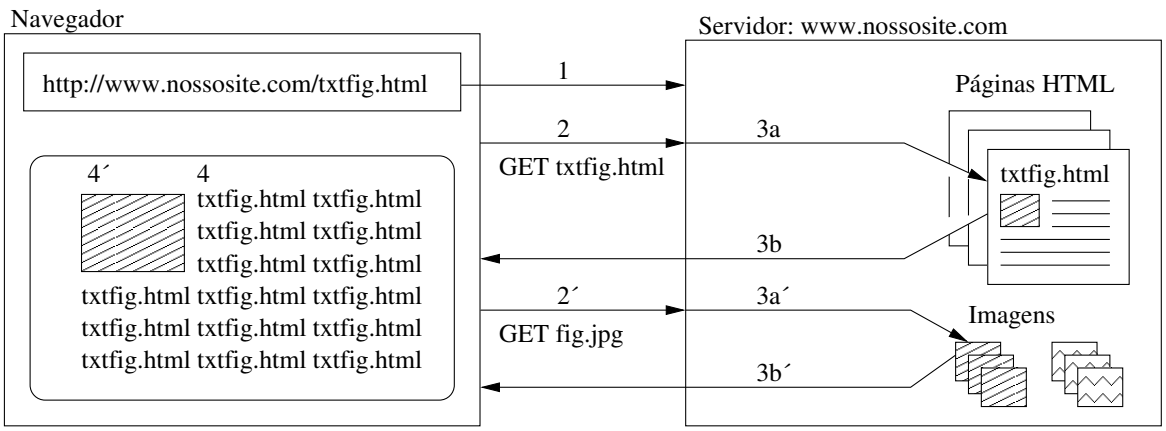


Figura 4.2: Transferência de um arquivo HTML com figuras através do protocolo HTTP

mesmo usuário, tratando-o como um novo cliente. Em outras palavras, isso significa que o usuário não tem que se identificar para o servidor (por meio de um *login*, por exemplo) e recupera vários documentos por meio de uma só sessão (identificada pelo *login*). Ao contrário, ele não precisa se identificar e deve fazer uma requisição separada para cada arquivo que deseja recuperar, seja um arquivo de texto, HTML, imagens ou qualquer outro tipo (no caso de *downloads*, *applets* Java (Mello, Chiara, and Villela 2002) e outros).

Isso não quer dizer que o HTTP seja um protocolo ruim. Pelo contrário, isso faz com que o protocolo seja simples de se entender e de se implementar. O único problema que isso implica está no caso de querermos aplicar técnicas de Mineração de Dados no *log* do servidor, pois informações muito úteis estarão faltando nos dados como, por exemplo, quem é o usuário que está acessando o servidor.

É importante notar que a única ação humana consiste em digitar um endereço no navegador ou, em outros casos, clicar[3] em um *link* de uma página. Todas as demais ações são interações computador-computador desencadeadas pelo processamento do documento que foi recuperado inicialmente (no caso de um documento que tenha figuras ou outros arquivos auxiliares).

4.2.1 Servidores *Proxy*

Quando um navegador faz uma solicitação HTTP, nem sempre ela é satisfeita pelo servidor *Web*, mas sim por um servidor *proxy*. Isso se deve ao fato de que muitos provedores de Internet utilizam servidores *proxy* para reduzir o tráfego na Internet. Esses servidores são utilizados para armazenar, em *cache*, os conteúdos freqüentemente solicitados. Na Figura 4.3 é ilustrado esse caso.

[3]Apertar o botão do *mouse* após colocar o cursor no ponto desejado.

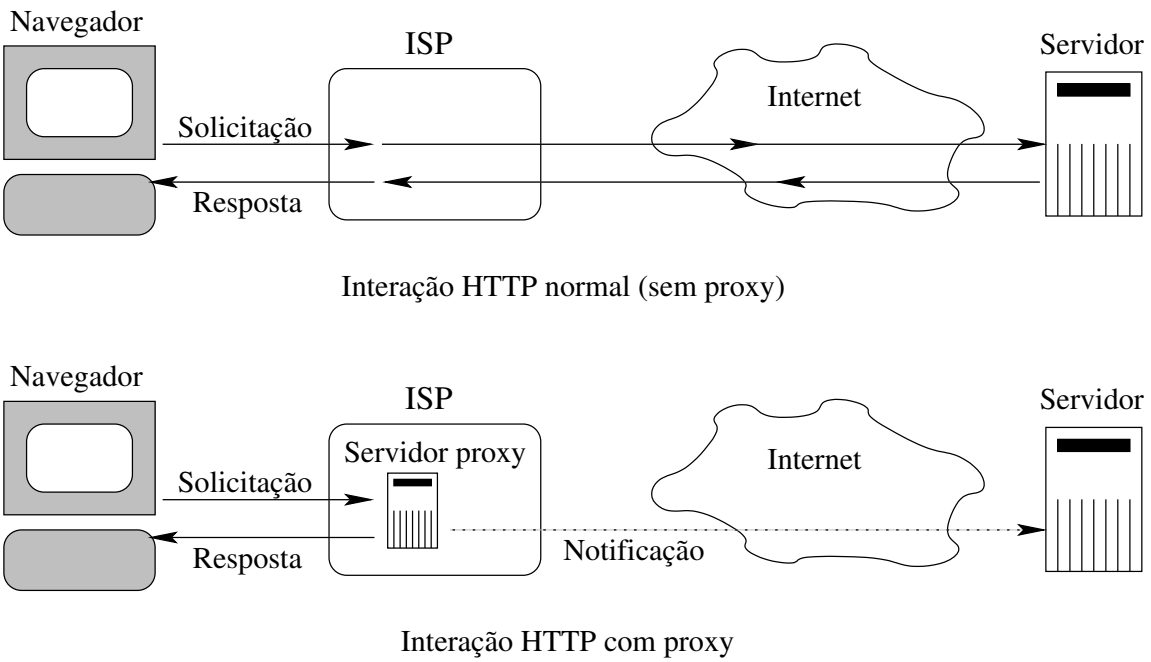


Figura 4.3: Funcionamento de um servidor *proxy*

A utilização de um servidor *proxy* introduz um problema no registro dos acessos no arquivo de *log* pois, muitas vezes, os servidores *proxy* não notificam adequadamente o servidor *Web* de que a requisição foi satisfeita por ele. Dessa forma, o servidor *Web* fica sem saber que um acesso foi feito e essa informação não é registrada no arquivo de *log*. Uma possível solução para isso é a utilização de *metatags* "não-*proxy*" no conteúdo HTML das páginas. Mas, neste caso, o servidor *proxy* perde a sua utilidade.

4.2.2 Caches de Navegador

Os caches do navegador também introduzem incertezas na tentativa de monitorar todos os eventos que ocorrem durante uma sessão de um usuário. A maioria dos navegadores armazena uma cópia de objetos recuperados recentemente (páginas, figuras, etc.) em um *cache* local. Assim, em vez de obter o arquivo do servidor, o navegador o recupera do *cache* e o servidor *Web* não fica sabendo disso, não podendo registrar o acesso.

Isso significa que não se pode ter certeza de se ter um mapa completo das ações do usuário. Na melhor das hipóteses é possível fazer uma representação da sessão por meio de uma árvore com cada folha sendo um arquivo transferido e marcado com a data e a hora em que foi solicitado.

Um outro tipo de incerteza surge quando um usuário executa vários navegadores para acessar um mesmo site. Nesse caso, a informação da seqüencia de acessos perde-se. Para entender o porquê isso acontece, basta imaginar dois navegadores abertos mostrando uma

mesma página do site. O usuário pode clicar, então, em dois *links* diferentes, um em cada página. Ao se olhar para o *log*, os dois acessos são vistos, mas não se sabe se o usuário clicou em um *link*, voltou para a página original (que seria recuperada do cache do navegador) e, em seguida, clicou no outro *link*. A situação tende a piorar porque, provavelmente, o usuário continuará a fazer acessos através dos dois navegadores independentes, ou então pode abrir novas janelas.

A conclusão disso tudo é a de que não se pode ter certeza de se ter um registro completo de todas as ações do usuário. Entretanto, apesar dessas incertezas, é possível extrair bastante informações dos arquivos de *log* de servidores Web (Kimball and Merz 2000). Em (Haight and Megarity 1998) pode ser encontrado um levantamento do que pode e o que não pode ser extraído de um arquivo de *log*.

4.3 *Logs* de Servidores *Web*

Todos os servidores *Web* têm a capacidade de registrar, em um arquivo de *log*, a sua interação com os navegadores dos usuários que acessam o *site*. Toda vez que um servidor responde a uma solicitação HTTP, ela é registrada no arquivo de *log*. É importante notar que, apesar de um registro ser feito para cada requisição, o servidor estará atendendo várias solicitações de vários usuários simultaneamente. Por isso, as entradas para uma sessão particular, *i.e.* todas as requisições feitas por um determinado usuário, não são contíguas. Os registros individuais de uma sessão estarão espalhadas por todo o arquivo de *log* do servidor e deverão ser reunidos antes que uma análise completa de uma sessão possa ser concluída. Na Figura 4.4 é mostrado um fragmento de um arquivo de log e parte de uma sessão de usuário (em cinza).

Tendo-se isso em mente e observando os passos necessários na transferência de um arquivo, ilustrados na Figura 4.1 na página 29, tem-se que as seguintes informações podem ser armazenadas no *log*[4]:

host endereço do computador do usuário. Essa informação vem do protocolo TCP/IP que, como já mencionado, é o protocolo utilizado na Internet e que possibilita a conexão entre dois computadores. O servidor precisa conhecer o endereço do cliente para poder enviar a resposta através da Internet.

A maioria dos computadores em rede não têm endereços de Internet fixos. Em vez disso, o endereço é atribuído dinamicamente para o computador quando o usuário

[4]Os termos utilizados para nomear as informações que são armazenadas no *log* estão em inglês porque essa é a forma usual de referenciá-los.

```
143.107.183.226 - - [02/Jul/2002:15:47:20 -0300] "GET /manuals/sce183/proc.html HTTP/
200.148.1.112 - - [02/Jul/2002:15:47:32 -0300] "GET /~poscomp/ HTTP/1.1" 200 4410 "-"
64.152.75.36 - - [02/Jul/2002:15:47:33 -0300] "GET /~wvlnunes/sma182/medias182.html H
143.107.183.226 - - [02/Jul/2002:15:47:34 -0300] "GET /manuals/sce183/proc5.html HTTP
64.152.75.36 - - [02/Jul/2002:15:47:35 -0300] "GET /~gnonato/ED/node28.html HTTP/1.0"
200.148.1.112 - - [02/Jul/2002:15:47:37 -0300] "GET /~poscomp/oque.html HTTP/1.1" 200
143.107.183.226 - - [02/Jul/2002:15:47:40 -0300] "GET /manuals/sce183/proc1.html HTTP
143.107.232.61 - - [02/Jul/2002:15:47:40 -0300] "GET /intranet HTTP/1.1" 301 326 "-"
143.107.232.61 - - [02/Jul/2002:15:47:40 -0300] "GET /intranet/ HTTP/1.1" 304 - "-" "
143.107.232.61 - - [02/Jul/2002:15:47:41 -0300] "GET /intranet/index.php HTTP/1.1" 30
143.107.232.61 - - [02/Jul/2002:15:47:41 -0300] "GET /intranet/sistemas.php HTTP/1.1"
64.152.75.36 - - [02/Jul/2002:15:47:43 -0300] "GET /manuals/UNIX/mail_five.3.2.html H
143.107.232.61 - - [02/Jul/2002:15:47:44 -0300] "POST /intranet/index.php HTTP/1.1" 2
200.221.176.22 - - [02/Jul/2002:15:47:50 -0300] "GET /manuals/HTML/internet.html HTTP
200.171.165.52 - - [02/Jul/2002:15:47:53 -0300] "GET /manuals/HTML/recursos.html HTTP
200.190.90.187 - - [02/Jul/2002:15:47:55 -0300] "GET /manuals/sce229/ HTTP/1.1" 200 2
200.221.176.22 - - [02/Jul/2002:15:47:57 -0300] "GET / HTTP/1.1" 200 20361 "-" "Mozil
143.107.183.226 - - [02/Jul/2002:15:48:05 -0300] "GET /manuals/sce183/proc2.html HTTP
200.210.171.23 - - [02/Jul/2002:15:48:06 -0300] "GET /~esteves/piraju/ HTTP/1.0" 404
200.171.249.188 - - [02/Jul/2002:15:48:15 -0300] "GET /~andre HTTP/1.1" 301 324 "-" "
200.171.249.188 - - [02/Jul/2002:15:48:16 -0300] "GET /~andre/ HTTP/1.1" 304 - "-" "M
64.152.75.36 - - [02/Jul/2002:15:48:18 -0300] "GET /manuals/ncsa/ HTTP/1.0" 200 1175
143.107.183.226 - - [02/Jul/2002:15:48:21 -0300] "GET /manuals/sce183/procrot2.html H
143.107.183.226 - - [02/Jul/2002:15:48:27 -0300] "GET /manuals/sce183/proc3.html HTTP
200.171.249.188 - - [02/Jul/2002:15:48:27 -0300] "GET /~andre/AULAS/POO/index.htm HTT
64.222.18.76 - - [02/Jul/2002:15:48:31 -0300] "GET /~mac/jbe/PiPh.html HTTP/1.0" 404
200.171.249.188 - - [02/Jul/2002:15:48:32 -0300] "GET /~andre/AULAS/POO/Notas.htm HTT
216.169.198.9 - - [02/Jul/2002:15:48:32 -0300] "GET /~mac/jbe/PiPh.html HTTP/1.0" 404
132.163.193.220 - - [02/Jul/2002:15:48:32 -0300] "GET /~mac/jbe/PiPh.html HTTP/1.0" 4
24.52.229.209 - - [02/Jul/2002:15:48:32 -0300] "GET /~mac/jbe/PiPh.html HTTP/1.0" 404
198.108.59.203 - - [02/Jul/2002:15:48:32 -0300] "GET /~mac/jbe/PiPh.html HTTP/1.0" 40
195.92.194.12 - - [02/Jul/2002:15:48:32 -0300] "GET /~mac/jbe/PiPh.html HTTP/1.1" 404
207.21.82.20 - - [02/Jul/2002:15:48:32 -0300] "GET /~mac/jbe/PiPh.html HTTP/1.0" 404
64.152.75.36 - - [02/Jul/2002:15:48:33 -0300] "GET /~szani/sma303/sma30302/node37.htm
128.251.168.91 - - [02/Jul/2002:15:48:35 -0300] "GET /~mac/jbe/PiPh.html HTTP/1.0" 40
200.171.165.52 - - [02/Jul/2002:15:48:39 -0300] "GET /manuals/HTML/tabelas.html HTTP/
143.107.183.226 - - [02/Jul/2002:15:48:39 -0300] "GET /manuals/sce183/proc4.html HTTP
200.171.229.137 - - [02/Jul/2002:15:48:42 -0300] "GET /manuals/HTML/okform.html HTTP/
200.171.165.52 - - [02/Jul/2002:15:48:43 -0300] "GET /manuals/HTML/tabasico.html HTTP
143.107.183.222 - - [02/Jul/2002:15:48:44 -0300] "GET / HTTP/1.1" 304 - "-" "Mozilla/
143.107.232.61 - - [02/Jul/2002:15:48:49 -0300] "POST /intranet/login.php HTTP/1.1" 3
143.107.232.61 - - [02/Jul/2002:15:48:49 -0300] "GET /intranet/sistemas.php HTTP/1.1"
```

Figura 4.4: Exemplo de log de um servidor Web

faz uma conexão com o seu provedor através do *modem*. Mesmo que o endereço seja dinâmico, ele permanece o mesmo durante uma sessão e pode ser utilizado para "amarrar" os eventos de uma sessão.

ident alguns clientes possuem, em sua máquina, um programa chamado `identd` (*identification daemon*), que possibilita que o servidor *Web* saiba informações sobre o usuário que o está acessando. Caso isso aconteça, essa informação será armazenada. No entanto, esse esquema de identificação raramente é utilizado.

authuser (*authenticated user*) o protocolo HTTP permite um tipo de segurança básica, mencionado na Seção 3.3.2 na página 19, na qual o usuário deve se identificar com um *login* e uma senha. Caso a requisição tenha sido de um arquivo protegido por esse tipo de autenticação, o nome do usuário será registrado e poderá ser utilizado para associar os registros de *log* desse usuário.

date também são registradas a data e a hora em que foi feita a solicitação de um arquivo.

request o nome do arquivo que foi requisitado é outra informação que é armazenada no *log*. Na verdade, o `request` é mais que isso: ele contém a *string* de requisição que inclui, além do nome do arquivo requisitado, o método que deve ser utilizado na recuperação do mesmo e a versão do protocolo sendo utilizada. Um exemplo de uma *string* de requisição seria `GET texto.html HTTP/1.1`, conforme visto[5] na Figura 4.1 na página 29.

`GET` é o método do HTTP sendo utilizado pelo navegador para a requisição de um arquivo. A string `texto.html` é a URL do arquivo sendo solicitado, ou seja, o endereço do arquivo. Por último, `HTTP/1.1` é a versão do protocolo que está sendo utilizada pelo navegador. Os dois métodos mais comumente utilizados do HTTP são o `GET` e o `POST`. Os dois diferem apenas na forma como as informações de uma página são enviadas para o servidor.

status o protocolo HTTP retorna para o cliente, além do arquivo requisitado, um código de *status*, indicando o sucesso da requisição ou, em outras situações, alguma anomalia como, por exemplo, "arquivo não encontrado", "autorização negada", e outros. Na Tabela 4.1 são listados os possíveis códigos que podem ser retornados. Uma descrição mais refinada dos códigos de *status* pode ser encontrada em (Group 1999a).

bytes o número de *bytes* retornado para o cliente, excluindo os cabeçalhos do protocolo HTTP, ou seja, o tamanho do arquivo requisitado.

[5]Na figura, a *string* de requisição está incompleta (faltando a versão do protocolo) devido ao espaço limitado.

Código	Descrição	Código	Descrição
100	Continuar	404	Não localizado
200	OK	405	Método não permitido
201	Criado	406	Não aceitável
202	Aceito	407	Autenticação de *proxy* exigida
203	Informações não autorizadas	408	Tempo limite da solicitação
204	Nenhum conteúdo	409	Conflito
205	Redefinir conteúdo	410	Feito
206	Conteúdo parcial	411	Comprimento exigido
300	Várias escolhas	412	Pré-condição falha
301	Movido permanentemente	413	Entidade da solicitação muito grande
302	Movido temporariamente	414	Solicitação-URI muito grande
303	Veja outro	425	Tipo de mídia não compatível
304	Não modificado	500	Erro interno do servidor
305	Utilizar *proxy*	501	Não implementado
400	Solicitação errada	502	*Gateway* errado
401	Não autorizado	503	Serviço indisponível
402	Pagamento exigido	504	Tempo limite do *gateway*
403	Proibido	505	Versão de HTTP não compatível

Tabela 4.1: Códigos de *status* de HTTP

user-agent nome e versão do navegador sendo utilizado pelo usuário. Esse dado também pode trazer a informação sobre qual sistema operacional o navegador está sendo executado.

referer quando clica-se num *link* de uma página que está sendo mostrada no navegador, este pode enviar para o servidor *Web* o endereço da página atualmente sendo mostrada. Desta forma, pode-se saber em que página havia um *link* para um arquivo armazenado no servidor que está realizando a requisição. Na Figura 4.5 é descrito um navegador mostrando a página `hub.html` que encontra-se no servidor `www.outrosite.com`. Essa página possui um *link* para o arquivo `texto.html` que encontra-se no servidor `www.nossosite.com`. Se o usuário clicar no *link*, uma requisição será feita para o servidor `www.nossosite.com` e ele poderá saber que esse *link* estava no endereço `http://www.outrosite.com/hub.html`.

Normalmente, os servidores Web estão configurados para armazenar as requisições num formato chamado CLF (*Common Log Format*) (Hallam-Baker and Behlendorf 1996). Um arquivo no formato CLF contém uma linha separada para cada requisição. Uma linha é composta por vários *tokens* separados por espaços. Esses *tokens* são exatamente as informações apresentadas anteriormente, ou seja:

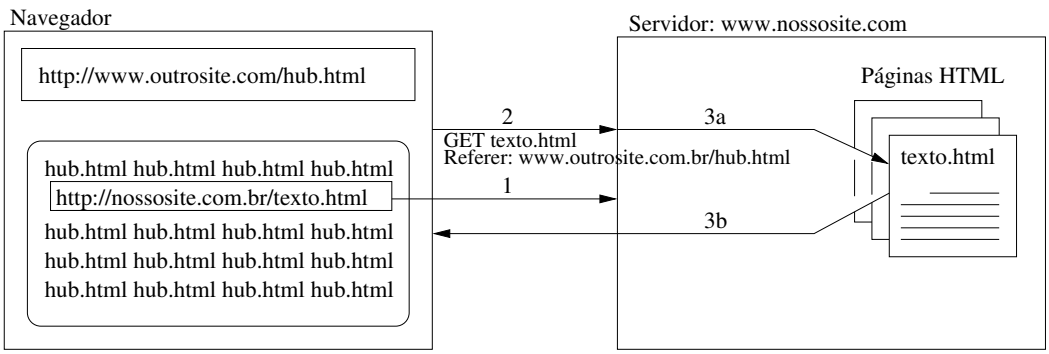


Figura 4.5: O protocolo HTTP também carrega outras informações, como o `referer`

```
host ident authuser date request status bytes
```

Se um *token* não possui valor definido, então esse valor é representado por um hífen (`-`). Normalmente, os valores de `ident` e `authuser` estarão faltando por serem pouco utilizados.

Muitos *webmasters* se utilizam de ferramentas como o Webalizer[6] para obter estatísticas de acessos aos *sites* pelos quais são responsáveis. Por isso, a maioria dos servidores *Web* são configurados para armazenar, também, os *tokens* `user-agent` e o `referer`, de forma que análises mais detalhadas possam ser feitas por esse tipo de ferramenta. Nesse caso, diz-se que o *log* está no formato ECLF (*Extended Common Log Format*). Na Tabela 4.2 é mostrado um resumo de alguns dados típicos registrados nos *logs* da maioria dos servidores *Web*.

Dado	CFL	ECFL	Descrição
host	√	√	Endereço IP do cliente
ident	√	√	Informações de identidade fornecidas pelo cliente
authuser	√	√	Se a solicitação foi para um documento protegido por senha, então esse é o identificador de usuário utilizado na solicitação
date	√	√	A data e a hora em que solicitação chegou no servidor
request	√	√	A primeira linha da solicitação do cliente
status	√	√	Código de *status* retornado para o cliente
bytes	√	√	Número de bytes retornado para o cliente excluindo os cabeçalhos de HTTP
referer		√	URL do servidor de referência
user-agent		√	Nome e versão do navegador utilizado pelo cliente

Tabela 4.2: Dados registrados no *log* de um servidor *Web*

[6] http://www.webalizer.org

4.4 *Strings* de Consulta

Uma solicitação HTTP pode ter a forma:

```
http://www.google.com/search?q=webmining
```

Tudo o que está à direita do ponto de interrogação é considerado uma *string* de consulta. Ela é, normalmente, utilizada para enviar para o servidor o conteúdo de um formulário[7] que foi preenchido no navegador. O conteúdo da *string* de consulta é expresso na forma *variavel* = *valor*, podendo haver vários desses pares separados pelo caractere "`&`".

A *string* de consulta torna-se mais valiosa quando o site de busca a retorna como parte do registro `referer`. A seguinte entrada de *log* exemplifica isso:

```
200.191.0.100 - - [02/Apr/2001:10:26:54 -0300] "GET / HTTP/1.1" 200 2275
 "http://www.google.com/search?hl=pt&safe=off&q=ciro+gomes&lr=" "Mozilla
/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)"
```

Essa entrada mostra que o *site* recebeu uma visita que veio a partir do *site* de busca Google[8] e que o usuário, para chegar ao *site*, procurou pelas palavras "`ciro`" e "`gomes`", pois o par `q=ciro+gomes` é a parte da *string* de consulta que corresponde às palavras de busca digitadas pelo usuário.

4.5 Modelos

Muitos *sites* que geram conteúdo dinâmico colocam esse conteúdo em um modelo (*template*), ou seja, existe uma página fixa para eles onde o que muda é o conteúdo, o qual dependendo dos parâmetros passados para a página, conforme mostrado na Figura 4.6.

Deve ser observado que uma página com conteúdo dinâmico recebe parâmetros de forma igual a das *strings* de consulta, ou seja, através pares *variavel* = *valor* separados por pelo caractere "`&`". O formato da página de uma notícia, por exemplo, é sempre o mesmo. O que muda é o conteúdo. Por exemplo:

```
http://www.meusite.com/noticias/mostra_noticia.php?codigo=16430
```

[7] Um formuário é um documento HTML que contém instruções para que o usuário entre com informações. Em um *site* de busca, por exemplo, existe uma página responsável por obter as palavras de busca que o usuário deseja encontrar. Essa página é um formulário.

[8] http://www.google.com

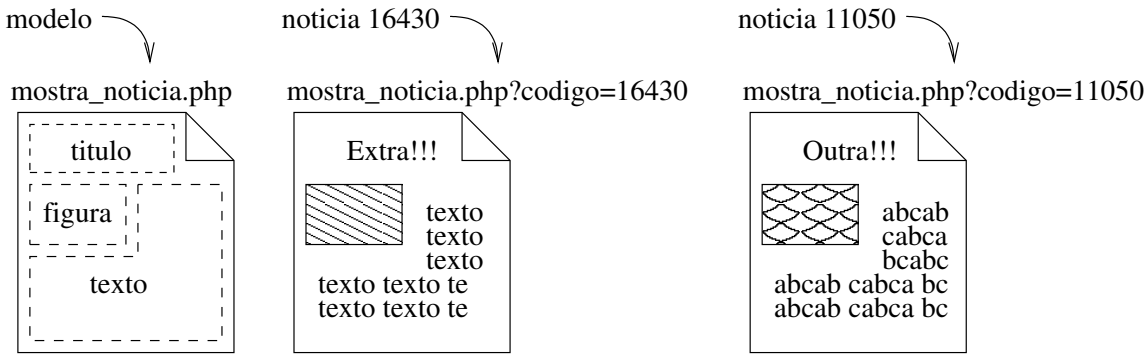


Figura 4.6: Exemplo de modelo (*template*)

identifica uma página que mostra uma notícia. É possível verificar que a notícia tem um código que é 16430. Pela análise desses tipos de endereços é possível recuperar o modelo da página e os identificadores do conteúdo.

4.6 O Processador de Sequências de Cliques

De várias maneiras, a sequência de cliques pode ser tratada por um processo do tipo

$$\text{extrair} \rightarrow \text{transformar} \rightarrow \text{carregar}$$

O processamento da sequência de cliques tem como objetivo final preparar os dados para que sejam carregados em um *Data Webhouse*. Esse processamento pode ser resumido em:

Filtrar registros não necessários: mesclar dados associados e excluir registros que não serão carregados no *Data Webhouse*. É necessários reduzir o volume de dados sem comprometer a integridade e a completitude dos mesmo.

Identificar sessões: marcar registros associados a uma única sessão e verificar se os tempos dos eventos são coerentes.

Identificar usuários: se possível, fazer a correspondência entre um usuário e um identificador de sessão. A diferença entre a a sessão e o usuário, é que a sessão sempre pertence a um único usuário enquanto que o usuário pode "criar" várias sessões visitando o *site* em dias diferentes, por exemplo.

Identificar *hosts*: Converter os endereços de IPs dos usuários e dos `referers` para o seu equivalente em texto. Dessa forma pode-se obter, por exemplo, o país de origem.

Colocar os dados em um formato único: transformar os dados das sequências de cliques para um formato bem defindo e que seja utilizável pelas ferramentas de KDD.

Na Figura 4.7 é ilustrada uma possível arquitetura, proposta por Kimball (Kimball and Merz 2000), de como pode ser feito esse processo.

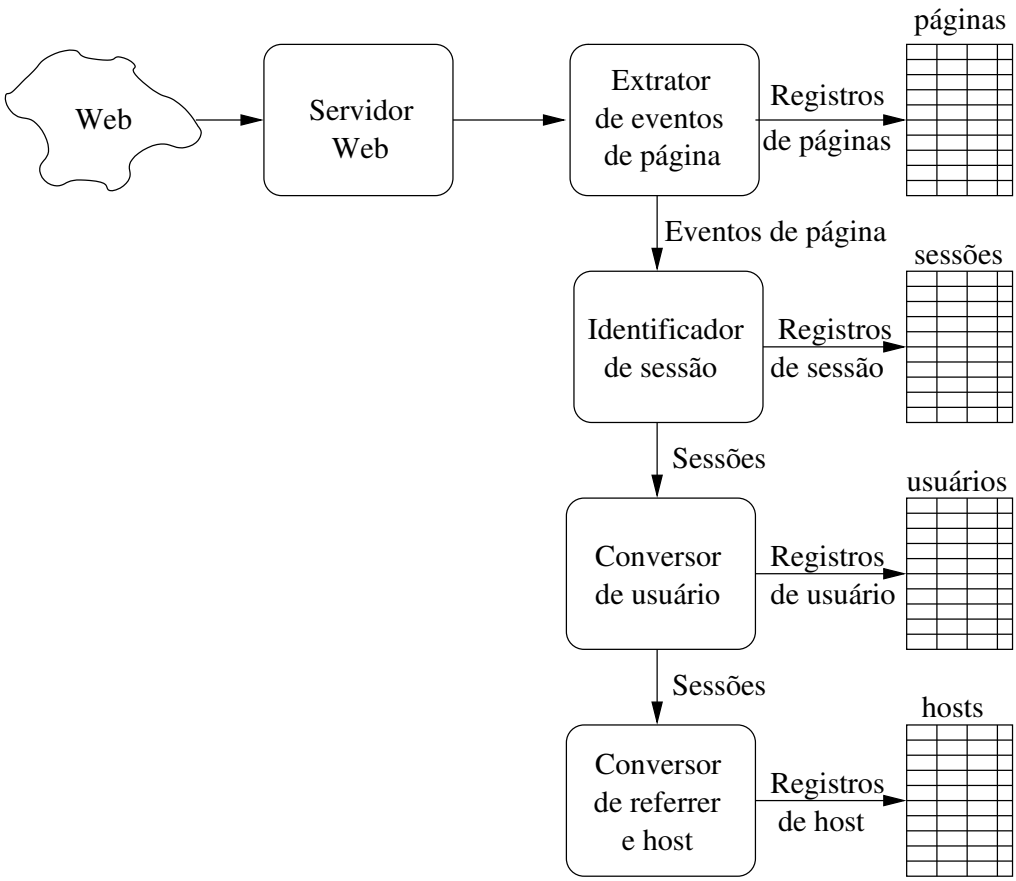


Figura 4.7: Possível arquitetura para o processador de sequências de cliques

4.6.1 Extrator de Eventos de Página

O extrator de eventos de página reúne registros do *log* e os transforma para eventos de página. Um evento de página pode ser, por exemplo, um clique em um *link*.

É importante não descartar registros de *log* que indiquem a recuperação de imagens e de outros arquivos. Eles são necessários para computar o tempo de permanência.

O extrator de eventos de página pode adicionar registros à sequência de cliques para adicionar eventos significativos como, por exemplo, a verificação dos itens do "carrinho de compras".

4.6.2 Conversor de Conteúdo

O conversor de conteúdo está relacionado com o extrator de eventos de página. Ele examina cada registro de evento de página e tenta relacionar o evento ao conteúdo específico do *site*. Mais especificamente, o conversor de conteúdo é responsável por gerar duas chaves na tabela de eventos de página: a chave da página e a chave de conteúdo, tratada na Seção 4.5 na página 37. A chave da página identifica uma página estática ou um modelo no caso de páginas geradas dinamicamente. A chave de conteúdo identifica um produto, uma notícia ou outro conteúdo específico.

4.6.3 Identificador de Sessão

O papel do identificador de sessão é reunir todos os eventos de página que ocorreram durante uma única sessão de um usuário. O identificador de sessão também é responsável por calcular o tempo de permanência do usuário. Todos esses cálculos contêm elementos de incerteza que o identificador de sessão deve tentar resolver.

É relativamente fácil identificar o tempo de início de uma sessão: para efeitos de análise, é o momento da primeira solicitação HTTP do usuário (identificado, por exemplo, pelo seu endereço). No entanto, como mencionado anteriormente, não há maneira de saber o momento exato que o usuário deixou o site. Assim, um tempo limite de inatividade determinado deverá ser utilizado para indicar o término da sessão.

Na Figura 4.8 é mostrado como pode ser calculado o tempo de permanência. Ele é calculado como o tempo entre as sucessivas solicitações HTTP para arquivos HTML menos o tempo que leva para servir a solicitação inteira, isto é, o tempo de envio da página mais o conteúdo auxiliar como, por exemplo, imagens.

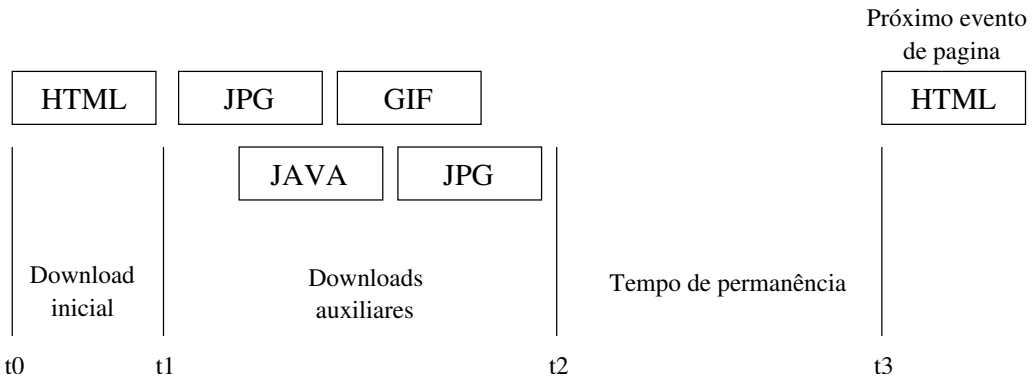


Figura 4.8: Calculando o tempo de permanência

A maioria dos navegadores e servidores permitem que várias solicitações sejam feitas ao mesmo tempo. Isso significa que os arquivos de conteúdo auxiliar serão processados em

paralelo. O tempo de envio de um arquivo pode ser estimado utilizando o número de bytes registrados na entrada do *log* e uma taxa de transferência arbitrária. Uma vez calculado o tempo de permanência para cada evento de página de uma sessão, é possível calcular o tempo de permanência total, conforme indicado na Figura 4.9.

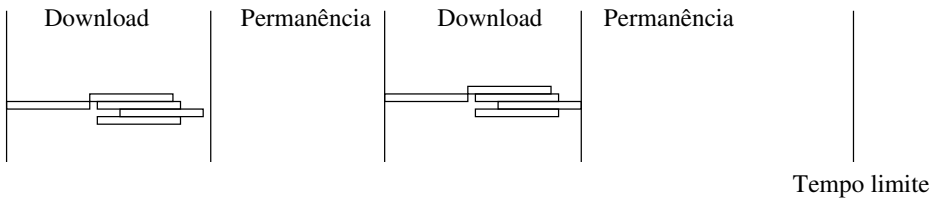


Figura 4.9: Calculando o tempo total de permanência

Quando um arquivo demora muito para ser recuperado é possível que o usuário interrompa a transferência. Se isso ocorrer, o tempo de permanência é zero, como mostrado na Figura 4.10.

