

UNIVERSIDADE FEDERAL DE PELOTAS
Bacharelado em Ciência da Computação



Trabalho de Conclusão de Curso

**Ferramenta para extração e armazenamento de
informações do Orkut**

Roberta Furtado Corrêa

Pelotas, 2008

ROBERTA FURTADO CORRÊA

**FERRAMENTA PARA EXTRAÇÃO E ARMAZENAMENTO DE INFORMAÇÕES DO
ORKUT**

Trabalho de Conclusão de Curso
apresentado ao Curso de
Bacharelado em Ciência da
Computação da Universidade
Federal de Pelotas, como requisito
parcial à obtenção do título de
Bacharel em Ciência da
Computação

Orientador: Msc. Anderson Priebe Ferrugem

Pelotas, 2008

Banca examinadora:

.....
.....
.....

Agradecimentos

Agradeço primeiramente a Deus, por ter me dado uma família tão amorosa e unida e por ter posto em meu caminho uma pessoa tão especial quanto o meu namorado Vinícius.

À minha mãe Solange, agradeço pela paciência e amor, pela criação, que embora em meio a tantas dificuldades soube sozinha fazer de seus três filhos pessoas trabalhadoras e responsáveis. A esta pessoa de tão bom coração, que é minha companheira para todas as horas e a pessoa mais importante da minha vida, a quem devo tudo que sou e que tenho hoje.

Não poderia deixar de agradecer a meus irmãos, Alexandre e Cristiano, sem o auxílio dos quais não teria sido possível dedicar cinco anos de minha vida única e exclusivamente à universidade.

E por fim, agradeço ao meu namorado pela paciência nos momentos de estresse, pela espera e compreensão. Sou principalmente grata por ter me apoiado e nunca me deixar desistir ou desanimar nos momentos difíceis, ou quando me sentia cansada.

A todos vocês o meu muito obrigado.

Resumo

Devido ao contínuo aumento da concentração de documentos e serviços disponíveis *online*, a *Web* pode ser considerada a maior fonte de informação existente. Entretanto, grande parte desta informação encontra-se implícita no meio de um grande volume de textos e hipertextos. Entre os serviços *Web* existentes estão os populares *sites* de relacionamento, que são compostos por páginas específicas, padronizadas e interligadas. Estes documentos contêm informações pessoais de milhões de usuários em seus perfis, os quais podem formar grupos por afinidade em comunidades e manter explicitamente relações de amizade, facilitando a descoberta de padrões e identificação de agrupamentos. Tais fatores fazem da *Web*, e conseqüentemente dos *sites* de relacionamento, fontes valiosas para Mineração de Dados. Sendo assim, este trabalho tem como objetivo a busca, extração e armazenamento automático de informações sobre comunidades e seus membros, visando um pré-processamento das informações para em um futuro estudo aplicar técnicas para identificação de padrões e agrupamentos.

Palavras-chave: Extração de Informação. Recuperação de Informação. Banco de Dados. Mineração na *Web*. Mineração de Dados. Redes Sociais.

Abstract

Due to the continuous increase of documents and services available online, the Web can be considered the greatest information source. However, most of that information stays implicit in a huge hypertext amount. Among the available services in the Web there are the very popular *social networks*: a set of interconnected standardized pages which contains personal information from millions people. The members of a social network are able to create affinity groups and establish friendship relations directly. Because of that feature we can discover behavioral patterns related with groups of people. So, social networking sites are really valuable data mining sources. In this work we aim to realize the automatic searching, extraction, storage and pre-processing of social networks groups information in order to allow application of pattern discovery approaches over the structured data.

Keywords: Information Extraction. Information Retrieval. Database. Web Mining, Data Mining. Social Networks.

Lista de Figuras

Figura 1 - Etapas da Mineração na <i>Web</i>	17
Figura 2 - Classificação da Mineração na <i>Web</i>	26
Figura 3 - Visão geral do algoritmo implementado pela ferramenta.....	37
Figura 4 - Diagrama de classes do módulo Model	42
Figura 5 - Diagrama de classes do módulo BD.	44
Figura 6 - Diagrama ER do banco de dados.	46
Figura 7 - Pesquisa manual por comunidades.....	49
Figura 8 - Comunidades capturadas.....	50
Figura 9 - Página de comunidade.	50
Figura 10 - Donos das comunidades capturadas.	51
Figura 11 - Membros de uma das comunidades capturadas.....	51
Figura 12 - Amigos de um dos membros de uma das comunidades capturadas.	52
Figura 13 – Comunidades relacionadas.....	53
Figura 14 - Relacionadas de uma das comunidades capturadas.	54

Lista de abreviaturas e siglas

Application Programming Interface - API

Análise de Redes Sociais - ARS

Banco de dados - BD

Extração de Informação - EI

Entidade Relacionamento - ER

File Transfer Protocol - FTP

HyperText Markup Language - HTML

Hypertext Transfer Protocol - HTTP

Inteligência Artificial - IA

Java Database Connectivity - JDBC

Java Virtual Machine - JVM

Processamento de Linguagem Natural - PLN

Recuperação de Informação - RI

Sistema Gerenciador de Banco de Dados - SGBD

Structured Query Language - SQL

Universal Resource Locator - URL

Sumário

1. Introdução	10
1.1 Motivação	11
1.2 Objetivos	14
2. Mineração na <i>Web</i>	16
2.1 Etapas	16
2.1.1 Recuperação de Informação (RI)	17
2.1.1.1 Motores de busca	17
2.1.1.2 Diretórios de Tópicos	18
2.1.1.3 Rastreadores <i>Web</i>	18
2.1.1.4 Indexação e pesquisa por palavra-chave	20
2.1.1.4.1 Relevance Ranking	21
2.1.1.4.1.1 Ranking baseado em hiperlink	22
2.1.1.4.1.2 Análise de Redes Sociais (ARS)	23
2.1.2 Extração da Informação (EI)	23
2.1.3 Generalização	24
2.1.4 Análise	25
2.2 Classificação	26
2.2.1 Mineração de Conteúdo	26
3.1 Linguagem de Desenvolvimento	29
4. Resultados	47
5. Conclusões	55
5.1 Trabalhos futuros	57
Referências	59

1. Introdução

Este trabalho é um estudo de caso sobre Recuperação, Extração e Pré-processamento da Informação em *sites* de relacionamento.

Conforme visto em Loh (2001), o problema conhecido como “sobrecarga de informações” (*information overload*) ocorre quando o usuário tem muita informação ao seu alcance, mas não tem condições de tratá-la ou de encontrar o que realmente deseja ou lhe interessa.

Para tentar minimizar este problema, surgiu a Mineração de Dados, que segundo Viglioni (2007) é o processo de varrer grandes bases de dados à procura de tendências e padrões, tais como: regras de associação, seqüências temporais e dados para classificação de itens ou agrupamento.

Neste contexto temos a Mineração na *Web*, que pode ser definida como uma extensão das técnicas de Mineração de Dados para descoberta e análise de informação útil originada na *Web*.

Entre os serviços *Web* existentes temos os *sites* de relacionamento, que são redes de comunicação interativa que usam a Internet como espaço de encontro entre os usuários. São portais onde os membros apresentam seus perfis pessoais descritivos, estabelecem publicamente ligações de amizade mútua e formam comunidades a partir de interesses comuns, onde trocam experiências e informações entre si, criando uma navegável rede de relações sociais *online*.

O serviço escolhido para o estudo de caso foi o Orkut, que é um *site* de relacionamento filiado ao Google, criado em 24 de Janeiro de 2004 com o objetivo de ajudar seus membros a criar novas amizades e manter relacionamentos. (ORKUT, 2007a)

Diante disso, o trabalho propõe o desenvolvimento de uma ferramenta para busca, extração e armazenamento de informações do Orkut.

Durante o texto é usado o termo *Web* para fazer referência à *World Wide Web*, que significa "rede de alcance mundial".

A *Web* não é sinônimo de internet, e sim um sistema de documentos em hipermídia que são interligados e executados nesta.

A internet é o conjunto de diversas redes de computadores que se comunicam através dos protocolos TCP/IP.

1.1 Motivação

O conteúdo da *Web* consiste de dados não-estruturados do tipo textos, de dados semi-estruturados do tipo documentos HTML (*HyperText Markup Language*) e dados estruturados tais como dados contidos em bancos de dados acessados pelas páginas.

Sua estrutura e linguagem são as mais variadas possíveis. Tal complexidade e a falta de organização formal levam à necessidade de ferramentas que busquem informações relevantes, de forma inteligente e automática, e as transformem em conhecimento útil.

Surge como alternativa a Mineração na *Web*, que conforme Pal (2000 apud LEANDRO, 2007) pode ser vista como a utilização de técnicas de Mineração de Dados para a recuperação automática, extração e avaliação de informação para a descoberta de conhecimento em documentos e serviços da *Web*.

O fato dos documentos que compõem a *Web* não possuírem uma estrutura padrão dificulta a extração de informações na mesma, problema este que não ocorre se utilizarmos como fonte de um serviço *online* específico que possua páginas padronizadas.

Os *sites* de relacionamento, que nada mais são do que uma rede social virtual, tornam-se um campo fértil para Mineração de Dados por concentrar em suas páginas informações e hábitos pessoais de milhões de pessoas, e principalmente por facilitar a descoberta de padrões e elementos-chave através das comunidades.

Outro fator que torna interessante a extração de informações de um *site* de relacionamento é o fato dos membros encontrarem-se ligados direta ou indiretamente

entre si por laços de amizade representados por *links*. Isto permitiria que estes dados fossem aproveitados na abordagem de Análise de Redes Sociais (ARS), a qual utiliza técnicas que se baseiam na teoria dos grafos para analisar comunidades e relações entre as pessoas.

O *site* de relacionamento escolhido para o trabalho foi o Orkut, pois segundo (ORKUT, 2007b), o Brasil é o país com o maior número de membros. Cerca de 53,31% dos usuários do sistema, declaram-se brasileiros. Na verdade, esse número não apresenta muita exatidão, já que muitos membros criam mais de um perfil por usuário ou declaram residir em outros países.

“O sistema possui atualmente mais de sessenta milhões (68.182.265 em 20 de agosto de 2007) de usuários cadastrados.” (ORKUT, 2007a).

De acordo com Alexa (2008), que registra a popularidade dos *sites* em todo o mundo, o Orkut é o endereço mais acessado pelos brasileiros.

Um exemplo interessante para aplicação deste trabalho, e que serviu inicialmente como motivação para realização do mesmo, seria auxiliar na investigação de crimes virtuais no Orkut. Pois conforme notícias publicadas na internet houve um crescimento no Brasil dos crimes cometidos via internet, especialmente os de pedofilia no Orkut.

A idéia seminal do trabalho era rastrear comunidades e usuários suspeitos de estarem relacionados a crimes de pedofilia no Orkut, mas por diversos fatores, como o tempo, foi necessário reduzir seu escopo transformando-o em um estudo de caso.

Em toda *Web*, páginas são atualizadas, incluídas ou retiradas a todo instante. Da mesma forma membros entram e saem do Orkut, e comunidades são criadas e removidas. Buscas manuais sobre os campos do *site* teriam que ser realizadas exaustivamente utilizando os mesmos termos, para que novas comunidades e integrantes estivessem incluídos nas mesmas.

“Em média, a cada 8 dias, 1 milhão de novos usuários ingressam no Orkut por meio de convites enviados por *e-mail* ou criando uma conta no Google.” (ORKUT, 2007a).

Tais fatores fortalecem a necessidade do uso de ferramentas para tratamento automatizado dessas informações para qualquer área de interesse da pesquisa.

Durante a pesquisa foram encontradas duas ferramentas que se assemelham ao trabalho aqui descrito, as quais têm suas características citadas a seguir.

Vizster é uma ferramenta interativa de código aberto para visualização de redes sociais *online* que possibilita explorar as ligações entre os membros da rede. Seu banco de dados é composto por mais de 1,5 milhões de perfis do friendster.com¹, os quais foram extraídos durante o período de outubro de 2003 a fevereiro de 2004. O rastreamento foi feito a partir de três contas distintas explorando os laços de amizade. Sendo assim, a rede é centrada em um único indivíduo e suas ligações diretas. Ligações entre seus amigos também são mostradas, ajudando a definir as várias comunidades em que a pessoa central é membro. O Vizster ainda suporta funcionalidades de pesquisa exploratória em cima dos atributos dos perfis extraídos. (JEFFREY, 2008)

Diferentemente do Vizster este trabalho utiliza informações do Orkut e estas são encontradas a partir da busca por comunidades através de palavras-chave. Estes dados não estão relacionados ao perfil, mas sim às ligações de amizade e às comunidades onde foram encontrados os usuários, bem como a verificação das comunidades relacionadas. Para tanto foi utilizado apenas um par usuário/senha criado exclusivamente para este fim.

Outra ferramenta encontrada foi o Web Scraper Plus +, um produto que extrai dados da *Web* e coloca-os em uma planilha Excel, ou banco de dados, incluindo suporte para criação automática de tabelas em Access, Microsoft SQL e MySQL. É capaz de efetuar automaticamente o *login* em um *site* seguro, submissão em formulários de pesquisa e rastreamento de resultados. O programa encontra-se na versão 5.0.17 que foi liberada em 17/05/2006, disponível para Windows 98/ME/ 2000/XP. Encontra-se disponível também uma versão para testes válida para 30 dias. (WEB, 2008)

A ferramenta desenvolvida possui as mesmas funcionalidades do Web Scraper Plus +, exceto pelo fato das informações extraídas serem colocadas exclusivamente em tabelas do banco de dados MySQL e de utilizar somente o Orkut como fonte. Em contrapartida, tem as vantagens de ser uma aplicação gratuita, multiplataforma e portátil, enquanto o produto citado é comercializável e pode ser utilizado apenas em

¹ Friendster é um site de relacionamento fundado em 2002 por Jonathan Abrams em Mountain View, California e foi o site pioneiro no gênero, que tem como exemplos mais famosos o hi5, Orkut e o MySpace.

algumas versões do sistema operacional Windows.

