POLINE LOTTIN

UMA APLICAÇÃO PARA CONSULTA ESTRUTURADA EM DADOS DA *WIKIPEDIA*

FLORIANÓPOLIS
Junho 2011

POLINE LOTTIN

UMA APLICAÇÃO PARA CONSULTA ESTRUTURADA EM DADOS DA *WIKIPEDIA*

Trabalho de conclusão de curso apresentado pela acadêmica Poline Lottin à banca examinadora do Curso de Graduação em Sistemas de Informação da Universidade Federal de Santa Catarina como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Orientadora:
Profª. Drª. Carina Friedrich Dorneles

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
SISTEMAS DE INFORMAÇÃO

FLORIANÓPOLIS
Junho 2011

POLINE LOTTIN

UMA APLICAÇÃO PARA CONSULTA ESTRUTURADA EM DADOS DA *WIKIPEDIA*

Orientadora:

_____
Profª. Drª. Carina Friedrich Dorneles

Banca Examinadora:

_____
Prof. Dr. Renato Fileto

_____
Prof. Dr. Ronaldo dos Santos Mello

FLORIANÓPOLIS
Junho 2011

AGRADECIMENTOS

Agradeço aos meus pais pelo incentivo aos meus estudos, vocês são os grandes culpados pelo meu sucesso. Agradeço aos meus amigos, em especial meu namorado Fernando, pelo grande apoio e presença nos momentos mais difíceis. Agradeço à minha orientadora Carina pela paciência em me orientar e por ter sido minha colega nesta caminhada.

RESUMO

Sistemas de busca para a *Web* foram desenvolvidos a partir da necessidade de encontrar documentos através da internet com maior facilidade, tendo em vista o grande crescimento do conteúdo disponível na rede. As ferramentas de consulta atuais baseiam-se em palavras-chave para recuperar e ranquear o conteúdo desejado pelo usuário. Porém, palavras-chave permitem pouca expressividade ao usuário que intuitivamente deseja efetuar consultas em linguagem natural. Este trabalho propõe um novo modelo de ferramenta de busca, que utiliza alguns princípios de linguagens de consulta a banco de dados relacionais para permitir consultas mais expressivas na *Web.* A expressividade da consulta efetuada com esta proposta é garantida através de uma interface que auxilia o usuário a formatar consultas em um formato semelhante à linguagem de consulta SQL. O modelo proposto foi implementado através de uma ferramenta de busca chamada *Find Me*. Esta ferramenta percorre documentos da *Wikipedia* e indexa dados semi-estruturados. A indexação é efetuada de forma que seja possível interpretar consultas estruturadas na *Web*, como se os dados estivessem em uma estrutura semelhante aos bancos de dados tradicionais. Alguns experimentos iniciais foram efetuados comprovando a eficácia do modelo e indicaram alguns trabalhos futuros para o projeto.

Palavras-chave: Recuperação de informação, ferramenta de busca, *Wikipedia*, SQL.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. Introdução

A popularização da *Web* proporcionou o aumento cada vez maior de dados espalhados pelo mundo em documentos digitais, em sua maioria, descritos em arquivos textuais. Hoje, a forma mais popular de recuperar estes dados é a busca através de palavras-chave que se mostra pouco expressiva, apresentando problemas como sinonímia e polissemia, por exemplo. A necessidade de encontrar informações mais relevantes em milhares de documentos na *Web* estimula pesquisas de novas técnicas mais eficazes para consultas textuais.

Atualmente o método de pesquisa mais conhecido é utilizado pelo *Google* em seu mecanismo de consulta. Este mecanismo procura coletar o maior número possível de recursos (URLs) através de softwares robôs e permite ao usuário consultar registros através de palavras-chave. Os motores de busca utilizam-se dos robôs, também chamados de *spiders* ou *crawlers*, para varrer páginas e capturar o maior número de informação possível, para então, indexar os links em sua base de dados. Atualmente, a maioria dos serviços que utilizam este mecanismo traz como resultado da consulta os links que são considerados mais relevantes com relação a um conjunto de palavras-chave informado pelo usuário. Os resultados são ordenados através de sua relevância, que pode ser definida de formas diferentes em cada serviço de buscas. A relevância de um link pode ser definida, por exemplo, a partir do número de repetições do termo pesquisado no documento, se o termo está contido no título da página e até mesmo pela popularidade do site, como feito pelo *PageRank* do Google [BRIN e PAGE, 1999].