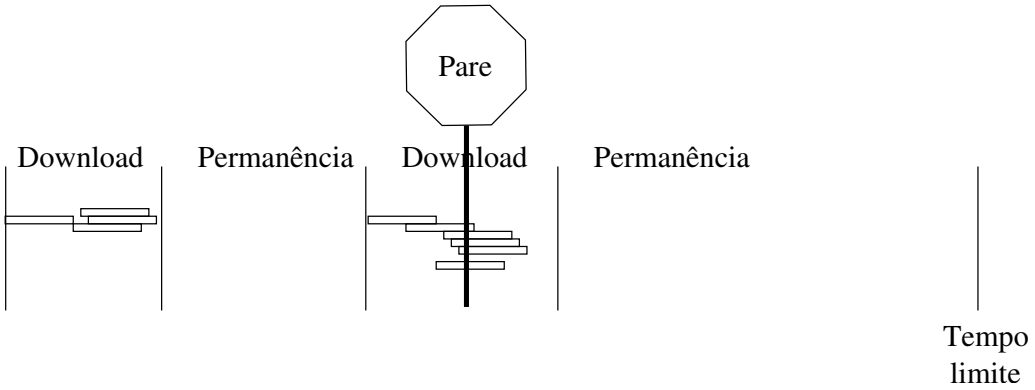


Figura 4.10: Exemplo de transferência cancelada

4.6.4 Conversor de *Hosts*

Os *hosts* e `referers` são armazenados no *log* como endereços de Internet numéricos. É interessante converter esses números para os nomes dos *hosts*. Converter nomes de *host* requer o uso de pesquisa inversa de DNS[9].

A maioria dos endereços IP não precisam ser convertidos para além da terceira parte do endereço. É suficiente saber que um endereço que começa com 152.176 pertence à America Online. O objetivo é, portanto, converter os nomes no nível necessário e reduzir a quantidade de pesquisa inversa de DNS, pois é um processo que faz uso intensivo dos recursos de rede e pode demorar alguns segundos para ser concluído.

[9] *Domains Name Service* — é um protocolo que relaciona nomes de domínios com os seus endereços. O DNS inverso serve para pesquisar o nome de domínio dado o endereço numérico.

4.7 Considerações Finais

Neste capítulo foi descrita uma arquitetura de um processador de sequência de cliques para que ele processe um *log* e o transforme numa base de dados adequada para que se possa aplicar técnicas de Mineração de Dados nessa nova base. Foram também descritos os campos que formam os registros das ações dos usuários, os quais fazem parte dos *logs* de servidores *Web*. Foi mostrado que há limitações na quantidade de informações que são registradas nos logs e, dessa forma, algumas dificuldades são encontradas ao se realizar uma mineração muito abrangente dos mesmos pois essa mineração será restringida pela limitação de informações nos dados.

No próximo capítulo são descritas alguma ferramentas existentes que processam *logs* de servidores *Web*.

Capítulo 5

Ferramentas para Análise de *Logs*

5.1 Considerações Iniciais

Nos capítulos anteriores foi descrito o processo de *Web Mining* e, mais precisamente, *Web Usage Mining*. Existem várias ferramentas *freeware* que auxiliam nesses processos. Neste capítulo são decritas duas dessas ferramentas, que têm características diferentes mas que possuem facilidades semelhantes às encontradas na maioria das ferramentas *freeware* pesquisadas. Algumas experiências realizadas com essas ferramentas utilizando *logs* do servidor *Web* do ICMC, bem como algumas alterações realizadas no código das ferramentas, também são descritas.

5.2 O Arquivo de *Log*

Com o intuito de mostrar as funcionalidades de algumas ferramentas *freeware* disponíveis para realizar análise de *logs*, bem como da ferramenta proposta neste trabalho, houve a necessidade de escolher algum arquivo de *log*. Como objeto de estudo, escolheu-se o arquivo de *log* do servidor *Web* do Instituto de Ciências Matemáticas e de Computação (ICMC[1]) da Universidade de São Paulo — São Carlos. Os motivos para essa escolha são, basicamente, dois:

1. A facilidade de se obter o arquivo propriamente dito, pois pôde-se interagir diretamente com os responsáveis pela manutenção do *site*.

2. O fato do *site* do ICMC ser bem conhecido das pessoas que estudam e trabalham no Instituto, tendo-se, então, um bom conhecimento do domínio. Deve ser lembrado que

[1] http://www.icmc.usp.br

esse é o primeiro passo para se realizar o processo de descoberta de conhecimento, conforme visto na Seção 2.3 na página 7.

Uma das primeiras decisões com relação ao arquivo de *log* é que não deveria ser realizada qualquer instrumentação do *site* para incluir informações adicionais nesse arquivo. O servidor foi configurado somente para armazenar o arquivo de *log* padrão no formato ECLF, que é o utilizado pela maioria dos *webmasters*.

Após essa decisão, foi solicitado às pessoas responsáveis pelo *site* do ICMC configurar o servidor *Web* do ICMC para armazenar os *log* no formato ECLF. Essa configuração foi feita no dia 02 de julho de 2002 e a captura prosseguiu até o dia 04 de setembro de 2002, completando um pouco mais de 2 meses de registros. Na Tabela 5.1 é mostrado um resumo das informações do arquivo de *log* capturado.

Data do primeiro registro:	02 de julho de 2002
Data do último registro:	04 de setembro de 2002

Mês	Tamanho (*bytes*)	Número de registros
Julho / 2002	467.686.905	2.475.166
Agosto / 2002	701.129.592	3.764.236
Setembro / 2002	72.524.380	382.428
Total	1.241.340.877	6.621.830

Tabela 5.1: Resumo das informações do arquivo de *log* original

Deve ser lembrado que o período escolhido coincide com o período de férias da Universidade (mês de julho) e com o primeiro mês de aula (agosto). Esse conhecimento deve ser levado em conta na análise dos resultados. Um outro evento que ocorreu nessa época foi o PosComp[2]. Esse evento, diferentemente do citado anteriomente, foi "observado" ou "descoberto" após realizar a experiência que envolve a plotagem dos grafos das sequências de cliques, como será visto posteriormente.

5.3 Ferramentas Escolhidas

São várias as ferramentas *freeware* disponíveis para realizar e/ou auxiliar no processo de *Web Usage Mining*. Essas ferramentas podem ser encontradas por meio de *sites* de busca, especialmente no Freshmeat[3], que é um *site* de catálogo de projetos.

[2]O PosComp é um tipo de "vestibular" utilizado por vários centros de Pós-Graduação no Brasil como parte do processo seletivo de cantidatos na Área de Ciências de Computação.

[3]http://www.freshmeat.net

Várias ferramentas foram analisadas, tais como Analog[4], AWStats[5], Webalizer[6], Yet Another Advanced Log Analyzer[7], Apache2Dot[8], Big Brother Log Analyzer[9], Sherlog[10], entre outras. Após essa análise, foram escolhidas duas ferramentas que possuem características diferentes. Uma delas possue funcionalidades semelhantes a maioria das outras ferramentas, mas foi escolhida entre elas por ser de execução mais rápida e gerar resultados de forma mais inteligível. Essas ferramentas são:

1. Apache2Dot
2. Webalizer

Nas próximas seções são mostradas as experiências realizadas com cada uma dessas ferramentas.

5.3.1 Apache2Dot

A ferramenta Apache2Dot consiste de um programa desenvolvido em Perl (Wall and Schwartz 1991) que tem como objetivo transformar um arquivo de *log* no formato ECLF em um arquivo com informações que permitem gerar um grafo direcionado. Esse arquivo pode ser usado como entrada para dois outros programas chamados `dot`[11] e `neato`[12], os quais fazem parte de uma ferramenta chamada GraphViz.

A ferramenta GraphViz é uma coletânea de programas capazes de gerar uma imagem que contém o desenho de um grafo descrito no arquivo de entrada gerado pelo programa Apache2Dot. O desenho de grafos enfoca o problema de visualização de informações estruturadas construindo representações geométricas dos grafos abstratos. A visualização efetiva dessas informações pode revelar características interessantes dos dados ao mesmo tempo que evita distrações e irrelevâncias.

Ambas são ferramentas *freeware* e podem ser encontradas em:

Apache2Dot: http://www.chaosreigns.com/code/apache2dot/

GraphViz: http://www.research.att.com/sw/tools/graphviz/

[4] http://www.analog.cx
[5] http://awstats.sourceforge.net
[6] http://www.webalizer.org
[7] http://verplant.org/yaala
[8] http://www.chaosreigns.com/code/apache2dot/
[9] http://bbla.sourceforge.net
[10] http://www.europeanservers.net/sherlog
[11] Cria disposições hierárquicas de grafos dirigidos.
[12] Cria disposições do tipo "mola" de grafos não direcionados.

A combinação dessas duas ferramentas permite a visualiazação dos *links* das páginas do *site* em estudo, conforme eles foram sendo clicados. Cada arquivo do *site* é representado por um nó do grafo e uma aresta orientada é desenhada partindo de cada `referer` para os arquivos que tenham sido clicados. A cor dessas arestas é proporcional à quantas vezes o *link* foi usado.

Inicialmente, ambas ferramentas foram executadas utilizando o arquivo de *log* original descrito na Tabela 5.1 na página 44. Esse arquivo de *log* foi, posteriormente, pré-processado para realizar uma limpeza retirando-se algumas informações consideradas irrelevantes, como descrito a seguir.

5.3.1.1 Limpeza

Após a execução de ambas ferramentas utilizando o arquivo de *log* original, alguns pontos foram observados. Entre eles, que seria necessário filtrar arquivos auxiliares das páginas HTML (figuras, entre outros) pois, para cada um desses arquivo, o servidor *Web* gera uma entrada tendo como `referer` a página principal, conforme pode ser visto na Figura 5.1. Nessa figura é mostrado o grafo gerado pela ferramenta onde a página principal "aponta" para figuras GIF que são objetos não interessantes para essa análise pois não são acessadas diretamente pelo usuário e sim carregadas automaticamente pelo navegador, já que fazem parte da página principal. Ou seja, essas entradas são geradas devido ao mecanismo de funcionamento do protocolo HTTP, conforme visto na Seção 4 na página 28. Alguns outros tipos de arquivos, que não influenciam na experiência, também foram retirados como, por exemplo, arquivos de documentos (pdf, doc, entre outros).

Essas informações, nesse caso consideradas sem utilidade, foram retiradas do arquivo de *log* utilizando como filtro o utilitário `grep`, presente na maioria das distribuições Linux. Os seguintes tipos de arquivos auxiliares foram retirados do *log*:

- medias (bmp, gif, jpg, png, avi, wav, mid)
- *scripts* (js, class, swf)
- *style sheets* (css)
- arquivos comprimidos (zip, gz, tar)
- documentos (ps, xml, txt, pdf, doc, ppt)
- outros (exe, pl, pcl, map)

Figura 5.1: Grafo mostrando uma página principal e a requisição de arquivos auxiliares

Mês	Tamanho (*bytes*)	Número de registros
Julho / 2002	74.974.175	413.681
Agosto / 2002	116.340.323	653.545
Setembro / 2002	11.787.867	65.611
Total	203.102.365	1.132.837

Tabela 5.2: Resumo das informações do arquivo de *log* após retirar os registros de arquivos auxiliares

Após essa limpeza, restaram no arquivo de *log* o número de registros mostrado na Tabela 5.2

Comparando o número total de registros no arquivo de *log* original — Tabela 5.1 na página 44 — e o número de registros após a primeira limpeza — Tabela 5.2 — pode-se observar que aproximadamente 5/6, *i.e.* 83% dos registros do arquivo de *log* representam, basicamente, arquivos auxiliares ou outros tipos de arquivos que não são páginas HTML geradas dinamicamente ou não.

Posteriormente, foi necessário realizar um outro tipo de pré-processamento do arquivo de *log* da Tabela 5.2, com o objetivo, dentre outros, de normalizar objetos idênticos mas identificados com nomes diferentes no arquivo de *log*, explicado a seguir:

- O nome de domínio do ICMC passou por modificações no último ano, passando de `icmc.sc.usp.br` para `icmc.usp.br`. Assim, no *log* estavam misturados os dois nomes de domínio. Além disso, algumas entradas continham o endereço IP do servidor *Web*, pois alguns usuários que conhecem esse endereço o digitam diretamente no navegador. Tendo isso em vista, foi necessário unificar esses nomes, sendo que todos eles foram nomeados como sendo `icmc.usp.br`.

- Alguns navegadores substituem alguns caracteres como "`~`" para seu equivalente em código hexa, "`%7E`" no caso do caractere "`~`". Esses códigos são reconhecidos pelos servidores *Web* sem nenhum problema. No entanto, a ferramenta utilizada não processa esses códigos. Dessa forma, uma mesma página como, por exemplo, "`~mdgvnune`" e "`%7emdgvnune`" será representada por dois nós diferentes no grafo gerado pela ferramenta. Assim, todos os códigos "`%xx`" foram substituídos pelos seus equivalentes em caracteres ASCII.

- Quando um usuário digita um endereço do tipo `http://www.site.com.br/um_diretorio/um_arquivo.html`, não há dúvidas de que ele quer acessar o arquivo `um_arquivo.html`. O usuário também pode digitar `http://www.site.com.br/um_diretorio/` e, nesse caso, ele está requisitando o arquivo *default* (normalmente `index.html`) que está dentro do diretório indicado. No entanto, se o usuário digita `http://www.site.com.br/um_diretorio`, sem a barra final, existirá uma ambiguidade pois

não se sabe se o nome `um_diretorio` é realmente um diretório ou se é um arquivo chamado "`um_diretorio`". Quando isso acontece, o servidor *Web* verifica so o nome indicado é um arquivo ou um diretório e, se for um arquivo, ele o envia. Mas, se for um diretório, ele envia uma instrução para o navegador refazer a requisição com uma barra final (isso é feito de forma transparente para o usuário).

Independente disso, a requisição original fica armazenada no *log* e a ferramenta utilizada distingue os dois caso, tratando-os como diferente e, consequentemente, gerando dois nós distintos no grafo. Assim, foi necessário normalizar essa situação e a forma escolhida foi retirar todas as barras finais, já que não se teria como descobrir, no caso de ambiguidades, se o arquivo requisitado é um diretório ou um arquivo.

A Tabela 5.3 mostra o arquivo de *log* após esse pré-processamento, o qual foi utilizado como entrada para a ferramenta Apache2Log.

Mês	Tamanho (*bytes*)	Número de registros
Julho / 2002	73.962.858	412.702
Agosto / 2002	114.880.047	652.367
Setembro / 2002	11.649.196	65.508
Total	200.479.116	1.130.513

Tabela 5.3: Resumo das informações do arquivo de *log* após a normalização de URLs

5.3.1.2 Execução

Durante as primeiras execuções da ferramentas, notou-se que era impraticável a visualiazação do grafo gerado devido ao enorme número de nós e arestas criadas. Mesmo após a limpeza, o grafo gerado tinha muitos nós e ligações, e mal podia-se ter a noção do todo.

Para contornar esse problema, decidiu-se alterar o código da ferramenta para que somente parte do grafo fosse gerado. O critério utilzado para isso foi o de que somente arestas com peso maior que 10% fossem geradas. O peso gerado pela ferramenta é calculado da seguinte forma: para cada par (página $\leftarrow$ `referer`) é gerada uma aresta de peso igual ao número de vezes que esse *link* ocorre no arquivo de *log*. Após todas as arestas serem geradas, normalizam-se os pesos tomando como base a de maior peso *i.e.*, dividem-se todos os pesos pelo maior peso e multiplica-se por 100, conforme pode ser visto na Figura 5.2. O primeiro grafo mostra o grafo gerado pela ferramenta na qual o maior peso é 350 para a aresta (A, B)[13]. Após a normalização, é gerado o grafo à direita no qual pode ser observado que a aresta (D, B) não aparece já que seu peso, *i.e.* $\frac{5}{350} \times 100 = 1.4$ é menor que 10% do maior peso nesse grafo.

[13]Par ordenado.

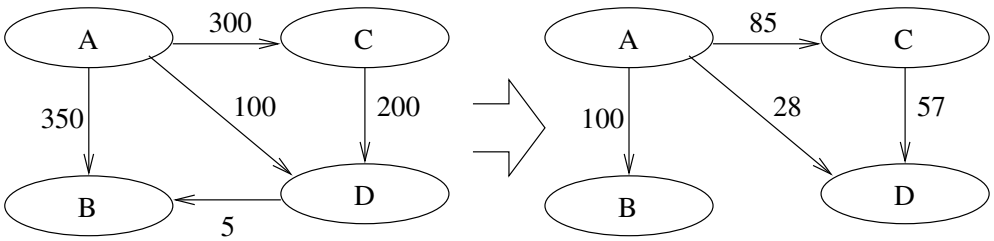


Figura 5.2: Cálculo dos pesos das arestas

Vale ressaltar a "importância" desse tipo de *software* ser *freeware* e *open source*. Se não fosse por isso, não podería se alterar a ferramenta a fim de atender necessidades específicas dos usuários e a sua utilidade seria limitada. Com a alteração feita, pôde-se restringir a visualização aos nós e arestas de maior visitação. O interessante dessa modificação é que ela gerou uma melhor aglutinação do conjunto de páginas visitadas. Após a geração do grafo com arestas de peso maior que 10%, modificou-se novamente a ferramenta, mas para gerar grafos cujas arestas tivesses peso entre 7% e 10% do máximo peso, com o objetivo de visualizar páginas menos visitadas.

Uma outra modificação feita na ferramenta foi a de incluir rótulos nas arestas, de forma que o peso fosse visualizado numericamente. No *software* original, essa indicação do peso é feita apenas pela cor das arestas, com preto representando o maior peso e, passando pela escala de cinza, o cinza mais claro representando o menor peso.

Com isso, três versões da ferramenta foram obtidas:

1. A original
2. A que gera grafos com $arestas >= 10\%$
3. A que gera grafos com $7\% \leq arestas < 10\%$

Na seção seguinte, são mostrados os resultados obtidos utilizando essa ferramenta no *log* do *site* do ICMC bem como a análise de alguns desses resultados.

5.3.1.3 Análise

Após a execução da ferramenta na versão que gera grafos com arestas maiores que 10%, obteve-se um grafo composto de três grandes áreas de acesso e várias outras pequenas partes. Feita uma análise melhor, verificou-se que a maioria das várias pequenas partes poderiam ser "aglutinadas" com outras maiores e as partes que restaram formavam, por si só, uma outra área de acesso distinta das demais. Ao final, pôde-se perceber quatro áreas gerais de acesso ao *site* do ICMC, mostradas nas Figuras 5.3, 5.4, 5.5 e 5.6. Vale observar

que essas figuras são as versões dos grafos com arestas maiores que 10% mesclados com as de arestas entre 7% e 10%, com a finalidade de facilitar a observação dos resultados. Para as primeiras análises, foram utilizadas as versões separadas dos grafos mas, depois, juntou-se os resultados para uma segunda análise, agora com uma visão geral dos resultados.

O primeiro grafo analisado foi o mostrado na Figura 5.3 na página seguinte, correspondente às páginas do Manual de HTML que o ICMC disponibiliza para o público. Foram observados os seguintes pontos:

- Grande parte das pessoas chega até esse manual por meio de *sites* de busca e/ou outras páginas de outros *sites*, uma vez que o nó `/manuals/HTML` está como ponto de partida dos acessos e não se observa uma ligação entre ele e algum nó da Figura 5.4 na página 53, na qual é mostrado o nó correspondente à página principal do ICMC. Essa hipótese é reforçada pelos resultados obtidos com a ferramenta Webalizer, analisada na Seção 5.3.2.

- Uma vez atingida a página principal do manual, a maioria das pessoas vai para a página `/manuals/HTML/zip.html` onde se encontra um arquivo compactado com todo o manual para *download*. Deve ser observado que por meio desta ferramenta não se pode analisar se as pessoas chegam até o manual, "passeiam" por ele e depois fazem o *download* do arquivo ou se fazem o *download* primeiro.

- Apesar de não ser de tanto interesse nesse caso, pode-se criar um *ranking* dos assuntos de interesse dentro do manual de HTML. Em primeiro lugar vem o nó `/manuals/HTML/intro.html` onde é mostrada uma introdução ao HTML; depois vem o nó `/manuals/HTML/tabelas.html` mostrando que as pessoas procuram esse tipo de informação, e assim por diante.

- Nota-se que os agrupamentos desconexos mostrados na Figura 5.3 na página seguinte também representam assuntos de interesse uma vez que, provavelmente, eles foram atingidos a partir de *sites* de busca. É dito provavelmente porque existem *links* que o ligam à página principal do manual e, no entanto, eles são pouco acessados ou não acessados (os *links*). Apesar disso, esses nós foram acessados com certa frequência e uma explicação pode ser os acessos por meio de *sites* de busca. Na Figura 5.7 são mostrados os *links* existentes entre esses pequenos agrupamentos.

Como já mencionado, o grafo da Figura 5.4 na página 53 corresponde à página principal do *site* do ICMC. No entanto, esse grafo mostra mais que isso: mostra, também, várias outras sub-seções que compõem o *site* do ICMC como, por exemplo, as páginas de graduação e pós-graduação, a página das pessoas que trabalham e estudam no ICMC, entre outras. Nesse grafo foram observados os seguintes pontos:

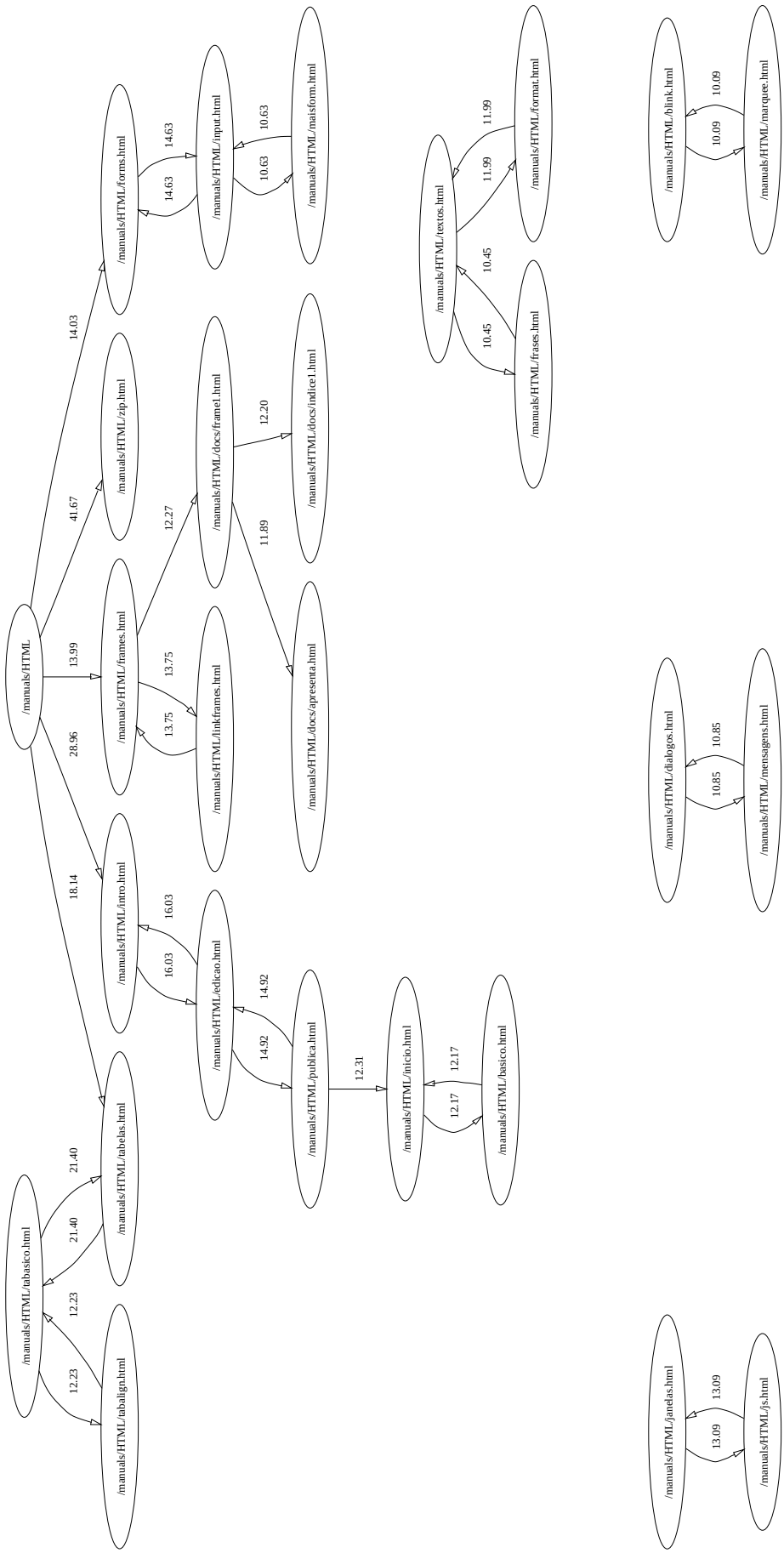


Figura 5.3: Parte do grafo correspondente às páginas do Manual de HTML

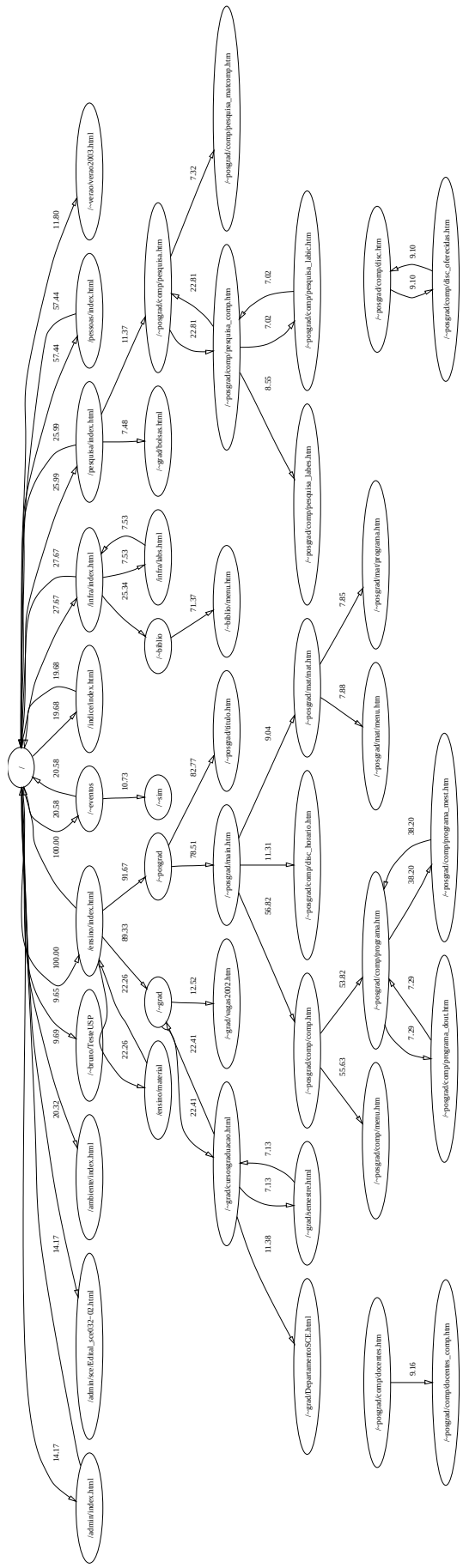


Figura 5.4: Parte do grafo correspondente à página principal do *site* do ICMC

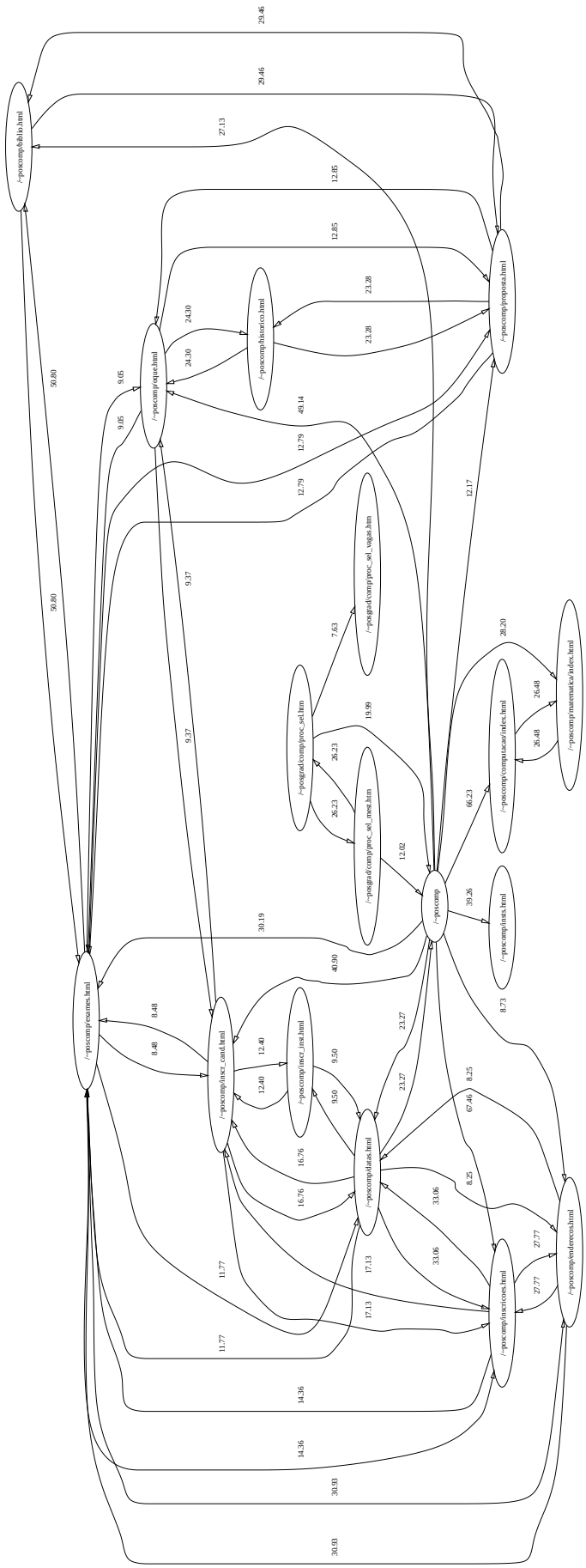


Figura 5.5: Parte do grafo correspondente às páginas do PosComp

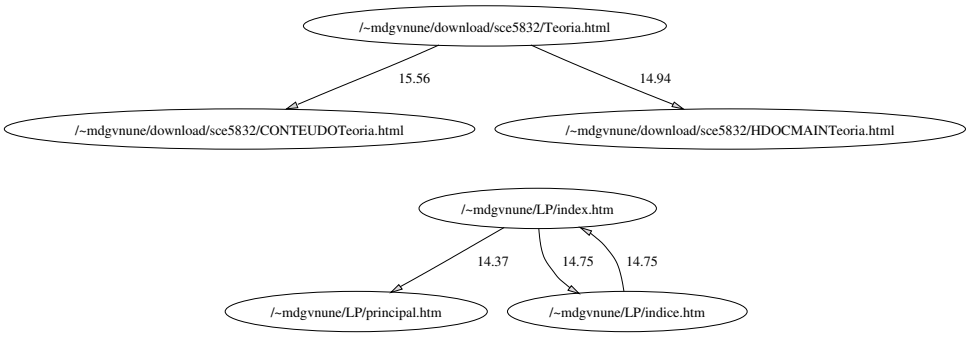


Figura 5.6: Parte do grafo correspondente às páginas de um docente

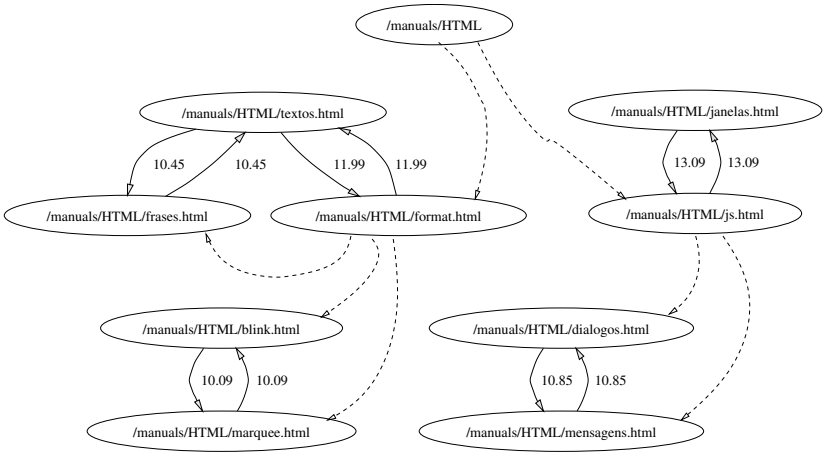


Figura 5.7: *Links* existentes no Manual de HTML mas pouco acessados ou não acessados

- A página principal do ICMC vem servindo bem ao seu objetivo: o de ser uma ponte para as outras seções do *site*. Isso pode ser observado pelo enorme número de *links* que saem do nó `/`.

- Para realizar uma análise mais apurada, pode-se criar um *ranking* das áreas do *site* do ICMC mais acessadas. Observa-se o seguinte *ranking*:

  1. **Nó `/ensino/index.html`:** contém o maior número de acessos dentre as demais áreas. Provavelmente, isso se deve ao fato das pessas entrarem no *site* do ICMC buscando informações sobre o ensino no ICMC. Na página principal do ICMC existe a seguinte descrição para esse *link*:

     Ensino:
     Graduação, Pós-graduação, Grupo PET, Projeto PAE e Material Didático

  2. **Nó `/pessoas/index.html`:** contém uma listagem de professores e alunos do ICMC. É "normal" que também seja uma página bem acessada.

  3. **Nó `/infra/index.html`:** contém informações sobre a infra-estrutura do ICMC como biblioteca e laboratórios.

  4. **Nó `/pesquisa/index.html`:** contém informações sobre as pesquisas realizadas nos laboratórios do ICMC, bem como informações sobre bolsas de Iniciação Científica. Na página principal existe o seguinte texto sobre o *link* de Pesquisas:

     Pesquisas:
     Iniciação Científica, Linhas de Pesquisa e Projetos

  5. **Nó `/~eventos`:** contém informações sobre os eventos sendo realizados no ICMC. Pelo grafo, percebe-se um *link* para a página `/~sing`, que é a página sobre o "7th International Workshop on Real and Complex Singularities". Existem outros eventos listados na página, mas esse último citado parece ter sido o mais procurado dentre os outros, uma vez que os outros *links* são pouco acessados ou não acessados.

  Com essa informação em mãos, um *webmaster* pode identificar falhas do *site* no sentido de saber se ele está ou não alcançando seus objetivos. No caso do *site* do ICMC, nota-se que seu objetivo vem sendo cumprido *i.e.* o de informar as pessoas sobre o ICMC.

  Um nó que chama a atenção é o `/~verao/verao2003.html`, uma vez que ele é pouco acessado em relação aos demais. Isso pode ser devido à baixa procura por cursos de verão no ICMC.

- Observando-se os *links* que saem do nó `/ensino/index.html` nota-se que as seções de graduação e pós-graduação são acessadas de forma praticamente iguais. No entanto, o nó `/ensino/material` é atingido menos vezes. Isso pode ser devido ao fato de que as pessoas que chegam à página sobre ensino estarem mais interessadas em saber sobre os cursos do que sobre o material de ensino disponível.

- Seguindo-se pelo nó `/~posgrad` tem-se *links* para os nós `/~posgrad/main.htm` e `/~posgrad/titulo.htm`. No entanto, essas duas páginas constituem *frames* da página `/~posgrad` *i.e.*, a página `/~posgrad` é formada pelas duas páginas citadas. Concentrou-se, então, a observação na página `/~posgrad/main.htm` que contém as informações de interesse.