Sendo assim, o trabalho propõe algoritmos para busca, extração e armazenamento de informações sobre comunidades do Orkut, possivelmente criadas por brasileiros, e seus respectivos membros.

1.2 Objetivos

O trabalho proposto tem como objetivo o desenvolvimento de uma ferramenta para pesquisa e extração de informações no Orkut, armazenando informações sobre comunidades possivelmente criadas por brasileiros e seus membros.

A cada pesquisa feita e comunidade encontrada são armazenadas as seguintes informações sobre:

- As comunidades encontradas: nome, URL(*Universal Resource Locator*), membros e comunidades relacionadas. Identificando entre os membros o dono da comunidade, quando este estiver especificado na página.
- As comunidades relacionadas: nome e URL.
- Os membros: nome, URL e amigos.
- Os amigos: nome e URL.

Para iniciar a execução do programa é solicitado ao usuário o fornecimento de suas credenciais do Orkut para que a aplicação possa conectar-se ao *site*, o que é indispensável para ter acesso às páginas.

A pesquisa por comunidades que possivelmente contêm informações de interesse é feita a partir de palavras-chave ou combinações de palavras, em português do Brasil, através do serviço de pesquisa do próprio *site*.

O termo de busca utilizado está embutido no código, sendo dispensada a inserção manual a cada execução do programa.

O objetivo é armazenar as informações extraídas em um banco de dados projetado de maneira a impedir a existência de dados redundantes, mas sendo possível identificar perfeitamente as relações entre usuários e comunidades, e seus respectivos papéis na pesquisa de interesse.

Ao iniciar a execução do programa é verificado se o banco de dados não está vazio, pois se contiver informações provenientes de pesquisas anteriores, estas

deverão ser apagadas antes mesmo da requisição de autenticação. Isto é necessário devido ao tempo que exigiria para uma ampliação do trabalho visando confrontar as informações armazenadas com as atuais. Pois poderia haver novos usuários ou comunidades, assim como novas relações terem sido estabelecidas. Ou nos piores casos, algumas comunidades ou usuários poderiam deixar de existir, como ligações poderiam ter sido desfeitas.

1.3 Organização do trabalho

No capítulo 2 trata de Mineração na *Web*. Inicialmente são feitas algumas considerações iniciais sobre área como, por exemplo, distinções entre esta e a Mineração de Dados tradicional. São descritas suas etapas, bem como alguns recursos que necessitam ou fazem parte das técnicas utilizadas nas mesmas, e suas categorias de acordo com os dados de interesse para mineração.

O capítulo 3 primeiramente cita e descreve de maneira breve, os recursos utilizados para desenvolvimento da ferramenta. Apresenta uma descrição detalhada do funcionamento geral do programa, abordando também as classes e métodos desenvolvidos, bem como os campos e relações entre as tabelas de banco de dados utilizadas. Descrevendo a necessidade de cada uma destas estruturas.

A metodologia adotada para validação do trabalho, e a apresentação dos resultados obtidos nesta, são relatados no capítulo 4.

Por fim, no capítulo 5 são abordadas as conclusões do trabalho e sugestões de trabalhos futuros.

2. Mineração na *Web*

Documentos são dados não estruturados, em oposição aos dados rigidamente estruturados que são manipulados em bancos de dados. Entende-se por dados estruturados, aqueles que são armazenados em estruturas bem definidas (esquemas), como é o caso dos bancos de dados relacionais, onde um esquema composto por nomes de tabelas e atributos (em cada tabela) é especificado e armazenado logo na criação de qualquer banco de dados. (SANDRA, 2007)

Utilizar e compreender os dados disponíveis na *Web* não é uma tarefa simples, pois seus dados são muito mais sofisticados e dinâmicos do que os sistemas de armazenamento de bancos de dados tradicionais. Enquanto estes últimos utilizam estruturas de armazenamento bem definidas e estruturadas, a *Web* não possui qualquer controle sobre a estrutura ou o tipo dos documentos que virtualmente armazena. (LEANDRO, 2007)

Outro aspecto que diferencia a Mineração de Dados tradicional da Mineração na *Web* é a existência de vínculos de hipertexto entre os seus documentos. Os vínculos de hipertexto são uma rica fonte de informações a ser explorada, pois dentre outras coisas, ajudam no processo de ranqueamento de páginas pelos motores de busca e na identificação de micro-comunidades na *Web*. (LEANDRO, 2007)

2.1 Etapas

De acordo com alguns autores como Etizone (1996 apud LEANDRO, 2007) e Sandra (2007), as principais tarefas da Mineração na *Web* representadas na Fig. 1 são: Recuperação de Informação, Extração da informação, Generalização e Análise.



Figura 1 - Etapas da Mineração na Web.
Fonte: LEANDRO, 2007.

2.1.1 Recuperação de Informação (RI)

A recuperação de informação trata da automatização do processo de recuperação de documentos relevantes, que inclui, principalmente, representação, indexação e busca por documentos. (LEANDRO, 2007)

De acordo com Markov (2007), técnicas de RI são aplicadas no desenvolvimento de motores de busca na Web, bem como na organização de páginas Web em diretórios de tópicos.

2.1.1.1 Motores de busca

Sites de busca exploram a estrutura livre da Web e tentam encontrar documentos que correspondam aos critérios de pesquisa do usuário. A idéia básica é usar um conjunto de palavras especificadas pelo usuário e recuperar documentos que incluem essas palavras. Esta é a abordagem de pesquisa por palavra-chave, bem conhecida na área de RI. (MARKOV, 2007)

Segundo Markov (2007), após recuperar um conjunto de documentos classificados por seu grau de correspondência com as palavras-chave pesquisadas, estes são classificados por importância (popularidade, Autoridade), geralmente

baseando-se na estrutura de *links* da *Web*.

2.1.1.2 Diretórios de Tópicos

As páginas da *Web* são organizadas em estruturas hierárquicas que refletem seus significados. Estes são conhecidos como diretórios de tópicos, ou simplesmente diretórios, e estão disponíveis em quase todos os portais de pesquisa na *Web*. (MARKOV, 2007)

Conforme visto em Markov (2007), estruturas de diretórios são freqüentemente utilizadas no processo de pesquisa na *Web* para melhor corresponder aos critérios do usuário ou especializar uma pesquisa dentro de um determinado conjunto de páginas de uma categoria específica.

As listas são normalmente criadas manualmente com a ajuda de milhares de criadores e editores de páginas *Web*, mas existem também abordagens para fazê-lo automaticamente através da aplicação de métodos de aprendizagem de máquina para classificação e agrupamento. (MARKOV, 2007)

Segundo Markov (2007), a coleta e armazenamento de documentos da *Web*, assim como sua indexação de acordo com as palavras que contêm, são de responsabilidade dos rastreadores *Web*. Sendo assim, estes são utilizados pelos motores de busca e na construção de diretórios de tópicos.

2.1.1.3 Rastreadores *Web*

Conforme Markov (2007), a coleta de documentos pode ser feita navegando-se na *Web* sistemática e exaustivamente, armazenando todas as páginas visitadas. Esta tarefa é feita por rastreadores (também chamados *Spiders* ou *robots*).

Idealmente o gráfico da *Web* seria uma árvore, ou seja, todas as páginas *Web* estariam interligadas, possuindo n páginas para $n-1$ *links* e existiria apenas um caminho entre duas páginas. Desta forma o trabalho de um rastreador seria simples, bastaria executar um algoritmo de busca em grafos, pela busca em profundidade ou pela busca em largura, armazenando todas as páginas visitadas. (MARKOV, 2007)

Segundo Markov (2007), um rastreador serve para visualizar e analisar a estrutura do grafo da *Web*. Como queremos estudar a estrutura da *Web* localmente, temos que impor alguns limites no rastreamento.

Tendo em vista a estrutura da *Web* pode-se limitar a profundidade de rastreamento pelo número de *links* a acompanhar ou pelo tamanho das páginas a serem obtidas. A região da *Web* a ser rastreada também podem ser especificada usando a estrutura de URLs. Assim, para todas as URLs com o mesmo nome de servidor o rastreamento se daria apenas dentro do limite de um servidor de páginas específico, enquanto que para todas as URLs com o mesmo prefixo de pasta como limite seriam rastreadas somente as páginas que estão armazenadas em suas subpastas. (MARKOV, 2007)

Outros limites são dinâmicos e refletem o tempo necessário para buscar uma página ou o tempo de execução do rastreador. Vários outros limites e restrições com relação ao conteúdo da página da *Web* também podem ser impostos. (MARKOV, 2007)

De acordo com Markov (2007), existe uma estrutura interessante chamada *hub*, que é a página no meio de um círculo de várias páginas. Uma página *hub* inclui um grande número de ligações e é geralmente algum tipo de diretório ou *site* de referência que aponta para muitas páginas *Web*.

Como a *Web* não é uma árvore, em geral, há mais de um *inlink*¹ a uma página. Na verdade, estes *inlinks* são muito importantes no momento de analisar a estrutura da *Web*, porque eles podem ser usados como uma medida de popularidade ou importância de uma página. Similar ao *hubs*, uma página da *Web* com um grande número de *inlinks* também é importante e é chamado de uma autoridade. (MARKOV, 2007)

Embora visualizar o grafo da *Web* seja um bom recurso de rastreadores, não é o mais importante. O papel fundamental de um rastreador que faz parte de um motor de busca é coletar informações sobre as páginas da *Web*, tais como conteúdo textual, títulos da página, cabeçalhos, estrutura de *tags*, ou estrutura de *links*. Estas informações são devidamente organizadas para um acesso eficaz e são armazenadas em um repositório local para serem usadas para indexação e pesquisa. (MARKOV, 2007)

¹ ocorrências da URL da página em outras páginas da *Web*.

Sendo assim, um rastreador não é apenas uma aplicação de um algoritmo de busca em grafo, mas também um *parser* e analisador HTML. (MARKOV, 2007)

Até agora, foi descrito o rastreamento baseado na sintaxe do grafo da *Web*, isto é, seguindo *links* e visitando páginas sem ter em conta a sua semântica. Para melhorar a eficiência da pesquisa na *Web*, ou para fins específicos, o rastreamento também pode ser feito como uma pesquisa guiada. (MARKOV, 2007)

2.1.1.4 Indexação e pesquisa por palavra-chave

Geralmente, há dois tipos de dados: estruturadas e não estruturadas. Dados estruturados têm chaves (atributos) associados a cada item de dados que refletem o seu conteúdo, significado, ou uso. Um exemplo típico de dados estruturados é uma tabela em um Banco de Dados Relacional. (MARKOV, 2007)

Dado nome e valor de um atributo (coluna), podemos obter um conjunto de tuplas (Linhas) que incluem esse valor com uma simples consulta. Em SQL (*Structured Query Language*) este tipo de consulta é expressa como **select * from** tabela **where** nome_do_atributo = valor_do_atributo. No entanto, para obter a mesma informação especificada como texto em parágrafos que descrevem o conteúdo de cada tabela, seria mais difícil e geralmente exigiria pessoas para ler e compreender o texto.

Segundo Markov (2007), o problema com a utilização de dados não estruturados é o custo associado ao processo de estruturação dos mesmos, pois a maior parte dos documentos são textos escritos em linguagem natural.

O processo que envolve o acesso ao conteúdo de base destes documentos e organização dos mesmos toma muito tempo e esforço por ser feito manualmente. Existem tentativas de uso computadores para este fim, mas o problema é que o conteúdo baseado em acesso assume compreensão do significado dos documentos, o que ainda é uma questão de investigação, estudo na área da Inteligência Artificial (IA) e Processamento de Linguagem Natural (PLN) em particular. (MARKOV, 2007)

De acordo com Markov (2007), existe uma solução que evita o problema do significado, mas ainda fornece alguns tipos de conteúdo baseado em acesso a dados não estruturados, que é a pesquisa por palavra-chave. A idéia é recuperar documentos, utilizando um simples critério booleano: a presença ou ausência de palavras específicas

nos documentos. Palavras-chave podem ser combinadas em disjunções e conjunções, dando assim mais expressividade às consultas.