Porém, estes serviços de consulta limitam o usuário a apenas buscar páginas através de palavras-chaves, obrigando-o a deduzir quais termos devem ser inseridos na consulta para se obter a página desejada como resultado. Existe a necessidade de atribuir significado às palavras ou frases informadas em consultas. Desta forma, seria possível contextualizar e relacionar conteúdos da Web, obtendo resultados mais significativos.

Nos mecanismos de consulta atuais, a consulta "*cidades que possuem prefeito chamado José*" não possui valor semântico e os documentos retornados serão aqueles que possuem os termos da consulta, mesmo que o documento retornado esteja relacionado ao prefeito da cidade de São José. Seria muito interessante a possibilidade de efetuar consultas mais sofisticadas que compreendessem que quando a consulta "*cidades que possuem prefeito chamado José*" é informada, na verdade o termo *José* está relacionado ao prefeito de uma cidade.

A *Wikipedia*[1] é fonte rica de informações e se encontra em constante crescimento, visto que seu conteúdo é alimentado colaborativamente. Os dados desta wiki seguem alguns guias de estilo, ou seja, alguns padrões definidos pelo site. Um dos padrões da *Wikipedia* é uma caixa informativa, chamada *infobox*, que disponibiliza dados semi-estruturados sobre cada documento.

1.1. Objetivos

A ideia geral consiste em previamente varrer páginas em HTML (inicialmente será utilizado um conjunto fixo de páginas, como as do *Wikipedia*) e através de algum mecanismo semântico[2], indexar e estruturar todas as relações entre os dados que forem encontrados. O principal objetivo do trabalho é desenvolver uma abordagem, e uma ferramenta de busca que a valida, capaz de permitir consultas estruturadas em páginas HTML, onde pouca ou quase nenhuma estrutura é encontrada. A proposta da ferramenta é permitir consultas que reflitam a abordagem definida, além de disponibilizar uma interface *Web* simples, facilitando a consulta até mesmo a usuários que não tenham conhecimento da sintaxe de consultas a banco de dados.

Para possibilitar a correta construção das consultas, foi desenvolvido um indexador capaz de capturar dados em textos e tabelas do documento, identificando seus relacionamentos e características (nome de objeto e seus atributos). Com os documentos relacionados entre si e características identificadas, estas informações são indexadas de forma a facilitar a construção de consultas estruturadas. Como *front-end* da aplicação (Figura 1), é disponibilizada uma página com campos que auxiliam o usuário a escrever uma consulta estruturada. A estrutura da consulta é semelhante às clausulas do SQL [CHAMBERLIN e BOYCE, 1992] informando o que se deseja consultar, qual o contexto da consulta e suas restrições para que um documento seja recuperado.

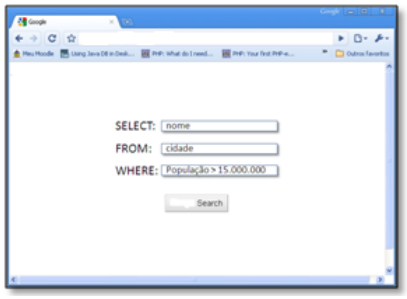


Figura 1. Protótipo do *front-end*

[1] http://www.wikipedia.org/
[2] Não é objetivo deste trabalho a construção ou uso de ontologias, ou qualquer estrutura semântica construída *a priori*

1.2. Motivação

Desde que a internet se popularizou, surgiu a preocupação de categorizar e indexar todas essas páginas para facilitar sua localização sem que o usuário tenha que navegar por outras páginas até encontrar o link desejado. Entre todas as páginas, que já devem ser indexadas por palavras-chave, existe um número considerável de documentos contendo dados estruturados (ou semi-estruturados) como tabelas, listas, URLs, etc. Novas pesquisas surgiram para estudar outras formas de indexação e consulta que aproveitem a estrutura encontrada em alguns destes documentos.