  Dessa página pode ser observado que saem três *links*:

  1. um para a página `/~posgrad/comp/comp.htm`, com grande quantidade de acessos. Essa página contém informações sobre a pós-graduação na área de Ciências de Computação.
  2. outro para a página `/~posgrad/comp/disc_horario.htm`, que contém informações sobre os horários das disciplinas de pós-graduação na área de Ciências de Computação.
  3. um para a página `/~posgrad/mat/mat.htm`, com um número menor de acessos. Essa página contém informações sobre a pós-graduação na área de Matemática.

  Com isso, nota-se que as pessoas que acessam o *site* do ICMC têm um maior interesse na pós-graduação em Ciências de Computação do que em Matemática. Na realidade, do total do número de alunos da pós-graduação do ICMC, aproximadamente 80% são da área de Computação.

- Indo-se, agora, para o nó `/~posgrad/comp/comp.htm`, tem-se *links* para os nós `/~posgrad/comp/menu.htm` e `/~posgrad/comp/program.htm`. De novo, essas duas páginas são *frames* que compõem a página `/~posgrad/comp/comp.htm`

  A página `/~posgrad/comp/menu.htm` contém um *menu* desenvolvido em Flash[14] que contém a seguinte estrutura:

  - Principal
  - Programa
  - Linhas de Pesquisa

[14] É uma tecnologia que permite o desenvolvimento de mini-aplicativos e animações para serem colocados dentro de páginas HTML.

  - Corpo Docente
  - Secretaria
  - Bolsas
  - Processo Seletivo
  - Disciplinas
  - Comissão Pós-Graduação
  - F.A.Q.
  - Download

  Nota-se, então, que a página `/~posgrad/comp/program.htm` corresponde ao item "Programa" do *menu*. Nota-se, também, que os nós relativos às outras páginas listadas no *menu* encontram-se desconexos como, por exemplo, os nós `/~posgrad/comp/docentes.htm` e `/~posgrad/comp/disc.htm`, conforme pode ser visto na Figura 5.8 na página oposta.

  O fato desses *links* não aparecerem como sendo utilizados deve-se à utilização de um *menu* desenvolvido em Flash. Uma vez que a responsabilidade de enviar o `referer` de uma página é o próprio navegador, nesse caso ele não tem esse controle. Assim, as requisições chegam até o servidor sem a informação do `referer`.

- Uma outra informação que pode ser obtida do grafo da Figura 5.4 na página 53 é a de que as informações sobre as pesquisas realizadas no laboratório LABES, de Engenharia de Software, e LABIC, de Inteligência Computacional, são as mais acessadas dentre todos os outros laboratórios. Isso pode ser verificado pelos *links* que saem do nó `/~posgrad/comp/pesquisa_comp.htm`.

  Entretando, na página `/~posgrad/comp/pesquisa_comp.htm` a seguinte lista de laboratórios está disponível:

  - Engenharia de Software e Sistemas de Informação
  - Inteligência Computacional
  - Banco de Dados
  - Sistemas Distribuídos e Programação Concorrente
  - Computação Gráfica e Processamento de Imagens
  - Hipermídia

  Assim, uma pergunta que surge é se as páginas do LABES e do LABIC são as mais visitadas por serem os dois primeiros dessa lista ou por um real interesse dos usuários nesses laboratórios.

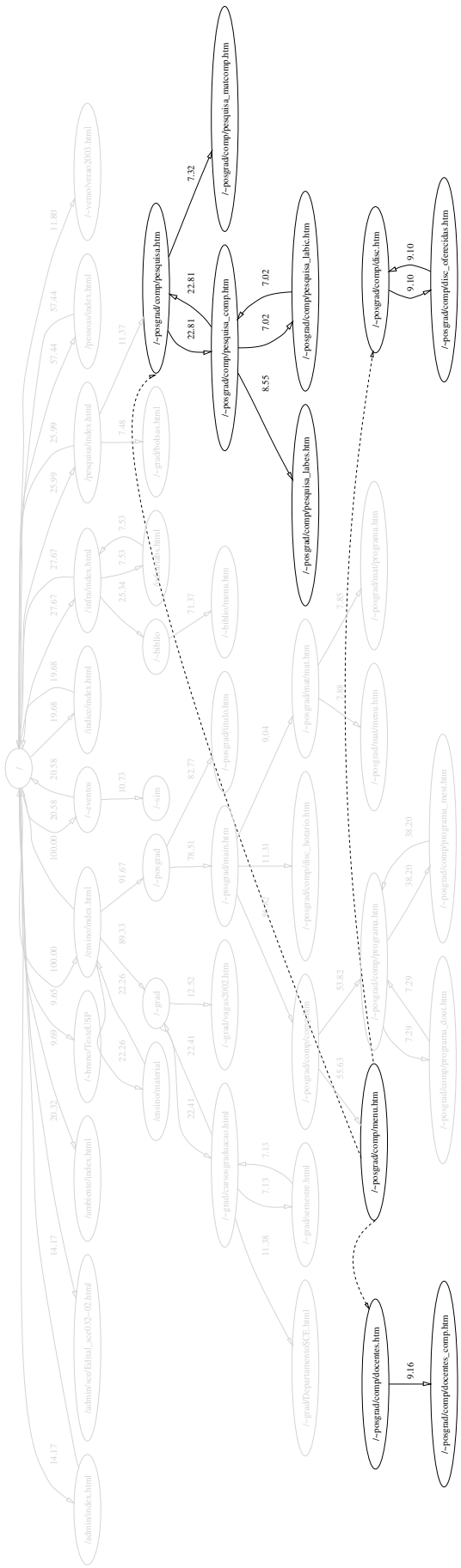


Figura 5.8: *Links* provenientes do nó `/~posgrad/comp/menu.htm`

- Como última análise do grafo da Figura 5.4, tem-se a observação de que o nó `/~posgrad/comp/disc.htm` tem um *link* apenas para o nó `/~posgrad/comp/disc_oferecidas.htm` e, no entanto, essa página possui outros *links*, tais como:

  - Disciplinas Oferecidas
  - Calendário-1º/2003
  - Horário 1º Semestre/2003
  - Orientações para Matrícula de Alunos Especiais

  Assim, apenas o primeiro *link* (Disciplinas Oferecidas) está sendo bem utilizado, mas não os outros. Provavelmente, o *link* sobre o Calendário não está num local adequado, já que a página trata sobre as disciplinas; o *link* sobre os horários, como foi visto, está sendo acessado a partir de outra página e o *link* sobre Matrícula de Alunos Especiais deve ter uma baixa procura mesmo, uma vez que a quantidade desses alunos no ICMC, perto dos regulares, é pequena.

O grafo da Figura 5.5 na página 54, correspondente às páginas com informações sobre o PosComp, foi outro grafo analisado. Conforme citado anteriormente, foi a partir dessa experiência que se "descobriu" sobre o evento PosComp. Nota-se uma alta quantidade de acessos por várias das páginas que compõem a seção do PosComp dentro do *site* do ICMC. Nesse grafo também foram observados os seguintes aspectos:

- O primeiro ponto observado é que o nó `/~posgrad/comp/proc_sel.htm` é proveniente da página `/~posgrad/comp/menu.htm` descrita na análise do grafo da Figura 5.4 na página 53 *i.e.*, do *menu* desenvolvido em Flash. O importante aqui é notar a frequência com que a página principal do PosComp é acessada, direta ou indiretamente, por meio da página `/~posgrad/comp/proc_sel.htm`. Uma experiência interessante é comparar com o acesso realizado por meio de outras páginas, principalmente por meio de *sites* de busca. Isso será visto na análise da ferramenta Webalizer — Seção 5.3.2 — onde é mostrado que alguns acessos às páginas do PosComp foram feitos a partir de *sites* de outras instituições de ensino.

- É interessante notar aqui, também, o *ranking* proveniente da página principal, Figura 5.5 na página 54:

  1. **Nó `/~poscomp/inscricoes.html`:** essa página contém informações sobre as inscrições no PosComp. O fato dessa página estar em primeiro lugar deve-se ao fato desse ser o maior interesse de quem quer fazer o Poscomp: verificar as condições necessárias para realizar a inscrição e inscrever-se.

2. **Nó `/~poscomp/computacao/index.html`:** essa página contém um modelo da prova de Computação. Ela parece ser muito visitada porque a maioria das pessoas estão interessadas na prova de Computação. Conforme foi visto em análise anterior, há uma busca muito grande por informações sobre esse curso.

3. **Nó `/~poscomp/oque.html`:** descreve o que é o PosComp. É interessante notar com que frequência há interesse em saber o que é o PosComp. Inclusive, a partir desse nó, há uma certa frequência na visita à página sobre a proposta (`/~poscomp/proposta.html`) diretamente, ou passando pela página sobre o histórico do PosComp (`/~poscomp/historico.html`).

   É interessante notar, também, que o acesso à página da proposta se dá mais pela página relacionada ao que é o PosComp do que diretamente pela página principal. Isso se deve, provavelmente, ao fato de que quem quer saber sobre o PosComp procura esse tipo de informação por "todas" as páginas que falam algo sobre o PosComp.

4. **Nó `/~poscomp/inscr_cand.html`:** descreve os procedimentos de inscrições para canditatos. Essa página contém algumas informações duplicadas em relação à página sobre inscrições que está em primeiro lugar no *ranking*. Nota-se, também, uma certa divisão de acessos entre essas duas páginas. Assim, essas duas páginas poderiam ser transformadas em apenas uma de forma a se ter uma organização mais simples e funcional.

5. **Nó `/~poscomp/insts.html`:** contém uma lista de instituições que utilizaram o PosComp.

6. **Nó `/~poscomp/exames.html`:** contém informações sobre os exames realizados para o processo seletivo.

7. **Nó `/~poscomp/matematica/index.html`:** contém um modelo da prova de Matemática.

8. **Nó `/~poscomp/biblio.html`:** contém a bibliografia indicada para os exames. É interessante o fato da página sobre a bibliografia não estar entre as mais acessadas. No entanto, percebe-se que a maioria das pessoas que visitou a página sobre os exames, foram até a página sobre bibliografia. Ou seja, ela foi uma página bastante acessada, só não o foi pela página principal.

9. **Nó `/~poscomp/datas.html`:** essa página tem uma lista das datas de realização das provas. É interessante observar que essa página é atingida diretamente pela página principal e por meio das duas páginas de inscrições, sendo esse último método o "preferido".

10. **Nó `/~poscomp/proposta.html`:** contém um texto que conta a proposta do PosComp. É diferente da página `/~poscomp/oque.html` pois esta descreve o que

é o PosComp e a outra o porquê dele existir.

11. **Nó `/~poscomp/enderecos.html`:** uma lista de endereços dos locais onde se realizaram os exames do PosComp. A maioria dos acessos a essa página chegam através das duas páginas de inscrições e dos exames. Também é interessante observar que o *link* da página `/~poscomp/inscr_cand.html` é pouco ou não utilizado, ou seja, provavelmente, após um certo tempo, as pessoas começaram a utilizar a página `/~poscomp/inscricoes.html` como ponte para as outras informações.

Na Figura 5.9 na próxima página são destacados alguns *links* comentados nessa análise.

O último grafo analisado foi o da Figura 5.6 na página 55. Ele contém páginas de dois cursos *on-line*: "Linguagens de Programação" e "Tópicos de Teoria da Computação", ambos mantidos por um docente da área de Computação.

O único tópico observado nesse grafo foi o de que esses cursos não estão sendo acessados pelos *links* existentes na página sobre material didático citada na análise do grafo da Figura 5.4 na página 53. Isso leva a crer que os cursos estão sendo acessados através de *links* em outros *site* e/ou *sites* de busca.

5.3.1.4 Conclusões

A ferramenta Apache2Dot auxiliou neste trabalho no sentido de entender melhor o que deveria ser pré-processado no arquivo de *log*. Entretanto, a ferramenta apresenta alguns problemas. Um ponto negativo da ferramenta é que ela não trata casos comuns como, por exemplo, a substituição de códigos hexas. Assim, essa substituição teve que ser implementada neste trabalho.

Um outro problema encontrado é o tamanho dos grafos gerados. Como são muitas as informações geradas, a visualização dos grafos fica comprometida. Apesar da ferramenta GraphViz permitir um certo controle sobre alguns aspectos do grafo como, por exemplo, tamanho de arestas, essa ajuda é insuficiente. No entanto, esse problema não é da ferramenta em si, mas sim uma dificuldade inerente do trabalho pois a quantidade de dados encontrada nos arquivos de *log* é muito grande.

Contudo, a ferramenta possibilita ao *webmaster* observar como os usuários do *site* estão "caminhando" por ele, permitindo que sejam tomadas ações como a mudança de lugar de *links* que não estão surtindo efeito, para páginas onde seriam melhor acessados.

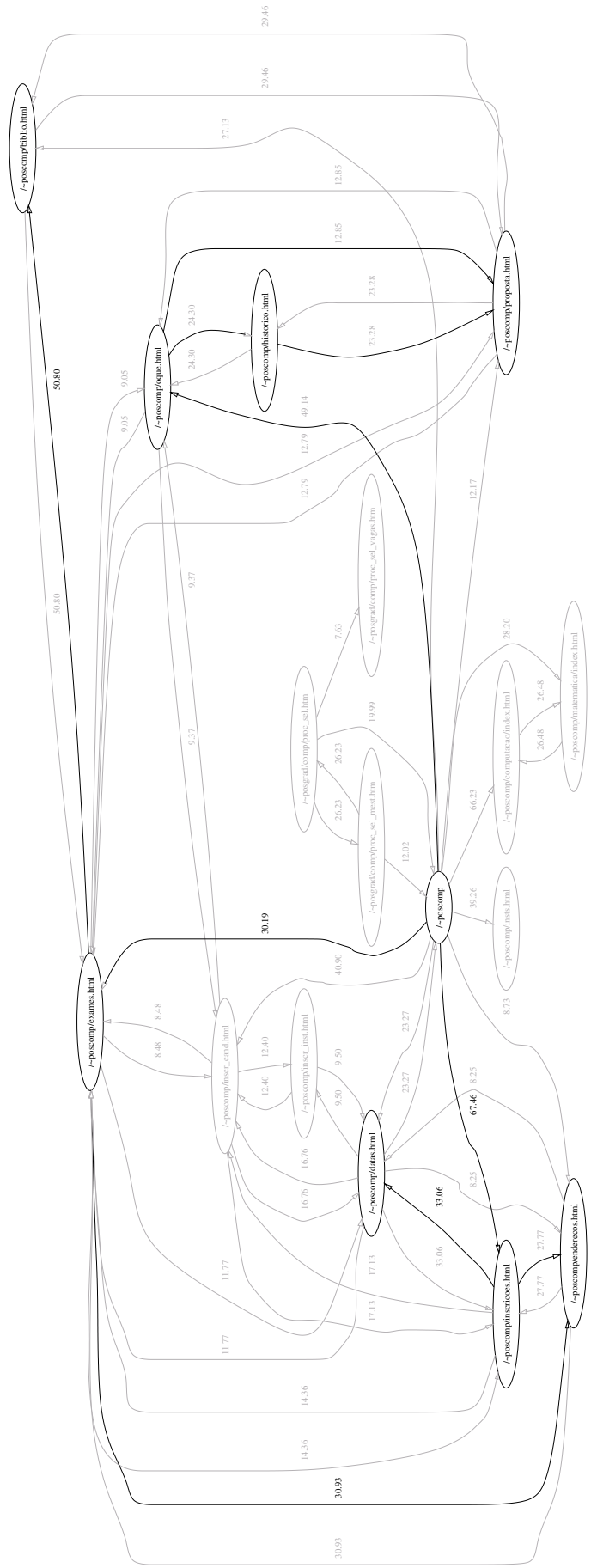


Figura 5.9: Alguns *links* considerados importantes, após a análise

5.3.2 Webalizer

O Webalizer é uma ferramenta para análise de arquivos de *log* de servidores *Web* que produz estatísticas de uso em formato HTML para visualização em navegadores. Os resultados são apresentados tanto numericamente, por meio de tabelas, como graficamente, o que facilita a interpretação dos dados.

Estatísticas de uso anual, mensal, diário e por hora são apresentados, assim como a possibilidade de mostrar o uso categorizado por *site*[15], URL, `referer`, tipo de navegador, palavras pesquisadas, página de entrada e de saída, usuário e país. Algumas dessas informações estão disponíveis somente se forem suportadas e estiverem presentes no arquivo *log* sendo processado.

O Webalizer suporta os formatos CLF e ECLF para arquivo de *log* de servidores *Web*, bem como outros formatos e outros tipos de *logs* como, por exemplo, de servidores *proxy*.

5.3.2.1 Limpeza

Analogamente ao trabalho realizado com a ferramenta Apache2Dot, o Webalizer foi executado algumas vezes, até que se obtivesse uma idéia do pré-processamento que deveria ser feito no arquivo de *log* para atingir os objetivos do trabalho. O Webalizer é uma ferramenta mais complexa que o Apache2Dot, de forma que alguns tipos de pré-processamento ele realiza automaticamente como, por exemplo, a substituição de códigos hexas, descritos na análise da ferramenta Apache2Dot — Seção 5.3.1 na página 45. A filtragem de arquivos que não são interessantes para a análise como, por exemplo, figuras, também é realizada pelo Webalizer, por meio de um arquivo de configuração.

No entanto, o pré-processamento dos nomes de domínios, mudando-os de `icmsc.sc.usp.br` para `icmc.usp.br` conforme descrito na Seção 5.3.1 na página 45, teve que ser feito pois, nesse caso, é uma situação específica do arquivo de *log* sendo analisado, não podendo ser uma situação prevista pelos desenvolvedores da ferramenta.

Um outro caso que houve necessidade de pré-processamento foi na normalização do campo de `request`. Notou-se que, em alguns resultados gerados pelo Webalizer, apareciam requisições à páginas chamadas `http://www.icmc.usp.br/~poscomp` e à páginas `/~poscomp` e outros casos semelhantes. Contudo, essas páginas são as mesmas. A explicação para isso é que alguns navegadores, ao invés de enviar na requisição apenas o nome da página desejada pelo usuários, envia o endereço completo e esse fato fica registrado no arquivo de *log*. Apesar do servidor *Web* lidar corretamente com esses casos, a ferramenta estudada

[15] O Webalizer utiliza o termo *site* para desginar o `host`, descrito na Seção 4.3 na página 32

não o faz.

5.3.2.2 Execução

Para a execução do Webalizer, foi necessário editar um arquivo de configuração para que se obtivessem os resultados desejados. Pode-se citar, como relevantes, as seguintes configurações realizadas:

`PageType` essa configuração instrui o Webalizer a consider como página somente alguns tipos de arquivos. Os tipos de arquivos escolhidos para essa configuração foram os arquivos com extensão `htm*`, `cgi` e `php*`. O símbolo '`*`' é um meta-caractere que indica para o Webalizer que ele pode ser substituido por qualquer combinação de letras. Assim, tanto as extensões `htm` como `html` são consideradas páginas.

`HideReferrer` esse comando permite que algumas páginas que aparecem no campo `referer` não sejam visualizadas, apesar de participarem das estatísticas. Seguindo o conselho do manual do Webalizer, foi colocado para esconder as páginas do próprio *site* já que elas tendem a ser a maioria dos `referers`.

`HideURL` esse comando, de forma similar ao `HideReferrer`, permite que os arquivos que aparecem no campo `URL` da requisição, *i.e.* os arquivos requisitados pelos usuários, não sejam visualizados. Aqui, escolheu-se esconder os arquivos de imagens e alguns outros arquivos auxiliares de páginas HTML tais como: *styles sheets* (css) e *scripts* (js, swf). Diferentemente do realizado com a ferramenta Apache2Dot, escolheu-se não esconder arquivos comprimidos, de documentos, entre outros, pois nesse caso quer-se ver como estes arquivos estão sendo acessados.

`GroupReferrer` essa configuração permite que o Webalizer agrupe tipos de `referers` semelhantes. Assim, agrupou-se endereços de *sites* de busca como o Yahoo, Altavista e Google. Isso se faz necessário pois nem sempre os endereços desses *sites* aparecem de forma igual no arquivo de *log*. O caso mais comum é quando existem dois *sites* em línguas diferentes para o mesmo serviço de busca. Assim, o *site* do Yahoo pode aparecer com os endereços `http://www.yahoo.com` e `www.yahoo.com.br`.

`GroupAgent` de forma similar ao `GroupReferrer`, esse comando permite que se agrupem os tipos de navegadores que são utilizados pelos usuários. Alguns navegadores já estavam configurados como *default* mas alguns outros, observados nas execuções preliminares do Webalizer, foram acrescentados.

`GroupDomains` esse comando permite que se agrupe os `hosts` dentro de seus domínios. Vale lembrar que os `hosts` consistem dos endereços das máquinas de onde os usuários

utilizam os seus navegadores. Esse comando permite que se agrupem, por exemplo, as máquinas `cust1.tnt.mia.uu.net` e `cust2.tnt.mia.uu.net` dentro do domínio `mia.uu.net`, ou seja, no resultado produzido pelo Webalizer será visualizado o domínio ao invés dos `hosts` separadamente. A configuração *default* do Webalizer é mostrar os `hosts` individualmente.

Um outro ponto importante a ser comentado é que utilizou-se uma versão modificada do Webalizer. Por ser um *software freeware* e *open source*, o Webalizer permite modificações no seu código de forma a atender à necessidades específicas. Diferentemente da ferrementa Apache2Dot na qual as modificações realizadas são de nossa autoria, essa versão foi modificada por Stanislaw Yurievich Pusep que modificou o Webalizer de forma que ele gerasse as estatísticas geográficas de forma mais rápida e confiável. Isso é feito utilizando-se uma biblioteca chamada GeoIP. Essa versão modificada do Webalizer pode ser encontrada em `http://sysdlabs.hypermart.net/proj/log.html`.

5.3.2.3 Análise

Conforme dito anteriormente, o Webalizer gera estatísticas de acesso ao *site* sendo analisado. A seguir serão mostrados alguns dos resultados gerados pela ferramenta e uma análise sobre algumas informações úteis que se podem tirar desses resultados.

A primeira estatística gerada pela ferramenta é a mostrada na Figura 5.10 na página 68. Ela oferece uma visão geral da quantidade de acessos ao *site* mês a mês. Os termos utilizados pelo Webalizer são explicados a seguir:

Hits representa o número total de requisições feitas ao servidor durando o perído de tempo considerado. Em última instância, é o número de registros observados no *log*. Observando-se o total de *hits* percebe-se que ele é igual ao número de registros mostrados na Tabela 5.1 na página 44.

Files representa o número total de *hits* que resultaram em algo sendo realmente enviado de volta ao usuário. Por exemplo, os *hits* que contêm requisições à arquivos que não existem ou que já estejam no *cache* do navegador, não são considerados *files*. Observando a diferença entre o número de *hits* e o número de *files* pode-se ter um indicativo da quantidade de usuários repetidos, uma vez que quanto maior a diferença maior é o número de pessoas requisitando páginas que já estão no *cache* do navegador, *i.e.* já tinham visitado a página anteriormente.

Pages são as requisições à arquivos que são considerados as reais páginas sendo vistas pelos usuários, e não os itens auxiliares tais como figuras. Algumas pessoas chamam

essa métrica de *page views*. Para saber quais requisições são consideradas *pages* o Webalizer se baseia no item de configuração chamado `PageType`, descrito anteriormente.

Visits ocorrem quando algum `host` requisita uma página pela primeira vez. Enquanto o mesmo `host` permanecer requisitando páginas dentro de um período de tempo pré-estabelecido, essas requisições serão consideradas parte da mesma visita. Se um `host` faz uma requisição e o período de tempo desde sua última requisição for maior que o período de tempo especificado na configuração (o *default* é de 30 minutos), uma nova visita será considerada.

Sites é o número total de endereços únicos de `hosts` que fizeram requisições. Essa métrica pode ser utilizado apenas como um indicativo do número de visitas ao *site*, pois usuários diferentes podem aparecer como vindo de um único *site*, conforme descrito na Seção 4.3 na página 32.

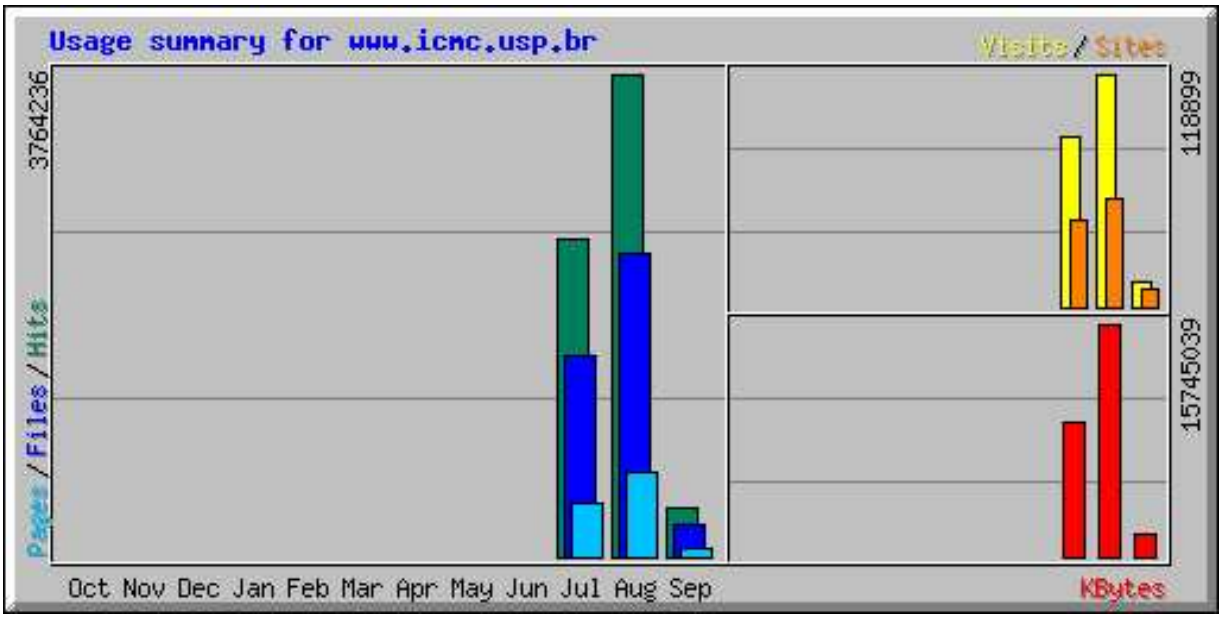
KBytes indica a quantidade de dados tranferida entre o servidor *Web* e a máquina do usuário.

Pode-se perceber, pela Figura 5.10 na página seguinte, que o mês de setembro contém poucos acessos devido ao perído de recolhimento de informações escolhido, *i.e.* de 02 de julho de 2002 a 04 de setembro de 2002. Devido a isso, o mês de setembro não foi considerado nas análises mais específicas. Outro ponto a ser observado nessa figura é que os acessos do mês de julho são menores que o do mês de setembro. Isso pode ser explicado pelo fato do mês de julho ser férias o que acarreta em um menor número de acessos ao *site* do ICMC.

As outras estatísticas geradas pelo Webalizer são as específicas para cada mês. Observando-se a Figura 5.10, percebe-se que os nomes dos meses são *links* para essas estatísticas específicas. São mostrados, a seguir, os resultados para o mês de julho e, quando for conveniente, os resultados de agosto são colocados juntos, de forma a poder comparar os resultados dos dois meses.

Ao clicar-se no *link* para os resultados do mês de julho, obtém-se uma página com vários resultados. O primeiro deles é o mostrado na Figura 5.11 na página 69. Aqui cabem mais algumas definições:

URL como mencionado na Seção 4.3 na página 32, a URL é o endereço do arquivo sendo requisitado. Na Figura 5.11, é dado o número total de URLs únicas, ou seja, quantos arquivos diferentes foram requisitados. O número alto de 30335 é justificado pelo grande número de imagens e outros arquivos auxiliares que existem nas páginas.



Summary by Month										
Month	Daily Avg				Monthly Totals					
	Hits	Files	Pages	Visits	Sites	KBytes	Visits	Pages	Files	Hits
Sep 2002	95607	62568	16363	3272	8626	1463787	13089	65455	250272	382428
Aug 2002	121426	76264	21040	3835	55187	15745039	118899	652248	2364212	3764236
Jul 2002	82505	52045	13687	2910	44504	9108101	87314	410626	1561372	2475166
Totals						**26316927**	**219302**	**1128329**	**4175856**	**6621830**

Figura 5.10: Resumo de uso para o *site* `www.icmc.usp.br`

Response Codes são as códigos de *status* descritos, também, na Seção 4.3.

Monthly Statistics for July 2002		
Total Hits		2475166
Total Files		1561372
Total Pages		410626
Total Visits		87314
Total KBytes		9108101
Total Unique Sites		44504
Total Unique URLs		30335
Total Unique Referrers		7362
Total Unique User Agents		2365
	Avg	Max
Hits per Hour	3437	13048
Hits per Day	82505	131433
Files per Day	52045	83047
Pages per Day	13687	20521
Visits per Day	2910	3932
KBytes per Day	303603	533916
Hits by Response Code		
Undefined response code		13
Code 200 - OK		1561372
Code 206 - Partial Content		17357
Code 301 - Moved Permanently		19896
Code 302 - Found		183
Code 304 - Not Modified		766374
Code 400 - Bad Request		94
Code 403 - Forbidden		1260
Code 404 - Not Found		108080
Code 405 - Method Not Allowed		285
Code 408 - Request Timeout		247
Code 416 - Requested Range Not Satisfiable		5

Figura 5.11: Resumo das estatísticas para o mês de julho de 2002

Apesar de ter-se várias métricas indicadas na Figura 5.11, foram observados os seguintes pontos considerados relevantes:

- A métrica *KBytes per Day* é uma métrica bastante útil para verificar se a largura de banda que tem-se na infra-estrutura do *site* suporta a demanda. Nesse caso, vê-se uma média de 303603 *KBytes* por dia. Transformando isso para *KBytes* por segundo, tem-se que o *site* do ICMC demanda uma largura de banda de 3.513 *KBytes* por segundo. Pode-se fazer o mesmo utilizando o máximo que foi registrado, *i.e.* 533916 *KBytes* por dia ou 6.179 *KBytes* por segundo.

- É interessante observar a quantidade de *Response Codes* 304. Ele indica a quantidade de requisições a arquivos que estavam nos *caches* dos navegadores. Nota-se que,

do total de *hits* que chegaram ao servidor, aproximadamente 30% são verificações requisitadas pelos navegadores para saber se podem usar os arquivos que estam em seus *caches*.

- Outro *Response Code* a ser observado é o 404. Ele indica o número de requisições à arquivos não existentes. Nota-se que aproxidamente 4% do *hits* são de arquivos não encontrados. Apesar de ser uma porcentagem pequena, o ideal é que esse número seja próximo de zero. Exigir que essa porcentagem seja zero não é realista, uma vez que um usuário pode requisitar qualquer arquivo diretamente, existindo ele ou não. É importante observar que o número de *hits* que retornam um código 404 não coincide com o número de páginas distintas que não são encontradas. Pode acontecer, por exemplo, de um *link* ser interessante e não levar a página alguma. Nesse caso, pelo *link* ser interessante, muitos usuários irão clicar nele e ficarão decepcionados, ao mesmo tempo que o número de *Response Codes* 404 irá aumentar devido a um único *link*.

  Seria interessante se a ferramenta fornecesse uma estatística de *Response Codes* 404 por URL requisitada. Assim, podería-se saber se existe algum *link* que está gerando a maioria desses erros.

- Como curiosidade, nota-se um alto número de navegadores diferentes, indicado pela métrica *Total Unique User Agents*.

Após esse primeiro resultado, tem-se um gráfico com o número de acessos feitos ao *site* em cada dia do mês. Esse gráfico é mostrado na Figura 5.12 na próxima página. Nessa mesma figura estão sendo mostrados os gráfico para o mês de julho e agosto, para que seja feita uma comparação do número de acessos. Podem-se observar os seguintes pontos:

- O primeiro dia de julho não tem acessos e o segundo dia tem poucos acessos. Isso se deve ao período de coleta dos dados. Portanto, deve-se desconsiderá-los das análises.

- Os dias correspondente aos finais de semana possuem menos acessos em relação aos outros dias da semana. Isso pode ser explicado pelo fato de ser um *site* de uma instituição de ensino. Se a análise tivesse sido feita em um *site* de jogos, por exemplo, pode ser que o inverso ocorresse.

- Esboçando-se uma curva geral para os acessos, obtem-se a Figura 5.13. Nela, podem ser observadas quatro áreas distintas:

  1. No início do mês de julho, tem-se um alto índice de acesso o qual cai rapidamente para um patamar mais baixo.

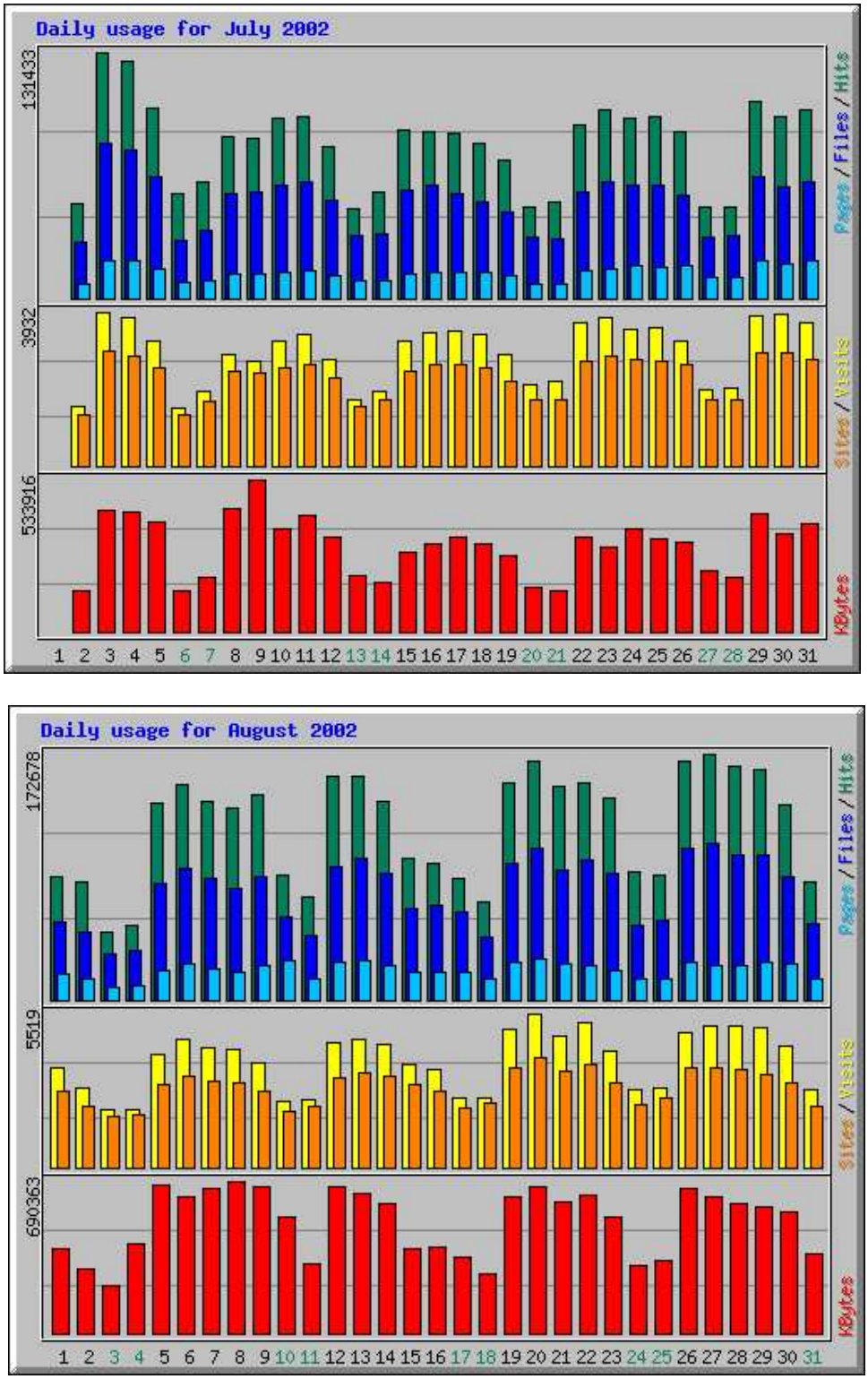


Figura 5.12: Acessos diários durante os mêses de julho e agosto de 2002

2. Esse patamar refere-se ao mês de férias, onde tem-se um número menor de acessos às páginas do ICMC.
3. No começo do mês de agosto, volta às aulas, tem-se uma queda no número de acessos, inclusive com índices menores que o do mês de julho. Não foi encontrada uma explicação para esse fato.
4. No decorrer do mês de agosto, os níveis do número de acessos vão aumentando gradativamente até o número médio de acessos normais.