No contexto de indexação, um índice é basicamente uma coleção de termos retirados dos documentos com ponteiros para os lugares onde as informações sobre os documentos podem ser encontradas. (PAL, 2000 apud LEANDRO, 2007)

A indexação de documentos na *Web* pode ser humana ou manual ou automática (PAL, 2000 apud LEANDRO, 2007), e está baseada nos modelos tradicionais de recuperação de informação: espaço vetorial, estatístico e lingüístico. (BAEZA-YATES, 1999; GIRARDI, 1998; SALTON, 1983 apud LEANDRO, 2007)

A indexação é essencialmente um processo de classificação onde é realizada uma análise conceitual do documento ou elemento de informação. Por exemplo, nas técnicas baseadas no modelo do espaço vetorial, a indexação envolve a atribuição de elementos de informação a certas classes, onde uma classe é o conjunto de todos os elementos de informação para o qual um termo de indexação (ou palavra-chave), em particular, tem sido atribuído. Os elementos de informação podem fazer parte de várias classes. Algumas técnicas atribuem pesos aos termos de indexação de um elemento de informação de forma a refletir sua relativa relevância (GIRARDI, 1998 apud LEANDRO, 2007). Nas técnicas baseadas no modelo estatístico, os termos de indexação são extraídos a partir de uma análise de frequência das palavras ou frases em cada documento e em toda a fonte de informação. Nas técnicas lingüísticas, os termos de indexação são extraídos utilizando técnicas de processamento da linguagem natural, por exemplo, análise morfológica, lexical, sintática e semântica. (GIRARDI, 1995 apud LEANDRO, 2007)

Uma consulta com base em palavra-chave retorna normalmente um grande número de documentos, não sendo possível identificar exclusivamente os documentos que a correspondem. Portanto, em RI há necessidade de se classificar documentos por sua relevância à consulta. (MARKOV, 2007)

2.1.1.4.1 Relevance Ranking

A pesquisa booleana por palavra-chave é simples e eficiente, mas retorna uma coleção desordenada de documentos. O tamanho médio de uma pesquisa na *Web* é de

dois termos. Obviamente, uma consulta deste tamanho não pode especificar precisamente as informações que o usuário procura, resultando em um grande conjunto de respostas que podem ou não corresponder às informações de interesse. (MARKOV, 2007)

Conforme visto em Markov (2007), a solução é classificar documentos em um conjunto de respostas por relevância à consulta e apresentar ao usuário uma lista ordenada, a qual inicia com os documentos do topo do *ranking*.

Embora nenhuma das técnicas existentes para avaliação da relevância de documentos tenha sido diretamente utilizada para desenvolvimento da ferramenta, é brevemente descrito o ranking baseado em hyperlink por ser o único critério visivelmente utilizado pelo motor de busca disponível no Orkut.

2.1.1.4.1.1 Ranking baseado em hyperlink

Segundo Markov (2007), papel dos *links* das páginas *Web* é semelhante ao papel de citações na literatura científica, por exemplo. Artigos populares são freqüentemente citados. Muitos *hiperlinks* apontando para uma página chamam a atenção dos usuários da *Web*, assim como citações de um artigo chamam a atenção de universitários.

As medidas de popularidade, autoridade e prestígio podem ser usadas para o *ranking* de páginas *Web* recuperadas por um motor de busca. A idéia é a de atribuir a cada página na *Web* uma classificação baseada na estrutura de *hyperlink*. (MARKOV, 2007)

Ao visualizar uma página de resultados do Orkut percebe-se que as comunidades retornadas apresentam-se de forma decrescente de acordo com o número de membros. No contexto de ranking por hyperlink, tal fato justifica-se por haver uma referência a estas URLs para cada membro que estas comunidades possuem. Isto porque para cada usuário do *site* existem páginas que exibem os *links* de todas as comunidades a que pertence.

2.1.1.4.1.2 Análise de Redes Sociais (ARS)

Uma rede social pode ser representada formalmente por um grafo dirigido com pesos atribuídos às arestas. Podemos supor que os nós representam documentos e as arestas representam citações de um documento a outros documentos. Neste cenário o conceito de prestígio pode ser associado com o número de arestas de entrada de um nó. Um pressuposto evidente em qualquer rede social é que o prestígio depende da autoridade das citações. Em outras palavras, o prestígio tem um caráter recursivo. Assim, a pontuação de prestígio de um nó não é simplesmente igual ao seu grau de entrada, mas tem que ser definido recursivamente usando as pontuações de prestígio dos nós que o citam. Isto é feito utilizando algumas noções de álgebra linear. (MARKOV, 2007)

2.1.2 Extração da Informação (EI)

Uma vez tendo sido os documentos recuperados, o próximo passo é transformar ou pré-processar esses documentos de forma que algoritmos de Mineração de Dados e aprendizagem de máquina possam ser aplicados de forma efetiva. (LEANDRO, 2007)

Sistemas de Extração de Informação atuam sobre um conjunto de dados não estruturados e objetivam localizar informações específicas em um documento ou coleção de documentos, extraí-las e estruturá-las com o intuito de facilitar o uso dessas informações. (ÁLVAREZ, 2007)

Segundo Pal (2002 apud Leandro, 2007) e Cordeiro (2003) as técnicas de Extração de Informação buscam derivar conhecimento de documentos recuperados, os quais já sabemos que contêm as informações de interesse, segundo a forma como um documento está estruturado e representado. Os elementos relevantes extraídos serão depois armazenados em alguma estrutura previamente definida, por exemplos numa tabela de uma Base de Dados.

De acordo com Álvarez (2007), os textos ou documentos dos quais são extraídas as informações de interesse podem apresentar algum nível de estruturação na apresentação dos dados, como também podem ser totalmente livres. O tipo de texto de onde é feita a extração tem grande influência sobre a escolha da técnica a ser utilizada

na construção de sistemas de EI, pois esta pode se basear apenas na estrutura do texto quando existente.

Estes métodos geralmente envolvem a escrita de código específico, e são popularmente chamados de *wrappers*, são responsáveis pelo mapeamento de um documento para algum modelo de representação do conhecimento. O problema é que para cada documento da *Web* temos que escrever um código específico, tornando o trabalho manual. Como os documentos da *Web* não possuem uma semântica agregada às informações que contém, e nem mesmo um padrão de como apresentar essas informações ao usuário, temos que aprender acerca da estrutura individual de cada documento e escrever um código para essa estrutura em particular. Daí a dificuldade de estendermos ou generalizarmos esse código para outros documentos. (LEANDRO, 2007)

No contexto da *Web*, o propósito de um *wrapper* é converter informações implícitas armazenadas em páginas HTML em informações explícitas estruturadas, para posterior processamento. (ÁLVAREZ, 2007)

2.1.3 Generalização

É nesta etapa que são utilizadas técnicas de Mineração de Dados e aprendizagem de máquina para descobrir novo conhecimento a partir do que já existe (LEANDRO, 2007).

O maior problema em aprender ou descobrir novos conhecimentos da *Web* é a falta de marcação semântica das informações. Muitos algoritmos de Mineração de Dados requerem como entrada exemplos positivos ou negativos de algum conceito. Se, por exemplo, tivéssemos um conjunto de páginas da *Web* marcadas como exemplos positivos e outras como negativos do conceito portal, seria fácil modelar um algoritmo classificatório para a classificação automática de novas páginas como portais ou não portais.

Agrupamento ou *clustering* é uma técnica de classificação que não requer entradas com marcação semântica, e por isso tem sido aplicada com sucesso em grandes conjuntos de documentos HTML (CUTTING, 1992 apud LEANDRO, 2007). No agrupamento, documentos são agrupados de acordo com a sua similaridade, portanto,

um novo documento é classificado de acordo com a sua similaridade com algum conjunto de documentos existente. (LEANDRO, 2007)

As Regras de associação também podem ser utilizadas nessa fase do processo. Regras de associação são basicamente expressões do tipo $X \Rightarrow Y$ onde X e Y são conjuntos de itens. $X \Rightarrow Y$ expressa que toda vez que uma transação T contiver X então ela provavelmente também conterá Y. A probabilidade ou confiança da regra é a porcentagem de transações contendo Y junto a X comparado ao total de transações contendo X. A idéia de minerar regras de associação se origina nos dados de supermercados e afins onde regras como “O cliente que compra o produto x também comprará o produto y com probabilidade (confiança) de c%”. (PAL, 2000 apud LEANDRO, 2007)

2.1.4 Análise

Uma vez os padrões tendo sido descobertos os analistas precisam de técnicas e ferramentas apropriadas de modo a entender, visualizar, interpretar e validar esses padrões. (LEANDRO, 2007)

Leandro (2007) cita alguns exemplos de ferramentas utilizadas para este fim.

O foco deste trabalho são as etapas de Recuperação de Informação e Extração da Informação. Como não foram diretamente utilizadas técnicas destas áreas recém descritas para desenvolvimento da ferramenta, optou-se por não abordá-las com detalhamento.

2.2 Classificação

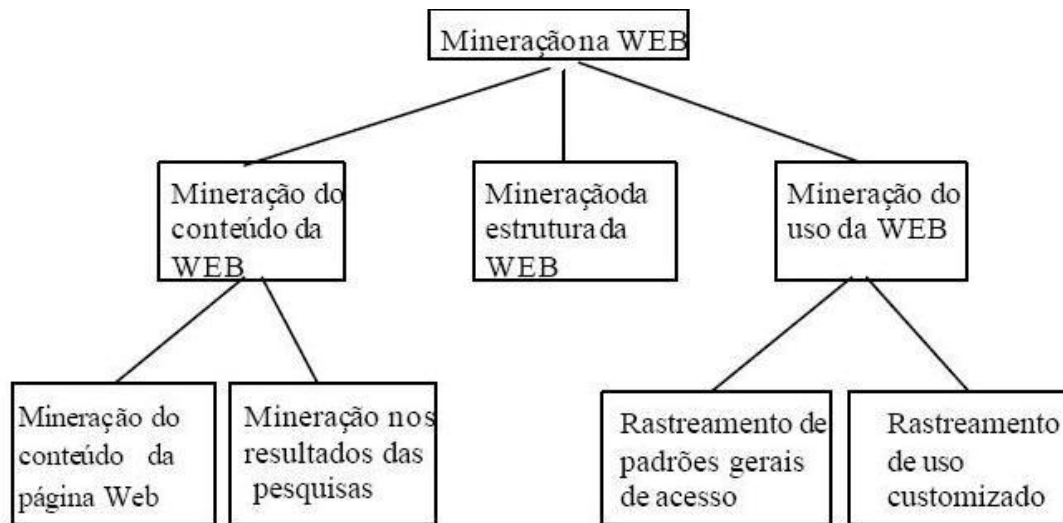


Figura 2 - Classificação da Mineração na Web.
Fonte: CAZELLA, 2001.

De acordo com Leandro (2007), Sandra (2007) e Cazella (2001), a Mineração na Web é subdividida em três categorias principais, as quais constituem as áreas de interesse onde minerar informação: Mineração do Conteúdo, Mineração de Uso ou Mineração da Estrutura.

2.2.1 Mineração de Conteúdo

Há basicamente duas estratégias para a Mineração de Conteúdo: uma realiza a mineração diretamente do conteúdo dos documentos e a outra incrementa o poder de busca de outras ferramentas e serviços. Na primeira estratégia, os documentos pretendidos já foram recuperados e já estão prontos para serem minerados. Na segunda estratégia, a Mineração de Conteúdo presta um grande “favor” às ferramentas e serviços de recuperação de informação, pois ajuda a realizar o processo de indexação e categorização dos documentos. Dessa forma, percebemos que quando a Mineração de Conteúdo utiliza a segunda estratégia, ela complementa o processo de recuperação de informação, sendo utilizada como uma ferramenta pelos motores e serviços de busca. (LEANDRO, 2007)

De acordo com Leandro (2007), a Mineração de Conteúdo pode seguir duas abordagens: baseada em agentes ou baseada em bancos de dados. A abordagem baseada em agentes envolve o desenvolvimento de sistemas de inteligência artificial que podem agir de forma autônoma ou semi-autônoma para a descoberta e organização de informações da *Web* de acordo com os interesses de um usuário em particular. (KOSALA, 2000 apud LEANDRO, 2007)

A abordagem de bancos de dados focaliza-se nas técnicas para transformar os dados semi-estruturados ou desestruturados da *Web* em modelos de dados estruturados onde mecanismos de consulta, como por exemplo a linguagem SQL, possam ser utilizados, assim como técnicas de Mineração de Dados para a análise. (SANDRA;LEANDRO, 2007)

Existem, portanto, duas grandes preocupações ou orientações de investigação neste sub-domínio: “Recuperação de Informação” e “Extração de Informação” (CAZELLA, 2001)

2.2.2 Mineração de Estrutura

Além dos conteúdos disponibilizados pelas páginas da *Web*, existe um conjunto de informação subjacente às mesmas que, para determinados fins, se reveste de uma grande importância. Este subdomínio da Mineração da *Web* foca, essencialmente, questões relacionadas com a estrutura topológica da *Web*. (CORDEIRO, 2003)

Enquanto que na Mineração de Conteúdo da *Web* estamos interessados no que há dentro dos documentos, na Mineração de Estrutura o interesse está nas informações que existem de forma implícita entre os documentos. Esta categoria envolve a Mineração da Estrutura que há por trás da interligação entre os documentos da *Web*. O que liga esses documentos são os vínculos de *hipertexto*, os quais são os principais objetos de estudo nesta categoria. (LEANDRO, 2007)

Como as páginas publicadas na *Web* são hipertextos, cada uma destas poderá conter várias ligações para outras páginas e uma determinada página poderá ser referenciada por outras. Podemos então modelar este universo por uma estrutura a que os matemáticos denominam de grafo, ou mais especificamente um grafo orientado.

(LEANDRO, 2007)

Conforme citado anteriormente a *Web* pode ser visualizada como um grafo orientado, apresentando uma forte semelhança com as chamadas redes sociais e que juntamente com a análise de citações, inspirou a pesquisa dessa categoria de mineração. (KUMAR, 2002 apud LEANDRO, 2007)

2.2.3 Mineração de Uso

Este sub-domínio tem como alvo descobrir os padrões comportamentais de determinados utilizadores, no que diz respeito à consulta de conteúdos na *Web*. Sabe-se que a maioria dos servidores que se encontra na Internet faz um registro dos acessos realizados aos mesmos. (CORDEIRO, 2003)

Enquanto a Mineração de Conteúdo e a Mineração de Estrutura utilizam os dados reais ou primários da *Web*, a Mineração de Uso lida com os dados secundários provenientes da interação do usuário com a *Web*. Estes incluem dados contidos em *logs* de servidores *Web*, *logs* de servidores *proxy*, *logs* de browsers, perfis de usuário, *cookies*, seções ou transações de usuários, pasta favoritos, consultas do usuário, cliques de mouse e qualquer outro dado gerado pela interação do usuário com a *Web*. (LEANDRO, 2007)

Segundo Leandro (2007), as aplicações da Mineração de Uso da *Web* podem ser classificadas em duas categorias principais: aprendizado de perfil de usuário ou modelagem em interfaces adaptativas (personalização) e aprendizado de padrões de navegação de usuário.