Os sistemas de busca por palavra-chave são bastante simples e tornaram-se a melhor opção para encontrar documentos pela internet. No entanto, não permitem ao usuário aplicar semântica aos termos da consulta, ou expressar filtros de busca. A pouca expressividade deste tipo de consulta é o principal ponto que motiva novas formas de consulta. O grande volume de dados estruturados (ou semi-estruturados) torna possível a construção de formas mais expressivas de consultas que possibilitam ao usuário especificar melhor sua busca, e conseqüentemente, tornam os resultados mais precisos.

Domínios colaborativos, tal como a *Wikipedia,* possuem padrões de *layout* e guias de estilo [WIKIPEDIA], onde todas as páginas possuem informações em uma estrutura padrão. Por exemplo, no cadastro da página de uma cidade na *Wikipedia*, algumas informações são de preenchimento obrigatório, como seu nome e estado pertencente. Estas informações serão sempre exibidas em uma tabela ao lado direito da página, chamada *infobox*.

1.3. Estrutura do trabalho

Os próximos capítulos se referem à fundamentação teórica do trabalho, Capítulo 2, seguida da apresentação da ferramenta *Find Me* Capítulo 4. A implementação do projeto é exposta no Capítulo 5 e a apresentação dos trabalhos correlacionados, no Capítulo 5.

Por fim, no Capítulo 6 serão descritos os experimentos efetuados a fim de descobrir a eficácia da ferramenta desenvolvida. No Capítulo 7 são apresentadas as conclusões obtidas e os trabalhos futuros do projeto.

2. Dados na Web

O princípio do armazenamento de dados em computadores iniciou na década de 50 através de sistemas de arquivos. Estes dados eram textos livres e não possuíam qualquer tipo de estrutura. Na década de 70 surgiram os bancos de dados relacionais, que facilitaram a leitura e escrita de dados através de linhas, colunas e tabelas. A Web surgiu na década de 90, baseada em linguagem de marcação, o HTML. Os dados armazenados em HTML são classificados em dados semi-estruturados, ou seja, não são textos totalmente livres, mas não possuem um esquema rígido como um banco de dados [MELLO et. al., 2000].

Atualmente os dados encontrados na *Web* se encontram em formatos bastante heterogêneos como o HTML, textos livres e até mesmo banco de dados gerenciados por um SGBD (disponíveis através de *Web Forms*). De forma geral, independentemente do formato, a grande maioria é acessada através de ferramentas de busca por palavra-chave.

No contexto deste trabalho, os dados na Web são classificados em três tipos [MADHAVAN et. al., 2009]:

- Não-estruturados: consiste em qualquer dado contido em um texto.
- Semi-estruturados: são os dados que possuem uma certa estrutura e estão armazenados normalmente em XML, HTML ou qualquer outra estrutura de marcação.
- Estruturados: dados contidos em um banco de dados e possuem um esquema definido.

O modelo mais utilizado para a recuperação de informação em dados não estruturados é o modelo vetorial [BAEZA-YATES e NETO, 1999]. Este modelo considera o texto como um vetor, sendo que cada termo deste texto possui um peso que determina sua relevância. O modelo vetorial é bastante eficaz para consultas por palavra-chave, muito utilizado para consulta a dados da *Web.* A indexação dos dados não estruturados é feita através de índices invertidos [BAEZA-YATES e NETO, 1999]. Os desafios da área de pesquisa a dados não estruturados englobam o desenvolvimento de algoritmos para extrair e estruturar os dados e uma interface de acesso mais sofisticada.

Os dados semi-estruturados são modelados através de modelos hierárquicos como o XML e grafos [ABITEBOUL et. al., 1999]. Como nestes dados é identificado algum tipo de estrutura, é possível utilizar linguagens de consulta mais expressivas como XQuery para o XML e *Lorel* para grafos OEM. Este tipo de dado não precisa estar associado a um esquema, como

nos bancos de dados relacionais. Por este motivo, um dos desafios das pesquisas sobre consultas a dados semi-estruturados é a capacidade de interpretar corretamente dados que poderiam ser interpretados de maneiras diferentes. Por exemplo, na Figura 2 a representação em XML de um dado que indica uma data de nascimento é feita por duas maneiras distintas: utilizando uma única *tag* como o valor "19/02/1990" e utilizando *tags* específicas para representar o dia, mês e ano.

```
1  <nascimento> 19/02/1990 </nascimento>
```

```
1  <nascimento>
2      <dia>19</dia>
3      <mês>fevereiro</mês>
4       <ano>1990</ano>
5  </nascimento>
```