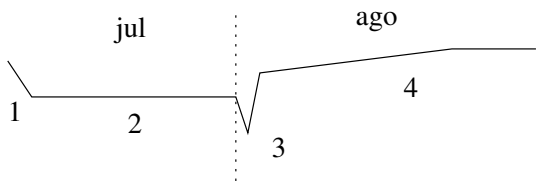


Figura 5.13: Curva geral dos acessos diários feitos ao *site* do ICMC

Outro gráfico de acesso gerado pelo Webalizer é o mostrado na Figura 5.14 na próxima página. Nessa figura podem ser observados os acessos por hora durante os meses de julho e agosto. O único aspecto a ser observado nessa figura é que ocorrem picos de acessos às 11 horas na parte da manhã e às 15 e 16 horas na parte da tarde. Para o *site* sendo analisado isso não tem muita importância mas, para um *site* de comércio, essa informação pode ser bastante útil.

Os próximos resultados gerados pela ferramenta estão mostrados nas Figuras 5.15 na página 74 e 5.16 na página 75, para os meses de julho e agosto, respectivamente. Os seguinte pontos podem ser observados nessas figuras:

- Pode-se notar que os endereços mais acessados correspondem àqueles citados na análise da ferramenta Apache2Dot. A diferença é que lá tem-se o número de acessos separado por *links* representados pelas arestas, enquanto que aqui tem-se uma visão global do número de acessos.
- Nota-se que, apesar da colocação (*ranking*) da página `/~poscomp/inscricoes.html` ter caído do mês de julho para o mês de agosto, é necessário atentar para o número de acessos que aumentou, indo de 3265 acessos em julho para 5247 acessos em agosto. Isso se deve à proximidade do realização do PosComp.
- Observa-se um interesse muito grande no arquivo `/~sma/sma302/sma302.pdf` no mês de agosto. Esse arquivo é uma apostila de Cálculo. Uma possível explicação é que, por ser o início das aulas, os alunos estão obtendo o material necessário para os seus estudos.

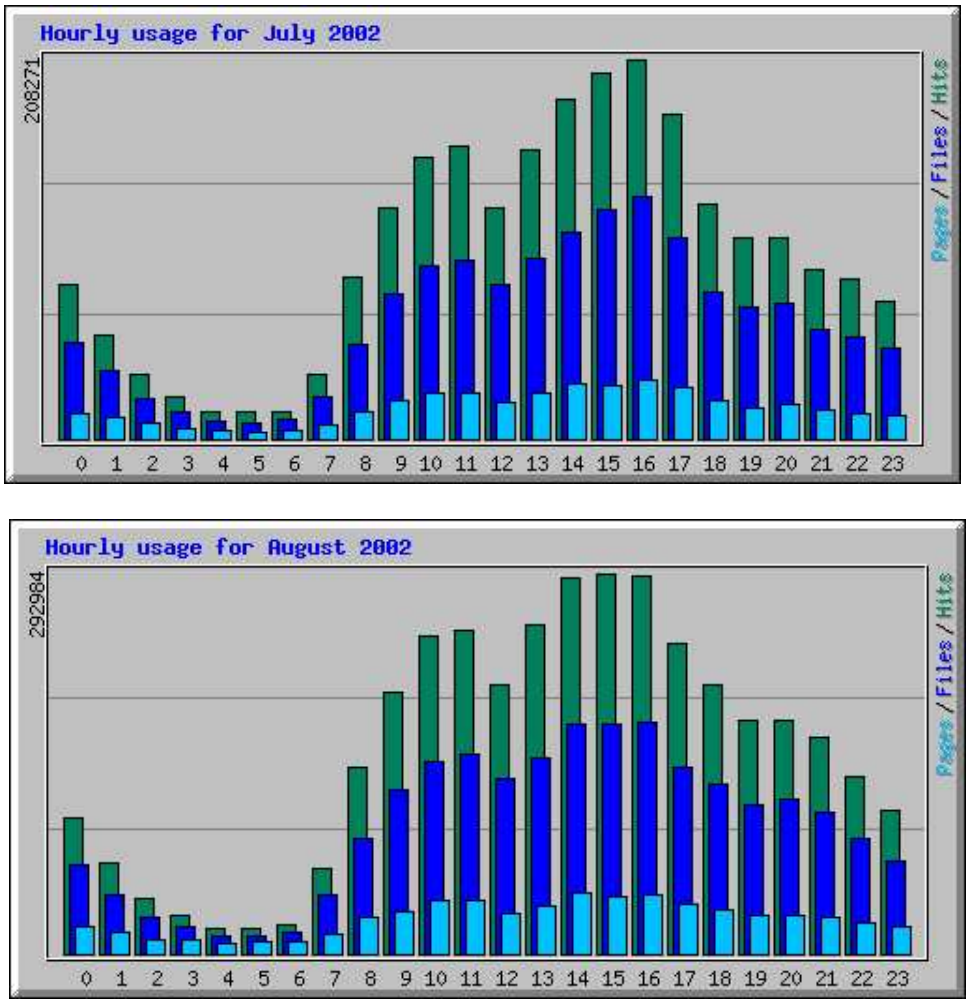


Figura 5.14: Acessos por hora durante os mêses de julho e agosto de 2002

Top 30 of 30335 Total URLs					
#	Hits		KBytes		URL
1	57347	2.32%	516297	5.67%	/
2	8653	0.35%	119737	1.31%	/manuals/HTML/
3	7399	0.30%	23298	0.26%	/~poscomp/
4	3265	0.13%	12183	0.13%	/~poscomp/inscricoes.html
5	3222	0.13%	6905	0.08%	/~biblio/
6	3061	0.12%	19914	0.22%	/ensino/
7	2869	0.12%	759	0.01%	/~posgrad/
8	2773	0.11%	21419	0.24%	/~poscomp/exames.html
9	2670	0.11%	45067	0.49%	/~poscomp/biblio.html
10	2560	0.10%	617	0.01%	/~posgrad/titulo.htm
11	2429	0.10%	5979	0.07%	/~posgrad/main.htm
12	2349	0.09%	6981	0.08%	/~poscomp/inscr_cand.html
13	2325	0.09%	65740	0.72%	/pessoas/
14	2179	0.09%	11621	0.13%	/~poscomp/oque.html
15	2099	0.08%	6694	0.07%	/~poscomp/datas.html
16	2094	0.08%	43989	0.48%	/~poscomp/computacao/
17	1915	0.08%	6879	0.08%	/~grad/
18	1893	0.08%	4072	0.04%	/~biblio/menu.htm
19	1772	0.07%	8309	0.09%	/~poscomp/proposta.html
20	1755	0.07%	7806	0.09%	/manuals/HTML/zip.html
21	1717	0.07%	26729	0.29%	/~poscomp/enderecos.html
22	1632	0.07%	507	0.01%	/~posgrad/comp/comp.htm
23	1627	0.07%	746	0.01%	/~posgrad/comp/menu.htm
24	1614	0.07%	20759	0.23%	/pessoas/profsMat.html
25	1590	0.06%	53661	0.59%	/~gbatista/magica/
26	1572	0.06%	125953	1.38%	/manuals/HTML/fontes/basicow95.zip
27	1550	0.06%	1716	0.02%	/~posgrad/comp/programa.htm
28	1459	0.06%	15977	0.18%	/~poscomp/insts.html
29	1389	0.06%	214809	2.36%	/~inbio/material/Apostila_Inbio_Biomol.pdf
30	1339	0.05%	23860	0.26%	/~poscomp/matematica/
View All URLs					

Figura 5.15: Endereços mais acessados durante o mês de julho de 2002

Top 30 of 30170 Total URLs					
#	Hits		KBytes		URL
1	127845	3.40%	1801130	11.44%	/
2	13614	0.36%	2069685	13.14%	/~sma/sma302/sma302.pdf
3	12418	0.33%	34595	0.22%	/~poscomp/
4	10339	0.27%	149066	0.95%	/manuals/HTML/
5	5836	0.16%	12182	0.08%	/~biblio/
6	5247	0.14%	20122	0.13%	/~poscomp/inscricoes.html
7	4452	0.12%	29820	0.19%	/ensino/
8	4193	0.11%	31424	0.20%	/~poscomp/exames.html
9	3989	0.11%	1021	0.01%	/~posgrad/
10	3881	0.10%	11281	0.07%	/~poscomp/inscr_cand.html
11	3880	0.10%	64488	0.41%	/~poscomp/biblio.html
12	3596	0.10%	832	0.01%	/~posgrad/titulo.htm
13	3385	0.09%	67408	0.43%	/~poscomp/computacao/
14	3377	0.09%	7758	0.05%	/~posgrad/main.htm
15	3345	0.09%	10365	0.07%	/~poscomp/datas.html
16	3343	0.09%	6808	0.04%	/~biblio/menu.htm
17	3111	0.08%	16234	0.10%	/~poscomp/oque.html
18	3053	0.08%	14671	0.09%	/~grad/
19	2911	0.08%	76948	0.49%	/pessoas/
20	2541	0.07%	11683	0.07%	/~poscomp/proposta.html
21	2412	0.06%	42819	0.27%	/~poscomp/enderecos.html
22	2375	0.06%	1075	0.01%	/~posgrad/comp/menu.htm
23	2372	0.06%	729	0.00%	/~posgrad/comp/comp.htm
24	2287	0.06%	39276	0.25%	/~poscomp/matematica/
25	2246	0.06%	2419	0.02%	/~posgrad/comp/programa.htm
26	2023	0.05%	8875	0.06%	/manuals/HTML/zip.html
27	1973	0.05%	64941	0.41%	/~gbatista/magica/
28	1886	0.05%	145045	0.92%	/manuals/HTML/fontes/basicow95.zip
29	1823	0.05%	12446	0.08%	/~poscomp/historico.html
30	1746	0.05%	227995	1.45%	/~inbio/material/Apostila_Inbio_Biomol.pdf
View All URLs					

Figura 5.16: Endereços mais acessados durante o mês de agosto de 2002

Ainda sobre as URLs acessadas, o Webalizer gera a tabela mostrada na Figura 5.17. Não é mostrada a tabela referente ao mês de agosto pelo fato dos tópicos relevantes serem os mesmo. Os tópicos que chamam a atenção nessa figura são os seguintes:

Top 10 of 30335 Total URLs By KBytes					
#	Hits		KBytes		URL
1	57347	2.32%	516297	5.67%	/
2	1389	0.06%	214809	2.36%	/~inbio/material/Apostila_Inbio_Biomol.pdf
3	1572	0.06%	125953	1.38%	/manuals/HTML/fontes/basicow95.zip
4	8653	0.35%	119737	1.31%	/manuals/HTML/
5	15	0.00%	109120	1.20%	/~eventos/icmcefei.ppt
6	57	0.00%	101760	1.12%	/~inbio/material/InBio_cursoIA_DM.PDF
7	146	0.01%	87743	0.96%	/manuals/HTML/fontes/gifcon32.exe
8	113	0.00%	87284	0.96%	/~sma/sma302/Calculo2.pdf
9	300	0.01%	82850	0.91%	/admin/telefones.html
10	532	0.02%	69801	0.77%	/~inbio/material/Apostila_biotec.pdf

Figura 5.17: Endereços com maior tráfego durante o mês de julho de 2002

- A maioria dos arquivos que estão sendo requisitados pelos usuários não está compactada, gerando um maior tráfego na rede. Esses arquivos deveriam ser comprimidos e, mesmo que não resultassem em uma diminuição significante de tamanho, ainda assim geraria menos tráfego.

- Nota-se que, entre os arquivos sendo requisitados, há uma página HTML `/admin/telefones.html`. Essa página corresponde a uma lista dos telefones do ICMC. Seria interessante que essa lista fosse separada em listas menores, indexadas por algum tópico como, por exemplo, departamentos do ICMC. Assim, quando algum usuário quisesse saber o telefone de apenas um departamento, não viria de todos, gerando uma quantidade menor de dados na rede.

Depois desses *rankings* sobre URLs, o Webalizer gera resultados sobre as páginas de entrada e saída, conforme é mostrado na Figura 5.18. Pode-se observar, nessa figura, os seguintes pontos:

- Observando-se que as páginas de entrada e de saída são praticamente as mesmas, pode-se supor que os usuários do *site* do ICMC o visitam sabendo o que querem, ou seja, não são usuários *hit-and-run* citados na Seção 3.3.2 na página 19.

- Outro ponto a ser observado refere-se às páginas relativas à seção do PosComp. Percebe-se que os usuários entram pela página `/~poscomp/` e saem pelas páginas

Top 10 of 5005 Total Entry Pages

#	Hits		Visits		URL
1	57347	2.32%	25421	25.42%	/
2	8653	0.35%	6960	6.96%	/manuals/HTML/
3	7399	0.30%	3747	3.75%	/~poscomp/
4	1590	0.06%	1542	1.54%	/~gbatista/magica/
5	1197	0.05%	804	0.80%	/ambiente/saocarlos/
6	3222	0.13%	772	0.77%	/~biblio/
7	670	0.03%	728	0.73%	/~andre/gene1.html
8	865	0.03%	706	0.71%	/ambiente/saocarlos/viagem.html
9	1117	0.05%	671	0.67%	/manuals/HTML/js.html
10	959	0.04%	669	0.67%	/ambiente/campus/

Top 10 of 5444 Total Exit Pages

#	Hits		Visits		URL
1	57347	2.32%	18461	17.58%	/
2	8653	0.35%	3131	2.98%	/manuals/HTML/
3	1590	0.06%	1772	1.69%	/~gbatista/magica/
4	2349	0.09%	1225	1.17%	/~poscomp/inscr_cand.html
5	1755	0.07%	1101	1.05%	/manuals/HTML/zip.html
6	7399	0.30%	1049	1.00%	/~poscomp/
7	3265	0.13%	1016	0.97%	/~poscomp/inscricoes.html
8	3222	0.13%	883	0.84%	/~biblio/
9	2094	0.08%	759	0.72%	/~poscomp/computacao/
10	865	0.03%	724	0.69%	/ambiente/saocarlos/viagem.html

Figura 5.18: Páginas de entrada e saída observadas no mês de julho de 2002

`/~poscomp/inscr_cand.html`, `/~poscomp/`, `/~poscomp/inscricoes.html` e `/~poscomp/computacao/`. Isso é interessante pois nota-se uma saída mais frequente pelas páginas `/~poscomp/inscricoes.html` e `/~poscomp/inscr_cand.html`, relativas às inscrições, ou seja, a maioria dos usuários que visitam as páginas do PosComp é para verificar informações relativas à inscrição. Resultado semelhante foi obtido durante a análise da ferramenta Apache2Dot.

Um outro resultado gerado pelo Webalizer é uma tabela onde é mostrado de quais `hosts` vêm a maioria dos acessos ao *site*. Conforme descrito anteriormente, a ferramenta foi configurada para agrupar os `hosts` dentro de seus domínios, resultando na tabela mostrada na Figura 5.19 na página oposta. O Webalizer também mostra, em outra tabela, a quantidade de dados em *KBytes* que foram transferidos para esses `hosts`. No entanto, essa tabela não é mostrada aqui por não ser relevante para as análises.

O único aspecto interessante na Figura 5.19 é poder observar que existem acessos feitos a partir de provedores como o IG e o Terra, além de serem observados acessos de dentro do próprio ICMC e da USP em geral. Os acessos feitos a partir de provedores podem ser de interesse de *sites* comerciais pois pode-se procurar parcerias.

Após os resultados sobre os `hosts`, tem-se uma tabela onde é mostrado de quais `referers` se tem mais acessos. Essa tabela pode ser visualizada na Figura 5.20 na página 80. O que pode ser observado nessa figura é que a grande maioria dos acessos ao ICMC provêm de *links* de *sites* de busca.

Um outro ponto a ser observado, e que também foi comentado na análise da ferramenta Apache2Dot é que a seção do *site* referente ao PosComp é acessada por meio de *sites* de outras instituições de ensino. Os `referers` `http://www.dcc.ufmg.br/pos/html/selecao.-html`, `http://www.dc.ufscar.br/posgrad/admissao2003.htm`, `http://www.pgcc.inf-.ufsc.br/`, `http://www.inf.ufrgs.br/pos/ppgc/mestrado/inscricoes.html`, `http:-//www.inf.ufpr.br/pos/selecao2003.html` e outros que podem ser observados na figura, são *sites* de outras instituições que utilizam o PosComp como processo seletivo para o programa de pós-graduação.

Outro resultado fornecido pela ferramenta é uma lista de palavras utilizadas nos *sites* de busca e que resultaram em um acesso ao *site* do ICMC. Essa lista é ilustrada na Figura 5.21. Essas palavras são obtidas pelo exame das *strings* de consulta, comentadas na Seção 4.4 na página 37, em busca dos padrões conhecidos de vários *sites* de busca. Esse *sites* e padrões podem ser especificados no arquivo de configuração do Webalizer.

O interessante de ser observado nessa figura é a palavra "`poscomp`", indicando que os acessos às páginas do PosComp não foram feitos somente por meio de *sites* de outras

Top 30 of 44504 Total Sites

#	Hits		Files		KBytes		Visits		Hostname
1	179846	7.27%	114172	7.31%	622923	6.84%	137	0.16%	telesp.net.br
2	160075	6.47%	107862	6.91%	519124	5.70%	154	0.18%	ig.com.br
3	70730	2.86%	48863	3.13%	244139	2.68%	291	0.33%	acessonet.com.br
4	53028	2.14%	33897	2.17%	143882	1.58%	330	0.38%	terra.com.br
5	52924	2.14%	39178	2.51%	178841	1.96%	308	0.35%	brasiltelecom.net.br
6	38135	1.54%	16031	1.03%	101688	1.12%	217	0.25%	icmc.usp.br
7	36180	1.46%	25611	1.64%	126427	1.39%	226	0.26%	speedyterra.com.br
8	23073	0.93%	8210	0.53%	48204	0.53%	286	0.33%	uspnet.usp.br
9	19881	0.80%	11226	0.72%	52759	0.58%	304	0.35%	brdterra.com.br
10	19429	0.78%	14147	0.91%	58176	0.64%	289	0.33%	telepar.net.br
11	17221	0.70%	10740	0.69%	101074	1.11%	178	0.20%	eesc.usp.br
12	12592	0.51%	10515	0.67%	47452	0.52%	269	0.31%	veloxzone.com.br
13	12478	0.50%	8787	0.56%	39401	0.43%	298	0.34%	ctbctelecom.com.br
14	12434	0.50%	8865	0.57%	66357	0.73%	169	0.19%	sc.usp.br
15	11857	0.48%	8054	0.52%	19794	0.22%	142	0.16%	nr.ip.pt
16	11621	0.47%	7611	0.49%	47631	0.52%	330	0.38%	proxy.aol.com
17	10894	0.44%	7748	0.50%	53108	0.58%	232	0.27%	telemar.net.br
18	9619	0.39%	7882	0.50%	38828	0.43%	236	0.27%	intelignet.com.br
19	9487	0.38%	6155	0.39%	24530	0.27%	209	0.24%	ipt.aol.com
20	8731	0.35%	5126	0.33%	17531	0.19%	34	0.04%	im.ufba.br
21	8541	0.35%	6032	0.39%	30561	0.34%	212	0.24%	speeduol.com.br
22	7958	0.32%	4661	0.30%	26923	0.30%	102	0.12%	dm.ufscar.br
23	7665	0.31%	6457	0.41%	32513	0.36%	240	0.27%	telesc.net.br
24	7354	0.30%	5941	0.38%	16212	0.18%	208	0.24%	prodigy.net.mx
25	7214	0.29%	5215	0.33%	23027	0.25%	213	0.24%	embratel.net.br
26	7112	0.29%	5270	0.34%	36617	0.40%	243	0.28%	virtua.com.br
27	7068	0.29%	2333	0.15%	12041	0.13%	65	0.07%	recad.usp.br
28	6679	0.27%	5099	0.33%	23215	0.25%	150	0.17%	pr7-ts.telepac.pt
29	6582	0.27%	201	0.01%	3665	0.04%	37	0.04%	x1crawler2-1-0.x-echo.com
30	4994	0.20%	4228	0.27%	13069	0.14%	111	0.13%	cache.rnp.br

View All Sites

Figura 5.19: `Hosts` de onde foram feitos acessos ao ICMC durante o mês de julho de 2002

Top 30 of 7362 Total Referrers			
#	Hits		Referrer
1	308527	12.46%	- (Direct Request)
2	26219	1.06%	Google
3	6356	0.26%	Yahoo!
4	3089	0.12%	MSN
5	2570	0.10%	AltaVista
6	1726	0.07%	Radar UOL
7	1611	0.07%	http://www2.usp.br/cgi-bin/wxis/wxis.cgi
8	1261	0.05%	http://lightning.prohosting.com/~goguega/acorde-c.html
9	1095	0.04%	http://www.dcc.ufmg.br/pos/html/selecao.html
10	1037	0.04%	http://www.dc.ufscar.br/posgrad/admissao2003.htm
11	998	0.04%	http://www2.usp.br/cgi-bin/insearch/insearch.cgi
12	744	0.03%	Miner BOL
13	583	0.02%	http://lightning.prohosting.com/~goguega/acorde-g.html
14	555	0.02%	http://www.pgcc.inf.ufsc.br/
15	463	0.02%	http://www.inf.ufrgs.br/pos/ppgc/mestrado/inscricoes.html
16	431	0.02%	http://coweb.icmc.sc.usp.br/coweb/mostra.php
17	406	0.02%	http://www.inf.ufpr.br/pos/selecao2003.html
18	362	0.01%	http://www.geocities.com/Colosseum/Gym/7220/midi.htm
19	304	0.01%	http://216.239.51.100/search
20	297	0.01%	http://www.geocities.com/Area51/Aurora/9541/cavaco.htm
21	282	0.01%	http://216.239.33.100/search
22	274	0.01%	http://www.ime.usp.br/dcc/posgrad/Admissao/
23	249	0.01%	http://216.239.35.120/translate_c
24	236	0.01%	http://www.inf.ufpr.br/pos/selecao2002.html
25	232	0.01%	http://busca.terra.com.br/busca/
26	225	0.01%	http://www.mcc.ufc.br/aviso_selecao.htm
27	202	0.01%	http://www.noronha.pro.br/ApostCha.htm
28	187	0.01%	http://www.construindoseusite.com.br/html/html.shtm
29	186	0.01%	http://search.zoom.globo.com/glbzoomSearch/engine
30	185	0.01%	http://www.dcc.unicamp.br/cpg/faq.html
View All Referrers			

Figura 5.20: `Referers` de onde foram feitos acessos ao ICMC durante o mês de julho de 2002

intituições de ensino, mas também por *sites* de busca. Outras palavras interessantes são as relacionada ao manual de HTML como "`html`", "`tutorial html`", "`javascript`", "`gif animado`" entre outras. Na analise da ferramenta Apache2Dot foi comentado que esse resultado complementaria o fato da página inicial do manual de HTML ser acessado diretamente, ou seja, sem ser por meio de outros *links* das páginas do *site* do ICMC. Também pode ser observado, na análise dos `referers`, na Figura 5.20 na página anterior, que o acesso ao manual de HTML também é feito por outros *sites* como, por exemplo, `http://www.construindoseusite.com.br/html/html.shtm`.

Top 20 of 13251 Total Search Strings			
#	Hits		Search String
1	1561	5.43%	html
2	630	2.19%	usp
3	491	1.71%	tutorial html
4	438	1.52%	javascript
5	263	0.91%	gif animado
6	263	0.91%	poscomp
7	150	0.52%	algoritmos
8	143	0.50%	maquina de turing
9	142	0.49%	baralho
10	125	0.43%	algoritmos geneticos
11	123	0.43%	redes neurais
12	119	0.41%	sao carlos
13	119	0.41%	são carlos
14	105	0.36%	magica
15	101	0.35%	estrutura de dados
16	100	0.35%	html tutorial
17	91	0.32%	frames
18	91	0.32%	tutorial
19	84	0.29%	imagens gif
20	77	0.27%	carlos ponce
View All Search Strings			

Figura 5.21: Palavras de busca que resultaram em acessos ao ICMC durante o mês de julho de 2002

O próximo resultado gerado pelo Webalizer é uma lista do navegadores mais utilizados, ilustrada na Figura 5.22 na página seguinte. Os seguintes pontos podem ser observados:

- Os navegadores Explorer e Netscape são os mais utilizados, sendo que o Explorer é muito mais utilizado que o Netscape.
- Os navegadores chamados "Google" e "Altavista" são, na realidade, os robôs de busca, que ficam navegando por vários *sites* e indexando o seu conteúdo (Apêndice C).
- Observa-se uma grande variedade de programas que fazem o *download* de *sites*, como o "Wget" (o mais utilizado segundo a Figura 5.22), "WebCopier", "WebZIP", entre

Top 15 of 2365 Total User Agents			
#	Hits		User Agent
1	2113136	85.37%	Explorer
2	276411	11.17%	Netscape
3	6939	0.28%	Wget
4	4747	0.19%	Google
5	4500	0.18%	Altavista
6	3272	0.13%	WebCopier
7	2959	0.12%	ia_archiver
8	2848	0.12%	FAST-WebCrawler/3.6 (atw-crawler at fast dot no; http://fast.
9	2326	0.09%	SiteSnagger
10	2147	0.09%	DA 5.0
11	2139	0.09%	WebZIP/4.1 (http://www.spidersoft.com)
12	1944	0.08%	Mercator-2.0
13	1829	0.07%	MSProxy/2.0
14	1770	0.07%	SlySearch/1.4 http://www.slysearch.com
15	1626	0.07%	contype
View All User Agents			

Figura 5.22: Navegadores mais utilizados para acessar o *site* do ICMC durante o mês de julho de 2002

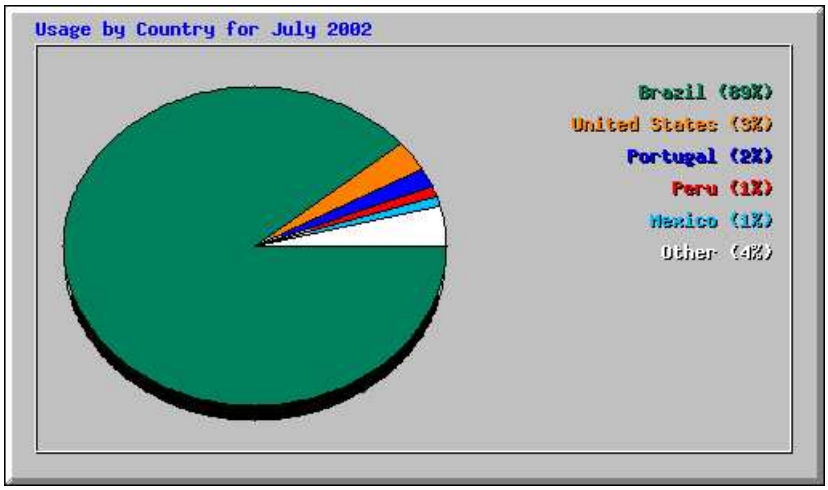
outros. Isso indica que alguns conteúdos do *site* do ICMC são interessantes ao ponto dos usuários quererem fazer uma cópia em sua máquina.

O último resultado gerado pelo Webalizer é uma lista dos países de onde provêm os acessos ao *site* sendo analisado. O único aspecto que foi notado nessa lista foi o fato do Peru e Chile terem menos acessos que o México, já que o ICMC tem mais contato com as instituições peruanas e chilenas. No entanto, isso foi verificado quando o Webalizer foi executado sem as alterações para se utilizar a biblioteca GeoIP. Após ser executada a versão modificada do Webalizer, verificou-se a lista mostrada na Figura 5.23 na próxima página, onde o Peru tem mais acesso que o México, apesar do Chile ainda continuar com poucos acessos. Talvez deve-se normalizar os dados pela população de cada país, uma vez que o México possui uma população maior e por isso aparece como tendo maior número de acessos.

5.3.2.4 Conclusões

Conforme foi visto, o Webalizer é uma boa ferramenta para obter-se algumas estatísticas básicas sobre os acessos feitos no *site* sendo analisado. Além disso, ela faz automaticamente a parte mais comum de pré-processamento, sendo que não é necessário alterar-se muito o arquivo de *log* para se obter resultados.

Um dos pontos negativos da ferramenta é que ela fornece algumas tabelas consideradas sem muito valor prático como, por exemplo, uma tabela com os valores observados no



Top 30 of 104 Total Countries

#	Hits		Files		KBytes		Country
1	2209902	89.28%	1388668	88.94%	7819397	85.85%	Brazil
2	79367	3.21%	54450	3.49%	439201	4.82%	United States
3	51458	2.08%	39225	2.51%	155423	1.71%	Portugal
4	34642	1.40%	19495	1.25%	203579	2.24%	Peru
5	16916	0.68%	14057	0.90%	44454	0.49%	Mexico
6	10946	0.44%	3433	0.22%	29699	0.33%	France
7	9264	0.37%	7632	0.49%	46428	0.51%	Spain
8	7646	0.31%	6229	0.40%	33508	0.37%	Great Britain (UK)
9	6424	0.26%	4841	0.31%	15027	0.16%	Venezuela
10	5720	0.23%	4690	0.30%	48063	0.53%	Germany
11	5596	0.23%	4528	0.29%	31472	0.35%	Japan
12	5209	0.21%	4085	0.26%	13829	0.15%	Colombia
13	3655	0.15%	3297	0.21%	15374	0.17%	Argentina
14	3039	0.12%	2842	0.18%	10763	0.12%	Chile
15	2563	0.10%	1959	0.13%	30112	0.33%	Italy
16	1971	0.08%	1727	0.11%	16146	0.18%	Canada
17	1460	0.06%	1095	0.07%	20366	0.22%	China
18	1349	0.05%	1112	0.07%	9421	0.10%	Bolivia
19	1328	0.05%	1127	0.07%	4532	0.05%	Europe
20	1143	0.05%	1002	0.06%	9453	0.10%	Netherlands
21	988	0.04%	819	0.05%	1782	0.02%	Panama
22	963	0.04%	828	0.05%	6600	0.07%	Switzerland
23	940	0.04%	867	0.06%	2505	0.03%	Ecuador
24	898	0.04%	728	0.05%	9697	0.11%	Israel
25	845	0.03%	682	0.04%	7496	0.08%	Belgium
26	835	0.03%	795	0.05%	2410	0.03%	Mozambique
27	744	0.03%	622	0.04%	6705	0.07%	Australia
28	724	0.03%	673	0.04%	2451	0.03%	Costa Rica
29	632	0.03%	538	0.03%	4010	0.04%	Sweden
30	537	0.02%	436	0.03%	2393	0.03%	Guatemala

Figura 5.23: Países de onde provêm os acessos ao *site* do ICMC durante o mês de julho de 2002

gráfico de acessos por dia — Figura 5.12 na página 71. Essa tabela não tem valor prático, pois contém apenas números que não podem ser analisados convenientemente na forma em que são apresentados. Um outro ponto negativo da ferramenta é que ela não possui uma forma prática de se comparar os resultados específicos entre os diversos meses. Por exemplo, se quer-se comparar os acessos diários nos meses de julho e agosto, como foi feito na Figura 5.12 na página 71, é necessário executar dois navegadores, cada um contendo as estatísticas de cada mês.

Apesar disso, a ferramenta oferece alguns resultados interessantes como, por exemplo, as palavras que foram utilizadas pelos usuários em *sites* de busca. Isso pode ser fundamental em um *site* de comércio. Um outro resultado importante é a tabela de `referers`. Com as informações dessa tabela, pode-se, por exemplo, verificar se um *banner* de propaganda está sendo bem sucedido, uma vez que pode-se ver a quantidade de acessos que está sendo feito ao *site* por meio desse *banner*.

5.4 Consideraçoes Finais

Um dos pontos importantes que foram observados realizando essas experiências, é que cada navegador segue um padrão diferente para o envio das requisições, por exemplo, substituindo ou não os códigos hexas. Isso torna complicada a análise dos arquivos de *log* se a ferramenta não faz um pré-processamento dessas requisições, como é o caso do Apache2Log.

No entanto, essas ferramentas permitem, entre outras utilidades, estruturar o *site* e sua infra-estrutura, fornecendo informações que darão o suporte necessário à tomada de certas decisões. Apesar de não se poder conhecer muito de cada tipo de usuário, é possível de se ter uma pequena idéia do tipo de informação que está sendo procurada no *site* sendo analisado. É importante notar que na ferramenta ApacheDot tem-se uma visão local, com número de acessos em cada *link*, enquanto que no Webalizer tem-se uma visão global. Assim, ambas ferramentas se complementam.

Seria interessante que esses tipos de ferramentas também pudessem aceitar um conhecimento de fundo como, por exemplo, a estrutura física do *site* e uma descrição das páginas. Assim, ao se analisar um grafo do tipo gerado pelo Apache2Dot, vería-se quais dos *links* existentes estão sendo utilizados e sobre qual assunto trata cada página. No caso do Apache2Dot, houve a necessidade de se visitar as páginas para se ter essa visão. Uma ferramenta complementar ao Apache2Dot seria uma que gerasse um grafo mostrando quais palavras de busca levaram a quais páginas do *site*.

Com relação à análise do *site* do ICMC, notou-se que os acessos concentram-se, basica-

mente, em três áreas: no manual de HTML, na seção do PosComp e em torno das seções de ensino de graduação e pós-graduação. No entanto, a seção do PosComp deve ser de acesso sazonal, ou seja, apenas em épocas próximas do evento será verificado um aumento no número de visitas.

Como novo conhecimento adquirido dos estudos das outras ferramentas que não foram analisadas, descobriu-se que alguns navegadores enviam uma requisição ao arquivo `favicon.ico` sempre que o usuário colocar uma página no *bookmark* do navegador, ou seja, apesar de não querer obter nenhum arquivo físico, ele fornece uma indicação da ação do usuário. Isso pode ser útil para descobrir quais os assuntos de maior interesse do usuário, dentro do *site* sendo analisado.

Capítulo 6

A Ferramenta Proposta

6.1 Considerações Iniciais

Ao se aplicar as técnicas de Mineração de Dados utilizando-se algoritmos de Aprendizado de Máquina enfrenta-se um problema: a maioria dos algoritmos de Aprendizado de Máquina tem um formato diferente para os arquivos de entrada. De forma a contornar esta dificuldade, foi desenvolvida uma sintaxe padrão para a representação desses arquivos de entrada (Batista 2001), os quais consistem num arquivo de dados (exemplos) propriamente dito, e num arquivo que descreve os atributos desses dados.

Esses arquivos, na sintaxe padrão, são utilizados pelo Sistema DISCOVER, que está sendo implementado no LABIC[1] — Laboratório de Inteligência Computacional. Esse sistema tem como objetivo auxiliar no processo de descoberta de conhecimento em bases de dados (KDD). Ele consiste em um conjunto de *scripts* Perl (Wall and Schwartz 1991) e em uma biblioteca de rotinas que são utilizadas pelos *scripts*. Esses *scripts* estão sendo integrados através de uma interface gráfica (Geromini 2002). Alguns desses *scripts* são filtros que transformam arquivos de dados na sintaxe padrão para a sintaxe dos arquivos de entrada dos diversos algoritmos de Aprendizado de Máquina suportados pelo Sistema DISCOVER— Figura 6.1.