A Mineração de Uso da *Web* despertou interesse especial no comércio eletrônico, principalmente pela sua necessidade de aprender acerca do comportamento dos clientes, perfis de compra, preferências e padrões de navegação. Alguns *sites* populares de comércio eletrônico já utilizam estas técnicas não só para a adaptação do *site* de acordo com o perfil do usuário, mas como para fazer recomendações de produtos de acordo com compras anteriores, ou baseadas na similaridade entre perfis de usuários. (LEANDRO, 2007)

3. Ferramenta Desenvolvida

Os algoritmos foram desenvolvidos na linguagem de programação Java com o auxílio de uma biblioteca para comunicação via protocolo HTTP (*Hypertext Transfer Protocol*).

Para armazenamento das informações extraídas foram utilizadas tabelas do Banco de Dados MySQL, assim como algumas ferramentas gráficas para facilitar a criação destas e o acesso aos dados.

3.1 Linguagem de Desenvolvimento

Os algoritmos que compõem a ferramenta foram escritos em Java, pois dentre outras vantagens é uma linguagem muito bem documentada e rodeada de API's (*Application Programming Interface*) gratuitas para os mais variados propósitos.

De acordo com Linguagem (2008), Java é uma linguagem de programação orientada a objetos desenvolvida pela Sun Microsystems e teve sua especificação concluída em 1991. Inicialmente elaborada para ser a linguagem-base de projetos de *software* para produtos eletrônicos, Java teve sua grande expansão em 1995, devido ao sucesso mundial da *Web*.

As principais características da linguagem de acordo com Java (2008) são:

- Orientação a objetos;
- Portabilidade - Independência de plataforma;
- Recursos de Rede - Possui extensa biblioteca de rotinas que facilitam a cooperação com protocolos TCP/IP, como HTTP e FTP (*File Transfer Protocol*);

- Segurança - Pode executar programas via rede com restrições de execução;
- *Bytecode* interpretado, ao invés de compilado.

3.1.2 Máquina Virtual Java

Programas Java não são traduzidos para a linguagem de máquina como outras linguagens estaticamente compiladas e sim para uma representação intermediária chamada de *bytecodes*. Estes *bytecodes* são interpretados pela JVM (*Java Virtual Machine*).

3.2 Ambiente de Desenvolvimento Integrado

O Eclipse foi a IDE escolhida para desenvolvimento da ferramenta, pois além de ser um *software* gratuito de código aberto disponível para desenvolvimento na linguagem Java, possui um amplo suporte ao desenvolvedor com uma vasta gama de *plug-ins*.

O Eclipse foi um projeto originalmente criado pela IBM em novembro de 2001, mas atualmente é controlado por uma organização sem fins lucrativos fundada em janeiro de 2004. (ECLIPSE, 2008)

3.3 Bibliotecas

Para desenvolvimento da ferramenta foi necessário utilizar uma biblioteca que possibilitasse a comunicação entre a aplicação e o servidor do Orkut através do protocolo HTTP.

3.3.1 HTTPClient

É uma biblioteca Java desenvolvida pela Apache Software Foundation para criação de *softwares* que utilizem o protocolo HTTP para comunicação. Em outros termos, é um pacote que executa o lado cliente do servidor, permitindo a utilização e

envio de informações via HTTP, criação de requisições com ou sem autenticação e manipulação de *cookies*. (HTTP, 2008)

De acordo com Jakarta (2008a), a `HttpClient` foi iniciada em 2001 como um subprojeto do Jakarta Commons. Atualmente seu desenvolvimento e manutenção são de responsabilidade do projeto `HttpComponents`.

Apesar do pacote `java.net`¹ fornecer funcionalidades básicas de acesso a recursos via HTTP, ele não prevê total flexibilidade e funcionalidades necessárias por muitos aplicativos. O componente `HttpClient` fornece um pacote que implementa do lado cliente das mais recentes normas e recomendações do protocolo HTTP. (JAKARTA, 2008a)

`HttpClient` suporta gerenciamento automático de *cookies*, permitindo que estes sejam automaticamente devolvidos para o servidor sempre que necessário. Também é possível configurar manualmente os *cookies* a serem enviados ao servidor. (JAKARTA, 2008b)

Dentre as diversas classes e interfaces que compõem a biblioteca, as utilizadas diretamente pela ferramenta desenvolvida foram:

- 1) Classe `Cookie`: Representa um *cookie*, que é a parte da informação de um estado que o agente HTTP e o servidor podem trocar com o objetivo de manter de uma sessão. (CLASS, 2008a)

O agente é quem faz a ligação ao servidor, normalmente navegador.

No método responsável pelo *login* é utilizado um vetor de objetos desta classe para armazenar os atributos de cada *cookie* obtido, como por exemplo nome e validade do *cookie*.

- 2) Classe `CookiePolicy`: Conforme (CLASS, 2008b) a classe `CookiePolicy` define as políticas de gerenciamento de *cookies* correspondente para determinado tipo ou versão de *cookie*.

São previstas as seguintes especificações:

- `BROWSER_COMPATIBILITY`: compatível com práticas comuns de gerenciamento de *cookie*.

¹ O pacote `java.net` provê as classes utilizadas para implementar aplicações de rede na linguagem Java.

- NETSCAPE: conforme especificação de *cookie* no projeto Netscape.
- RFC_2109: em conformidade com RFC2109 (*default*).
- IGNORE_COOKIES: não processar automaticamente os *cookies*. Impede o recebimento e envio de *cookies*.

No trabalho é utilizada a especificação BROWSER_COMPATIBILITY. De acordo com Class (2008b) esta especificação proporciona alto grau de compatibilidade com os gerenciamentos de *cookie* comuns de navegadores populares.

3) Interface CookieSpec: Define a especificação de gerenciamento de *cookie*.

(INTERFACE, 2008a)

Deve definir para um determinado *host*, porta e o caminho de origem:

- Regras de *parsing* do cabeçalho "Set-Cookie".
- Regras de validação dos *cookies* analisados.
- Formatação do cabeçalho "Cookie".

De acordo com Cookies (2008), os *Cookies* funcionam da seguinte forma:

- Quando o servidor deseja ativar um *cookie* no cliente, envia uma linha no cabeçalho HTTP iniciada por Set-Cookie: ...
- A partir desse momento, conforme as opções especificadas pelo *cookie*, o cliente irá enviar no seu cabeçalho HTTP dos pedidos uma linha contendo os *cookies* relevantes, iniciada por Cookie: ...

4) Classe Header: Representa um cabeçalho HTTP. (CLASS, 2008d)

5) Classe HttpClient: Representa um agente usuário (cliente) HTTP, contendo um objeto da classe HTTPState e uma ou mais instâncias da classe HttpConnection, para que possam ser aplicados os métodos da interface HTTPMethod. (CLASS, 2008e)

Neste contexto torna-se necessário descrever as seguintes classes e interface:

- A Classe `HTTPState` armazena atributos HTTP que podem persistir a partir de uma requisição, como *cookies* e credenciais para autenticação. (CLASS, 2008h)
 - A Classe `HttpConnection` é uma abstração de um par `InputStream` e `OutputStream`, que são superclasses de todas as classes que representam um fluxo de entrada ou fluxo de saída de *bytes* (CLASS, 2008j), todas juntamente com seus respectivos atributos. (CLASS, 2008f)
 - A Interface `HttpMethod` representa uma requisição a ser enviada, através de um objeto da classe `HTTPConnection`, e sua correspondente resposta. (INTERFACE, 2008b)
- 6) Classe `HttpStatus`: Possui constantes enumerando os códigos de *status* HTTP. (CLASS, 2008i)
- 7) Classe `NameValuePair`: Uma simples classe para encapsular um par nome/valor. (CLASS, 2008g)
- 8) Classe `GetMethod`: Implementa o método HTTP GET. (CLASS, 2008c)
- Tipicamente o método `get` é usado para fazer o *download* de um documento a partir de um servidor *Web*. Isto pode ser conseguido com os métodos `getResponseBody`, `getResponseBodyAsStream` ou `getResponseBodyAsString`. No caso do trabalho em questão foi utilizado o método `getResponseBodyAsString`, obtendo como retorno uma *string* com o código-fonte das páginas. (JAKARTA, 2008c)
- 9) Classe `PostMethod`: Implementa o método HTTP POST. (CLASS, 2008k)
- Há dois importantes passos para usar o método POST, em primeiro lugar fornecer os dados para a requisição e, em segundo lugar a leitura da resposta do servidor. (JAKARTA, 2008d)

Segundo Jakarta (2008d) os dados são fornecidos por requisição de uma das variantes de `setRequestBody` que tanto podem ter um `InputStream`, um vetor de

objetos `NameValuePair` ou uma *string*.

O corpo da resposta ao POST pode ser lido por qualquer um dos métodos `getResponseBody` do método GET da classe `GetMethod`. (JAKARTA, 2008d)

3.4 Banco de Dados

Um projeto de Banco de Dados foi construído visando não só armazenar os dados extraídos, como também para caracterizar os relacionamentos existentes no Orkut, ao mesmo tempo que impede a existência de dados redundantes.

A ferramenta MySQL Administrator foi utilizada para iniciar a execução do MySQL e criação das tabelas do banco de dados, bem como para visualização de estrutura e conteúdo das mesmas.

Já o MySQL Query Browser foi utilizado na etapa de validação da ferramenta para execução das consultas que são descritas no capítulo Resultados, mais precisamente em Metodologia de Validação.

3.4.1 Sistema Gerenciador de Banco de Dados (SGBD)

O MySQL é um SGBD de gratuito de código aberto que utiliza a linguagem SQL como interface. (WHY, 2008)

Segundo [Recursos \(2008\)](#), o MySQL possui as seguintes características:

- Rapidez
- Consistência
- Alta performance
- Facilidade de instalar, usar e administrar

3.4.2 Ferramentas Gráficas

No *site* do MySQL estão disponíveis gratuitamente várias ferramentas, gráficas ou não, que proporcionam maior aproveitamento dos recursos na utilização do SGBD. (FERRAMENTAS, 2008)

3.4.2.1 MySQL Administrator

MySQL Administrator é uma ferramenta visual que permite administrar mais facilmente um ambiente MySQL. (MYSQL, 2008d)

Com ele é possível executar facilmente todas as operações da linha de comandos de maneira visual, incluindo configuração de servidores, administração de usuários e monitoramento dinâmico do estado do banco de dados. Outras tarefas administrativas comuns como monitoramento do estado de replicação, *backup* e restauração, e visualização de *logs* também podem ser feitos pelo console gráfico do MySQL Administrator. (MYSQL, 2008a)

Segundo MySQL (2008a), MySQL Administrator está disponível para as plataformas Windows, Linux e Mac OS X 10.3, 10.4, e é compatível com as versões a partir do MySQL 4.0.

3.4.2.2 MySQL Query Browser

MySQL Query Browser é uma ferramenta visual para criação, execução e otimização de consultas SQL para um servidor de banco de dados MySQL. (MYSQL, 2008d)

No MySQL Query Browser é possível optar entre usar as ferramentas visuais para realizar as consultas ou manter o controle completamente manual. (MYSQL, 2008c)

3.5 Implementação

A seguir, serão descritos os passos fundamentais do algoritmo implementado pela ferramenta desenvolvida. Uma visão geral do algoritmo pode ser vista na Fig. 3.

Conforme pode ser visto na Fig. 3, após ter efetuado o *login*, o primeiro passo é recuperar as comunidades que podem conter a informação de interesse através da pesquisa por palavra-chave.

Inicialmente, todas as comunidades retornadas têm seu nome e seu código *cmm*¹ extraídos e armazenados temporariamente em uma lista.

Após todas as páginas de resultados terem sido varridas, a lista que contém as informações iniciais é percorrida e a cada comunidade obtém-se o *cmm* necessário para montar a URL da página principal da comunidade, da qual serão extraídas as informações sobre o dono e as comunidades relacionadas que formaram uma nova lista.

Posteriormente, a partir da geração da URL das páginas de membros da comunidade que está sendo analisada, uma lista é formada com nome e seu código *uid*² dos mesmos. A seguir esta é percorrida e a partir do *uid* de cada membro recupera-se uma lista de amigos.

Ao alcançar o final da lista de comunidades encontradas pelo motor de busca, terão sido extraídas todas as informações desejadas.

Finalmente, todas as informações encontradas a partir de uma comunidade pertencente à lista de resultados são salvas no BD, isto até que a lista de comunidades chegue ao fim.

¹ O *cmm* é um código identificador único para comunidades no Orkut, é a parte que diferencia a URL de uma comunidade das demais.

² O *uid* é semelhante ao *cmm*, mas diferencia a URL do perfil de um usuário das demais.