Figura 2 - Mesmo dado expresso em maneiras distintas

A *Web* também armazena dados estruturados, como os dados que são acessados na *Deep Web* [MADHAVAN et. al., 2009] e dados produzidos na *Web, como o GoogleBase* [HSIEH, et. al., 2006]. O modelo destes dados segue a estrutura de um registro ou objeto que possui atributo e se relacionam entre si. As linguagens de consultas para este tipo de dados são bastante expressivas, como o SQL.

Atualmente existe a grande necessidade de conseguir integrar todas as tecnologias de consulta para poder acessar os dados da *Web* armazenados em diferentes tipos de formato em um único lugar. Para isto, surgiram várias propostas, como a consulta a dados estruturados e semi-estruturados através de palavra-chave [LI et. al., 2011] e a consulta a dados não estruturados através de consultas estruturadas [CAFARELLA et. al., 2007].

3. Find Me

Este capítulo apresenta o funcionamento do *Find Me*, uma ferramenta de busca que permite consultas estruturadas em dados encontrados em domínios wiki. A principal contribuição da ferramenta é permitir a construção de consultas mais expressivas sobre documentos em HTML na *Web*, utilizando princípios de consultas estruturadas com dados não necessariamente estruturados.

O capítulo está organizado como segue. Na Seção 3.1 é apresentada a visão geral do funcionamento do *Find Me* e as heurísticas utilizadas para identificar os índices. Na Seção 3.3 são apresentadas as etapas para a indexação dos documentos e o processo de consulta disponibilizado ao usuário final.

3.1. Visão geral

A arquitetura geral do sistema está apresentada na Figura 3. O Find Me é composto por quatro componentes principais:

- **Crawler**: este componente é encarregado de percorrer a *Web* automaticamente, identificar documentos em HTML e encaminhar para o identificador de índices. O ponto inicial da varredura do *Crawler* na *Web* é dado através de uma ou mais páginas *Web* que são denominadas sementes.

- **Identificador de índices:** recebe do *Crawler* os documentos varridos utilizando um *XML Parser* para identificar as características do documento e indexá-las de acordo com as regras definidas nas Seções 3.2.1 e 3.2.2.

- **Índice**: é encarregado de armazenar informações sobre cada um dos documentos varridos pelo *Crawler* em uma estrutura simples e que facilite o processamento de consultas.

- **Interpretador de consultas:** este componente deve receber a consulta estruturada do usuário e interpretá-la para uma consulta compatível com os campos armazenados no índice.