Dessa forma, para que os arquivos de *logs* de servidores *Web* possam ser utilizados no Sistema DISCOVER com o objetivo de extrair conhecimentos desses *logs*, neste trabalho foi desenvolvido um filtro que transforma esses *logs* em arquivos de dados na sintaxe padrão do DISCOVER.

[1] http://labic.icmc.sc.usp.br

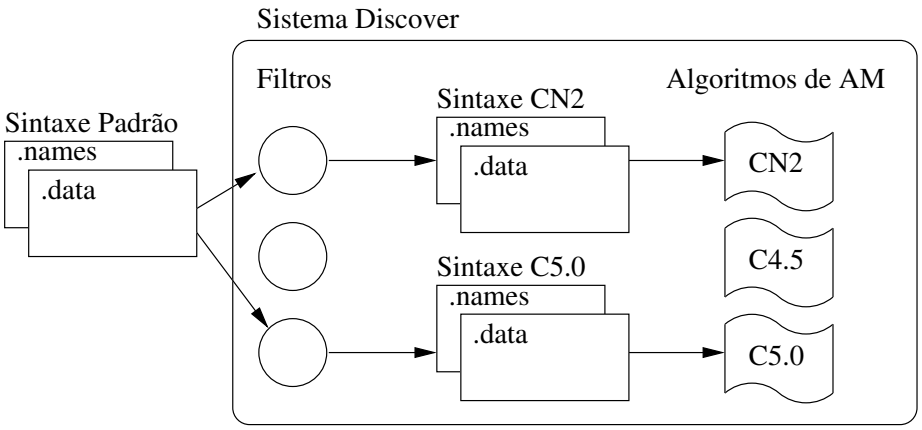


Figura 6.1: Exemplo de utilização de filtros no Sistema DISCOVER

6.2 O Projeto Discover

A descrição aqui apresentada está baseada em diversos trabalhos realizados por integrantes da equipe de desenvolvimento do Sistema DISCOVER. Assim, maiores informações podem ser obtidas em (Batista 2001), (Kemp, Batista, and Monard 2002) e (Prati 2003), (Prati, Baranauskas, and Monard 2002).

O processo de KDD requer experimentação, análise e comparação de diversos modelos de extração de conhecimento, na busca por um resultado satisfatório. Na condução desses experimentos, atividades como transformações de formato, adaptações, execução de diferentes algoritmos de Aprendizado de Máquina, medições da precisão, entre outras, devem ser executadas um grande número de vezes.

Muitas dessas tarefas podem ser automatizadas utilizando-se sistemas integrados comerciais. Geralmente, essas ferramentas tem um caráter mais exploratório e fazem uso de algoritmos e ferramentas proprietários, o que dificulta o seu uso por pesquisadores que pretendem analisar e desenvolver novos algoritmos e ferramentas. Uma alternativa é o uso de ferramentas de domínio público. Entretanto, essas ferramentas também têm algumas características não apropriadas para realizar pesquisas mais abrangentes.

Esses fatores conduzem, diversas vezes, à necessidade de se utilizar os algoritmos de aprendizado tal como foram implementados pelos seus idealizadores e, por conseguinte, todas as atividades necessárias para a execução de experimentos devem ser feitas para cada algoritmo particular. Essa necessidade implica, muitas vezes, no desenvolvimento de programas para automatizar essas tarefas.

Nos últimos anos, diversos pesquisadores de nosso laboratório de pesquisa (LABIC[2]) têm utilizado algoritmos de Aprendizado de Máquina em suas pesquisas, desenvolvendo, na

[2] http://labic.icmc.usp.br

forma de *scripts* Perl, uma série de ferramentas para facilitar a configuração e execução de experimentos (Prati, Baranauskas, and Monard 2001b), (Prati, Baranauskas, and Monard 2001a), (Batista 2001), (Kemp, Batista, and Monard 2001), (Baranauskas 2001), (Baranauskas and Monard 2003), (Bernardini 2002), (Gomes 2002), (Imamura 2001), (Milaré 2000), (Paula 2003), (Pila 2001), (Pugliesi 2001).

Surgiu, então, a proposta de desenvolver um projeto conjunto no qual todos os membros do laboratório que trabalham na área de KDD estariam envolvidos: o projeto Discover (Baranauskas and Batista 2000). A princípio, o projeto Discover consistiria apenas de um repositório de *scripts*. Por meio da combinação desses *scripts* independentes seria possível a realização de tarefas mais complexas. Posteriormente surgiu a proposta de se criar um ambiente integrado, no qual os *scripts* seriam substituídos por bibliotecas de classes e essas classes empacotadas como componentes, com a composição dos componentes sendo feita através de uma interface gráfica (Geromini 2002).

6.2.1 O Sistema Discover

A idéia central do Sistema Discover consiste em utilizar os algoritmos de aprendizado implementados pela comunidade e as ferramentas com finalidades específicas desenvolvidas pelos pesquisadores relacionados ao projeto, tais como ferramentas de pré-processamento de dados e pré-processamento de texto, tanto para aprendizado supervisionado quanto para aprendizado não-supervisionado, amostragem e avaliação de erro, mesclagem de regras, cobertura de regras, qualidade de regras, entre outros.

De uma maneira geral, o Sistema Discover pode ser entendido como um conjunto de métodos que são aplicados sobre os dados ou sobre o conhecimento extraído a partir dos dados. Dessa forma, é muito importante que o sistema ofereça um base sólida para manipular dados e conhecimento. Essa base é composta por sintaxes padrões para a representação de dados e de conhecimento, e por bibliotecas que oferecem um conjunto de funcionalidades básicas de manipulação de dados e de conhecimento. Atualmente, existem definidas sintaxes padrões para a representação de dados no formato atributo-valor e para a representação do conhecimento induzido por diversos indutores simbólicos, bem como bibliotecas que oferecem funcionalidades sobre essas sintaxes padrões. Futuramente, novas sintaxes padrão devem ser especificadas, principalmente para a representação de regras de regressão (Dosualdo 2002), regras de associação (Melanda 2002) e *clusters* (Martins 2001). Na Figura 6.2 é mostrado, de uma forma simplificada, como os filtros, sintaxes e bibliotecas interagem uns com os outros, para o caso de algoritmos simbólicos de aprendizado supervisionado.

A vantagem do Sistema Discover como ferramenta de apoio à pesquisa em KDD, em

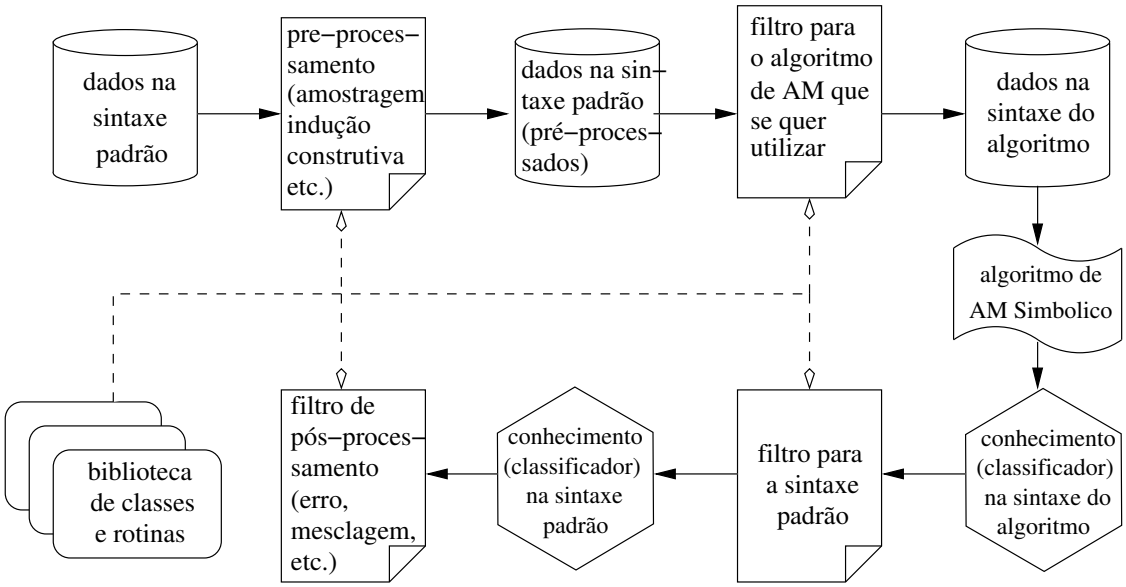


Figura 6.2: Interação entre filtros, sintaxes e bibliotecas

relação a outros sistemas, é a visão unificada que os formatos padrões oferecem para quem está desenvolvendo novos componentes, além de um conjunto de ferramentas para a manipulação dos mesmos.

Uma outra vantagem que se espera é que, a partir da conclusão dos diversos trabalhos que estão sendo realizados, o Sistema Discover ofereça uma série de vantagens em relação à composição das ferramentas desenvolvidas em nosso laboratório, evitando que essas implementações se percam e, além disso, reunindo-as em um ambiente integrado.

6.2.2 A Sintaxe Padrão para Arquivos de Dados

Como já mencionado, o Sistema Discover visa, entre outras coisas, integrar alguns dos algoritmos mais conhecidos de Aprendizado de Máquina simbólico em uma única ferramenta, além de realizar estudos comparativos entre eles.

Contudo, os diversos algoritmos de classificação usados (C4.5, C5.0, CN2, entre outros) possuem sintaxes diferentes para seus arquivos de entrada, tanto o arquivo de atributos quanto o de dados. Isso faz com que seja necessário definir uma sintaxe padrão para representar os arquivos que serão utilizados nesse ambiente. Uma vez gerados tais arquivos nessa sintaxe padrão, o ambiente possui filtros para converter os arquivos dessa sintaxe para a sintaxe de um algoritmo em particular (C4.5, por exemplo). Assim, novos algoritmos de Aprendizado de Máquina podem ser facilmente adicionados ao Discover por meio da implementação desses filtros, como ilustrado na Figura 6.1 na página 88.

6.2.2.1 Arquivos

A sintaxe padrão dos dados, definida em (Batista 2001), utiliza arquivos do tipo texto para declarar os atributos (e seus respectivos tipos) e os valores que esses atributos assumem em um determinado conjunto de exemplos. Os atributos são declarados em um arquivo com a extensão `.names` e os valores que esses atributos assumem em um conjunto de exemplos são declarados em um outro arquivo com a extensão `.data`. Os dois arquivos devem possuir o mesmo nome, se diferenciando apenas pela extensão.

No caso de Aprendizado de Máquina supervisionado, a primeira declaração em um arquivo `.names` define qual deve ser o atributo classe. O atributo classe pode ser qualquer atributo presente no conjunto de exemplos. Após a declaração do atributo classe, são declarados os demais atributos. Cada atributo possui um identificador e um tipo de dado associado a ele. São válidos os identificadores que são combinações de números, letras e "_" (underscore), em qualquer seqüência. Para identificadores mais complexos que envolvem outros caracteres que não sejam os especificados anteriormente (como espaços, letras acentuadas, etc.) é necessário colocar o identificador entre aspas. Desta forma, são identificadores válidos: abc, 1, 1a, _1a, " _12a ", "válido".

No arquivo `.data` são declarados os valores que os atributos presentes no arquivo `.names` assumem para um conjunto de exemplos. Cada linha de um arquivo `.data` representa um exemplo. Desta forma, o "separador de registros" é o caractere de nova linha (representado em muitas linguagem de programação por `\n`). Cada linha possui uma seqüência de valores separados por vírgula, ou seja, a vírgula é o separador de campos. Cada valor presente em uma linha está associado a um atributo do arquivo `.names`. Sendo assim, a ordem em que os valores são declarados em uma determinada linha deve ser a mesma ordem na qual os atributos foram declarados no arquivo `.names`.

6.2.2.2 Tipos de Dados

Foram definidos os seguintes tipos de dados[3], que podem ser associados aos identificadores de atributos:

Nominal O tipo `nominal` é utilizado para declarar um atributo que pode assumir um grupo restrito de valores.

Enumerated O tipo de dado `enumerated` é semelhante ao tipo de dado `nominal`. A principal diferença é que com o tipo `enumerated` é possível identificar uma ordem

[3]Os termos utilizados nas implementações de cada componente do Sistema Discover estão em inglês pois pretende-se compartilhar os resultados do projeto com a comunidade científica.

entre os valores que o atributo pode assumir. Entretanto, não existe uma definição clara de distância entre esses valores. Um exemplo de tipo `enumerated` é um atributo que pode assumir os valores `pequeno`, `médio` e `grande`.

Integer O tipo de dado `integer` é utilizado para declarar um atributo que pode assumir um valor inteiro.

Real O tipo `real` é semelhante ao tipo de dado `integer`, com a diferença que um atributo `real` pode armazenar números com ou sem parte fracionária.

String Um atributo `string` pode assumir como valor uma string de tamanho indefinido e que pode conter quaisquer caracteres incluindo quebra de linha (`\n`). Para identificar os limites de um string é necessário inserir o símbolo de aspas (`"`) no início e no fim da string.

Date O tipo de dado `date` permite declarar um atributo que pode conter uma data (dia, mês e ano). A princípio, os valores das datas devem estar no formato `mm/dd/aaaa` (formato americano).

Time O tipo de dado `time` permite declarar um atributo que pode conter um horário (hora, minuto e segundo). A princípio, os valores dos horários devem estar no formato `hh:mm:ss`.

6.2.2.3 A Gramática da Sintaxe Padrão

A gramática que define a sintaxe do arquivo `.names` é mostrada na Figura 6.3 na próxima página.

Para implementar *scripts* usando a sintaxe definida para o Sistema Discover, foi desenvolvida uma biblioteca orientada a objetos, em Perl, que implementa uma série de métodos para a manipulação de arquivos na sintaxe padrão. A descrição da organização dessa biblioteca, bem como sua principais funções, podem ser encontradas em (Batista 2003a).

6.3 O Filtro

Neste trabalho foi projetado e desenvolvido o filtro `log2discover.pl` com a finalidade de transformar arquivos de *log* de servidores *Web* em arquivos na sintaxe padrão do Sistema Discover, com o objetivo de extrair informações e adquirir conhecimento utilizando os dados desses arquivos de *log* e as ferramentas que constituem o Discover.

```
S ::= <class-defs> | <feature-defs>

<class-defs> ::= <feature-name> . | null .

<feature-name> ::= <identifier>

<feature-defs> ::=
    <feature-name> : <feature-type> .
  | <feature-name> : <feature-type> : <extended-defs> .
  | <feature-name> : <feature-type> := <expression> : <extended-defs> .

<feature-type> ::=
    real
  | integer
  | boolean
  | nominal
  | nominal (<list>)
  | enumerated (<list>)
  | date
  | time
  | string

<extended-defs> ::= <extended-def> | <extended-def> : <extended-defs>

<extended-def> ::= <identifier> | <identifier> (<list>)

<list> ::= <identifier> | <identifier> , <list>
```

Figura 6.3: Gramática da sintaxe do arquivo `.names`.

No desenvolvimento do filtro `log2discover.pl`, foi utilizada a linguagem *freeware* Perl, que é a mesma utilizada no desenvolvimento do Sistema DISCOVER. Além disso, a própria linguagem Perl foi desenvolvida como uma ferramenta para processamento de *strings*, o que facilita muito o desenvolvimento de um filtro como o proposto neste trabalho.

6.3.1 Definição do arquivo `.names`

Antes de iniciar o desenvolvimento do filtro, a estrutura do arquivo `.names` teve que ser definida. Procurou-se, para isso, incluir os campos que um arquivo de *log* possui e que foram considerados valiosos para um primeiro estudo na área de *Web Usage Mining*. Também tomou-se o cuidado de separar os campos em campos atômicos, ou seja, que não possuem duas informações juntas, como é o caso da data e hora, as quais foram separados em campos distintos. Também foram incluídos campos que não necessariamente têm dados neste primeiro instante, ou seja, não têm campos equivalentes nos *logs*, mas que consideramos importantes para serem estudados em trabalhos futuros.

Para nomear os campos do arquivo `.names`, foi utilizada uma variação da nomenclatura proposta pelo World-Wide-Web Consortium (W3C) (Hallam-Baker and Behlendorf 1996). Nessa nomenclatura, os nomes dos campos podem ter uma das seguintes formas:

- **<identifier>** O nome do campo está relacionado com a transação HTTP como um todo. O identificador `data`, por exemplo, indica o momento em que a transação, como um todo, foi executada.
- **<prefix>_<identifier>** Identifica uma informação específica que é transferida na transação. O prefixo indica de quem é essa informação ou o fluxo que é seguido. O IP do cliente (c), por exemplo, é indicado por `c_ip` e o do servidor (s) por `s_ip`.
- **<prefix>h_<header>** Identifica uma informação específica que é transferida através do cabeçalho (**h**eader) do protocolo HTTP. O prefixo tem o mesmo significado que no formato anterior.

Os prefixos que podem ser utilizados são os seguintes:

- **c** Cliente.
- **s** Servidor.
- **cs** Do cliente para o servidor.
- **sc** Do servidor para o cliente.

- **x** Identificador de uma aplicação específica.

Assim, um campo com nome `cs_method` indica qual o método de requisição enviado do cliente para o servidor enquanto que `csh_referer` refere-se ao campo `referer` transferido, do cliente para o servidor, no cabeçalho do protocolo HTTP.

A seguir, é dada uma lista dos campos que estão presentes no arquivo `.names` por nós definido:

c_ip O endereço de Internet do cliente que fez a requisição. Esse é o endereço para o qual a resposta do servidor será enviada. A maioria dos computadores em rede não têm endereços de Internet fixos. Em vez disso, o endereço é atribuído dinamicamente para o computador quando o usuário faz uma conexão com o seu provedor através do *modem*. Mesmo que o endereço seja dinâmico, ele permanece o mesmo durante uma sessão e pode ser utilizado para "amarrar" os eventos de uma sessão.

c_dns O filtro está preparado para traduzir os endereços IPs para o respectivo nome de domínio. O nome traduzido é armazenado neste campo. Nem sempre é possível fazer essa tradução. Nesses casos, o campo conterá um valor desconhecido, indicado pelo símbolo '`?`'.

c_userid O nome de usuário da pessoa requisitando o documento, conforme determinado pelo protocolo de autenticação do HTTP (Group 1999b). Se o documento sendo requisitado não está protegido por senha, então este campo armazenará um valor desconhecido.

date A data na qual a transação HTTP foi completada.

time A hora na qual a transação HTTP foi completada.

cs_method Este campo armazena o nome do método utilizado para solicitar um documento do servidor, conforme descrito na Seção 4.3 na página 32.

cs_uri URIs (*Uniform Resource Identifiers*) são *strings* que identificam recursos (documentos, imagens, arquivos para download, serviços, caixas de e-mail, etc.) na *Web*. No caso do campo `cs_uri`, ele indica a localização do arquivo que o cliente deseja obter do servidor.

cs_uri_stem A URI sem os parâmetros.

cs_uri_query Somente os parâmetros da URI.

cs_version A versão do protocolo HTTP sendo utilizado. Nota-se que quem indica qual versão do protocolo utilizar é o cliente, pois essa informação trafega do cliente para o servidor.

sc_status O resultado da transação, como definido no protocolo HTTP ((Group 1999a)).

sc_bytes O número de *bytes* transferidos do servidor para o cliente.

csh_referer O `referer` é uma informação que trafega no cabeçalho do protocolo e indica a URI de origem da URI (`cs_uri`) sendo requisitada, como descrito na Seção 4.3 na página 32.

csh_referer_stem A URI do `referer` sem os parâmetros.

csh_referer_query Somente os parâmetros passados para o `referer`.

csh_user_agent O nome e/ou a versão do navegador sendo utilizado pelo cliente.

Os próximos campos contêm somente o valor "`?`" (valor desconhecido na sintaxe padrão) pois foi decidido incluí-los para serem considerados em trabalhos futuro.

csh_cookie Algumas versões Apache (nome de um servidor *Web* muito utilizado) têm a possibilidade de ativar um módulo que permite rastrear um usuário através de *cookies*. *Cookies* são arquivos enviados pelo servidor e que o navegador guarda no disco da máquina cliente. Depois, toda vez que o navegador visitar *esse site*, ele envia esse arquivo de volta para o servidor. Assim, o campo `csh_cookie` serve para guardar o número utilizado para rastrear o usuário, na tentativa de se identificar, no *log*, qual a origem da requisição. O motivo de considerar este campo como trabalho futuro é que o módulo que permite rastrear um usuário não é ativado por *default* no Apache.

x_session Este campo não está presente no arquivo de *log*, mas é uma informação secundária que tem que ser extraída de uma combinação de registros do *log*. A sessão é uma forma de identificar o usuário que está acessando o servidor. Se o campo `csh_cookie` for usado, não há necessidade de se usar o `x_session`. Mas, no caso do `csh_cookie` não puder ser usado ou o servidor *Web* não suportar esta funcionalidade, o campo `x_session` é uma solução para rastrear os usuários. O problema com este campo é que não é fácil extrair esta informação, normalmente sendo infererida por heurísticas aplicadas no arquivo de *log*, não sendo, assim, uma informação precisa. No presente momento, o filtro não infere esta informação (trabalho futuro).

x_uri_type Este campo indica qual o tipo de recurso que foi requisitado. Este recurso pode ser um arquivo HTML, uma figura, um arquivo de *script* (como uma página dinâmica), entre outros. Esta informação não está no arquivo de *log*, devendo ser obtida de uma base de informações secundária, criada, por exemplo, pelo *webmaster* do *site*.

x_uri_subtype Este campo indica o sub-tipo do recurso sendo requisitado. Ele é uma informação que complementa o campo `x_uri_type`. Assim, se no campo `x_uri_type` tem-se a indicação de que o recurso requisitado é uma página HTML, o campo `x_uri_subtype` pode indicar que esta página é uma página de descrição de um produto, por exemplo.

x_referer_type Semelhante ao campo `x_uri_type`, mas aplicado ao `referer`.

x_referer_subtype Semelhante ao campo `x_uri_subtype`, mas aplicado ao `referer`.

6.3.2 Implementação

O filtro implementado lê um arquivo de *log* (entrada) e produz os dois arquivos de saída: o arquivo `.names` e o arquivo `.data` na sintaxe do Sistema DISCOVER, conforme mostra a Figura 6.4.

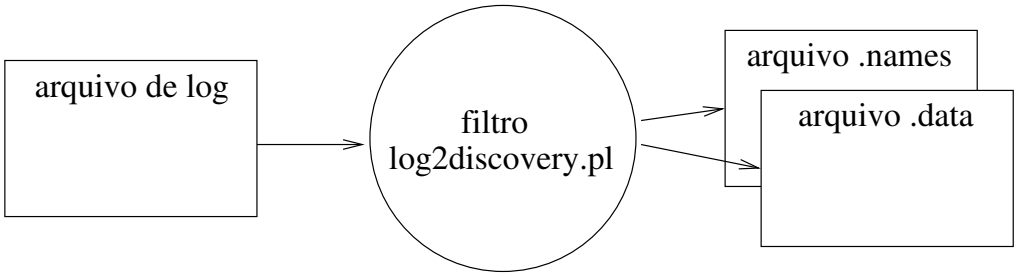


Figura 6.4: Entrada e saídas do filtro `log2discover.pl`.

Assim, na implementação, o código do filtro foi dividido, basicamente, em três partes:

1. Obtenção dos parâmetros de entrada.
2. Criação do arquivo `.names`.
3. Criação do arquivo `.data`.

Essas partes estão descritas nas seções seguintes.

6.3.2.1 Obtenção dos parâmetros de entrada

O filtro `log2discover.pl` aceita como entrada os seguintes parâmetros, listados a seguir:

-if (**i**nput **f**ile). Este é um parâmetro obrigatório que especifica qual o nome do arquivo de *log* que deve ser lido e processado.

-pr (**pr**efix). O prefixo indica qual o nome dos arquivos de saída que serão gerados. Se nada for especificado, o filtro utilizará o nome `log` e os arquivos de saída serão: `log.names` e `log.data`.

-id (**i**nitial **d**ate). Se uma data for especificada, serão processados somente os registros do arquivo de *log* com data posterior à especificada.

-fd (**f**inal **d**ate). Se uma data for especificada, serão processados somente os registro do arquivo de *log* com data anterior à especificada.

Para fazer o *parsing* da linha de comando, foi utilizado um pacote para Perl chamado `Getopt::Long`.

6.3.2.2 Criação do arquivo `.names`

Como já mencionado, o arquivo `.names` é o arquivo que descreve, no Sistema Discover, os campos dos registros que serão processados pelos algoritmos de Aprendizado de Máquina. Não há nada de especial na codificação dessa parte do filtro: ele cria um arquivo cujo nome encontra-se especificado no parâmetro de entrada `-pr` e escreve nele um texto contendo os nomes e os tipos dos campos no formato padrão para arquivos de entrada do Sistema Discover.

O conteúdo final desse arquivo é mostrado na Figura 6.5 na página oposta:

6.3.2.3 Criação do arquivo `.data`

Para criar o arquivo `.data`, o filtro processa cada registro do arquivo de *log*, transformando esses registros em registros no formato padrão do Sistema Discover. O processamento de cada linha do arquivo de *log* envolve os seguintes passos:

1. tenta casar a linha do registro atual do *log* com o padrão que foi configurado no filtro. Esse padrão é, na verdade, uma expressão regular que casa com um registro do *log* gerado pelo servidor Apache (configuração padrão).

```
| Class Attribute
null.

| Attributes
c_ip:              nominal.
c_dns:             string.
c_userid:          nominal.
date:              date.
time:              time.
cs_method:         nominal.
cs_uri:            string.
cs_uri_stem:       nominal.
cs_uri_query:      string.
cs_version:        nominal.
sc_status:         nominal.
sc_bytes:          integer.
csh_referer:       string.
csh_referer_stem:  nominal.
csh_referer_query: string.
csh_user_agent:    string.
csh_cookie:        string.
x_session:         nominal.
x_uri_type:        nominal.
x_uri_subtype:     nominal.
x_referer_type:    nominal.
x_referer_subtype: nominal.
```

Figura 6.5: Conteúdo do arquivo `.names` gerado pelo filtro.

2. uma vez que o registro casou com a expressão regular, pode-se saber a data em que esse registro foi criado, filtrando-o de acordo com as datas iniciais e finais especificadas nos parâmetros `-id` e `-fd` de linha de comando.

3. além da data, os outros campos que formam um registro do *log* são separados e transformados para o formato padrão, utilizando-se funções apropriadas. Essas funções são específicas para cada tipo de campo e agem como mini-filtros, lidando, também, com valores desconhecidos. Por exemplo, para um campo do tipo `nominal` é chamada a função que sabe como lidar com valores nominais.

4. finalmente, depois de processados todos os campos do registro atual, o resultado é escrito no arquivo de saída.

6.3.3 Exemplos de Entradas e Saídas

Nesta seção é mostrada a ação do filtro sobre alguns registros de *log*, na criação do arquivo `.data`. Para isso, foi criado um arquivo de *log* de nome `log.test` com o seguinte conteúdo:

1. `143.107.183.226 - - [02/Jul/2002:15:47:20 -0300] "GET /manuals/sce183/proc.html HTTP/1.1" 200 1329 "http://www.icmc.sc.usp.br/manuals/sce183/conteudo.html" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"`

2. `143.107.183.67 - - [02/Jul/2002:15:47:10 -0300] "GET /imagens/pessoas1.gif HTTP/1.1" 304 - "http://www.icmc.sc.usp.br/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)"`

3. `200.131.15.49 - - [02/Jul/2002:19:46:21 -0300] "GET /~poscomp/ HTTP/1.1" 200 4410 "http://mail.passosuemg.br/webmail/src/read_body.php?mailbox=INBOX&passed_id=9&startMessage=1&show_more=0" "Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)"`

4. `143.107.231.61 - - [02/Jul/2002:17:02:57 -0300] "GET /intranet/serv/altser.php?rowid=715 HTTP/1.1" 200 7545 "http://www.icmc.sc.usp.br/intranet/serv/pesqser.php" "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98; Win 9x 4.90)"`

A seguir, o filtro foi executado com a seguinte linha de comando:

```
./log2discover.pl -if log.test
```

criando, assim, os arquivos `log.names` e `log.data`. O arquivo `log.names` é o mostrado na Figura 6.5 na página 99 e o arquivo `log.data` gerado contém os seguintes quatro registros correspondentes:

1. `143.107.183.226, ?, ?, 2002/07/02, 15:47:20, GET, "/manuals/sce183/proc.html", /manuals/sce183/proc.html, ?, HTTP/1.1, 200, 1329, "http://www.icmc.sc.usp.br/manuals/sce183/conteudo.html", http://www.icmc.sc.usp.br/manuals/sce183/conteudo.html, ?, "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)", ?, ?, ?, ?, ?, ?`

2. `143.107.183.67, ?, ?, 2002/07/02, 15:47:10, GET, "/imagens/pessoas1.gif", /imagens/pessoas1.gif, ?, HTTP/1.1, 304, ?, "http://www.icmc.sc.usp.br/", http://www.icmc.sc.usp.br/, ?, "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)", ?, ?, ?, ?, ?, ?`

3. `200.131.15.49, ?, ?, 2002/07/02, 19:46:21, GET, "/~poscomp/", /~poscomp/, ?, HTTP/1.1, 200, 4410, "http://mail.passosuemg.br/webmail/src/read_body.php?mailbox=INBOX&passed_id=9&startMessage=1&show_more=0", http://mail.passosuemg.br/webmail/src/read_body.php, "mailbox=INBOX&passed_id=9&startMessage=1&show_more=0", "Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)", ?, ?, ?, ?, ?, ?`

4. `143.107.231.61, ?, ?, 2002/07/02, 17:02:57, GET, "/intranet/serv/altser.php?rowid=715", /intranet/serv/altser.php, "rowid=715", HTTP/1.1, 200, 7545, "http://www.icmc.sc.usp.br/intranet/serv/pesqser.php", http://www.icmc.sc.usp.br/intranet/serv/pesqser.php, ?, "Mozilla/4.0 (compatible; MSIE 6.0; Windows 98; Win 9x 4.90)", ?, ?, ?, ?, ?, ?`

6.4 Considerações Finais

Neste capítulo foi descrito sucintamente o Sistema Discover, mostrando a idealização do projeto e a motivação de se criar uma sintaxe padrão para os arquivos de entrada dos algoritmos de Aprendizado de Máquina utilizados no processo de KDD.

Foi também descrita a implementação de um filtro que transforma *logs* de servidores *Web* em arquivos de dados na sintaxe padrão do Sistema Discover, o que permitirá gerenciar a grande quantidade de dados que um arquivo de *log* armazena. Após realizar a transformação, é possível iniciar o processo de extração de conhecimento desses *logs*

utilizando as facilidades implementadas e a serem implementadas futuramente no Sistema Discover.

Atualmente, no Sistema Discover, já se encontra implementado o *Discover Learning Environment* (DLE) (Batista and Monard 2003a), (Batista and Monard 2003b) que permite algumas facilidades, tais como: manipulação de atributos e dados, filtro de exemplos, estatísticas descritivas, métodos de *resampling*, entre outros. Essas funcionalidades permitem que se faça um pré-processamento do arquivo de *log* como, por exemplo, a retirada de registros de requisições à arquivos auxiliares (figuras, entre outros) e, por meio das facilidades de manipulação de atributos e dados, o cálculo das sessões dos usuários.

Capítulo 7

Experiências com a Ferramenta Proposta

7.1 Considerações Iniciais

Uma vez que existe a possibilidade de se transformar um arquivo de *log* no formato padrão do Sistema DISCOVER, podem-se utilizar as facilidades já implementadas nesse sistema para que experiências envolvendo Aprendizado de Máquina sejam realizadas, com o objetivo de tentar extrair conhecimento útil utilizando as informações fornecidas nos arquivos de *logs*.

Utilizando o filtro criado, o arquivo de *log* do ICMC, citado no Capítulo 5, foi transformado em um arquivo `.data` e um arquivo `.names`, contendo a descrição dos atributos do arquivo `.data` criado. Neste capítulo, são descritas algumas das experiências realizadas com algoritmos de Aprendizado de Máquina que induzem conceitos proposicionais e relacionais, com o intuito de tentar extrair algum conhecimento do arquivo de *log*.

7.2 Aprendizado de Máquina Indutivo

O interesse atual em Aprendizado de Máquina (AM) mostra-se grande porque, embora não haja uma definição universal da natureza de inteligência, há um consenso de que a capacidade de aprender é essencial para um comportamento inteligente (Russell and Norvig 2003), (Rezende 2003). Assim, AM se justifica do ponto de vista científico enquanto método para estudar a cognição humana. Além disso, ferramentas computacionais de aquisição de conhecimento têm utilizado com sucesso técnicas de AM, como por exemplo na construção de sistemas especialistas.

O objetivo de AM é desenvolver métodos, técnicas e ferramentas para construir *máquinas inteligentes*, que se modificam para realizar cada vez melhor sua(s) tarefa(s) (Monard and Baranauskas 2003a). Um dos paradigmas de AM é o aprendizado indutivo[1]. A indução pode ser vista como um raciocínio do específico para o geral. Um problema de AM pode ser formulado como uma tarefa de aprendizado de conceitos a partir de exemplos, conhecido como *aprendizado indutivo de conceitos*, no qual regras que descrevem um determinado conceito são induzidas a partir de exemplos positivos e/ou negativos daquele conceito.

7.2.1 Linguagens de Descrição

Para a expressão de qualquer paradigma de aprendizado são necessárias linguagens que descrevam objetos (exemplos) assim como linguagens que descrevam os conceitos aprendidos. Vários formalismos lógicos têm sido utilizados em sistemas de aprendizado indutivo para a representação de exemplos e conceitos. Em geral, distinguem-se dois tipos de descrição: *descrição baseada em atributos* e *descrição relacional*.

Em uma descrição baseada em atributos, objetos são descritos em termos de atributos e valores desses atributos, conforme pode ser visto na Tabela 7.1. Alguns dos algoritmos/sistemas de aprendizado indutivo que têm sido utilizados com relativo sucesso em um número razoável de aplicações usam linguagens baseadas em atributos para a representação de exemplos e conceitos. Essas linguagens conseguem descrever objetos e conceitos por meio de proposições (Monard and Baranauskas 2003b).

a_1	a_2	...	a_m	*classe*
$x_{1,1}$	$x_{1,2}$	...	$x_{1,m}$	y_1
$x_{2,1}$	$x_{2,2}$	...	$x_{2,m}$	y_2
...	...	...	...	...
$x_{n,1}$	$x_{n,2}$	...	$x_{n,m}$	y_n

Tabela 7.1: Formato atributo-valor para dados

Porém, apesar do relativo sucesso dos sistemas de aprendizado proposicionais, tais sistemas são fortemente limitados, em função de alguns problemas inerentes à linguagem usada na descrição de exemplos e conceitos, tais como:

- Representação restrita: não conseguem representar problemas de conhecimento relacional, ou que envolvem relacionamentos estruturais arbitrariamente complexos;
- Não fazem uso de conhecimento prévio sobre o domínio: se restringem a utilizar informações sobre os atributos do objeto sendo estudado, quando muito combinando-os para encontrar novos atributos mais descritivos;

[1] Outros paradigmas seriam, por exemplo, aprendizado genético, conexionista, entre outros.

No caso de uma descrição relacional (também chamada de descrição estrutural), um objeto é descrito em termos de seus componentes e de relações entre eles e outros objetos.