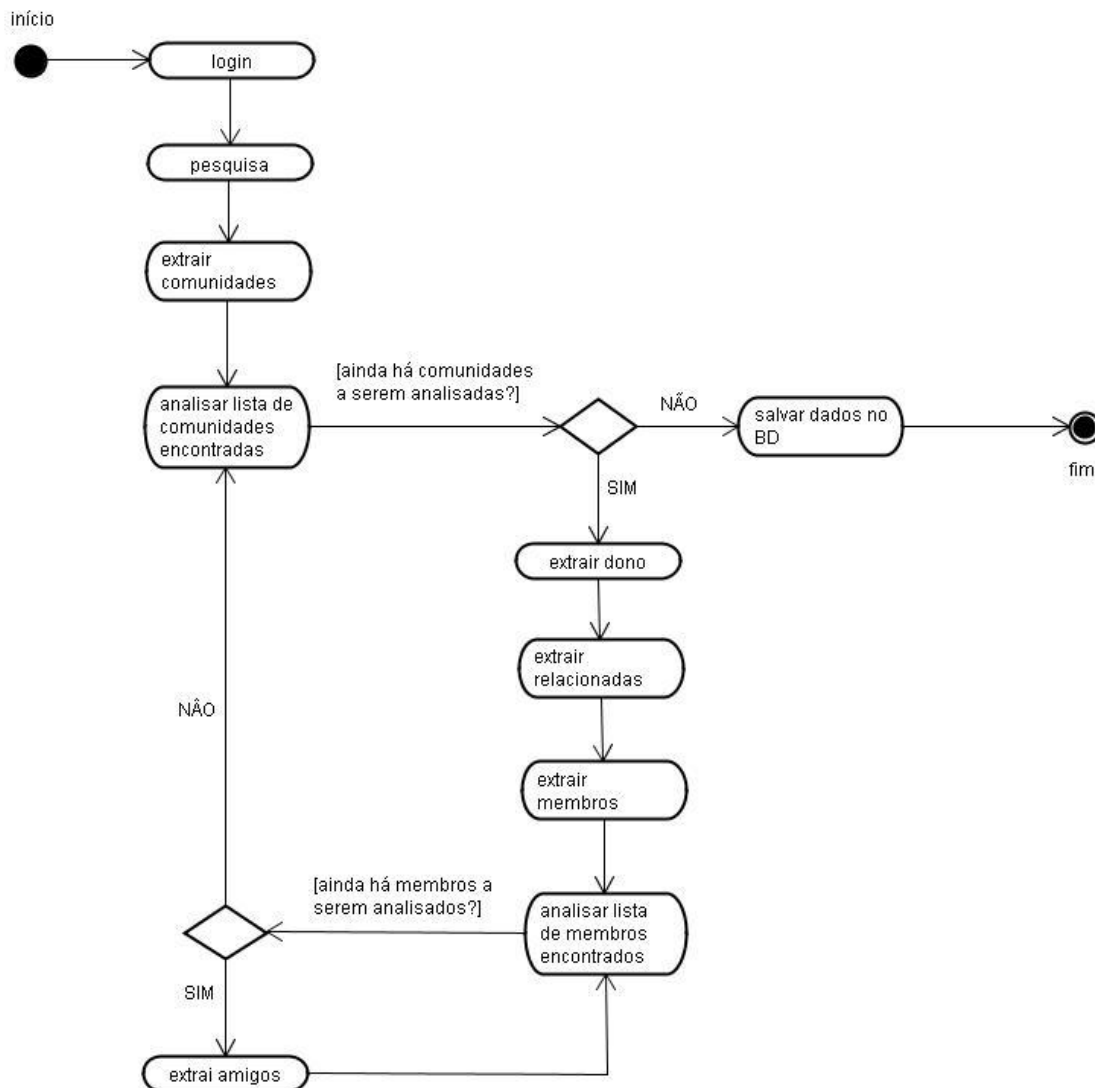


Figura 3 - Visão geral do algoritmo implementado pela ferramenta.

Sendo assim, se considerarmos o Orkut como um grafo, pode-se dizer que foi realizada uma busca em largura. Pois a cada página visitada, seja ela de resultados, membros ou amigos, as informações contidas em todos os *hiperlinks* são extraídas antes de se explorar as relações de cada um destes que levam às demais informações de interesse. Nesta abordagem, cada página é analisada completamente, uma de cada vez, e posteriormente descartada, evitando o acúmulo de dados desnecessários na memória por mais tempo do que o necessário.

A busca foi limitada por profundidade, pois só se obtém os membros das comunidades encontradas pela busca e seus amigos diretos.

3.5.1 Módulo Model

Este é o módulo principal do programa, onde estão as todas as classes responsáveis por compor a lógica da aplicação. A seguir serão descritas as quatro classes do módulo.

1) Classe Acessador

Responsável por realizar a autenticação do usuário no *site* do Orkut, o que é indispensável para que a aplicação tenha acesso ao conteúdo de suas páginas.

Embora o método *login* tenha sido baseado em exemplos para requisições com autenticação disponíveis na documentação da biblioteca HTTPClient, não seria possível efetuar a autenticação somente a partir da alteração dos parâmetros sugeridos neste.

Sendo assim, foi necessária a realização de uma pesquisa para tomar conhecimento de todas as modificações necessárias para adaptação do código às necessidades do trabalho.

Neste método um objeto da classe NameValuePair é instanciado toda vez que um par nome/valor é associado, os quais são enviados como parâmetro pelo método POST da classe PostMethod na requisição de autenticação.

O método POST foi utilizado para enviar os parâmetros necessários para realização do *login* no *site* do Orkut, os quais estão armazenados em um vetor de objetos NameValuePair.

A classe HttpStatus foi utilizada no método *login* para verificar se a requisição com autenticação foi bem sucedida, ou seja, se resultou em um redirecionamento para outra URL. Sendo assim, o código de *status* retornado foi comparado com todos códigos de redirecionamento existentes, que segundo Interface (2008,c) são:

- SC_MOVED_PERMANENTLY, correspondente ao código 301 (HTTP/1.0 - RFC 1945)
- SC_MOVED_TEMPORARILY, correspondente ao código 302

(Sometimes Found) (HTTP/1.0 - RFC 1945)

- SC_SEE_OTHER, correspondente ao código 303 (HTTP/1.1 - RFC 2616)
- SC_TEMPORARY_REDIRECT, correspondente ao código 307 (HTTP/1.1 - RFC 2616)

Já no método `pegaConteudo` o *status* esperado é SC_OK, que segundo Interface (2008,c), correspondente ao código 200 (HTTP/1.0 - RFC 1945). Caso seja retornado um *status* diferente a execução do programa é interrompida.

O método `pegaConteudo` é utilizado para recuperar os códigos-fontes HTML através do método GET da classe `GetMethod`, sendo invocado na classe `Analisador` cada vez que deseja-se obter o conteúdo de uma página.

De acordo com Códigos (2008), o código 200 significa que o servidor processou a solicitação com sucesso. Geralmente isso indica que o servidor forneceu uma página que foi solicitada, indicando no caso do trabalho que o arquivo foi recuperado com sucesso.

Entretanto não há garantias de que o código-fonte obtido é correspondente à página que esperávamos obter. Durante os testes iniciais da ferramenta ao interagir com o servidor do Orkut, foi constatado que há situações em que um redirecionamento seja efetuado a partir de um *script* intermediário, não sendo reconhecido pelo protocolo HTTP como tal, e sim como uma requisição bem sucedida (*status* 200) como foi o caso especial que exigiu a criação do método `locationReplace`.

O método `locationReplace` trata o redirecionamento contido no código Javascript retornado ao tentar obter o conteúdo da primeira página de resultados de uma pesquisa. Ele é responsável por extrair a URL existente no código e tratá-la, substituindo alguns códigos pelos símbolos correspondentes.

Em ambos os casos de redirecionamento é necessário invocar novamente o método `get` da classe `GetMethod` passando como parâmetro a nova URL.

2) Classe `Analisador`

Nesta classe encontram-se todos os *parsers* utilizados para localizar e extrair informações sobre as comunidades retornadas na pesquisa por palavra-chave, bem como acerca de seus membros e comunidades relacionadas.

Como os documentos *Web* são formatados através das *tags* da linguagem HTML, as páginas do Orkut possuem uma estrutura padrão para páginas com a mesma função. Logo, foi possível desenvolver *parsers* específicos para páginas de resultados, de comunidades, de membros e de amigos.

Antes de dar início à implementação destes *parsers* foi realizado um estudo sobre a estrutura dos documentos *Web* a serem utilizados pelos mesmos, identificando assim, fragmentos de código capazes de garantir aos *parsers* a localização exata das informações que deseja-se extrair.

Assim como na estrutura, foi constatado que as URLs também respeitam um padrão para páginas semelhantes. Ou seja, URLs de páginas de mesmo tipo só se diferenciam pelo número da página, *cmm* ou *uid*.

Para conhecer a URL padrão das páginas de resultados para diferentes buscas por comunidades foram realizadas pesquisas manuais, descobrindo assim que códigos representam os símbolos mais utilizados nas mesmas. Estes são substituídos pelo código ASCII correspondente em hexadecimal precedido pelo caractere '%', com exceção do espaço que é substituído pelo símbolo '+'. A regra não se aplica a letras acentuadas e cedilha, assim como para caracteres que não são reconhecidos como operadores, os quais simplesmente não sofrem alteração.

Sendo assim foi possível obter o código-fonte dessas páginas diretamente através do método GET, dispensando a submissão de informações pelo formulário de pesquisa do *site*.

Os *parsers* foram desenvolvidos seguindo a mesma seqüência de passos, diferenciando-se entre si basicamente no que diz respeito às *substrings* utilizadas para localização das informações de interesse dentro dos códigos retornados.

Primeiramente há a necessidade de obter-se o número total de resultados, membros ou amigos, os quais estão contidos nas suas respectivas primeiras páginas. Então é calculado o número total de páginas em que estes encontram-se organizados para que os demais *parsers* saibam quantas destas precisarão percorrer. Tendo como exceção apenas as comunidades relacionadas, que devido à existência de um número limite apresentam-se todas na página principal da comunidade.

Em seguida, de acordo com as informações a serem armazenadas na forma de atributos, um novo objeto da classe Comunidade ou da classe Usuário é estanciado e adicionado a uma lista. Após o término de execução dos *parsers* existirão listas formadas por comunidades, comunidades relacionadas, membros e amigos.

Por fim, é montada a URL da página seguinte e através do método pegaConteudo da classe Acessador é obtido o código-fonte a ser utilizado na próxima execução do laço de repetição.

Para cada pesquisa, a seguinte ordem de métodos do Analisador é chamada:

- pegaComunidades;
- para cada comunidade na lista de comunidades encontradas:
 - pegaDono;
 - pegaRelacionadas;
 - pegaMembros;
 - para cada membro na lista de membros capturados:
 - pegaAmigos

3) Classe Comunidade

Esta classe tem como objetivo representar, dentro do programa, uma comunidade do Orkut, incluindo seu nome, cmm, dono, lista de relacionadas e lista de membros. Estão também disponíveis os métodos para retornar a URL da comunidade bem como a URL da sua página de membros, as quais são montadas a partir do cmm.

4) Classe Usuario

Esta classe tem como objetivo representar, dentro do programa, um usuário do Orkut, incluindo seu nome, uid e sua lista de amigos. Também estão disponíveis métodos para acessar as URLs do seu perfil e da sua lista de amigos, que são montadas a partir do seu uid.

A Fig. 4 mostra uma visão geral do módulo model através de um diagrama de classes.

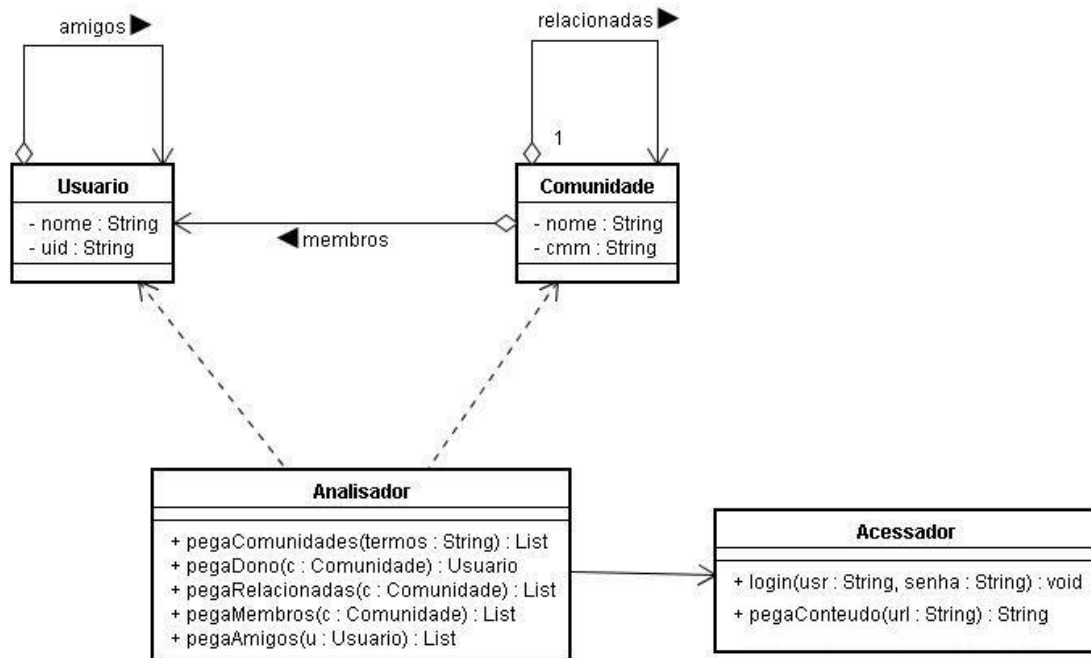


Figura 4 - Diagrama de classes do módulo Model

3.5.2 Módulo BD

Neste pacote encontram-se todas as classes responsáveis pela manipulação de informações nas tabelas do banco de dados.