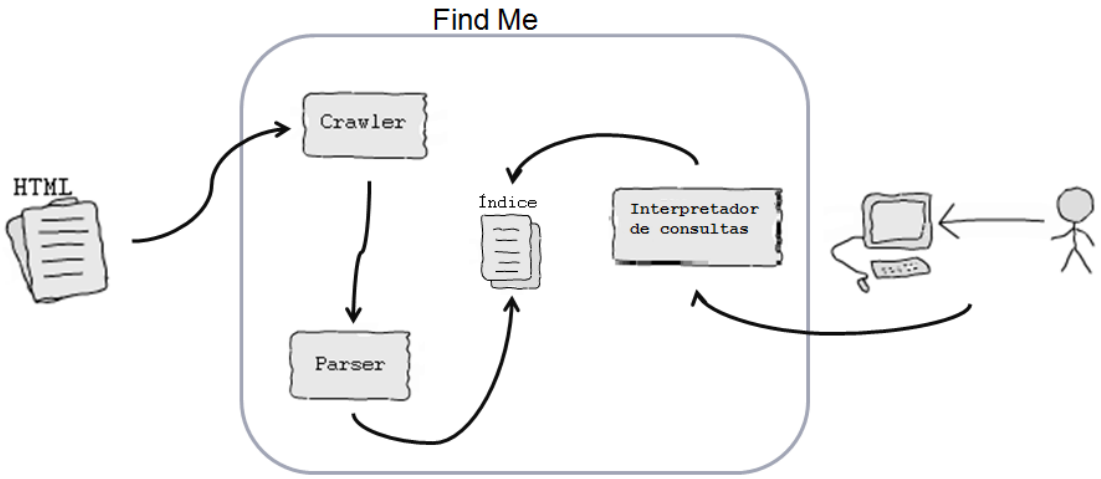


Figura 3 - Arquitetura da ferramente Find Me

3.2. Heurísticas para indexação

De forma geral, o processo de indexação de documentos HTML parte de duas premissas: A primeira premissa determina que todo documento HTML é um objeto, todo objeto pertence a alguma classe, todo objeto deve possuir atributos e todos os atributos devem possuir um valor. Um exemplo da aplicação desta premissa é representado na Figura 4.

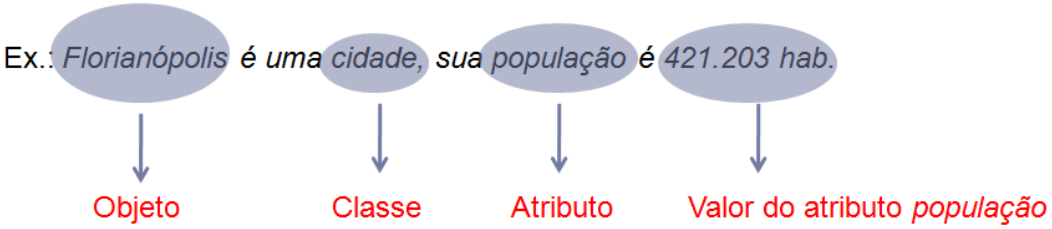


Figura 4 - Modelo de dados do *Find Me*

A segunda premissa define que os objetos se relacionam através de atributos que contêm em links que referenciam outro documento. Desta forma, um atributo pode ser atômico ou uma referência a outro objeto.

Figura 5 - Atributos do objeto *Florianópolis*. Fonte: http://pt.wikipedia.org/wiki/Florian%C3%B3polis

Para que os documentos sejam encontrados através de consultas estruturadas, é necessário identificar, além da classe deste objeto, quais atributos compõem este objeto. Por exemplo, para que a consulta "SELECT * *FROM cidades* `WHERE população > 5.000`" recupere documentos sobre cidades que possuem população maior que 5 mil, é preciso primeiro identificar que o documento a ser indexado possui informações sobre uma *cidade,* ou seja, identificar que a classe deste objeto é *cidade*. Após isso, também é necessário identificar que o documento possui um atributo *população* com um valor maior que 5 mil.

Encontrar relações do tipo "atributo-valor" no conteúdo de um documento é o principal desafio do *Find Me*. Para tanto, foram criadas algumas heurísticas que identificam tipos de documentos (as classes) e atributos. Estas heurísticas são descritas respectivamente nos itens 3.2.1 e 3.2.2. É muito importante destacar que todas as heurísticas consideradas são dependentes do conjunto de elementos da linguagem HTML, pois o objetivo do trabalho é efetuar consultas em dados semi-estruturados contidos em páginas HTML. Assim, as heurísticas são definidas através de caminhos HTML que representam o componente desejado (classe, objeto, atributo ou valor de atributo).

3.2.1. Heurísticas para determinação da classe de um objeto

Um *crawler* inicia a visita de páginas a partir de uma ou mais sementes. As sementes definidas para o *crawler* desta ferramenta foram páginas da *Wikipedia* que contêm listas de

artigos também da *Wikipedia*, conforme Figura 6 onde se encontra uma lista de municípios brasileiros.

Figura 6 - Lista de municípios. Fonte: http://pt.wikipedia.org/wiki/Lista_de_munic%C3%ADpios_do_Brasil

O *crawler*, então, visita cada uma das páginas referenciadas na lista de municípios. Os objetos são identificados através de sua URL e o nome deste objeto será o título da página, conforme exposto na Figura 7.