Linguagens baseadas em lógica de primeira ordem são usadas há mais de 30 anos (Sammut 1993). No entanto, aprendizado empírico de primeira ordem só começou a atrair atenção maciça no início dos anos 90. A adoção, por sistemas de aprendizado, de linguagens lógicas de primeira ordem como linguagens de representação, permite que relações ou predicados possam ser induzidos. Isto faz com que o espaço dos conceitos passíveis de serem aprendidos seja aumentado.

7.2.2 Programação Lógica Indutiva

A Programação Lógica Indutiva – PLI – é uma abordagem relativamente recente de aprendizado indutivo de máquina, que busca contornar algumas das limitações das abordagens que utilizam a descrição de objetos baseada em atributos por meio do uso de teoria do domínio (ou conhecimento de fundo) e uma linguagem de descrição de conceitos baseada em lógica de primeira ordem. Ela foi inicialmente definida como a intersecção de aprendizado de máquina e programação lógica (Muggleton 1991). Embora a adoção dessa linguagem viabiliza o aprendizado de um conjunto muito maior de conceitos, traz também uma série de problemas relacionados à busca no espaço de hipóteses, uma vez que este é muito maior que no caso proposicional (Lavrač and Džeroski 1994), (Caulkins 2000).

O conhecimento de fundo utilizado por um sistema de PLI pode ser expresso de forma intensional ou extensional. Uma teoria extensional é representada apenas por fatos, enquanto que uma teoria intensional contém tanto fatos quanto cláusulas com variáveis, permitindo reduzir a descrição do conceito. A forma extensional pode ser indesejável por várias razões, incluindo o fato de que o conceito pode ter um número extremamente grande de exemplos. Portanto, é preferível descrever conceitos intencionalmente proporcionando uma forma compacta de declaração.

7.2.3 Técnicas Básicas de PLI

De uma forma geral, PLI pode ser descrita a partir de uma teoria de conhecimento de fundo inicial $\mathcal{K}$ e algum conjunto de exemplos $E = E^+ \cup E^-$, onde E^+ representa os exemplos positivos e E^- os exemplos negativos. O objetivo consiste em induzir uma hipótese $\mathcal{H}$ que junto com $\mathcal{K}$ *explica* os exemplos E.

Sistemas de PLI podem ser agrupados em duas famílias. A primeira utiliza uma estratégia de revisão sucessiva e a outra faz uso da estratégia "dividir-e-conquistar".

Nas técnicas de revisão sucessiva, a cada exemplo tratado erroneamente, faz-se um exame para descobrir o defeito, talvez com a ajuda de um especialista e tenta-se consertá-lo. Já nas técnicas que fazem uso da estratégia "dividir-e-conquistar", todos os exemplos de treinamento são consideradas em conjunto (aprendizado não-incremental) e, a cada iteração, uma cláusula da teoria que cobre alguns exemplos positivos e nenhum (ou pouco) exemplo negativo é encontrada. Os exemplos cobertos pela cláusula são descartados e o processo é repetido até que todos os exemplos positivos são cobertos por pelo menos uma cláusula.

Levando-se em consideração a busca realizada para encontrar cláusulas da teoria, esta família que faz uso da estratégia "dividir-e-conquistar" pode ser sub-dividida ainda em "*top-down*" e "*bottom-up*". Sistemas "*top-down*", começam com a cabeça da cláusula mais geral contendo apenas o literal da relação meta e adicionam literais ao seu corpo até que todos os exemplos negativos, ou a sua maioria, sejam excluídos. Sistemas "*bottom-up*" formam a generalização de um sub-conjunto pequeno dos exemplos positivos, e em seguida a generalizam mais, descartando literais enquanto a cláusula não cobre algum exemplo negativo, ou cobre muito poucos exemplos negativos. Ambos os métodos têm-se mostrado muito mais rápido que os sistemas baseados em revisão sucessiva, obtendo também resultados bastante expressivos.

7.3 O *Data Webhouse*

Conforme citado na Seção 4.6 na página 38 do Capítulo 4, o pré-processamento da sequência de cliques, envolve algumas etapas, tais como:

- Filtrar registros não necessários
- Identificar sessões
- Identificar usuários
- Identificar *hosts*
- Colocar os dados em um formato único

Como explicado na Seção 5 na página 43, neste trabalho é utilizado o arquivo de *log* padrão, no qual se encontram as informações fornecidas pelo servidor *Web* sem a necessidade de instrumentar o *site*. Nesse caso, não existe uma forma eficaz de se identificar o usuário, mas sim uma sessão. As outras etapas foram implementadas conforme pode-se observar na Figura 7.1. Primeiramente o arquivo de *log* foi transformado para o formato padrão

do Sistema DISCOVER, ao mesmo tempo em que os *hosts* foram identificados. O filtro `log2discover` possibilita, também, a filtragem por data. Uma vez que o *log* encontra-se no formato padrão do Sistema DISCOVER, é possível utilizar as facilidades implementadas no ambiente computacional *Discover Learning Environment* (DLE) (Batista and Monard 2003a), (Batista and Monard 2003b) do DISCOVER para criar filtros que transformam o *log* que está na sintaxe padrão para o formato do algoritmo de Aprendizado de Máquina escolhido para a experiência. Nesses filtros foi implementado o identificador de sessão do usuário e, por meio do DLE podem ser filtrados os registros não relevantes para as experiências.

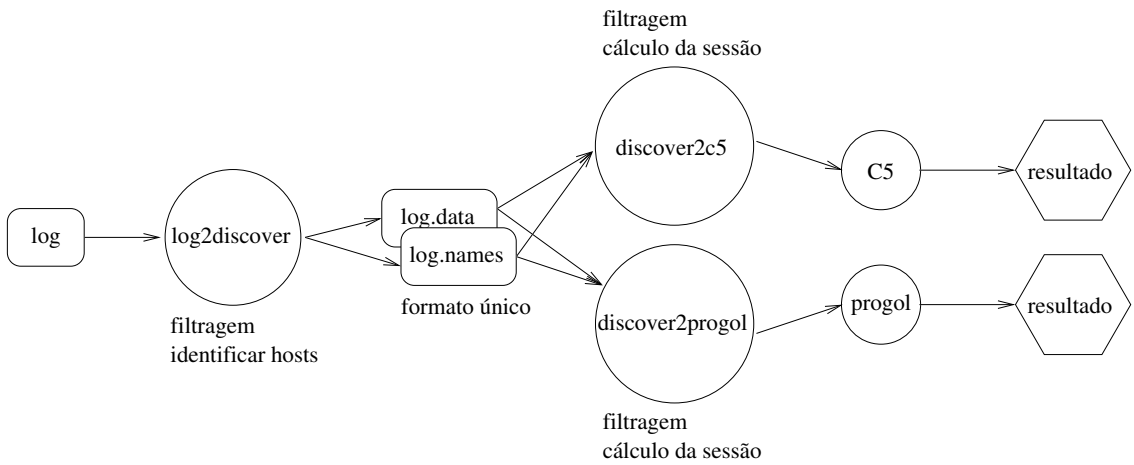


Figura 7.1: O processo proposto para se analisar os arquivos de *log*

Apesar das etapas básicas terem sido implementadas, outras etapas podem ser requeridas, dependendo do tipo de análise que se quer realizar. No entanto, uma vez que se tem o arquivo de *log* no formato da sintaxe padrão do DISCOVER, a manipulação dos dados é facilitada pelo uso do DLE.

7.4 Experiências com Indução de Conceito Proposicional

O primeiro tipo de experiência realizada consiste na aplicação de algoritmos de Aprendizado de Máquina proposicionais. Foi escolhido o algoritmo C5.0[2] (Quinlan 1987). O C5.0 é um indutor de árvores e de regras de decisão. Para os experimentos foi escolhida a indução de regras. Foram propostos e realizados três experimentos, descritos a seguir:

1. Nessa experiência, as sessões foram categorizadas como tendo alto, médio ou baixo número de páginas acessadas. A intenção foi a de descobrir regras que caracterizem

[2]Especificamente, a versão implementada em See5.

o usuário de acordo com o número de páginas acessadas por ele. As sessões que continham menos de 15 acessos foram consideradas de baixo acesso, enquanto que as que tinham mais de 40 acessos foram consideradas de alto acesso. As outras são consideradas de médio acesso.

2. Essa experiência é similar à anterior, com a diferença que se rotulou as sessões como tendo ou não alto número de páginas acessadas. Na construção do conjunto de dados não foram colocadas sessões com número de acessos médio. Assim, obteve-se um conjunto de dados com um maior contraste entre sessões.

3. Nessa experiência, tentou-se achar regras que determinem, a partir das URLs sendo requisitadas, se o usuário é um robô de busca ou não. Os robôs considerados foram os dos *sites* de busca do Altavista e do Google, pois foram os que apareceram nos resultados apresentados na Seção 5.3.2 na página 64 do Capítulo 5.

Os experimentos que utilizam indução de conceito proposicional não requerem muitas modificações no formato do arquivo de *log*, pois esse já se encontra descrito na linguagem de atributo-valor utilizado por esse tipo de algoritmo de Aprendizado de Máquina—Tabela 7.1 na página 104. O Sistema Discover já possui um filtro que transforma os dados que estão no formato da sintaxe padrão para o formato do C5.0. Para identificar as sessões do arquivo de *log* foi implementado um filtro específico.

7.4.1 Descrição dos Conjuntos de Dados

Como a quantidade de registros existentes no arquivo de *log* do ICMC, capturado do dia 02 de julho de 2002 a 04 de setembro de 2002 é muito grande, escolheu-se utilizar os registros de um dia de captura. Esse dia foi 05 de agosto de 2002 e o motivo dessa escolha é que ele representa a primeira segunda-feira após as férias de julho. Para o conjunto de dados de teste, utilizou-se os registros do dia seguinte *i.e.*, do dia 06 de agosto de 2002. Na Tabela 7.2 é mostrado um resumos dos dados utilizados nos experimentos.

Os experimentos foram realizados utilizando um subconjunto dos atributos do arquivo de *log*. Incluíram-se os atributos mais relevantes para essas experiências. Na Tabela 7.3 são mostrados os atributos considerados em cada experimento. É importante observar que as sessões não foram consideradas, pois há uma correspondência muito direta entre ela e a classe atribuída aos exemplos. Ou seja, o atributo que especifica a sessão é considerado o mais relevante pelo algoritmo, o qual induz regras do tipo

if sessão = 3798 *then* classe = alta.

[3] N/A – Não Aplicável

Conjunto de treinamento (dia 05 de agosto de 2002)

Experiência	Número de Sessões	Classes	Sessões por Classe	Exemplos por Classe	%	Erro na Classe Majoritária	Total de Exemplos
1	30	alto	10	1358	73.1	26.9	1858
		médio	10	360	19.4		
		baixo	10	140	7.5		
2	94	alto	34	2570	76.1	23.9	3377
		não_alto	60	807	23.9		
3	140	robô	70	1539	29.4	29.4	5228
		não_robô	70	3689	70.6		

Conjunto de teste (dia 06 de agosto de 2002)

Experiência	Número de Sessões	Classes	Sessões por Classe	Exemplos por Classe	%	Erro na Classe Majoritária	Total de Exemplos
1	30	alto	10	2285	81.5	N/A[3]	2802
		médio	10	377	13.4	N/A	
		baixo	10	140	5.0	N/A	
2	114	alto	54	4688	85.2	N/A	5500
		não_alto	60	812	14.8	N/A	
3	140	robô	70	1858	26.1	N/A	7136
		não_robô	70	5278	73.9	N/A	

Tabela 7.2: Características dos Conjunto de Dados utilizados com C5.0

que cobrem perfeitamente os exemplos de treinamento mas não conseguem cobrir os exemplos de teste.

Experiência	`session`	`domain`	`url`	`referer`	`agent`
1		•	•	•	•
2		•	•	•	•
3			•	•	

Tabela 7.3: Atributos utilizados nas experiências com C5.0

7.4.2 Resultados e Análises

O C5.0 foi primeiramente executado utilizando o conjunto de treinamento com 10k *cross validation*[4] para encontrar uma estimativa do erro. Após isso, todo o conjunto de treinamento foi utilizado para induzir as regras e o erro aparente, *i.e* o erro utilizando o mesmo conjunto de dados, foi medido. Finalmente, o erro do classificador foi medido utilizando-se o conjunto de teste. Na Tabela 7.4 são mostrados os resultados obtidos.

Observando-se os resultados, podem ser notados os seguintes pontos:

- Na experiência 1, o número de regras geradas, quando se executa *cross validation* para estimar o erro, está em torno de 18 com um desvio padrão pequeno. Entretanto,

[4]Utilizando 10k *cross validation* um classificador é induzido em cada passo utilizando 90% dos exemplos do conjunto de treinamento e os 10% restantes são usados como exemplos de teste.

Experiência	10k *Cross Validation*		Classificador Induzido		Teste
	Nº de Regras	Erro (%)	Nº de Regras	Erro Aparente (%)	Erro (%)
1	18.5 ± 0.5	10.0 ± 2.7	48	5.6	21.3
2	51.0 ± 20.7	14.9 ± 1.5	40	14.7	66.2
3	-	-	0	29.4	-

Tabela 7.4: Resultados obtidos com C5.0

quando se utiliza todo o conjunto de treinamento para induzir o classificador final, o número de regras cresce para 48. Isso significa que para cobrir 10% a mais de exemplos, o número de regras necessárias incrementa em mais de 70%, o que mostra uma irregularidade muito grande.

- Ainda na experiência 1, pode ser observado que o erro obtido utilizando-se 10k *cross validation* (10.0 ± 2.7) não é coerente com o erro obtido utilizando-se o conjunto de teste (21.3) o qual é duas vezes maior, confirmando a irregularidade já observada anteriormente.

- Na experiência 2, pode ser observado que o desvio padrão do número de regras induzidas utilizando-se 10k *cross validation* é muito alto (± 20.7). Pode, também, ser observado que o erro do classificador (14.9 ± 1.5) não tem relação com o erro obtido utilizando-se somente o conjunto de teste (66.2). Na realidade, esse último erro é muito maior que o erro de 23.9 da classe majoritária — Tabela 7.2 na página precedente. Ou seja, essa experiência também mostra uma grande irregularidade.

- Na experiência 3, o indutor não conseguiu criar nenhuma regra nova, mas somente a regra *default* que cobre os exemplos da classe majoritária `não_robô`.

Na realidade, este tipo de resultado já era esperado pois os algoritmos de Aprendizado de Máquina que induzem conceitos proposicionais "enxergam" cada um dos exemplos isoladamente. Mas os exemplos provenientes de arquivos de *log* não são constituídos de exemplos isolados, e sim de exemplos que pertencem a sessões do usuário.

7.5 Experiências com Indução Relacional

O outro tipo de experiência realizado consiste na aplicação de algoritmos de Aprendizado de Máquina relacional. Foi escolhido o algoritmo de PLI Progol 4.5 (Muggleton 1995), por este permitir que os conceitos sejam descritos de forma intencional. As experiências realizadas foram semelhantes às realizadas com o C5.0. Contudo, como o Progol também aprende utilizando apenas exemplos positivos (Muggleton 1996), mais três experiências

nesse sentido foram realizadas. Essas experiências são uma variação da segunda experiência, mas utilizando apenas exemplos positivos.

O Sistema DISCOVER não possui, ainda, um filtro capaz de transformar dados que estejam no formato da sintaxe padrão para arquivos de entrada do Progol. Dessa forma, um filtro que transforma os dados na sintaxe padrão para o formato do Progol foi desenvolvido.

De forma a ter-se uma noção da descrição relacional utilizada pelo Progol, na Figura 7.2 na página seguinte é mostrada uma parte do arquivo utilizado como entrada para o Progol. As seis primeiras cláusulas apresentadas, nomeadas como '`r`', são sessões do arquivo de *log* transformado para fatos do Progol. O primeiro parâmetro de uma cláusula '`r`' é o identificador de sessão e é por meio dele que se dá o relacionamento entre os fatos '`r`'. Em seguida, observa-se duas cláusulas chamadas '`visits`'. A primeira delas é um exemplo positivo, ou seja, diz que a sessão com '`id`' 3219 é uma sessão com alto número de acessos, enquanto que a segunda é um exemplo negativo, ou seja, a sessão de '`id`' 51 *não* é uma sessão com alto número de acessos. A última parte da figura apresenta o conhecimento de fundo que será utilizado, criado por meio de regras. Assim, pode-se ter uma regra do tipo `domain(S, X) :- r(S, _, X, _, _, _, _, _, _, _).` que diz que o domínio de uma sessão `S` é `X`, caso exista algum fato '`r`' cuja sessão, indicada pelo primeiro parâmetro, é `S` e o domínio, indicado pelo terceiro parâmetro, é `X`. Dessa forma, o Progol irá relacionar todos os fatos de uma mesma sessão `S`. Uma outra regra ilustrada na Figura 7.2 é a regra para `link`, que diz que existe um *link* de uma página `U1` para uma página `U2` caso haja um registro no *log* dizendo que um usuário acessou uma página `U2` a partir da página `U1` (`link(S, U1, U2) :- r(S, _, _, _, _, _, U2, _, _, U1, _)`). É importante observar que essas regras foram dadas de forma intencional, ou seja, não houve necessidade de enumerar os casos de domínios nem os de *links*.

7.5.1 Descrição dos Conjuntos de Dados

Os conjuntos de dados utilizados nas experiências com o Progol são os mesmos utilizados nas experiências com o C5.0. No entanto, a forma de visualizá-los é diferente pelo fato de agora os exemplos não serem as requisições propriamente ditas, mas sim as sessões. Na Tabela 7.5 na página 113 são mostradas as informações sobre esses conjuntos de dados.

Deve ser observado que o conjunto de treinamento e teste utilizados nas experiências 1, 2 e 3 são os mesmos utilizados em C5.0 —Tabela 7.2 na página 109. Entretanto, Progol enxerga os exemplos de forma diferente. Por exemplo, na experiência 1, tem-se 30 sessões, cada uma delas com 10 exemplos positivos. Os 20 exemplos negativos referem-se a exemplos negativos dessas classes, retirados das outras duas classes. Em outras palavras, para os 10 exemplos positivos da classe "alto", existem 10 exemplos negativos da classe "médio" e 10

```
r('3219', '143.107.183.53', '?', '2002-8-5', '18:53:26', 'GET',
    '/', '200', '20307', '?',
    'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)').
r('3219', '143.107.183.53', '?', '2002-8-5', '18:53:32', 'GET',
    '/pessoas/index.html', '200', '30821', '/',
    'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)').
r('3219', '143.107.183.53', '?', '2002-8-5', '18:53:37', 'GET',
    '/pessoas/profsComp.html', '200', '18787', '?',
    'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)').

r('51', '200.157.148.254', '?', '2002-8-5', '0:24:51', 'GET',
    '/manuals/HTML/compframes.html', '200', '4720',
    'http://www.google.com.br/search',
    'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)').
r('51', '200.157.148.254', '?', '2002-8-5', '0:25:1', 'GET',
    '/manuals/HTML/trabframes.html', '200', '3206',
    '/manuals/HTML/compframes.html',
    'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)').
r('51', '200.157.148.254', '?', '2002-8-5', '0:25:9', 'GET',
    '/manuals/HTML/tabelas.html', '200', '3348',
    '/manuals/HTML/trabframes.html',
    'Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)').

%%% Positive and negative examples

visits('3219', 'high').
:- visits('51', 'high').

%%% Background Information

ip(S, X)      :- r(S, X, _, _, _, _, _, _, _, _, _).
domain(S, X)  :- r(S, _, X, _, _, _, _, _, _, _, _).
date(S, X)    :- r(S, _, _, X, _, _, _, _, _, _, _).
time(S, X)    :- r(S, _, _, _, X, _, _, _, _, _, _).
method(S, X)  :- r(S, _, _, _, _, X, _, _, _, _, _).
url(S, X)     :- r(S, _, _, _, _, _, X, _, _, _, _).
status(S, X)  :- r(S, _, _, _, _, _, _, X, _, _, _).
bytes(S, X)   :- r(S, _, _, _, _, _, _, _, X, _, _).
referer(S, X) :- r(S, _, _, _, _, _, _, _, _, X, _).
agent(S, X)   :- r(S, _, _, _, _, _, _, _, _, _, X).

link(S, U1, U2) :- r(S, _, _, _, _, _, U2, _, _, U1, _), U1 \= '?'.
```

Figura 7.2: Exemplo de arquivo de entrada para o Progol – Experiência 2

Conjunto de treinamento (dia 05 de agosto de 2002)

Experiência	Número de Sessões	Classes	Exemplos		Erro na Classe Majoritária
			Positivos (%)	Negativos (%)	
1	30	alto	10 (11.1%)	20 (22.2%)	66.7%
		médio	10 (11.1%)	20 (22.2%)	
		baixo	10 (11.1%)	20 (22.2%)	
2	94	alto	34 (36.2%)	60 (63.8%)	36.2%
3	140	robô	70 (50.0%)	70 (50.0%)	50.0%
4	80	alto	40 (50.0%)	0	N/A
		baixo	40 (50.0%)	0	N/A
5	80	baixo	80 (100.0%)	0	N/A
6	80	baixo	80 (100.0%)	0	N/A

Conjunto de teste (dia 06 de agosto de 2002)

Experiência	Número de Sessões	Classes	Exemplos		Erro na Classe Majoritária
			Positivos (%)	Negativos (%)	
1	30	alto	10 (11.1%)	20 (22.2%)	N/A
		médio	10 (11.1%)	20 (22.2%)	N/A
		baixo	10 (11.1%)	20 (22.2%)	N/A
2	114	alto	54 (47.4%)	60 (52.6%)	N/A
3	140	robô	70 (50.0%)	70 (50.0%)	N/A
4	80	alto	40 (50.0%)	0	N/A
		baixo	40 (50.0%)	0	N/A
5	80	baixo	80 (100.0%)	0	N/A
6	80	baixo	80 (100.0%)	0	N/A

Tabela 7.5: Características dos Conjunto de Dados utilizados nas experiências com Progol

exemplos negativos da classe "baixo". Procedimento análogo é utilizado para declarar os exemplos negativos das classes "médio" e "baixo". Na experiência 2, das 94 sessões, 34 são exemplos positivos da classe "alto" e 60 são exemplos negativos dessa classe. Finalmente, na experiência 3, das 140 sessões, 70 são exemplos positivos da classe "robô" e 70 são exemplos negativos dessa classe.

Já as experiências 4, 5 e 6 são aquelas nas quais são apresentados ao Progol somente exemplos positivos. É importante observar que é necessário indicar isso para o Progol utilizando a cláusula '`set(posonly)?`', não bastando apenas mostrar os exemplos para que ele verifique que só há positivos (Ferro 2002). Em decorrência disso, não é possível determinar o erro na classe majoritária, já que há apenas uma classe sendo apresentada.

A diferença entre as experiências 5 e 6 está no fato de utilizarem atributos diferentes. Na experiência 5 somente a URL é considerada no aprendizado, enquanto que na experiência 6, alguns outros atributos são utilizados. Na Tabela 7.6 são mostrados os atributos que foram utilizados em cada uma das experiências. Nota-se que existe mais um atributo chamado `link` que foi criado como conhecimento de fundo, utilizando uma descrição intencional.

Experiência	session	domain	url	referer	agent	link
1	•	•	•	•	•	•
2	•	•	•	•	•	•
3	•		•	•		•
4	•	•	•	•	•	•
5	•		•			
6	•	•	•	•	•	•

Tabela 7.6: Atributos utilizados nas experiências com Progol

7.5.2 Resultados e Análises

Na Tabela 7.7 são mostrados os resultados obtidos:

Experiência	Treinamento				Teste
	Nº de Cláusulas	Nº de Fatos	Nº de Regras	Erro (%)	Erro (%)
1	19	11	8	8.89	30.00
2	23	4	19	20.21	41.23
3	60	3	57	40.71	51.43
4	14	11	3	4.05	17.50
5	27	24	3	6.25	46.25
6	17	17	0	1.25	31.25

Tabela 7.7: Resultados obtidos com Progol

O tamanho das hipóteses (número total de cláusulas) considera tanto as regras induzidas quanto os fatos, os quais, simplesmente, descrevem os exemplos positivos que não são cobertos por essas regras. Um número grande de fatos na hipótese induzida indica que o Progol não conseguiu achar regras que cobrem esses exemplos.

Sem considerar, ainda, a qualidade individual das regras induzidas, é possível observar na Tabela 7.7 que o erro do classificador no conjunto de teste é somente aceitável na experiência 4.

A seguir, são comentadas algumas das regras induzidas nas experiências realizadas.

- Na experiência 1, duas das regras induzidas são:

```
[p=8, n=1] visits(A,medium) :- url(A,'/~poscomp/inscricoes.html').
[p=4, n=1] visits(A,low) :- url(A,'/~poscomp/inscr_cand.html').
```

  onde `[p=8, n=1]` indica que essa regra cobre 8 (oito) exemplos positivos e 1 (um) negativo, enquanto que a outra cobre 4 (quatro) positivos e 1 (um) negativo.

  A informação contida nessas duas regras já foi observada na Seção 5.3.1.3 na página 50. O que acontece é que existem duas páginas diferentes, com conteúdo semelhante, que

explicam como fazer a inscrição no PosComp. Ambas as páginas tem o endereço externo ao *site* do ICMC para realizar efetivamente a inscrição.

Essas duas regras indicam que a primeira dessas páginas é mais visitada que a segunda. Entretanto, se ambas as páginas forem substituídas por somente uma página, as sessões seriam, provavelmente, da classe que representa um alto número de acessos.

- Na experiência 2, tem-se um certo número de domínios desconhecidos, indicado pelo símbolo "?". Assim, uma das regras induzidas é:

```
[p=4, n=0] visits(A,high) :- domain(A,?), url(A,'/~andre').
```

que diz "uma sessão tem muitas páginas acessadas se ela vem de um domínio desconhecido e a página `/~andre` foi acessada.".

Entretanto, como o domínio é desconhecido, o relacionamento com essa página, na regra, é artificial. Deve ser lembrado que o domínio é desconhecido quando, na preparação do *Data Webhouse*, não é possível encontrar, através do serviço de DNS reverso, o domínio correspondente ao endereço IP armazenado no arquivo de *log*.

Uma outra regra induzida é

```
[p=3, n=0] visits(A,high) :- url(A,'/manuals/HTML/dialogos.html').
```

que diz "uma sessão tem muitas páginas acessadas se o usuário acessou, em algum momento, a página `/manuals/HTML/dialogos.html`.". Essa regra pode ser explicada pelo fato de que alguns usuários que estão acessando as páginas referentes ao manual de HTML navegam por muitas páginas dele, como observado na Seção 5.3.1.3 na página 50, em especial se ele tiver passado pela página `dialogos.html`.

- Na experiência 3, duas das três regras induzidas pelo Progol são:

```
[p=5, n=0] robot(A) :- url(A,'/~kaori/lunah/reply.php').
[p=5, n=0] robot(A) :- url(A,'/~kaori/lunah/template.php').
```

Assim, pode ser observado que as páginas referentes ao usuário `kaori` cobrem 10 exemplos do total de 140 exemplos utilizados no treinamento. Isso pode ser um indicativo de que essas páginas são frequentemente acessadas por robôs de busca e uma maior investigação deve ser feita para descobrir-se as razões desse fato.

- Na experiência 4, uma regra com suporte razoável é

  ```
  [p=24, n=7] visits(A,low) :- url(A,'/pessoas/index.html').
  ```

  indicando que pessoas que acessaram poucas páginas passaram pela página `/pessoas/index.html`. Isso pode ser um indício de que as pessoas que procuraram essa página sabiam o que estavam procurando no *site* do ICMC.

- Nas experiências 5 e 6, nas quais são utilizados somente exemplos positivos, não foram encontradas regras de interesse. Na realidade, a presença de exemplos negativos auxilia o Progol no processo de especialização.

7.6 Considerações Finais

Para extrair conhecimento de *logs* de servidores *Web*, os algoritmos de Aprendizado de Máquina devem ser capazes de encontrar um relacionamento entre os registros de uma mesma sessão. Dessa forma, há necessidade de se utilizar algoritmos de PLI, como Progol por exemplo, para descobrir essas relações, as quais não podem ser expressas por algoritmos que induzem conceitos proposicionais, como o C5.0.

Porém, deve ser observado que muito da arte do PLI está na seleção e formulação apropriada do conhecimento de fundo, já que conhecimento de fundo irrelevante pode piorar os resultados. Entretanto, selecionar esse conhecimento de fundo não é uma tarefa trivial pois, especialmente no caso de *Web Usage Mining*, há inúmeras formas de ligar as informações das diversas requisições realizadas pelo usuário.

Neste capítulo foram apresentadas algumas experiências com Progol, as quais serão incrementadas futuramente com conhecimento de fundo que inclua a estrutura de *links* do *site*. Isso pode ser realizado descrevendo-se intencionalmente um conhecimento de fundo que permita fazer a ligação entre requisições utilizando o fato de que algumas páginas requisitadas numa sessão encontram-se no campo `referer` de outras requisições da mesmas sessão.

Um outro aspecto a ser explorado é utilizar indução construtiva (Lee 1999) para construir outros atributos em função dos atributos encontrados no arquivo de *log* original.

Também deve ser observado que as experiências realizadas com Progol consomem muito tempo (às vezes dias) de processamento. Como Progol é um sistema geral de PLI, uma idéia a ser melhor explorada é o desenvolvimento de um sistema de PLI restrito ao domínio

de *logs* de servidores *Web*, com o intuito de diminuir drasticamente o tempo de processamento.

Capítulo 8

Conclusões

8.1 Considerações Iniciais

A área de *Web Mining* tem grande potencial pois a *Web* se mostra um domínio fértil para a utilização de diversas técnicas de Mineração de Dados. Essas técnicas podem incluir estatísticas diretas como, por exemplo, a frequência de acessos às páginas (Cooley, Mobasher, and Srivastava 1999). Técnicas de Mineração de Dados, como métodos estatísticos, *clustering*, entre outros (Tveit 2000) ou, ainda, outras formas de análises, como encontrar caminhos comuns de usuários, também podem ser utilizadas.

Este trabalho, que envolve a utilização de algumas dessas técnicas em *logs* de servidores *Web*, ou seja, *Web Usage Mining*, é o primeiro desse tipo a ser realizado pelo grupo de Inteligência Artificial do LABIC. Neste capítulo é feita uma conclusão geral do trabalho realizado.

8.2 Conclusão

Os estudos realizados demonstraram que as ferramentas *freeware* e *open source* existentes para análise de *logs* permitem que sejam levantadas algumas informações úteis sobre a utilização geral de um *site*. Essas informações podem ser utilizadas para identificação de páginas interessantes para os usuários, exploração da organização e do interrelacionamento das páginas dentro do *site*, verificação da eficácia de propagandas *on-line*, melhoria de resultados de busca por páginas (uma vez que é possível de se observar quais as palavras de busca que são utilizadas pelos usuários e que resultam numa visita ao *site*), entre outros.

Em geral, o uso combinado dessas ferramentas proporciona um melhor entendimento de

como o *site* está sendo utilizado pelos seus visitantes. Esse melhor entendimento da utilização do *site* é imprescindível para uma melhoria contínua da arquitetura dos *sites* que existem atualmente, bem como para projetar melhores *sites* no futuro.

Também deve ser ressaltada a importância da existência de *softwares open source* para análise de *logs*, uma vez que isso possibilita a alteração das ferramentas para a obtenção de outros resultados não necessariamente obtidos com as implementações originais. No entanto, essas ferramentas oferecem uma visão global da utilização do *site*, não permitindo que uma análise mais específica dos usuários que acessam o *site* seja feita. Com uma análise desse tipo, pode-se fazer o levantamento dos caminhos dos usuários no site, de forma a se identificar caminhos problemáticos e caminhos interessantes. Mas o real interesse desse tipo de análise é o da identificação do perfil do usuário de forma a se fazer, por exemplo, um melhor direcionamento de propaganda, ou então recomendação de páginas de dentro do *site* que sejam de provável interesse do visitante. Esse tipo de conhecimento a respeito do *site* pode ser obtido pela aplicação de algoritmos de Aprendizado de Máquina nos *logs*.

Neste trabalho foi implementada uma ferramenta que transforma os arquivos de *logs* de servidores *Web* em arquivos no formato da sintaxe padrão do Sistema Discover. Esse sistema permite contornar o problema de se ter diferentes formatos para arquivos de entrada dos diversos algoritmos de Aprendizado de Máquina. Assim, uma vez que os dados do arquivo de *log* estão no formato da sintaxe padrão do Sistema Discover, é possível utilizá-lo como entrada para esses algoritmos. Além disso, no Sistema Discover encontram-se implementadas algumas facilidades, tais como: manipulação de atributos e dados e filtro de exemplos. Essas funcionalidades permitem que se faça um pré-processamento do arquivo de *log* como, por exemplo, a retirada de registros de requisições à arquivos auxiliares (figuras, entre outros).

Utilizando a ferramenta implementada, realizaram-se alguns experimentos utilizando os dados do *log* do *site* do ICMC-USP como entrada para algoritmos de Aprendizado de Máquina que induzem conceitos proposicionais e relacionais. Os resultados obtidos com os algoritmos de indução de conceitos proposicionais não foram muito bons, devido ao fato de que, apesar dos *logs* de servidores *Web* terem o formato atributo-valor, existe um relacionamento entre os seus registros. Esse relacionamento é, justamente, as sessões dos usuários do *site*. O pior resultado foi o da experiência que tentava aprender se um determinado usuário é um robô de busca, somente observando as páginas requisitadas. Em (Kohavi 2001) é dito que esta tarefa de identificar robôs é um problema em aberto.

Os experimentos realizados com algoritmos de Aprendizado de Máquina que induzem conceitos relacionais também não foram muito satisfatórios, obtendo-se muitas regras não confiáveis. Esses resultados devem-se à necessidade de uma investigação mais profunda

sobre a utilização dos sistemas de PLI e na seleção e formulação apropriada do conhecimento de fundo. No entanto, ficou claro que para extrair conhecimento de *logs* de servidores *Web*, os algoritmos de Aprendizado de Máquina devem ser capazes de encontrar um relacionamento entre os registros de uma mesma sessão. Dessa forma, há necessidade de se utilizar algoritmos de PLI e essas experiências foram um primeiro contato com a utilização de algoritmos de Aprendizado de Máquina no domínio de *logs*. No entanto, deve ser observado que as experiências realizadas com Progol consumiram muito tempo de processamento, sendo esse um problema de escalabilidade dos sistemas de PLI para aplicações que envolvem *logs* de servidores *Web* (Tveit 2000).

Como conclusão geral, fica o fato de que explorar a área de *Web Usage Mining* requer um bom planejamento do que se deseja realizar e uma formulação clara dos objetivos a serem atingidos. Assim, pode-se tomar as providência necessárias como, por exemplo, uma possível instrumentação do *site* a ser analisado, a escolha cuidadosa das ferramentas e algoritmos a serem utilizados e, dependendo do caso, um bom preparo na criação do conhecimento de fundo ou de atributos que sejam relevantes para as análises.

8.3 Trabalhos Futuros

A partir deste trabalho inicial, podem-se identificar várias idéias sobre *Web Mining* que sugerem a realização de futuros trabalhos. Um desses trabalhos está relacionado com uma análise mais aprofundada de ferramentas *freeware* e *open source* existentes, tendo-se como resultado um *survey* desse tipo de ferramenta. Um outro tipo de experimento relacionado com essas ferramentas seria fazer a análise de algum *site* e, a partir dos resultados obtidos, propor modificações no mesmo e verificar o impacto dessas alterações.

Um outro trabalho futuro, relacionado com os experimentos realizados com os algoritmos de Aprendizado de Máquina é aprofundar os estudos em PLI e Progol. Uma das idéias é a de se implementar um sistema de PLI voltado apenas para o domínio de *logs*, de forma que o tempo de execução seja reduzido. Além disso, o uso de indução construtiva, isto é, a criação de outros atributos em função dos originalmente encontrados nos arquivos de *log*, merece uma maior investigação. Existe, também, a possibilidade de se aplicar outras técnicas de aprendizado (por exemplo, *clustering* e regras de associação), como descrito em trabalhos relacionados nessa área (Ling), (Fu, Sandhu, and Shih 1999b), entre outros.