Com exceção apenas da classe `JdbcConnection`, todas as classes deste pacote implementam os comandos SQL básicos a serem utilizados pelo programa, sendo que cada classe o faz para uma tabela específica. As seis classes que formam o pacote `bd` são descritas a seguir:

1) Classe `JdbcConnection`

Esta classe é responsável por estabelecer conexões e manter a comunicação com o banco de dados.

Esta classe utiliza um *driver* JDBC (*Java Database Connectivity*), que converte chamadas JDBC para o protocolo de rede usado pelo banco de dados MySQL, permitindo que desenvolvedores que trabalham com a linguagem de programação Java

possam construir programas que interagem com este SGBD. (MYSQL, 2008b)

Os métodos `getConnection` e `closeConnection` são responsáveis por iniciar/abrir e encerrar/fechar uma conexão com o banco de dados.

Através do método `getResultSet` as demais classes do pacote são capazes de implementar o comando `select`, passando como parâmetro a cláusula “where” do comando em forma de *string*.

O método `doQuery` permite que as demais classes deste pacote implementem os comandos `insert`, `update` e `delete` através do envio das consultas como parâmetro na forma de *strings*.

2) Classe `ComunidadeBD`

Utilizada basicamente para implementar os comandos SQL necessários para manipulação de informações na tabela `comunidades`.

O método `existe` verifica se a comunidade a ser inserida já faz parte de um registro na tabela.

A função do método `tiraDono` é deixar nulo/vazio o campo de identificação do membro que criou a comunidade.

3) Classe `ComunidadeRelacionadaBD`

O método `temRelacao` tem função similar a do método `existe` nas classes `ComunidadeBD` e `UsuarioBD`, porém verifica se um registro com a relação entre uma comunidade encontrada e uma relacionada já existe na tabela `comunidades_relacionadas`.

4) Classe `ComunidadeUsuarioBD`

Semelhante à classe `ComunidadeRelacionadaBD` o método `temRelacao` confere se uma relação entre uma comunidade e seu membro já existe na tabela `comunidades_usuarios`.

5) Classe `UsuarioAmigoBD`

Segue o mesmo princípio que as classes `ComunidadeRelacionadaBD` e `ComunidadeUsuarioBD` para manipulação das informações relativas à tabela `usuarios_amigos`.

6) Classe UsuarioBD

Análoga à classe ComunidadeBD, porém para manipulação de informações na tabela usuarios.

Na Fig. 5, são ilustradas as classes pertencentes ao módulo bd.

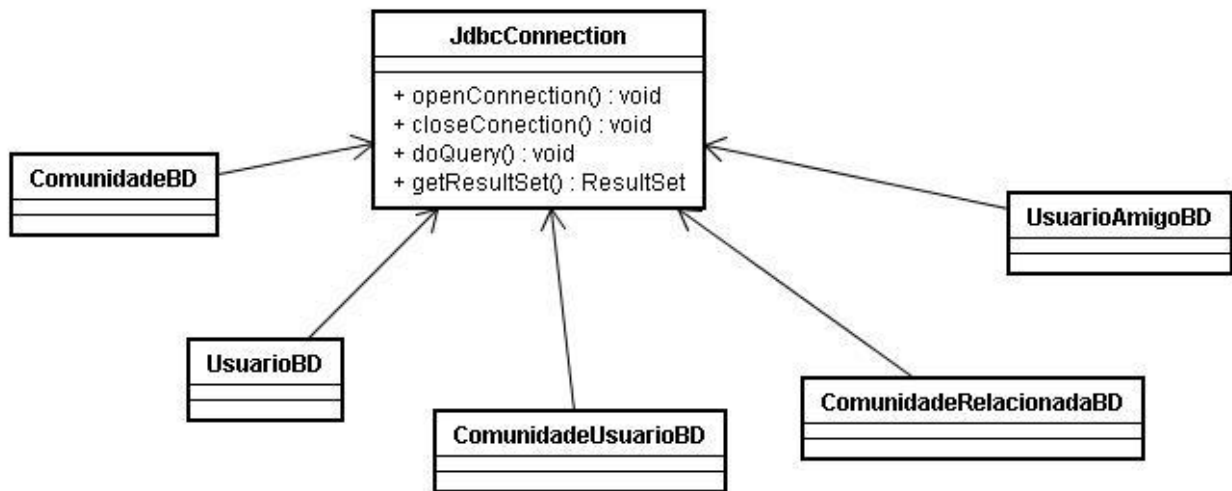


Figura 5 - Diagrama de classes do módulo BD.

3.5.3 Projeto do banco de dados

Foram necessárias cinco tabelas no banco de dados para armazenamento das informações e controle das relações entre comunidades e membros. A seguir, serão descritas as tabelas criadas.

1) Tabela comunidades

Armazena os dados das comunidades encontradas na pesquisa por palavra-chave e de suas comunidades relacionadas.

Onde o id_dono é dado pela chave estrangeira da tabela usuário, a qual corresponde ao índice do registro onde está armazenado o membro que criou a comunidade. Para as comunidades que não tiverem sido encontradas na busca este campo da tabela permanece nulo.

Foi utilizada uma *flag* com o nome *so_relacionada* para distinguir as comunidades encontradas na lista de resultados de uma pesquisa e as que simplesmente pertencem a uma ou mais listas de comunidades.

2) Tabela *comunidades_usuarios*

Responsável por relacionar comunidades com seus membros a partir de seus índices nas tabelas *comunidades* e *usuarios*, respectivamente.

3) Tabela *usuários*

Contém as informações dos membros das comunidades encontradas na pesquisa e dos seus amigos.

De forma semelhante à tabela *comunidades*, foi utilizada a *flag so_amigo* para identificação dos usuários que foram encontrados em listas de amigos, mas que não pertencem a nenhuma comunidade resultante da pesquisa por palavra-chave.

4) Tabela *comunidades_relacionadas*

Onde são armazenadas as relações entre as comunidades encontradas na pesquisa por palavra-chave e suas comunidades relacionadas através de seus índices na tabela *comunidade*, ou seja, um auto-relacionamento da tabela *comunidades*.

5) Tabela *usuarios_amigos*

Tabela onde é possível identificar as ligações entre usuários e seus amigos, tendo como indicadores a chave estrangeira que corresponde aos índices dos registros onde se encontram na tabela *usuarios*, caracterizando um auto-relacionamento da mesma.

A Fig. 6 ilustra a modelagem do BD com o diagrama ER (entidade-relacionamento) do banco de dados.

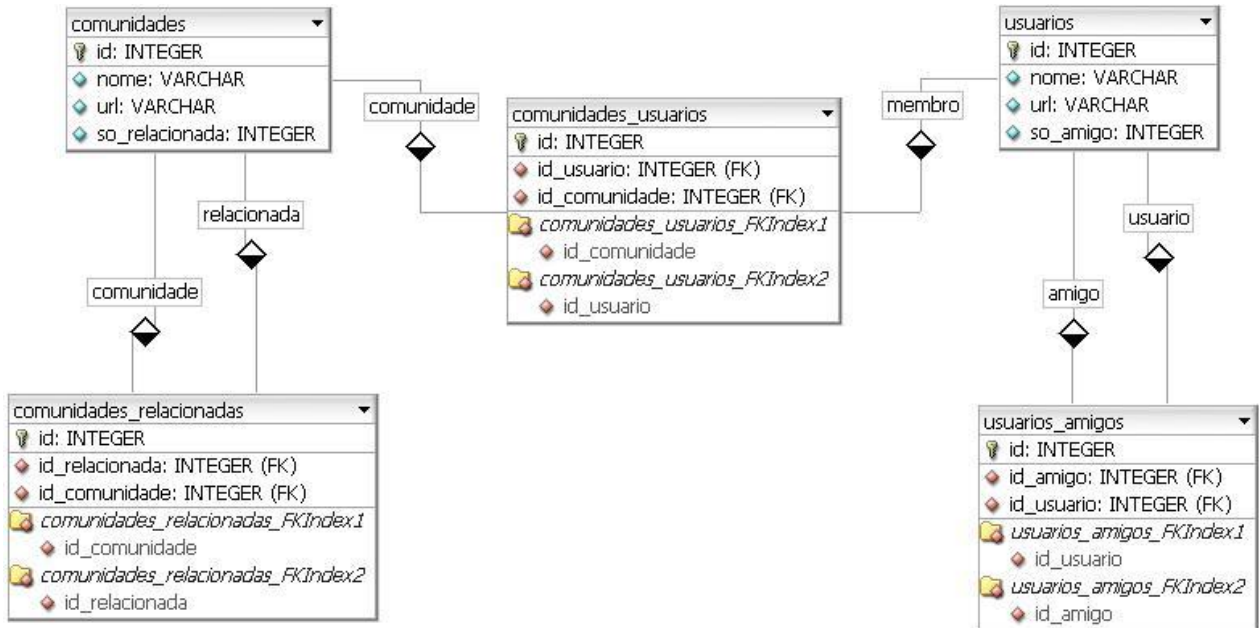


Figura 6 - Diagrama ER do banco de dados.

4. Resultados

Como trata-se de um estudo de caso, o termo de pesquisa utilizado para execução do programa na validação pôde ser escolhido aleatoriamente.

Levando em consideração os dados a serem extraídos, principalmente no que diz respeito ao número de membros de cada comunidade e de amigos destes, preocupou-se em utilizar na busca uma chave que retornasse poucas comunidades para melhor visualização dos resultados para validação da ferramenta.

Sendo assim, após pesquisas manuais chegou-se aos termos: ciencia da computacao ufpel, os quais foram utilizados em uma mesma busca separados apenas por espaço, sem acentuação e cedilha.

4.1 Metodologia de validação

Para validação da ferramenta desenvolvida foi considerado suficiente utilizar um método que possibilitasse confrontar as informações obtidas com as disponíveis no Orkut, visando comprovar que todas as informações foram extraídas e armazenadas sucesso.

Para isto foram realizadas as seguintes consultas no BD construído:

- 1) **select * from** comunidades **where** so_relacionada=0; Deve retornar as informações sobre todas as comunidades encontradas pelo motor de busca.
- 2) **select * from** usuario, comunidades **where** usuarios.id= comunidades.id_dono; Deve retornar as informações sobre os donos das comunidades encontradas pelo mecanismo de pesquisa.
- 3) **select * from** usuarios, comunidades_usuarios **where**

comunidades_usuarios.id_usuario=usuarios.id and comunidades_usuarios.id_comunidade=85; Deve retornar as informações sobre todos os membros da comunidade retornada que possui o menor número de membros.

4) **select** usuarios.url, usuarios.nome **from** usuários, usuarios_amigos **where** id_usuario=22610 and id_amigo=usuarios.id; Deve retornar nome e URL de todos os amigos do usuário com menor número de amigos.

5) **select** comunidades.url, comunidades.nome **from** comunidades, comunidades_relacionadas **where** id_comunidade=78 and id_relacionada=comunidades.id; Deve retornar nome e URL de todas as comunidades relacionadas da única comunidade encontrada que possui comunidades relacionadas.

Como pode ser observado nas consultas 3 e 4 acima, não foram recuperadas todas as informações extraídas para exibição dos resultados, pois resultaria em um número muito grande de membros e amigos. Então para fins de amostragem foi feita a escolha de um usuário e de uma comunidade com base nos resultados da busca manual, verificando-se em seguida os seus respectivos índices nas tabelas comunidades e usuários. Da mesma forma foi identificado visualmente o índice da comunidade escolhida para a consulta 5.

Embora não tenha sido possível exibir todas as informações armazenadas, as consultas acima foram consideradas suficientes para comprovação da eficiência da ferramenta desenvolvida. Isto se deve ao fato de que algoritmos que funcionem corretamente para extração e armazenamento de informações de uma comunidade também o farão para as demais comunidades rastreadas, sendo que o mesmo aplica-se para validação das informações sobre membros e seus amigos.

4.2 Análise dos resultados

O programa iniciou sua execução às 23h37min41s do dia 10 de agosto de 2008 e encerrou-se após 760,047s. A execução utilizada na validação foi realizada em um computador com processador Intel Pentium 4 e 256M de RAM, dispondo de uma conexão de 400Kbps.

Todas as informações a que a ferramenta se propunha foram extraídas com sucesso, totalizando 8 comunidades e 11922 usuários. Incluindo comunidades encontradas somente em listas de comunidades relacionadas e usuários recuperados através de listas de amigos.

O resultado da execução de cada uma das consultas listadas anteriormente são ilustradas a seguir, juntamente com os resultados obtidos na busca manual para efeito de comparação. Visando a privacidade de integrantes do Orkut não são apresentadas imagens do *site* para lista de membros e lista de amigos.