É importante salientar que na *Wikipedia* existem diversos artigos com o título "São Paulo" que fazem referência a assuntos diferentes, tais como estado, time de futebol, santo, etc. (tais assuntos equivalem às classes definidas pelo *Find Me*). Portanto, a identificação de objetos deve ocorre a partir de sua URL e não por seu nome.

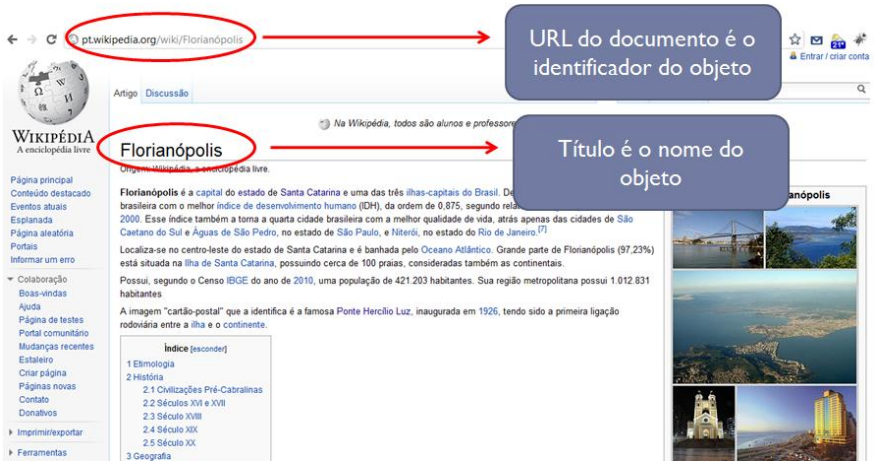


Figura 7 - Objeto Florianópolis. Fonte: http://pt.wikipedia.org/wiki/Florian%C3%B3polis

A determinação da classe dos objetos é feita através dos dados contidos na caixa informativa do documento. Caixa informativa (Figura 8), também chamada de *infobox* é um padrão dos documentos wiki, onde são informados vários atributos sobre o objeto.

Figura 8 – *Infobox* do objeto Florianópolis. Fonte: http://pt.wikipedia.org/wiki/Florian%C3%B3polis

A heurística que define a classe de um objeto parte do princípio de que o texto encontrado no título de um *infobox*, geralmente, inicia com o nome da classe do objeto. Um exemplo é a Figura 8. Considerando que a Figura 9 é representação simplificada em HTML do *infobox* da Figura 8, a classe de um objeto é obtida através do seguinte caminho na árvore DOM do HTML: `/body/table[@class='infobox']//tr//td`. Por exemplo, no caso do documento da Figura 9, a primeira linha da tabela possui o texto *Município de Florianópolis*.

```
<table class="infobox">
    <tbody>
        <tr><td>Município de Florianópolis</td></tr>
        <tr>
            <th>Distância até a capital</th>
            <td>1 692 km</td>
        </tr>
        <tr>
            <th><a href="http://...">Área</a></th>
            <td>433,317km²</td>
        </tr>
        <tr>
            <th><a href="http://...">População</a></th>
            <td>421 203 hab.</td>
        </tr>
        <tr>
            <th><a href="http://...">Mesorregião</a></th>
            <td><a href="http://...">Grande Florianópolis</a></td>
        </tr>
    </tbody>
</table>
```

Figura 9 – Código HTML do *infobox* do objeto Florianópolis (adaptado)

Para agrupar páginas de tipos iguais, é utilizada uma tabela de sinônimos. Nestes casos, mesmo quando os textos recuperados possuírem palavras diferentes, mas sinônimas, os documentos serão classificados em um mesmo tipo, desde que a sinonímia seja conhecida, como, por exemplo, as palavras "*cidade*" e "*município*". A Figura 8 mostra um *infobox* com palavras que determinam o tipo de um documento serão comparadas, como o texto `Município de Florianópolis`. A palavra *Município* indica que a classe do objeto é *cidade*.

3.2.2. Heurísticas para identificação de atributos

Como citado, todo objeto possui atributos e todos os atributos devem possuir um valor associado. O desafio deste trabalho é conseguir identificar relações entre o nome do atributo e a *string* que determina o valor do atributo em dados não estruturados.