Como forma de se ter uma melhor confiança na identificação das sessões e, possivelmente, a identificação dos usuários, existe a possibilidade de se instrumentar o *site*. Ferramentas como phpOpenTracker[1] ajudam na tarefa de rastrear os usuários através do *site* e em

[1] http://www.phpopentracker.de/

medir sua utilização. Também o servidor Apache[2] possui um módulo capaz de gravar de forma transparente e automática um identificador de usuário nos arquivos de *log*. Isso poderia ser utilizado para se identificar o usuário sem o auxílio de ferramentas extras e, de certa forma, garantiria a veracidade da informação.

Além da análise de *logs* de servidores *Web*, existem outros tipos de *logs* como os gerados por servidores *proxy* e por *firewalls*. Esses *logs* são passíveis de análises similares às efetuadas nos *logs* de servidores *Web*.

E, por último, existe a possibilidade de se averiguar as outras sub-áreas de *Web Mining*, como *Web Content Mining*. Um projeto interessante seria construir um *site* de busca que utilizasse técnicas de *Web Content Mining* para indexar o conteúdo dos *sites*.

8.4 Considerações Finais

Os *logs* de servidores *Web* têm alguns problemas, tais como: não identificar sessões automaticamente, não conter dados transacionais (de compras, por exemplo, os quais estão em um banco de dados separado), não registrar eventos críticos, como "adicionar ao cesto de compras", entre outros problemas. Apesar de ser possível inferir os eventos citados, essa inferência é uma tarefa árdua e dependente do *site* sendo analisado. Em (Talagala, Asami, and Patterson 1999), é mostrado uma aplicação de *Web Usage Mining* que utiliza esses tipos de eventos), entre outros problemas.

No entanto, quando se trata de *Web Usage Mining*, *logs* de servidores se apresentam como bons candidatos à aplicação de técnicas de Mineração de Dados pois apresentam uma grande quantidade de dados, muitos atributos (com projeto apropriado do *site*), dados relativamente limpos, domínio acionável, ou seja, é possível de se alterar a estrutura do *site* com relativa facilidade e observar os resultados dessas mudanças, entre outras características. Tecnicamente, o *e-commerce* é um excelente domínio para área de *Web Mining* e a área de *Web Mining* é de grande importância para o *e-commerce*, fazendo da *Web* um laboratório experimental.

[2] http://www.apache.org

Referências

Aumann, Y., O. Etzioni, R. Feldman, M. Perkwitz, and T. Shmiel (1998). Predicting event sequences: Data mining for prefetching web-pages. In *KDD'98*.

Baranauskas, J. A. (2001). Extração Automática de Conhecimento por Múltiplos Indutores. Tese de Doutorado, ICMC-USP, http://www.teses.usp.br/teses/disponiveis/55/55134/tde-08102001-112806.

Baranauskas, J. A. and G. E. A. P. A. Batista (2000). O projeto Discover: Idéias iniciais. (comunicação pessoal).

Baranauskas, J. A. and M. C. Monard (2000, Feb). Reviewing some Machine Learning Concepts and Methods. Technical Report 102, ICMC-USP, São Carlos, SP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_102.ps.zip.

Baranauskas, J. A. and M. C. Monard (2003). Combining Symbolic Classifiers from Multiple Inducers. *Knowledge Based Systems 16*(3), 129–136. Elsevier Science.

Batista, G. E. A. P. A. (2001). Sintaxe padrão do arquivo de exemplos do projeto Discover. http://www.icmc.sc.usp.br/~gbatista/Discover/SintaxePadraoFinal.htm.

Batista, G. E. A. P. A. (2003a). A biblioteca da sintaxe padrão para arquivos de exemplos do Sistema Discover. Technical report, ICMC-USP. Trabalho em Andamento.

Batista, G. E. A. P. A. (2003b). Pré-processamento de dados em aprendizado de máquina supervisionado. Tese de Doutorado, ICMC-USP / a ser defendida.

Batista, G. E. A. P. A. and M. C. Monard (2003a). Descrição da Arquitetura e do Projeto do Ambiente Computacional Discover Learning Environment — DLE. Technical Report 187, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_187.pdf.

Batista, G. E. A. P. A. and M. C. Monard (2003b). The Discover Object Library (DOL) User's Manual. Technical report, ICMC-USP. (in press).

Beeferman, D. and A. Berger (1999). Agglomerative clustering of a search engine query log. In *Proceedings of 6th International Conference on Knowledge Discovery and*

Data Mining, pp. 407–416.

Bernardini, F. C. (2002). Combinação de Classificadores Simbólicos para Melhorar o Poder Preditivo e Descritivo de *Ensembles*. Dissertação de Mestrado, ICMC-USP.

Blum, A. L. and P. Langley (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 245–271.

Boyer, R. S. and J. S. Moore (1977). A fast string searching algorithm. *Communications of the ACM 20*(10), 762–772.

Caulkins, C. W. (2000, agosto). Aquisição de conhecimento utilizando aprendizado de máquina relacional. Dissertação de Mestrado, ICMC-USP.

Cooley, R., B. Mobasher, and J. Srivastava (1999). Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems 1*(1), 5–32.

Corporation, T. C. (1999). Introduction to data mining and knowledge discovery.

Craven, M., D. DiPasquo, D. Freitag, A. K. McCallum, T. M. Mitchell, K. Nigam, and S. Slattery (1998). Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, Madison, US, pp. 509–516. AAAI Press, Menlo Park, US.

Craven, M., S. Slattery, and K. Nigam (1998). First-order learning for web mining. In *Proceedings of the 10th European Conference on Machine Learning*, pp. 250–255.

Deogun, J. S., H. Sever, and V. V. Raghavan (1998). Structural abstractions of hypertext documents for web-based retrieval. In *Proceedings of the 9th International Workshop on Database and Expert Systems Applications DEXA*, pp. 385–390.

Dosualdo, D. G. (2002). Investigação de regressão para data mining. Monografia para o Exame de Qualificação de Mestrado, ICMC-USP.

Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth (1996). Knowledge discovery and data mining: Towards a unifying framework. In *Proceedings of the Second International Conference on Data Mining and Knowldege Discovery*, pp. 82–88. AAAI Press, Menlo Park, US.

Ferro, M. (2002, Outubro). Aquisição de conhecimento utilizando aprendizado de máquina relacional. Monografia para o Exame de Qualificação de Mestrado, ICMC-USP / a ser defendida.

Fu, Y., K. Sandhu, and M. Shih (1999a). Clustering of web users based on access patterns. In *Proceedings of the 1999 KDD Workshop on Web Mining*.

Fu, Y., K. Sandhu, and M. Shih (1999b). Clustering of web users based on access patterns.

Gale, W. A. (1986). *Artificial Intelligence and Statistics*. Addison-Wesley.

Geromini, M. R. (2002). Projeto e desenvolvimento de uma interface gráfica para o ambiente de descoberta de conhecimento DISCOVER. Monografia para o Exame de Qualificação de Mestrado, ICMC-USP.

Gomes, A. K. (2002). Análise do Conhecimento Extraído de Classificadores Simbólicos utilizando Medidas de Avaliação e Interessabilidade. Dissertação de Mestrado, ICMC-USP, http://www.teses.usp.br/teses/disponiveis/55/55134/tde-04072002-144610.

Group, N. W. (1999a). *RFC2616 - Hypertext Transfer Protocol – HTTP/1.1*.

Group, N. W. (1999b). *RFC2617 - HTTP Authentication: Basic and Digest Access Authentication*.

Haight, S. and J. Megarity (1998, August). Measuring web site usage: Log file analysis. Technical Report 57, National Library of Canada. http://www.nlc-bnc.ca/publications/1/p1-256-e.html.

Hallam-Baker, P. M. and B. Behlendorf (1996). Extended log file format. Technical report, W3C. http://www.w3c.org/pub/WWW/TR/WD-logfile.html.

Horst, P. S. and M. C. Monard (2000, novembro). Um sistema computacional para avaliação de regras induzidas por algoritmos de aprendizado de máquina. In *Proceedings of IBERAMIA-SBIA 2000 Open Discussion Track*, pp. 167–176.

Imamura, C. Y. (2001). Pré-processamento para Extração de Conhecimento de Bases Textuais. Dissertação de Mestrado, ICMC-USP.

John, G., R. Kohavi, and K. Pfleger (1994). Irrelevant features and the subset selection problem. In M. Kaufmann (Ed.), *Proceedings of the Eleventh International Conference on Machine Learning*, San Francisco, CA, pp. 167–173.

Kemp, A. H., G. E. A. P. A. Batista, and M. C. Monard (2001, maio). Descrição da implementação dos métodos estatísticos de resampling do ambiente DISCOVER. Technical Report 143, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_143.ps.zip.

Kemp, A. H., G. E. A. P. A. Batista, and M. C. Monard (2002, fevereiro). Descrição da implementação dos filtros para recuperação da taxa de erro dos algoritmos de aprendizado de máquina usado no ambiente DISCOVER. Technical Report 175, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_175.ps.zip.

Kimball, R. and R. Merz (2000). *The Data Webhouse Toolkit*. John Wiley and Sons, Inc.

Knuth, D. E., J. H. Morris, Jr., and V. R. Pratt (1977). Fast pattern matching in strings. *SIAM Journal on Computing 6*, 323–350.

Kohavi, R. (2001). Mining e-commerce data: the good, the bad, and the ugly. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 8–13. ACM Press.

Kosala, R. and H. Blockeel (2000). Web mining reseach: A survey. In *SIGKDD Explorations*, Volume 2, pp. 1–15.

Langley, P. (1996). *Elements of Machine Learning*. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Lavrač, N. and S. Džeroski (1994). *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood.

Lee, H. D. (1999, Abril). Seleção e Construção de Features Relavantes para o Aprendizado de Máquina. Monografia para o Exame de Qualificação de Mestrado, ICMC-USP.

Lee, H. D. (2000, abril). Seleção de features relevantes para aprendizado de máquina. Dissertação de Mestrado, ICMC-USP.

Lee, H. D. and M. C. Monard (2000). A practical approach for knowledge-driven constructive induction. In *Proceedings of Argentine Symposium on Artificial Intelligence, ASAI'2000, 29th International Conference SADIO*, pp. 71–86.

Ling, L. S. Mining the most interesting web access associations.

Loh, S., L. K. Wives, and J. P. M. Oliveira (2000, July). Concept-based knowledge discovery in texts extracted from the web. In *Proceedings of ACM SIGKDD Explorations*, Volume 2, pp. 29–39.

Martins, C. A. (2001). Interpretação de clusters em Aprendizado de Máquina. Monografia para o Exame de Qualificação de Doutorado, ICMC-USP.

Martins, C. A. and M. C. Monard (2000). Interpretação de clusters usando aprendizado de máquina simbólico. In *Proceedings of the 21st Iberian Latin American Congress on Computational Methods in Engineering, CILAMCE 2000*.

Melanda, E. (2002). Pós-processamento de conhecimento de regras de associação. Monografia para o Exame de Qualificação de Doutorado, ICMC-USP.

Mello, R., R. Chiara, and R. Villela (2002). *Aprendendo Java 2*. Novatec Editora.

Milaré, C. R. (2000, Março). Extração de Conhecimento de Redes Neurais. Monografia para o Exame de Qualificação de Doutorado, ICMC-USP.

Mitchell, T. M. (1997). *Machine Learning*. WCB McGraw-Hill.

Monard, M. C. and J. A. Baranauskas (2003a). *Conceitos sobre Aprendizado de Máquina* (1 ed.), Chapter 4, pp. 89–114. Volume 1 of *1* Rezende (2003).

Monard, M. C. and J. A. Baranauskas (2003b). *Indução de Regras e Árvores de Decisão* (1 ed.), Chapter 5, pp. 115–140. Volume 1 of *1* Rezende (2003).

Muggleton, S. (1995). Inverse Entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming 13*(3-4), 245–286.

Muggleton, S. (1996). Learning from positive data. In *Proceedings of the 6th International Workshop on Inductive Logic Programming*, Volume 1314, pp. 358–376.

Muggleton, S. H. (1991). Inductive Logic Programming. *New Generation Computing 8*, 295–318.

Murray, D. and K. Durrell (1999). Inferring demographic attributes of anonymous internet users. In *Proceedings of ACM Workshop on Web Usage Analysis and User Profiling (WEBKDD'99)*.

Paula, M. F. (2003). Ambiente para Disponibilização de Conhecimento. Dissertaçao de Mestrado, ICMC-USP / a ser defendida.

Pila, A. D. (2001, abril). Seleção de Atributos Relevantes para Aprendizado de Máquina utilizando a Abordagem de Rough Sets. Dissertação de Mestrado, ICMC-USP, http://www.teses.usp.br/teses/disponiveis/55/55134/tde-13022002-153921/publico/dissertacao_ADP.pdf.

Prati, R. C. (2003). Projeto e implementação do framework de integração do Sistema Discover. Dissertação de Mestrado, ICMC-USP / a ser defendida.

Prati, R. C., J. A. Baranauskas, and M. C. Monard (2001a, junho). Extração de informações padronizadas para a avaliação de regras induzidas por algoritmos de aprendizado de máquina simbólico. Technical Report 145, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_145.ps.zip.

Prati, R. C., J. A. Baranauskas, and M. C. Monard (2001b, março). Uma proposta de unificação da linguagem de representação de conceitos de algoritmos de aprendizado de máquina simbólicos. Technical Report 137, ICMC-USP. ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/RT_137.ps.zip.

Prati, R. C., J. A. Baranauskas, and M. C. Monard (2002, Setembro). Padronização da Sintaxe e Informações sobre Regras Induzidas a Partir de Algoritmos de Aprendizado de Máquina Simbólico. *Revista Eletrônica de Iniciação Científica 2*(3). http://www.sbc.org.br/reic/edicoes/2002e3.

Pugliesi, J. B. (2001, Março). O Pós-Processamento em Extração de Conhecimento de Bases de Dados. Monografia para o Exame de Qualificação de Mestrado, ICMC-USP.

Quinlan, J. R. (1987). Generating Production Rules from Decision Trees. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Italy, pp. 304–307.

Rezende, S. O. (2003). *Sistemas Inteligentes: Fundamentos e Aplicações*. Barueri, SP, Brasil: Editora Manole.

Rezende, S. O., J. B. Pugliesi, E. A. Melanda, and M. F. Paula (2003). *Mineração de Dados* (1 ed.), Chapter 12, pp. 307–336. Volume 1 of *1* Rezende (2003). ISBN 85-204-1683-7.

Russell, S. J. and P. Norvig (2003). *Artificial Intelligence: A Modern Approach* (2 ed.). Prentice Hall.

Sammut, C. A. (1993). The Origins of Inductive Logic Programming: A Prehistoric Tale. In *Proceedings Third International Workshop on Inductive Logic Programming*, Bled, Slovenia, pp. 127–147. Springer Verlag.

Slattery, S. and M. Craven (1998). Combining statistical and relational methods for learning in hypertext domains. In D. Page (Ed.), *Proceedings of ILP-98, 8th International Conference on Inductive Logic Programming*, Madison, US, pp. 38–52. Springer Verlag, Heidelberg, DE.

Slattery, S. and T. Mitchell (2000). Discovering test set regularities in relational domains. In *Proceedings of the 17th International Conf. on Machine Learning*, pp. 895–902. Morgan Kaufmann, San Francisco, CA.

Spiliopoulou, M., C. Pohle, and L. Faulstich (1999). Improving the effectiveness of a web site with web usage mining. In *Proceedings of the Workshop on Web Usage Analysis and User Profiling, WEBKDD '99*, pp. 51–56.

Talagala, N., S. Asami, and D. A. Patterson (1999). Usage patterns of a web-based image collection. In *IEEE Symposium on Mass Storage Systems*, pp. 203–214.

Tveit, A. (2000, June). Web mining with inductive logic programming. WWW.

Wall, L. and R. Schwartz (1991). *Programming PERL*. O´Reilly & Associates.

Weiss, S. M. and N. Indurkhya (1998). *Predictive Data Mining: A Practical Guide*. San Francisco, CA: Morgan Kaufmann Publishers.

Witten, I. H. and E. Frank (2000). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers.

Apêndice A

Clustering

A.1 Considerações Iniciais

O objetivo do *clustering* é encontrar classes que são diferentes umas das outras e cujos membros sejam similares entre si, segundo alguma medida de similaridade. Diferentemente dos modelos preditivos, não se sabe, a priori, quais serão essas classes, as quais não possuem um significado conceitual como no caso de aprendizado supervisionado.

A.2 Técnicas e Algoritmos

As diferentes técnicas de *clustering* são classificadas de acordo com os resultados obtidos pelos algoritmos. Na Figura A.1 são sumarizadas essas técnicas, descritas a seguir.

Técnicas de otimização: tentam forma uma K-partição ótima sobre os dados, isto é, divide o conjunto em K grupos mutuamente exclusivos onde K é dado pelo usuário. As técnicas de otimização são computacionalmente caras pois fazem uma busca exaustiva pela K-partição ótima. Por isso, seu uso é restrito a pequenas quantidades de dados e/ou pequenos valores de K.

Técnicas de grupo (*clumping*): retornam grupos nos quais seus membros podem se sobrepor. O problema com essas técnicas é que vários *clusters* para um mesmo objeto podem ser obtidos.

Técnicas probabilísticas: associam, para cada dado, a probabilidade dele estar em cada grupo.

Técnicas hierárquicas: criam árvores comumente chamadas de dendrogramas. As folhas da árvore representam exemplos individuais e os nós internos representam grupos de objetos. Esses métodos podem ser subdivididos em aglomerativos e divisivos. Os métodos aglomerativos constróem a árvore de baixo para cima (das folhas para a raiz), enquanto os divisivos constróem a árvore de cima para baixo. As técnicas hierárquicas são computacionamente mais baratas que as técnicas de otimização.

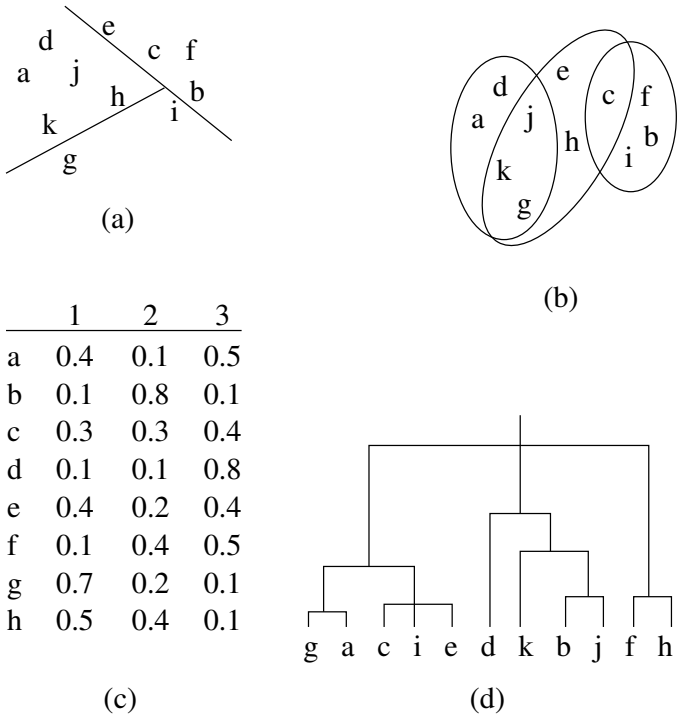


Figura A.1: Métodos de *clustering*: (a) otimização (b) *clumping* (c) probabilístico (d) hierárquico

Segue uma descrição de alguns algoritmos de *clustering* (Witten and Frank 2000) e (Gale 1986).

A.2.1 Algoritmo K-means

O algoritmo K-means é um dos mais simples. Ele encontra-se na classe das técnicas de otimização, ou seja, tem como saída K partições. Como entrada, o usuário deve especificar o número K de partições.

Inicialmente, o algoritmo escolhe, aleatoriamente, K pontos que serão os centros dos grupos iniciais. O próximo passo é atribuir, para cada exemplo do conjunto de dados, ao grupo a qual ele pertence. Isso é feito calculando-se a distância, por exemplo a euclidiana, entre o exemplo e cada centro. O exemplo irá pertencer ao grupo do qual ele estiver mais perto do centro.

Depois, o centróide (meio) de cada grupo é calculado. Deve ser observado que os pontos iniciais não são o centro definitivo de cada grupo; eles são, apenas, uma tentativa inicial. O centróide calculado, para cada grupo, passa a ser o novo centro. O processo se repete até que a iteração termina quando os centros dos grupos estabilizam, isto é, o mesmo ponto é escolhido como centro durante algumas iterações.

Esse algoritmo tem alguns problemas. Os centros dos grupos finais nem sempre representam uma solução ótima. Isso porque diferentes configurações de grupos podem ser obtidas a partir de diferentes escolhas para os centros dos grupos iniciais. Não é difícil imaginar o que acontece quando a situação mostrada na Figura A.2 acontece.

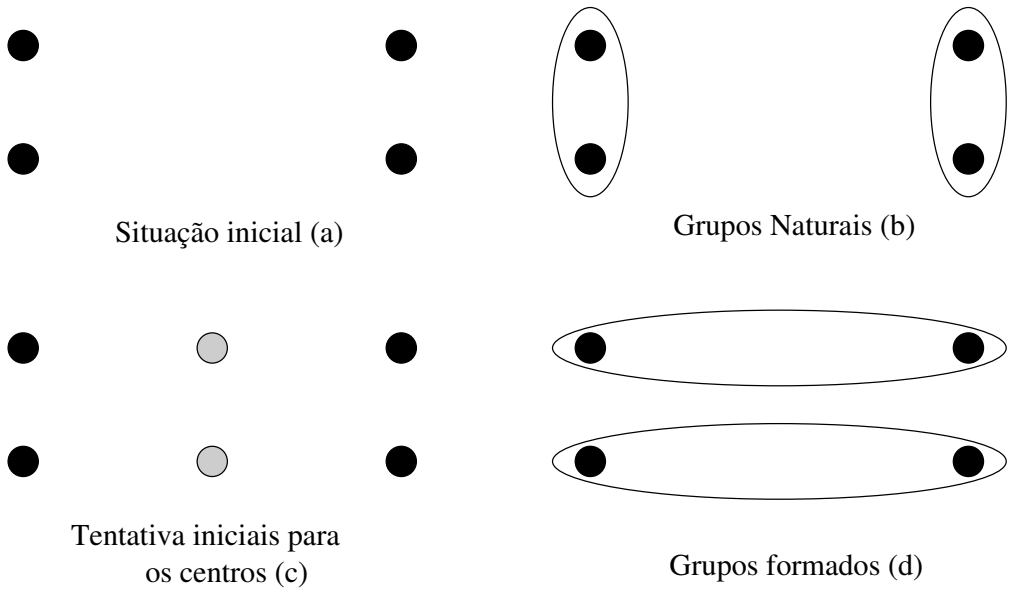


Figura A.2: Um dos problemas do algoritmo K-means

Nessa situação (a), tem-se, como dados, quatro pontos que formam um retângulo. Os agrupamentos naturais seriam os dois formados pelos pontos dos lados menores do retângulo (b). Mas, supondo que os centros iniciais caiam no meio dos lados maiores (c), essa configuração forma uma situação estável, fazendo com que os grupos finais sejam os dois formados pelos pontos dos lados maiores do retângulo (d).

Ou seja, os grupos formados pelo algoritmo podem não ser os naturais. Deve ser observado que o resultado mostrado não está errado mas, para os seres humanos, não é o agrupamento natural. Entretanto, caso os centros escolhidos inicialmente fossem outros, os grupos naturais seriam formados.

Assim, para encontrar os melhores agrupamentos deve-se executar o algoritmo algumas vezes, com diferentes centros iniciais, e escolher o melhor resultado.

A.2.2 *Clustering* Incremental

Esse algoritmo trabalha olhando para cada exemplo do conjunto de dados, um a um, formando uma árvore na qual cada folha é um objeto e a raiz representa todo o conjunto de dados. Ou seja, o algoritmo pertence à classe das técnicas hierárquicas.

No começo, a árvore consiste apenas da raiz. Os exemplos são adicionados um de cada vez, formando um novo grupo ou juntando-se a um grupo já existente. O algoritmo também prevê reestruturações maiores na árvore.

Para decidir onde cada exemplo será colocado, um valor chamado de *utilidade da categoria* é calculado. Esse valor mede a qualidade de um agrupamento.

Cada novo exemplo é processado tentando colocá-lo em cada um dos nós filhos da raiz calculando-se a utilidade da categoria para cada possibilidade de agrupamento. Se não for oportuno, o exemplo é colocado como um novo filho da raiz. Mas, se a utilidade da categoria for boa, então o exemplo forma um novo grupo com aquele nó filho. Caso esse nó já seja um grupo, o algoritmo é chamado recursivamente com esse grupo sendo a raiz da subárvore. Na Figura A.3 é mostrada uma visão melhor desses passos.

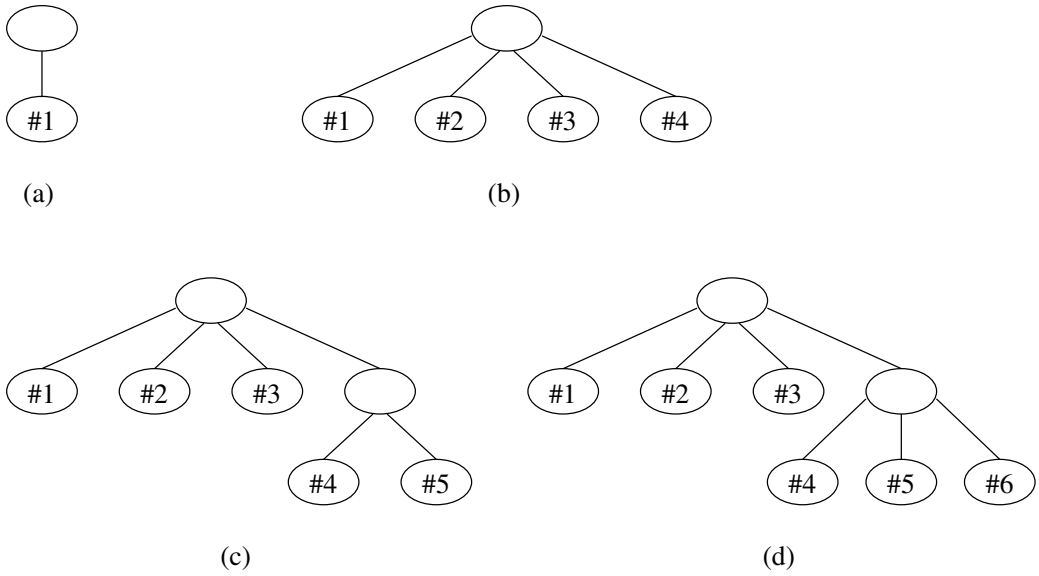


Figura A.3: (a) situação inicial (b) alguns exemplos são adicionados, mas não forman grupos (c) um primeiro grupo é formado (d) mais um exemplo é adicionado ao novo grupo

Mas, se o algoritmo seguisse sempre esses passos, a estrutura da árvore seria fortemente dependente da ordem dos exemplos. Para contornar esse problema, o algoritmo, ao calcular a utilidade da categoria para cada um dos nós, guarda os dois melhores valores. O melhor valor é utilizado para formar um novo grupo (a menos que seja melhor criar um novo grupo separado para o novo exemplo). No entanto, antes de formar o grupo, o algoritmo verifica a possibilidade de juntar o segundo melhor valor com o primeiro. Na Figura A.4

é descrita essa etapa.

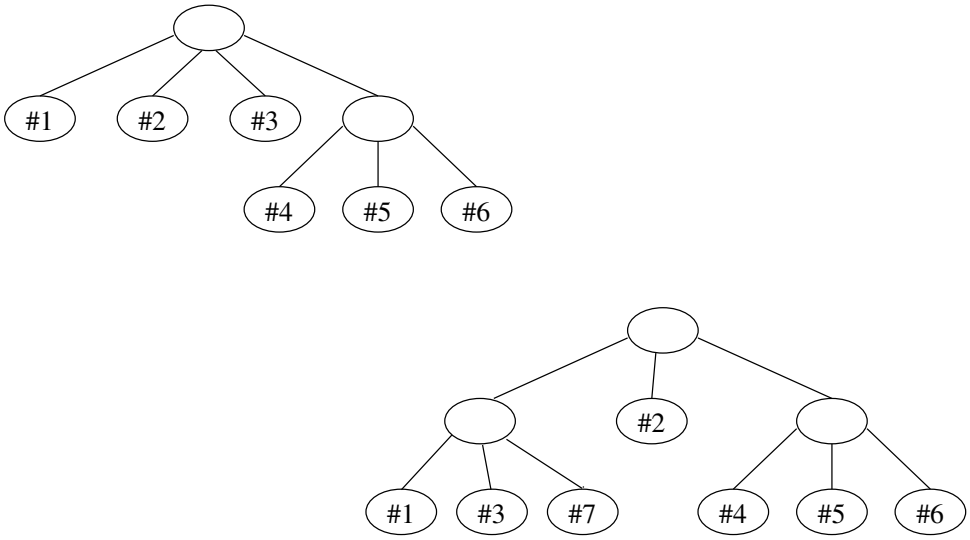


Figura A.4: Reestruturação da árvore

A operação inversa também pode ser executada, ou seja, a divisão de um grupo. Sempre que o melhor valor para a utilidade da categoria é encontrado e a união de dois grupos (melhor valor e segundo melhor) não for indicada, o algoritmo verifica a possibilidade de dividir o grupo de melhor valor.

As operações de união e divisão provêm uma maneira de compensar escolhas erradas causadas pela ordem dos dados.

Um problema que ocorre com esse algoritmo é que ele cria uma árvore de hierarquia sobrecarregada, com cada exemplo do conjunto de dados em suas folhas. Para evitar isso, um parâmetro de corte é dado para suprimir o crescimento. O corte é especificado em termos da função de utilidade da categoria. Sempre que o aumento na utilidade da categoria, causado pela adição de um novo nó, for suficientemente pequeno, esse nó não é considerado.

A.2.3 Algoritmo EM

Normalmente, não é possível determinar com certeza absoluta o grupo ao qual pertence cada um dos exemplos considerados, mas somente a probabilidade de um exemplo pertencer a um ou outro grupo. O algoritmo EM (*Expectation-Maximization*) faz justamente isso. Portanto, ele pertence à classe das técnicas probabilísticas.

O fundamento desse algoritmo baseia-se num modelo estatístico chamado *misturas finitas*. Uma mistura é um conjunto de k distribuições de probabilidade, representando k grupos, que governam os atributos dos membros do grupo. Ou seja, se for conhecido que um

determinado exemplo pertence a um grupo A, a função de distribuição de probabilidade desse grupo daria a probabilidade desse exemplo pertencer à esse grupo. É importante notar que cada exemplo pertence, de verdade, a somente um grupo. Simplesmente não se sabe qual.

O exemplo mais simples do modelo de misturas finitas ocorre quando tem-se apenas um atributo numérico que possui uma distribuição normal para cada agrupamento, mas com diferentes médias e variâncias. O problema de *clustering* resume-se em ajustar os parâmetros das distribuições em cima dos dados.

A fim de exemplificar, sejam dois grupos A e B tal que cada um tenha uma distribuição normal com parâmetro μ_a e δ_a para A e μ_b e δ_b para B. Alguns exemplos são amostrados dessas distribuições, usando as probabilidades p_a e p_b (onde $p_a + p_b = 1$), resultando num conjunto de dados formado pela classe de onde ele veio e o valor que ele assume. Na Figura A.5 é ilustrado o problema para um conjunto de exemplos.

Dados

A 51	B 62	B 64	A 48	A 39	A 51
A 43	A 47	A 51	B 64	B 62	A 48
B 62	A 52	A 52	A 51	B 64	B 64
B 64	B 64	B 62	B 63	A 52	A 42
A 45	A 51	A 49	A 43	B 63	A 48
A 42	B 65	A 48	B 65	B 64	A 41
A 46	A 48	B 62	B 66	A 48	
A 45	A 49	A 43	B 65	B 64	
A 45	A 46	A 40	A 46	A 48	

Modelo

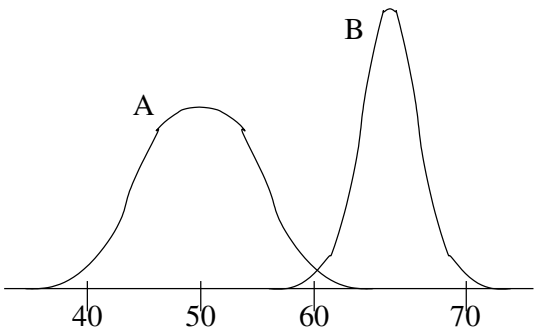


Figura A.5: Um modelo de mistura com duas classes. $\mu_a = 50, \delta_a = 5, p_a = 0.6; \mu_b = 65, \delta_b = 2, p_b = 0.4$

Supõe-se, agora, que somente esses valores sejam dados (sem a classe) e que deseja-se determinar os seis parâmetros que caracterizam o modelo (μ_a, δ_a, μ_b, δ_b, p_a, p_b). Esse é o chamado *problema da mistura finita*.

Se fossem conhecidos os dados com as classes, achar os parâmetros seria uma questão de

estimar a média e o desvio padrão para as amostras da classe A e B, separadamente. E, se tivéssemos os parâmetros, para achar a probabilidade de que um exemplo tenha vindo da distribuição A bastaria calcular $P(A|x)$.

O problema é que não se sabe nem de quais grupos vieram os exemplos e nem quais os parâmetros das distribuições. O que o algoritmo EM faz é utilizar o mesmo procedimento utilizado no algortimo K-means: faz-se uma tentativa inicial para os parâmetros, utilizam-se esses parâmetros para calcular as probabilidades dos exemplos pertencerem a cada grupo (*expectation*), reestimam-se os parâmetros com base nas probabilidades calculadas (*maximization*) e repete-se o processo.

Como mencionado anteriormente, o algoritmo K-Means pára quando os centros se estabilizam. Já o algoritmo EM tem um problema: ele converge para um valor fixo, mas nunca o atinge. Para contornar isso, o algoritmo calcula um valor chamado *probabilidade geral*. Esse valor, que aumenta a cada iteração, mede a qualidade dos agrupamentos. Com isso, o processo de iteração se repete até que o aumento da probabilidade geral se torne desprezível.

Deve ser ressaltado que o algoritmo EM não calcula os agrupamentos ótimos, sendo que todo o processo deve ser repetido algumas vezes para se obter o melhor resultado.

A.2.4 *Clustering* Conceitual

Com a finalidade de construir os grupos, os algoritmos de *clustering* utilizam uma função que mede a similaridade entre objetos ou grupos de objetos. Ou seja, a similaridade entre dois objetos é o valor de uma função numérica aplicada às descrições dos objetos, que podem ser vistas como vetores, ou seja:

$$Similaridade(A, B) = f(\vec{A}, \vec{B})$$

Esse conceito de similaridade é chamado *livre de contexto*, uma vez que ela é independente da relação que A ou B tem com os outros objetos. A distância euclidiana utilizada no algoritmo K-means é um exemplo disso.

Há, também, as medidas *dependentes de contexto*. Elas medem a similaridade entre dois objetos levando em conta os outros objetos (O).

$$Similaridade(A, B) = f(\vec{A}, \vec{B}, \vec{O})$$

Como exemplo disso, tem-se que os número 1 e 9 são similares no contexto dos números de 1 a 100, mas são muito diferentes se for levado em conta uma faixa de valores entre 1 e 10.