Seguindo a mesma numeração das consultas quando descritas na seção Metodologia de Validação foram obtidas:

1) Comunidades resultantes da pesquisa por palavra-chave:

Resultados de pesquisa para ciencia da computacao ufpel

Início > Pesquisar

[todos os resultados](#) [usuários](#) [comunidades](#) [tópicos](#)

Pesquisar novamente: [pesquisar](#)

Resultados de pesquisa para **ciencia da computacao ufpel** [refinar os resultados](#)

Resultados 1 - 2 de 2

Resultados em **meu país** (Brasil):

	<p>Ciência da Computação -UCPel-</p> <p>Categoria: Alunos e Escolas (33)</p> <p>Local: Brasil</p> <p>Comunidade criada à todos aqueles q fazem Ciência da Computação na UCPel e também para os professores!!!!</p>
	<p>Ciência da Computação - UFPEL</p> <p>Categoria: Alunos e Escolas (3)</p> <p>Local: Brasil</p> <p>POOORA NÃO AXEI UMA COMUNIDADE DA CIENCIA DA COMPUTAÇÃO ...</p> <p>eu passei fui procura e nada ... tive que faze uma ...</p> <p>;@@@</p>

Figura 7 - Pesquisa manual por comunidades.



Figura 8 - Comunidades capturadas.

- 2) Donos das comunidades encontradas. Porém para comprovação foi utilizada somente a imagem da página de umas das comunidades retornadas.

The screenshot shows the community page for 'Ciência da Computação -UCPel-'. The page includes a sidebar with navigation options and a main content area with community details and a forum list.

Ciência da Computação -UCPel-
 Início > Comunidades > Ciência da Computação -UCPel-

descrição: Comunidade criada à todos aqueles q fazem Ciência da Computação na UCPel e também para os professores!!!!

idioma: **Português**

categoria: Alunos e Escolas

dono: ▼ Clarissa ▼ Ribeiro Assis

moderadores:

tipo: pública

privacidade do conteúdo: aberta para não-membros

fórum: anônimo

local: Pelotas, Rio Grande do Sul, Brasil

criado em: 11 de março de 2006 17:10

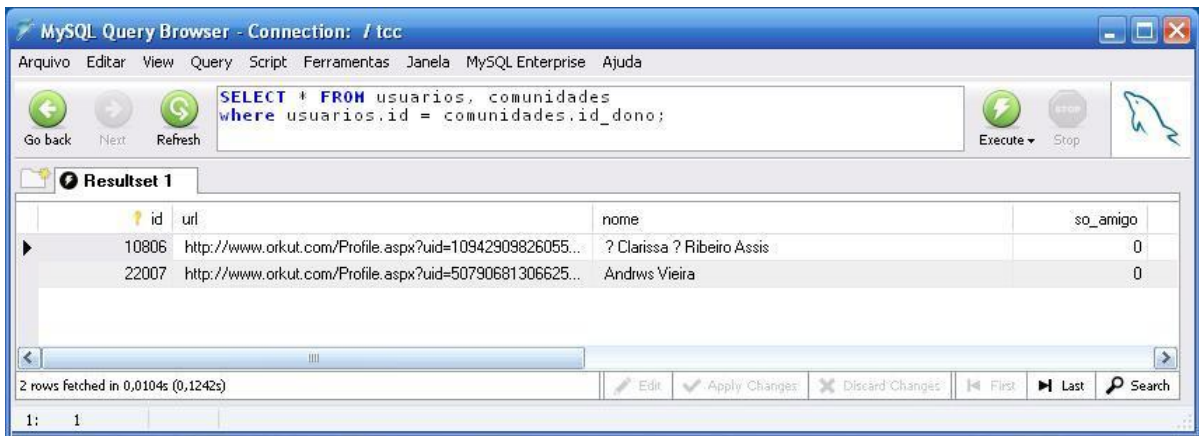
membros: 33

fórum

tópico	postagens	última postagem
<input type="checkbox"/> hey, I'm horny!	1	05/08/08
<input type="checkbox"/> Símbolo do curso para formatura	1	10/02/08
<input type="checkbox"/> Grupo de Estudo em Desenvolvimento de Jogos	2	13/12/06
<input type="checkbox"/> Em que ano vc iniciou o curso?	5	13/12/06

[ver todos os tópicos](#) [novo tópico](#) [denunciar spam](#)

Figura 9 - Página de comunidade.



MySQL Query Browser - Connection: / tcc

Arquivo Editar View Query Script Ferramentas Janela MySQL Enterprise Ajuda

Go back Next Refresh

```
SELECT * FROM usuarios, comunidades
where usuarios.id = comunidades.id_dono;
```

Execute Stop

Resultset 1

id	url	nome	so_amigo
10806	http://www.orkut.com/Profile.aspx?uid=10942909826055...	? Clarissa ? Ribeiro Assis	0
22007	http://www.orkut.com/Profile.aspx?uid=50790681306625...	Andrws Vieira	0

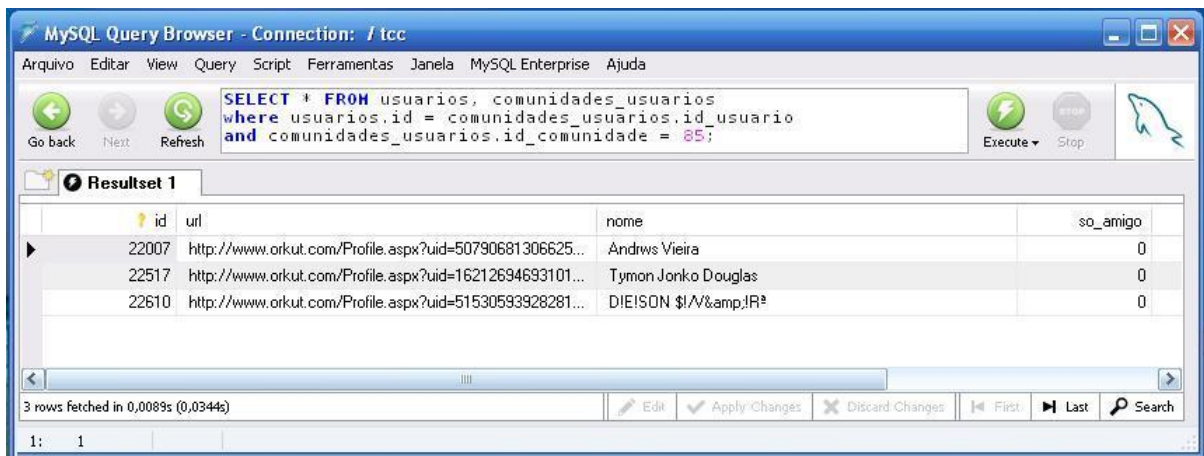
2 rows fetched in 0,0104s (0,1242s)

Edit Apply Changes Discard Changes First Last Search

1: 1

Figura 10 - Donos das comunidades capturadas.

- 3) Todos os membros de uma comunidade. Pode-se comparar o número de membros, já que optou-se por não divulgar imagens.



MySQL Query Browser - Connection: / tcc

Arquivo Editar View Query Script Ferramentas Janela MySQL Enterprise Ajuda

Go back Next Refresh

```
SELECT * FROM usuarios, comunidades_usuarios
where usuarios.id = comunidades_usuarios.id_usuario
and comunidades_usuarios.id_comunidade = 85;
```

Execute Stop

Resultset 1

id	url	nome	so_amigo
22007	http://www.orkut.com/Profile.aspx?uid=50790681306625...	Andrws Vieira	0
22517	http://www.orkut.com/Profile.aspx?uid=16212694693101...	Tymon Jonko Douglas	0
22610	http://www.orkut.com/Profile.aspx?uid=51530593928281...	DIEISON \$!/\$&IRª	0

3 rows fetched in 0,0089s (0,0344s)

Edit Apply Changes Discard Changes First Last Search

1: 1

Figura 11 - Membros de uma das comunidades capturadas.

4) Nome e URL de todos os amigos de um usuário.

The screenshot shows the MySQL Query Browser interface. The query executed is:

```
SELECT usuarios.url, usuarios.nome
FROM usuarios, usuarios_amigos
where id_usuario = 22610
and id_amigo = usuarios.id;
```

The results are displayed in a table with two columns: 'url' and 'nome'. The table contains 11 rows of data, representing the friends of a user with ID 22610.

url	nome
http://www.orkut.com/Profile.aspx?uid=13394914680279...	Èveline Kenes
http://www.orkut.com/Profile.aspx?uid=18241654738013...	giandra !!!!
http://www.orkut.com/Profile.aspx?uid=11334988546022...	Helena Szortika Quadros
http://www.orkut.com/Profile.aspx?uid=43450577573647...	?Silvane?? Soares
http://www.orkut.com/Profile.aspx?uid=47796575249716...	~*Vaninha*~ *
http://www.orkut.com/Profile.aspx?uid=38898821897695...	Igor Antoni
http://www.orkut.com/Profile.aspx?uid=71164757753244...	'S?????? ?
http://www.orkut.com/Profile.aspx?uid=11345390165641...	Grazi Gouvea
http://www.orkut.com/Profile.aspx?uid=43002769761438...	Cris Kenes
http://www.orkut.com/Profile.aspx?uid=17936577886333...	J&f&r\$on K&n&\$ Nun&\$
http://www.orkut.com/Profile.aspx?uid=85185665233320...	Fernando Duarte

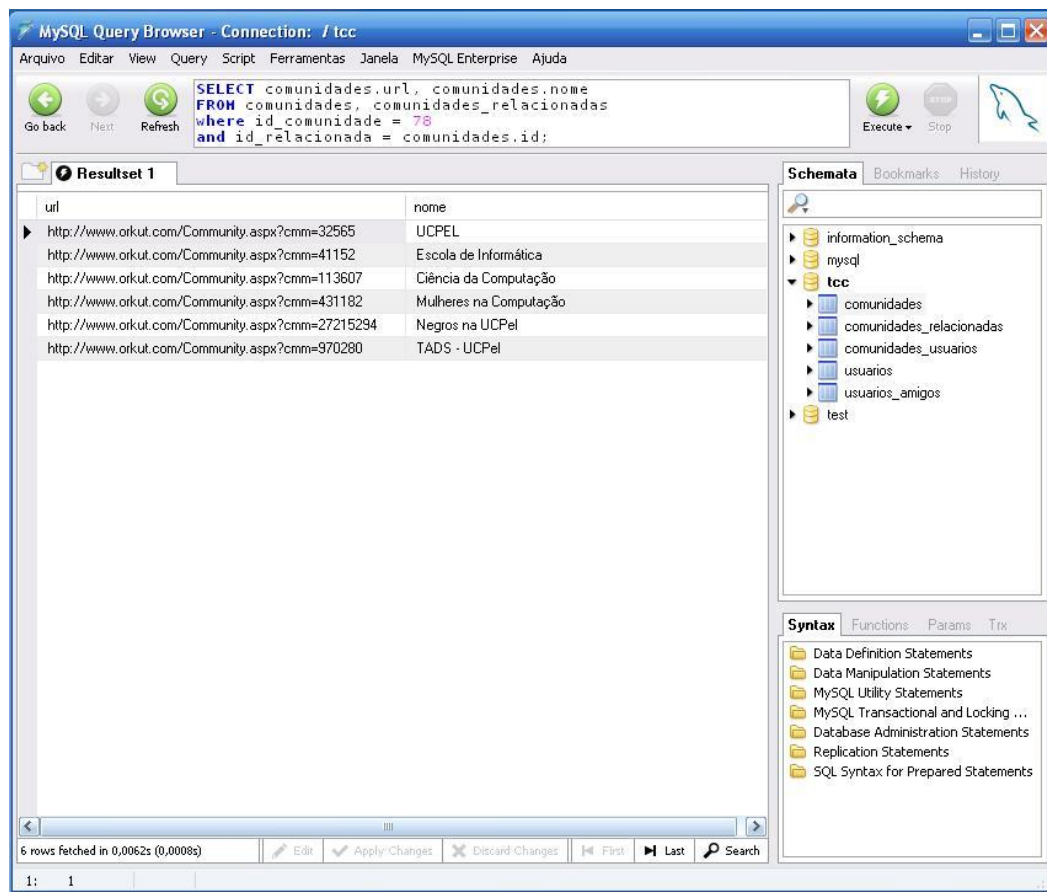
The interface also shows a 'Schemata' panel on the right with a tree view of the database structure, including 'information_schema', 'mysql', and 'tcc'. The 'tcc' database contains tables like 'comunidades', 'comunidades_relacionadas', 'comunidades_usuarios', 'usuarios', and 'usuarios_amigos'. The 'Syntax' panel at the bottom right lists various SQL statement categories.

Figura 12 - Amigos de um dos membros de uma das comunidades capturadas.

5) Nome e URL de todas as comunidades relacionadas de uma comunidade.



Figura 13 – Comunidades relacionadas.



The screenshot displays the MySQL Query Browser interface. The main window shows a query executed against a database named 'tcc'. The query is:

```
SELECT comunidades.url, comunidades.nome  
FROM comunidades, comunidades_relacionadas  
where id_comunidade = 76  
and id_relacionada = comunidades.id;
```

The result set, titled 'Resultset 1', contains 6 rows of data:

url	nome
http://www.orkut.com/Community.aspx?cmm=32565	UCPEL
http://www.orkut.com/Community.aspx?cmm=41152	Escola de Informática
http://www.orkut.com/Community.aspx?cmm=113607	Ciência da Computação
http://www.orkut.com/Community.aspx?cmm=431182	Mulheres na Computação
http://www.orkut.com/Community.aspx?cmm=27215294	Negros na UCPel
http://www.orkut.com/Community.aspx?cmm=970280	TADS - UCPel

The interface also shows a 'Schemata' panel on the right, listing databases like 'information_schema', 'mysql', and 'tcc'. The 'tcc' database is expanded, showing tables such as 'comunidades', 'comunidades_relacionadas', 'comunidades_usuarios', 'usuarios', 'usuarios_amigos', and 'test'. A 'Syntax' panel is also visible at the bottom right, listing various SQL statement types.

Figura 14 - Relacionadas de uma das comunidades capturadas.

5. Conclusões

Os resultados obtidos ao final deste trabalho foram satisfatórios, pois a ferramenta desenvolvida foi capaz de desempenhar todas as tarefas propostas conforme pôde ser verificado na seção de análise dos resultados.

O trabalho aqui descrito envolve Mineração de Estrutura e Mineração de Conteúdo, pois as informações a serem extraídas são rastreadas explorando-se a estrutura de *hiperlinks* existente a partir das comunidades encontradas na pesquisa por palavra-chave. E indiretamente através dos motores de busca, os quais também utilizam-se de técnicas de Mineração de Estrutura e Mineração de Conteúdo para rastreamento e indexação de documentos.

Embora o foco do trabalho seja Recuperação e Extração de Informação, não foram diretamente aplicadas técnicas de RI para desenvolvimento da ferramenta, visto que foi possível utilizar-se do motor de busca disponível no Orkut. Nesta etapa os algoritmos encarregaram-se de encontrar dentro das páginas de resultados o nome e URL de cada comunidade encontrada, e a partir daí os demais documentos são encontrados através dos diversos tipos de relações permitidas no *site*.

Da mesma forma, para Extração de Informação não foi utilizada nenhuma técnica específica, apenas foram aplicados e aprimorados os conhecimentos de programação para construção dos *parsers*, contando com o auxílio de recursos da linguagem Java e classes da biblioteca HTTPClient.

Para o estudo de caso foi realizada apenas uma busca, pois foi considerado suficiente para mostrar que foram alcançados os objetivos propostos. A maneira como uma possível ampliação para utilizar diversas palavras-chave não interferiria nos

algoritmos de extração e armazenamento das informações, que são o foco do trabalho.

Conectar-se ao *site* do Orkut programaticamente torna-se uma tarefa complexa pelo fato deste fazer uso de *cookies*. As primeiras tentativas foram baseadas em pequenos tutoriais da biblioteca HTTPClient e códigos encontrados na *Web*, nos quais não era previsto gerenciamento de *cookies* e tratamentos para códigos de *status* de retorno de uma requisição. Sendo assim foi necessário pesquisar algum código mais completo já existente.

Foi muito útil poder contar com os exemplos de código para *login* da biblioteca, embora não fossem suficientes sozinhos, pois caso contrário este tomaria muito mais tempo para estudo da mesma, suas classes, interfaces e respectivos métodos. Além do tempo exigido fugiria ao escopo do trabalho, o qual não consistia no estudo da biblioteca HTTPClient.

Outra dificuldade encontrada foi com relação à dinamicidade da *Web*, que não diz respeito somente ao conteúdo disponível, mas também aos recursos utilizados pelos desenvolvedores *Web* e estrutura dos documentos. Algumas dessas modificações visam barrar ou dificultar a extração automática de informações, como foi o caso do tratamento citado na seção Implementação do capítulo 3.

Mineração na *Web* é de natureza interdisciplinar, abrangendo áreas como Recuperação de Informação, Processamento de Linguagem Natural, Extração de Informação, Aprendizagem de Máquina, Banco de Dados, Mineração de Dados, *Data Warehousing*, abrindo um leque de possibilidades para trabalhos futuros.

Este trabalho não foi desenvolvido para fins lucrativos, e sim visando oferecer uma contribuição científica e acadêmica. Sendo assim, a ferramenta foi executada exclusivamente para validação do trabalho. Para isto foram utilizados termos de pesquisa que retornassem poucas comunidades como resultado e explorando um tema que não comprometesse ou prejudicasse qualquer usuário do Orkut se identificado por seu nome ou apelido na rede social.

Diante do item 5.3 descrito em Termos (2008), para aplicação da ferramenta aqui descrita para fins específicos, comerciais ou não, seria necessária uma autorização prévia concedida pela empresa Google Inc.

5.1 Trabalhos futuros

Pensando simplesmente na otimização de código e da ferramenta tem-se como opções:

- Construir uma interface gráfica para o usuário, permitindo visualização dos conteúdos em tabelas, onde o usuário teria como opções realizar uma nova consulta ou consultar o banco de dados.
- Permitir que um usuário leigo em programação insira novas palavras-chave quando necessitar. Isto poderia ser feito pelo usuário diretamente em arquivos de texto ou através de uma interface de usuário, sendo que um algoritmo se encarregaria de atualizar o arquivo além de ler do mesmo.
- Manter e atualizar as informações obtidas, contemplando as alterações nas relações entre os indivíduos, e alterações nas comunidades, bem como as que se formaram o ou deixaram de existir após a última pesquisa. Isto se o interesse for manter somente informações atuais.
- Como no trabalho foi realizada somente uma pesquisa por comunidades, não foi criada uma tabela para armazenamento das palavras-chave ou composições utilizadas, a qual necessitaria de outra tabela para relacioná-la com a tabela comunidades. Esta tabela seria semelhante às tabelas comunidades_usuarios, comunidades_relacionadas e usuarios_amigos. A criação destas tabelas seria conveniente para saber com que termos foi encontrada a comunidade.
- Estender os *parsers* para que sejam capazes de extrair informações textuais inseridas pelos usuários em seus perfis, nos fóruns das comunidades ou em outros pontos da rede social.
- Incorporar o uso de expressões regulares para tornar os *parsers* mais genéricos, e possivelmente estender a aplicação de modo a abranger outros *sites* de relacionamento. O que poderia necessitando apenas de códigos para autenticação específicos para cada serviço.

Um exemplo interessante como trabalho futuro seria a construção de um *Data Warehouse*, pois além de possibilitar a análise de grandes volumes de dados, permitiria

as chamadas séries históricas, as quais possibilitam uma melhor análise de eventos passados. Isto seria relevante, pois além das informações da *Web* mudarem constantemente, permitiria aplicação de algoritmos de Mineração de Dados.

Outra abordagem possível seria utilizar técnicas de Análise de Redes Sociais, que conforme foi dito no trabalho baseia-se na teoria dos grafos para analisar fenômenos sociais.

Referências

ALEXA, The Web Information Company. Disponível em: <http://www.alexa.com/site/ds/top_500>. Acesso em: 12 fev 2008.

ÁLVAREZ, Alberto C. **Extração de Informação de Artigos Científicos: uma abordagem baseada em indução de regras de etiquetagem**. 2007. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e Computação, Universidade de São Paulo, São Carlos.

CAZELLA, Sílvio. **Mineração de dados na Web e agentes**. 2001. Trabalho Individual (Programa de pós-graduação em Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

CLASS Cookie. Disponível em: <<http://hc.apache.org/httpclient-3.x/apidocs/org/apache/commons/httpclient/Cookie.html>>. Acesso em: 19 jul 2008.

CLASS CookiePolicy. Disponível em: <<http://hc.apache.org/httpclient-3.x/apidocs/org/apache/commons/httpclient/cookie/CookiePolicy.html>>. Acesso em: 19 jul 2008.

CLASS GetMethod. Disponível em: <<http://hc.apache.org/httpclient-3.x/apidocs/org/apache/commons/httpclient/methods/GetMethod.html>>. Acesso em: 19 jul 2008.

CLASS Header. Disponível em: <<http://hc.apache.org/httpclient-3.x/apidocs/org/apache/commons/httpclient/Header.html>>. Acesso em: 19 jul 2008.

CLASS HttpClient. Disponível em: <<http://hc.apache.org/httpclient-3.x/apidocs/org/apache/commons/httpclient/HttpClient.html>>. Acesso em: 19 jul 2008.

CLASS HttpConnection. Disponível em: <<http://hc.apache.org/httpclient-3.x/apidocs/org/apache/commons/httpclient/HttpConnection.html>>. Acesso em: 19 jul 2008.

CLASS NameValuePair. Disponível em: <<http://hc.apache.org/httpclient-3.x/apidocs/org/apache/commons/httpclient/NameValuePair.html>>. Acesso em: 19 jul 2008.

CLASS HttpState. Disponível em: <<http://hc.apache.org/httpclient-3.x/apidocs/org/apache/commons/httpclient/HttpState.html>>.

3.x/apidocs/org/apache/commons/httpclient/HttpState.html>. Acesso em: 19 jul 2008.
CLASS HttpStatus. Disponível em: <<http://hc.apache.org/httpclient-3.x/apidocs/org/apache/commons/httpclient/HttpStatus.html>>. Acesso em: 19 jul 2008.

CLASS OutputStream. Disponível em:
<<http://java.sun.com/j2se/1.5.0/docs/api/java/io/OutputStream.html>>. Acesso em: 19 jul 2008.

CLASS PostMethod. Disponível em: <<http://hc.apache.org/httpclient-3.x/apidocs/org/apache/commons/httpclient/methods/PostMethod.html>>. Acesso em: 19 jul 2008.

CÓDIGOS de status HTTP. Disponível em:
<<http://www.google.com/support/webmasters/bin/answer.py?hl=br&answer=83040>>.
Acesso em: 24 jul 2008

COOKIES. Disponível em: <<http://www.wesk.org/textos/outros/3.html>>. Acesso em: 19 jul 2008.

CORDEIRO, João P.C. **Extracção de Elementos Relevantes em Texto/Páginas da World Wide Web**. 2003. Tese (Mestrado em Inteligência Artificial e Computação) - Departamento de Ciência de Computadores, Faculdade de Ciências da Universidade do Porto, Porto.

FERRAMENTAS. Disponível em: <<http://www.mysqlbrasil.com.br/?q=node/7>>. Acesso em: 24 jul 2008

HTTP Components. Disponível em: <<http://hc.apache.org/>>. Acesso em: 19 jul 2008.

INTERFACE CookieSpec. Disponível em: <<http://hc.apache.org/httpclient-3.x/apidocs/org/apache/commons/httpclient/cookie/CookieSpec.html>>. Acesso em: 19 jul 2008.

INTERFACE HttpMethod. Disponível em: <<http://hc.apache.org/httpclient-3.x/apidocs/org/apache/commons/httpclient/HttpMethod.html>>. Acesso em: 19 jul 2008.

INTERFACE HttpStatus. Disponível em: <<http://hc.apache.org/httpcomponents-core/httpcore/apidocs/org/apache/http/HttpStatus.html>>. Acesso em: 24 jul 2008

JAKARTA Commons HTTPClient. Disponível em: <<http://hc.apache.org/httpclient-3.x/>>. Acesso em: 19 jul 2008.

JAKARTA Commons HTTPClient. Disponível em: <http://hc.apache.org/httpclient-3.x/cookies.html>>. Acesso em: 19 jul 2008.

JAKARTA Commons HTTPClient. Disponível em: <http://hc.apache.org/httpclient-3.x/methods/get.html>>. Acesso em: 19 jul 2008.

JAKARTA Commons HTTPClient. Disponível em: <http://hc.apache.org/httpclient-3.x/methods/post.html>>. Acesso em: 19 jul 2008.

JEFFREY Heer: Vizster - Visualizing Online Social Networks. Disponível em: http://www.cs.berkeley.edu/~jheer/vizster/early_design/>. Acesso em: 16 fev 2008.

LEANDRO Balby Marinho, ROSARIO Girardi: Mineração na Web. Disponível em: <http://www.sbc.org.br/reic/edicoes/2003e2/tutoriais>>. Acesso em: 18 jul 2007.

LOH, Stanley ; GARIN, Ramiro Saldaña. Web Intelligence: Inteligência Artificial para Descoberta de Conhecimento na Web. In: Oficina de Inteligência Artificial, 5., 2001, Pelotas. **V Oficina de Inteligência Artificial**. Pelotas : EDUCAT - Editora da UCPEL, 2001. p. 11-34.

MARKOV, Zdravko ; LAROSE, Daniel T. **Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage**. 3.ed. Hoboken, New Jersey: John Wiley & Sons, Inc., 2007. 218p.

MYSQL Administrator. Disponível em: <http://www.mysql.com/products/tools/administrator/>>. Acesso em: 24 jul 2008.

MySQL® Connector/J. Disponível em: <http://www.mysql.com/products/connector/j/>>. Acesso em: 24 jul 2008

MYSQL Query Browser. Disponível em: <http://www.mysql.com/products/tools/query-browser/>>. Acesso em: 24 jul 2008.

MYSQL Tools. Disponível em: <http://www.mysql.com/products/tools/>>. Acesso em: 24 jul 2008.

ORKUT. Disponível em: <http://pt.wikipedia.org/wiki/Orkut>>. Acesso em: 18 jul. 2007.

ORKUT - notícias - dados demográficos. Disponível em: <http://www.orkut.com/MembersAll.aspx>>. Acesso em: 23 jul 2007.

RECURSOS do MySQL. Disponível em: <http://www.mysqlbrasil.com.br/?q=node/3>>.

Acesso em: 24 jul 2008

SANDRA de Amo: Curso de Data Mining – Aula 17. Disponível em: <<http://www.deamo.prof.ufu.br/arquivos>>. Acesso em: 18 jul 2007.

TERMOS de Serviço do Google. Disponível em: <<http://www.google.com/accounts/TOS>>. Acesso em: 27 ago 2008.

VIGLIONI, Giovanni M.C. **Metodologia para previsão de demanda ferroviária utilizando Data Mining. 2007.** Dissertação (Mestrado em Engenharia de Transportes) - Instituto Militar de Engenharia, Rio de Janeiro.

WEB Scraper Plus+. Disponível em: <<http://www.web-scraper-plus-.com-about.com/>>. Acesso em: 16 fev 2008.

WHY MySQL?. Disponível em: <<http://www.mysql.com/why-mysql/>>. Acesso em: 24 jul 2008