No *clustering* conceitual, não há interesse em apenas agrupar os dados, mas sim em identificar descrições conceituais para os grupos e usar esses conceitos para guiar a busca de um melhor agrupamento. A similaridade entre objetos é dependente de um conjunto de conceitos (C) (um conjunto de regras) que podem ser utilizados para descrever estruturas dentro dos grupos. Assim, tem-se que:

$$Similaridade(A, B) = f(\vec{A}, \vec{B}, \vec{O}, C)$$

A qualidade de um grupo é dependente da qualidade do conceito que descreve o grupo. No *clustering* conceitual, os exemplos são agrupados de forma a maximizar a qualidade do conceito que descreve o grupo.

Como exemplo, considere-se o conjunto de dados (a) da Figura A.6.

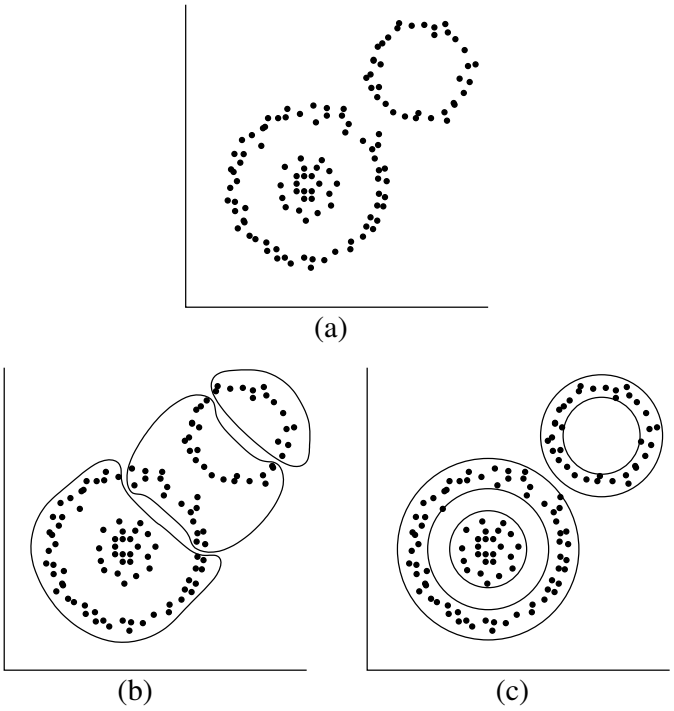


Figura A.6: (a) conjunto de dados (b) grupos obtidos por *clustering* não conceitual (c) grupos obtidos por *clustering* conceitual

Cada objeto é definido por duas variáveis. Se fosse utilizado, por exemplo, o algoritmo K-means, poderíam-se encontrar os grupos mostrados em (b). Mas, supondo que tem-se o seguinte conceito:

$$r1 \leq \sqrt{(x - c_1)^2 + (y - c_2)^2} \leq r2$$

Então, interpretando, o interesse está em grupos que formem anéis. Para isso, o algoritmo deve encontrar as constantes r_1, r_2, c_1 e c_2 de forma a maximizar o conceito. Com isso, obtêm-se os grupos mostrados em (c) da Figura A.6.

A vantagem do *clustering* conceitual é que restringem-se os possíveis grupos que podem ser formados.

A.3 Considerações Finais

Os algoritmos de *clustering* são bem variados e dão uma noção de como os dados podem ser agrupados, possibilitando algum discernimento sobre os mesmos. Muitas vezes, *clustering* é seguido de uma etapa na qual técnicas preditivas são utilizadas, inferindo regras sobre os *clusters* descobertos. Isso pode ajudar a ter uma compreensão sobre os exemplos que pertencem a cada *cluster* (Martins and Monard 2000).

Apêndice B

Regras de Associação

B.1 Considerações Iniciais

As regras de associação ajudam a identificar relações entre os dados. Normalmente, são utilizadas para analisar compras de supermercados. Por exemplo: "se uma pessoa compra um martelo então ela compra pregos". Mas as regras de associação podem ser utilizadas para outros fins, tais como:

- Serviços opcionais de telecomunicações assinados pelos clientes podem ajudar a determinar quais desses serviços podem ser oferecidos em um único pacote. O mesmo pode ser feito para serviços bancários.
- Combinações não usuais de compras podem indicar uma fraude no cartão de crédito e merecer uma investigação.
- Os históricos de pacientes de um hospital podem identificar combinações de tratamentos que levam à complicações.

Muitas vezes, as regras de associação são utilizadas para dar uma idéia sobre dados que estão disponíveis mas que não se sabe o que procurar neles.

B.2 Tipos de Regras

Em geral, são geradas três tipos de regras pelo algoritmo de regras de associação:

As úteis: são as de alta qualidade no que tange à sugestão de ações a serem tomadas. O exemplo clássico é a regra: "nas quintas-feiras, fraldas e cerveja são compradas

juntas". Essas regras não são de difícil explicação. Nesse exemplo, descobriu-se que nas quintas, os casais jovens preparavam-se para o final-de-semana estocando fraldas para as crianças e cerveja para os pais.

As triviais: são aquelas já conhecidas das pessoas que são familiares com o negócio. O exemplo de "quem compra martelo compra pregos" pertence a essa categoria. As regras triviais são as mais frequentemente geradas pelos algoritmos de regras de associação, principalmente depois de uma promoção. Nesse caso, as regras podem estar medindo o sucesso de uma campanha de *marketing*.

As inexplicáveis: são as que têm difícil explicação. Elas não sugerem nenhuma ação imediata. Um exemplo seria: "quando uma nova loja de ferragens é aberta, o produto mais vendido é anel de torneira". Ou seja, um padrão de venda foi descoberto mas não se sabe como lucrar com ele. Para essas regras, uma maior investigação sobre o fato deve ser feita.

B.3 O Processo Básico

Dada as transações a serem analisadas, a criação das regras começa com a geração da matriz de co-ocorrências. Essa matriz mostra o número de vezes que pares de itens ocorrem em conjunto. Nas Tabelas B.1 e B.2 é mostrado um exemplo.

Cliente	Itens comprados
1	suco de laranja, refrigerante
2	leite, suco de laranja, limpador de janelas
3	suco de laranja, detergente
4	suco de laranja, detergente, refrigerante
5	limpador de janelas, refrigerante

Tabela B.1: Exemplos de transações

	SL	LJ	L	R	D
SL	4	1	1	2	1
LJ	1	2	1	1	0
L	1	1	1	0	0
R	2	1	0	3	1
D	1	0	0	1	2

Tabela B.2: Matriz de co-ocorrências. SL = Suco de Laranja; LJ = Limpador de Janelas; L = Leite; R = Refrigerante; D = Detergente

Com a matriz de co-ocorrências pode-se verificar, por exemplo, que há duas transações nas quais refrigerante e suco de laranja aparecem juntos. Com isso, pode-se criar regras do tipo: "se um cliente compra suco de laranja então ele também compra refrigerante" ou então "se um cliente compra refrigerante então ele também compra suco de laranja". Os valores na diagonal representam o número de transações em que aparece o respectivo item. A idéia de matriz de co-ocorrências pode ser estendida para qualquer número de combinações, ou seja, regras do tipo "se A e B então C" também podem ser geradas.

B.4 Medidas

O número de regras que podem ser geradas a partir da matriz de co-ocorrências é enorme. Por isso, é necessário escolher as melhores regras. Para medir a qualidade das regras, têm-se algumas medidas, tais como:

Suporte: é a frequência com que todos os itens da regra aparecem nas transações. Por exemplo, para a regra "se um cliente compra suco de laranja então ele também compra refrigerante", duas das cinco transações a suportam. A regra tem um suporte de 40%.

Confidência: mede com que frequência o resultado da regra aparece nas transações dado que a condição é satisfeita. Ou seja, em uma regra qualquer "se A então B", é a frequência[1] condicional de B dado A.

$$Confidência = F(B|A) = \frac{F(A \text{ e } B)}{F(A)} = \frac{Suporte}{F(A)}$$

No exemplo considerado, a confidência da regra "se um cliente compra suco de laranja então ele também compra refrigerante" é de 50% (2 transações em que aparece suco de laranja e refrigerante dividido por 4 transações em que aparece suco de laranja). É interessante observar que a outra regra "se um cliente compra refrigerante então ele também compra suco de laranja" tem uma confidência maior (66.6%).

Confidência esperada: em uma regra "se A então B", é a frequência com que B aparece nas transações, ou seja, o quanto se espera ter de B no conjunto de dados.

***Lift*:** mede o quanto a parte condicional da regra influência (ou alavanca) a parte do resultado.

[1] Não se diz que é a probabilidade porque é calculada sobre o conjunto de dados, e não com o número de dados tendendo à infinito.

$$Lift = \frac{Confidência}{F(B)} = \frac{Confidência}{Confidênciaesperada}$$

O *lift* mostra o quanto a regra é melhor em predizer o resultado do que simplesmente escolher aleatoriamente. Quando o *lift* é maior que 1, então tem-se uma boa regra. Quando o *lift* é menor que 1, negar o resultado produz uma regra melhor. Exemplo: se a regra "se A então B" tem uma confidência de 33%, a regra "se A então não B" tem uma confidência de 67%.

A seguir são mostrados alguns exemplos numéricos dessas medidas.

Número de transações: 1000
Número de transações que têm martelo: 50
Número de transações que têm pregos: 80
Número de transações que têm madeira: 20
Número de transações que têm martelo e pregos: 15
Número de transações que têm pregos e madeira: 10
Número de transações que têm martelo e madeira: 10
Número de transações que têm martelo, pregos e madeira: 5

Suporte para martelo e pregos: 1.5% (15/1000)
Suporte para martelo, pregos e madeira: 0.5% (5/1000)

Confidência para "martelo ⇒ pregos": 30% (15/50)
Confidência esperada para "martelo ⇒ pregos": 8% (80/1000)
Lift para "martelo ⇒ pregos": 3.75 (30%/8%)

Confidência para "pregos ⇒ martelo": 19% (15/80)
Confidência esperada "pregos ⇒ martelo": 5% (50/1000)
Lift para "pregos ⇒ martelo": 3.8% (19%/5%)

Confidência para "martelo e pregos ⇒ madeira": 33% (5/15)
Confidência esperada para "martelo e pregos ⇒ madeira": 2% (20/1000)
Lift para "martelo e pregos ⇒ madeira": 16.5 (33%/2%)

Confidência para "madeira ⇒ martelo e pregos": 25% (5/20)
Confidência esperada para "madeira ⇒ martelo e pregos": 1.5% (15/1000)
Lift para "madeira ⇒ martelo e pregos": 16.6% (25%/1.5%)

B.5 Considerações Finais

As regras de associação são úteis para descrever relações entre os dados. Mais que isso, elas sugerem a tomada de ações. Mas, algumas vezes, essas ações nem sempre são de fácil decisão. Por exemplo, na descoberta de que dois itens são vendidos juntos, colocá-los perto um do outro pode reduzir o total de compras feitas pelos clientes pois, frequentemente, as pessoas levam produtos que não tinham planejado comprar ao andar pelo supermercado.

Apêndice C

Recuperação de Informação

C.1 Considerações Iniciais

Antigamente, as bibliotecas enfrentavam o problema de indexar e recuperar textos manualmente. A utilização da tecnologia da computação nessas bibliotecas, permitiu a automatização dessas tarefas. Catálogos foram automatizados e bancos de dados de bibliografias foram criados. As técnicas de recuperação automática de documentos relevantes passaram a ser utilizadas, também, nos escritórios de patentes, em jurisprudência, em escritórios de negócios e em outras áreas.

C.2 O Processo Básico

Basicamente, a Recuperação de Informação possui três atividades distintas:

Indexação: refere-se aos métodos utilizados para representar os documentos com a finalidade de recuperá-los.

Busca: é o processo de examinar os documentos ou as suas representações em busca de uma palavra ou frase.

Colocação (*Ranking*): refere-se ao processo de ordenar os documentos recuperados em ordem de relevância.

Após a busca, vários documentos podem ser recuperados, alguns relevantes e outros não. A eficiência de um sistema de Recuperação de Informação é medida por três estimadores estatísticos: precisão, *recall* e *fallout*. Quando uma busca termina, tem-se duas coleções de

documentos: os documentos que foram selecionados (ou recuperados) pela busca e os que não foram selecionados. Dentro de cada um desses grupos tem-se, ainda, uma subdivisão em dois grupos: os documentos que são relevantes e os que não são. Na Figura C.1 é ilustrada melhor essa situação.

	Relevantes	Não Relevantes	
Recuperados	a	b	(a + b) Documentos Recuperados
Não Recuperados	c	d	(c+d) Documentos Não Recuperados
	(a+c) Todos Documentos Relevantes	(b+d) Todos Documentos Não Relevantes	(a+b+c+d) Todos os Documentos

Precisão = a / (a + b)
Recall = a / (a + c)
Fallout = b / (b + d)

Figura C.1: Precisão, *recall* e *fallout*

Precisão: é a parte dos documentos que foram recuperados e que são relevantes em relação à palavra que se está buscando.

***Recall*:** é a parte dos documentos relevantes que foi recuperada. Isto é, a "chamada pelos documentos relevantes".

***Fallout*:** é a parte dos documentos não relevantes que foi recuperada. Ou seja, a "queda na precisão".

Assim, o objetivo é maximizar a precisão e o *recall* e minimizar o *fallout*. É importante notar que só a precisão não é suficiente como medida pois, para maximizá-la, basta recuperar 1 documento relevante e nenhum não relevante. Nesse caso, tería-se 100% de precisão. Com o *recall* tem-se uma situação parecida: basta recuperar todos os documentos que existem na base. Todos os documentos relevantes necessariamente estarão aí. Portanto, a precisão e o *recall* devem ser utilizados em conjunto. Como a precisão e o *recall* são insensíveis ao número total de documentos, utiliza-se o *fallout* como índice de rejeição.

As técnicas para Recuperação de Informação podem ser dividas em duas:

1. força bruta

2. indexação

discutidas a seguir.

C.2.1 Força Bruta

A maneira mais óbvia de recuperar os documentos que contêm uma palavra determinada é procurar, um a um, em todos os documentos, por aquela palavra.

Um algoritmo simples para fazer isso é considerar o documento como uma grande e única cadeia de caracteres e ter dois ponteiros: um para o documento e outro para a palavra a ser encontrada. Compara-se, então, as letras apontadas pelos ponteiros e, se casarem, incrementam-se os dois ponteiros. Se não casarem, reposiciona-se o ponteiro da palavra se ele não estiver na primeira letra ou incrementa-se somente o ponteiro do documento. Ou seja, desliza-se a palavra a ser encontrada pelo texto. O problema com esse algoritmo é que ele é muito lento: se m é o tamanho da palavra e n é o tamanho do documento, temos $O(n \times m)$ comparações.

Knuth, Morris e Pratt (Knuth, Morris, Jr., and Pratt 1977) e Boyer e Moore (Boyer and Moore 1977) propuseram duas modificações diferentes para esse algoritmo de forma que ele precisasse de $O(n + m)$ comparações. No primeiro caso, a palavra a ser procurada é deslizada pelo documento mais que um incremento por vez. No segundo caso, a palavra é comparada de trás para frente.

Apesar da simplicidade, esses algoritmos são redundantes pois, se procura-se pela mesma palavra duas vezes, todos os documentos serão varridos duas vezes. Ou seja, como não é feita uma indexação, a única representação para os documentos são os próprios documentos. Contudo, isso traz a vantagem de não haver necessidade de utilizar espaço extra para armazenar os índices, por exemplo.

C.2.2 Indexação

Ficou claro para a comunidade de pesquisadores na área de Recuperação de Informação que a indexação é a chave para o sucesso desse tipo de sistema. Assim, vários métodos de indexação foram criados sendo que cada um desses métodos tem a sua própria técnica e forma de representação (estrutura de dados) específicas.

É importante notar que nesses métodos nem todas as palavras são indexadas. A seguinte lista mostra alguns exemplos de palavras que não são indexadas por esses métodos:

***Stop words*:** palavras que ocorrem com muita frequência normalmente são eliminadas antes da indexação. Exemplos dessas palavras são os artigos e as preposições.

Plurais: palavras no plural são convertidas para o singular.

Normalização: técnicas de normalização utilizadas em Processamento de Língua Natural (PLN) também são utilizadas para converter palavras com significados semelhantes em um único conceito. Uma técnica bem comum é o *stemming*, que consiste em reduzir as palavras às suas raízes retirando os afixos.

***Thesauri*:** os métodos de indexação baseados na ocorrência de uma única palavra não funcionam adequadamente. Métodos baseados em conceitos, ao contrário, casam palavras com significados semelhantes. Uma maneira de fazer isso é com a ajuda dos *thesauri*, que correlacionam termos em um dicionário. Além de reduzir o tamanho dos índices, essas técnicas aumentam a eficiência da seleção de documentos.

A desvantagem dos métodos de indexação é que eles consomem espaço, podendo chegar a necessitar 300% do espaço utilizado para os documentos. Segue uma descrição de alguns métodos de indexação.

C.2.2.1 *Clustering*

A idéia por trás do *clustering* é que documentos semelhantes tendem a serem relevantes para a mesma palavra de busca. Agrupando documentos similares em *clusters*, o espaço de busca pode ser reduzido e a busca acelerada.

As técnicas de *clustering* agem sobre vetores. Assim, cada documento é representado por um vetor t-dimensional, onde t é o número de termos escolhidos para serem indexados. Os t termos são escolhido entre todos os termos de todos os documentos. No vetor, um 0 representa a ausência da palavra e um 1 representa a presença da mesma. Pode-se, também, indicar a presença da palavra por um peso, calculado ou pela frequência do termo no documento ou pela frequência relativa a todos os documentos.

A maioria dos métodos de *clustering* requer que uma função de similaridade seja especificada. É ela quem vai dar o grau de similaridade entre os vetores que representam os documentos (Apêndice A).

Uma aplicação interessante do *clustering* é aplicá-lo aos termos. Termos similares tendem a serem relacionados. Dessa forma, pode-se reduzir a dimensão dos vetores que representam os documentos, onde os grupos de termos relacionados formam conceitos que representam os termos.

C.2.2.2 Assinaturas

No método de recuperação por assinaturas cada documento é assinado com uma string de *bits*. Mais precisamente, tem-se um arquivo que armazena essas assinaturas.

A assinatura é gerada por meio da aplicação de funções de espalhamento[1] nos termos dos documentos e concatenando os valores gerados. Aplicando a mesma operação na palavra que se quer encontrar e comparando o valor obtido com as assinaturas, é possível encontrar os documentos que contenham a palavra de busca. Como a coleção de assinaturas é menor que a de documentos, a busca é mais rápida. O método é mais eficiente quando a distribuição de 1's na cadeia de *bits* é uniforme.

O problema com esse método é que, como as funções de espalhamento não são precisas (colisões), tem-se a geração de ruído, ou seja, documentos que não sejam relevantes podem ser recuperados.

Uma vantagem é que o espaço necessário para o índice (o arquivo de assinaturas) não é muito grande se comparado aos outros métodos.

C.2.2.3 Índices Invertidos

O índice invertido é um arquivo que contém cada termo de um conjunto de documentos associado a uma lista de documentos na qual ele aparece. Na Figura C.2 é mostrado um exemplo dessa estrutura.

Uma das vantagens dessa estrutura é que ela é fácil de ser implementada. Usando um árvore B+ acoplada ao arquivo de índice invertido, o processo de busca torna-se extremamente rápido. O maior problema dessa estrutura é o espaço que ela ocupa.

A simplicidade do índice invertido possibilita que outras informações sejam adicionadas ao arquivo de nomeação como, por exemplo, a localização da palavra dentro do arquivo. Na Figura C.3 é ilustrada essa possível modificação. Dessa forma, pode-se utilizar essas informações extras para ajudar na colocação (*ranking*) dos documentos após a busca.

C.3 Uso na Internet

As técnicas de Recuperação de Informação também podem ser utilizadas na Internet. De fato, existem muitos serviços da Internet que as utilizam, como as famosas máquinas de busca. Segue-se uma breve descrição desses serviços.

[1] Funções *hash*.

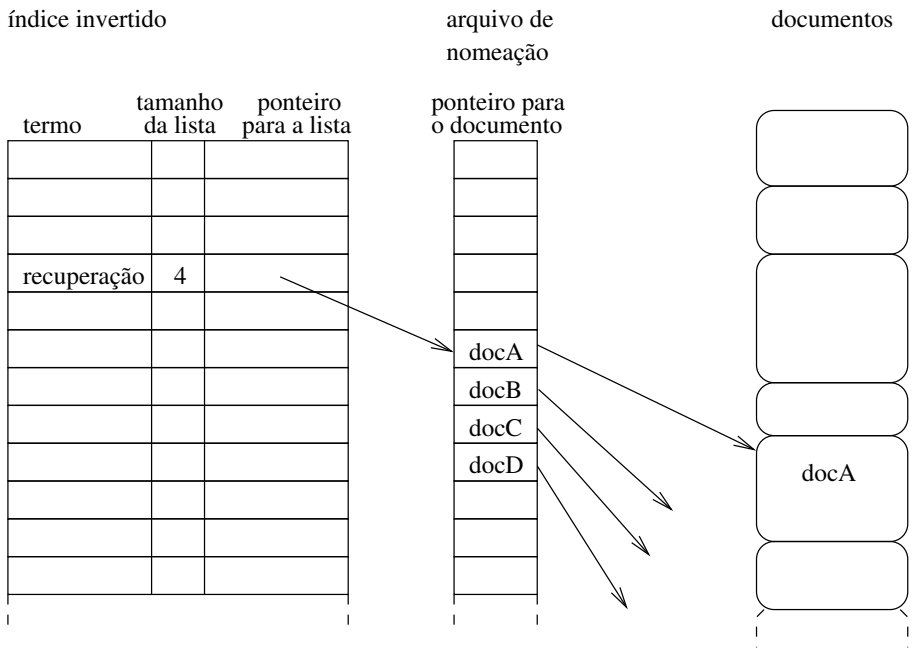


Figura C.2: Estrutura de um índice invertido

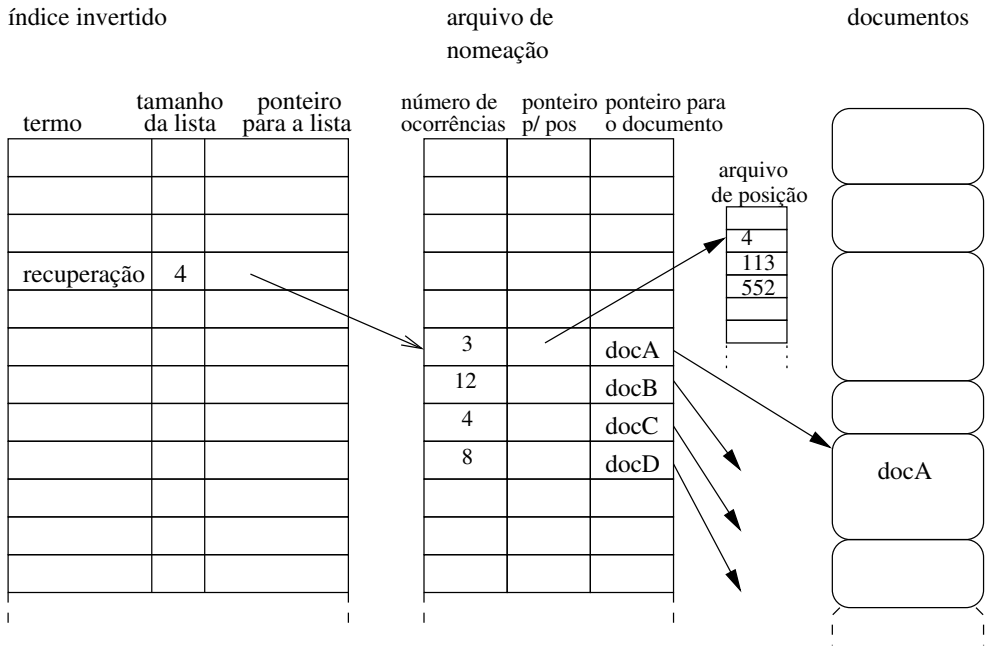


Figura C.3: Modificação para armazenar as posições dos termos nos documentos em que ele aparece

C.3.1 Serviços da Internet

O número de serviços que a Internet oferece é muito grande: desde serviços de comunicação até de distribuição de arquivos. Um dos problema que surge é o de indexar as informações que esses serviços disponibilizam.

A seguinte lista cita alguns serviços oferecidos na Internet e os métodos utilizados para indexar a informação:

***E-mail*:** é comum, nos ambiente de negócio, a troca de documentos através de *e-mails*. Tais arquivos se acumulam e procurar por um documento específico pode ser algo complicado. O uso das técnicas de Recuperação de Informação pode ser de grande ajuda nessa tarefa.

USENET: também é conhecida como lista de discussões. Cada lista contém milhares de mensagens sobre um determinado assunto e obter somente as mensagens relevantes a um tema específico dentro desse assunto é muito difícil. Aqui, também, as técnicas de Recuperação de Informação ajudam na seleção das mensagens que sejam interessantes para um determinado usuário.

FTP: permite o acesso a arquivo em computadores remotos. Um servidor de FTP disponibiliza parte da sua estrutura de diretórios para a troca de arquivos. Cada servidor oferece, normalmente, arquivos relacionados a um ou mais tópicos. Cada diretório acessível contém um arquivo de texto chamado `readme` (esse nome não é padronizado podendo ser `index`, `readme`, `read.me`, `dir`, etc.) explicando o conteúdo de cada arquivo no diretório.

Gopher: é um sistema distribuído de documentos. Possui uma interface simple de *menu* para acesso aos vários documentos.

Alex: é uma sistema de arquivos que possibilita o acesso aos arquivos de servidores FTP. O Alex permite ver os arquivos do FTP como parte da estrutura de diretórios local. Com isso, pode-se utilizar algumas ferramentas do Unix, como o `grep` e o `find`, para encontrar documentos em *sites* remotos sem ter que se fazer uma cópia local dos arquivos.

Archie: é um serviço que permite a localização de arquivos nos *sites* de FTP. Ele, periodicamente, cria uma lista dos arquivos disponíveis visitando recursivamente a estrutura de diretórios. Essa listagem é, então, armazenada em um banco de dados que é disponibilizado na Internet. O problema com o Archie é que só os nomes dos arquivo são indexados, o que nem sempre fornece informação sobre o conteúdo desses arquivos.

Veronica: ou *Very Easy Rodent-Oriented Net-wide Index to Computerized Archives*. É um sistema que indexa o conteúdo do Gopher. Ele, periodicamente, visita recursivamente o sistema de *menus* de vários servidores Gophers registrados, indexando todos os títulos em cada texto de *menu*.

WAIS: ou *Wide Area Information Servers*. Em contraste com o Archie, que indexa somente os nomes dos arquivos, os índices do WAIS contém palavras-chave para o conteúdo dos arquivos. Mais que isso, o índice do WAIS é descentralizado. Os primeiros *sites* de busca usavam o WAIS para indexar a informação recuperada pelos seus *spiders* (ver Seção C.3.2).

Netfind: é um serviço que tenta localizar dinamicamente endereços de *e-mail*, e outras informações, sobre os usuários da Internet. Ele usa um conjunto de heurísticas para localizar as máquinas em que o usuário que se está procurando pode ter acesso, ou uma conta de *e-mail*.

Catálogos e Diretórios: no começo da WWW, o número de páginas disponíveis era pequeno e muitos usuários construíam listas dos *links* mais interessantes (os famosos *bookmarks*), disponibilizando-as para o público. Contudo, o crescimento rápido da WWW tornou essas listas obsoletas. Numa tentativa de tornar essas listas mais atualizadas, criaram-se sistemas de coleta automática dessas listas. Mas esses sistemas criavam listas enormes, de forma que era necessário que eles possibilitassem uma busca por palavras nessas listas. A busca se processava percorrendo a lista e tentando casar a palavra procurada. Deve ser observado que não há a criação real de índices e que os sistemas de catálogos e diretórios são diferentes das máquinas de busca. No Yahoo[2], por exemplo, os *links* são coletados e classificados manualmente pelos editores. Mais precisamente, os autores de páginas enviam seus endereços para os *sites* de catálogos juntamente com um título e uma descrição dos seus *sites*. O *site* é, então, classificado e colocado no local certo da estrutura de diretórios.

***Webrings*:** os *webrings* são um tipo especializado de diretórios. Os *links* nesses diretórios formam uma lista circular que contém *sites* relacionados a um mesmo tópico. Uma "autoridade" central gerencia a lista em um servidor, aceitando novos membros e mantendo uma lista de todos os *sites* que compõem o *webring*. Na Figura C.4 é ilustrada a estrutura de um *webring*.

[2] http://www.yahoo.com

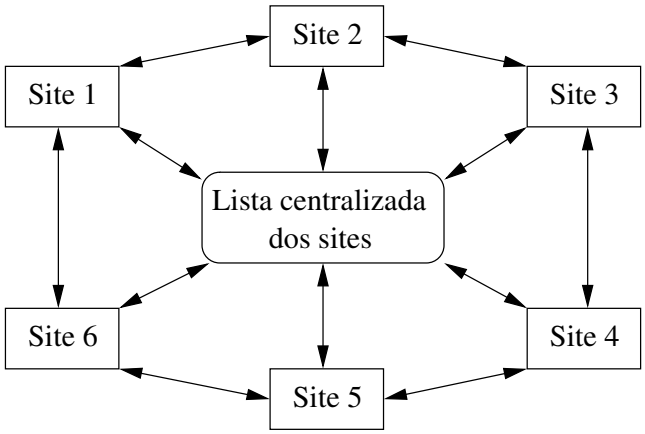


Figura C.4: Exemplo de um *webring* com 6 *sites*

C.3.2 Máquinas de Busca

As máquinas de busca (*search engines*) são programas que caminham pela *World Wide Web*, indexando o seu conteúdo, para uma busca posterior. Os documentos são encontrados explorando-se o grafo formado pelos *links* dos documentos. De um documento inicial, todos os seus *links* são extraídos e colocados em uma fila. O processo é repetido escolhendo-se um endereço da fila, visitando-o e extraindo seus *links*.

A arquitetura desses sistemas é composta de três componentes:

1. o *spider*
2. o *parser*
3. o indexador

O *spider* é o programa que caminha pela *Web*. É importante observar que o *spider* não caminha por toda a Web pois não é garantido que, a partir de um documento, todos os documentos da *Web* estejam conectados a ele direta ou indiretamente. Portanto, um conjunto de *links* iniciais ajuda a fazer uma cobertura melhor.

O *parser* é o programa que extrai os *links* dos documentos que o *spider* visitou. Esses novos *links* são devolvidos para o *spider* colocar na fila de *links* a visitar.

Quando um documento é analizado pelo *parser*, termos são extraídos para a indexação. A forma de indexação pode variar: pode-se indexar todo o texto ou partes dele como, por exemplo, os títulos. Também varia a forma como o índice é construído.

Na Figura C.5 é mostrado como esses componentes interagem, bem como as três listas que o *spider* utiliza:

LTV (*List To Visit*) é a lista de *links* que ainda serão visitados. Inicialmente essa lista contém um conjunto de *links* cuidadosamente escolhidos.

LV (*List Visited*) quando uma página da lista LTV é visitada, ela é removida da LTV e inserida na LV. Isso permite que o *spider* verifique se a página que ele está para visitar já não foi visitada.

LNV (*List Not to Visit*) o *spider* explora continuamente a *Web*, recuperando documentos inteiros e descartando-os depois de reter parte de seu conteúdo. Isso faz com que haja um aumento no tráfego da Internet e uma sobrecarga nos servidores. Esse cenário tende a piorar com o aumento do número de máquinas de busca disponíveis. Baseado nesses fatos, alguns guias de como construir programas do tipo *spider* foram propostos. Esses guias dão sugestões para os autores de páginas de como indicar para o *spider* qual conteúdo deve ser indexado e qual não deve. Os *spiders* que levam em conta essas indicações utilizam a LNV justamente para isso.

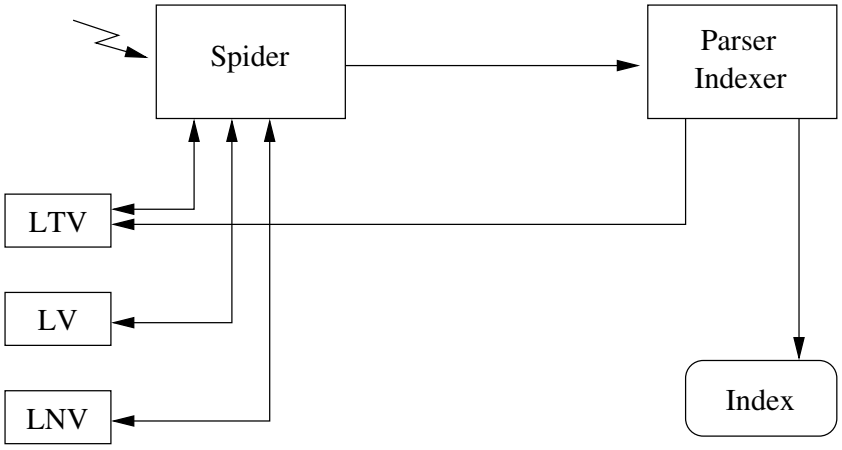


Figura C.5: Arquitetura de uma máquina de busca

Uma vez que a indexação tenha sido feita, os *sites* de busca disponibilizam uma interface para que palavras ou frases possam ser procuradas nos índices. Depois que a busca finaliza, é feita a colocação (*ranking*) dos documentos de acordo com alguma(s) estratégia(s). A seguinte lista mostra alguns dos critérios que podem ser utilizados:

Frequência: documentos nos quais a palavra sendo procurada aparece com mais frequência são melhores colocados.

Localização: a localização da palavra de busca pode influenciar na colocação. Palavras que aparecem nos títulos, por exemplo, são consideradas mais importantes. Outros locais importantes são os cabeçalhos, os endereços ou o começo dos documentos.

Totalidade: quando busca-se por mais de uma palavra, os documentos nos quais todas aparecem são melhores colocados.

Tamanho: documentos menores são mais favorecidos.

Diretório: se o *site* de busca contém um catálogo, eles tendem a favorecer os *sites* que aí se encontram.

***Links*:** páginas que são muito referenciadas por outras páginas são mais importantes. *Sites* populares são favorecidos.

Metadados: a linguagem HTML possibilita que o autor especifique metadados. Com eles, o autor pode especificar as palavras chaves e a descrição da página. Os *sites* de busca normalmente levam essas informações em consideração quando fazem a colocação das páginas recuperadas.

Dinheiro: as companhias podem pagar para terem seus *sites* aparecendo entre os 10 ou 20 melhores colocados quando seus *sites* forem recuperados por uma palavra de busca.

Sabendo os fatores que levam a uma melhor colocação, alguns *sites* tentam "trapacear" para estarem entre os melhores colocados. Isso é chamado de *spamming*. Por exemplo, uma página pode conter uma mesma palavra-chave muitas vezes dentro do seu conteúdo. Essas palavras são invisíveis para o usuário (porque estão com a mesma cor de letra e fundo ou porque estão entre tags de comentário) mas são visíveis para as máquinas de busca. Contudo, algumas máquinas de busca detectam o *spamming* e penalizam esses *sites* fazendo com que tenham uma baixa colocação.

C.4 Considerações Finais

As técnicas de Recuperação de Informação na *Web* são muitas e abrangem desde a utilização de técnicas de Banco de Dados até a aplicação de algoritmos de Aprendizado de Máquina.

Contudo, o que se tem hoje nos *sites* de busca é, ainda, a falta de relevância dos documentos recuperados. Isso se deve, em parte, pela dificuldade de se indexar documentos pouco estruturados como são as páginas HTML. A crecente utilização de XML pode resolver parte desse problema, uma vez que ele oferece uma melhor estruturação e adição de conteúdo semântico aos documentos.