A estrutura do *infobox* normalmente apresenta valores separados em uma tabela de duas colunas, onde a primeira coluna pode indicar o nome de um atributo e a segunda coluna o seu valor. Portanto, a heurística usada para identificar um atributo atômico é definida através do caminho HTML: `/body/table[@class='infobox']//tr/th`. Consultando a *tag* irmã da *tag* <TH>, será encontrada uma tag <TD>, onde o valor da característica é encontrado. Assim, a heurística usada para identificar o valor de um atributo é `/body/table[@class='infobox']//tr//td`.

Nas *tags* <TH>, o texto encontrado é o nome do atributo, como área, população e mesorregião. Cada *tag* <TH> possui uma *tag* irmã, a *tag* <TD> onde o seu conteúdo apresenta

o valor da característica. Portanto, seguindo as heurísticas definidas para encontrar o nome de um atributo e o seu valor, pode-se concluir que o *infobox* representado pela Figura 9 representa um objeto sobre uma cidade que está há 1.692 km de distância da capital.

Como cada atributo pode estar referenciando um novo documento (estar contido em um *hyperlink*), uma página que descreve a cidade de Florianópolis, por exemplo, tem atributos atômicos como "distância até a capital" e atributos que serão outros objetos como "população" que faz referência à página sobre a palavra "população" na *Wikipedia*. Desta forma, a heurística que determina um atributo no qual seu nome faz referência outro objeto é definida da seguinte forma no HTML: `/body/table[@class='infobox']//tr/th/a`. Da mesma forma é definida a heurística que identificados os valores dos atributos que referenciam outros objetos: `/body/table[@class='infobox']//tr//td/a`. Com base nestas heurísticas, é possível identificar que os atributos "*área*", "*população*" e "*mesorregião*" encontrados na Figura 9 não são atributos atômicos, são atributos que referenciam outros documentos da *Wikipedia* (outros objetos).

3.3. Processo de Indexação

No contexto do trabalho, todas as URLs percorridas pelo *crawler* são consideradas um novo objeto e indexadas. Os dados identificados através das heurísticas são armazenados em uma estrutura de índice que deve facilitar a consulta efetuada pelo usuário. As etapas do processo de indexação encontradas na Figura 10 são descritas conforme apresentado nas seções a seguir.

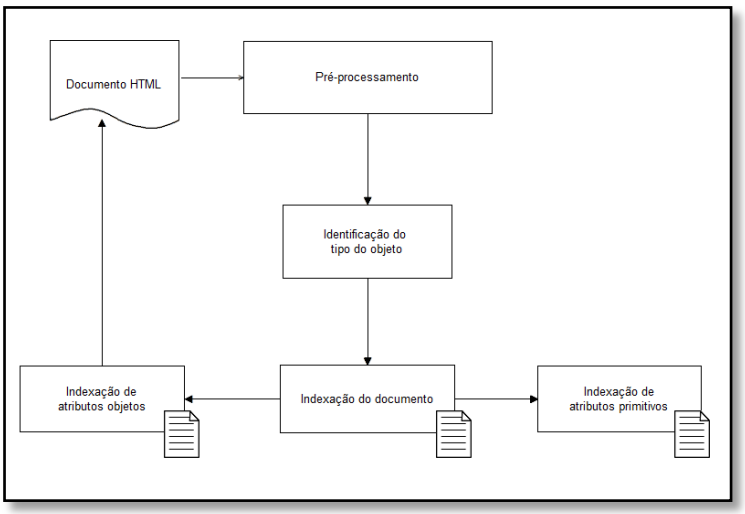


Figura 10. Etapas do processo de indexação

3.3.1. Pré-processamento

A etapa de pré-processamento consiste na transformação dos documentos recuperados pelo *crawler* em um documento XML bem formado. Um documento XML é dito "bem formado" quando segue exatamente a definição da estrutura proposta pela *W3 Consortium*[3]. A estrutura impõe regras como o fechamento de todas as *tags* abertas e o envolvimento completo de cada *tag* por seus elementos pais. Um *parser* XML necessita que todas as *tags* de um documento estejam bem formadas para poder percorrer a árvore DOM.

As páginas disponibilizadas na *Web* não garantem a estrutura de um XML bem formado, o que afeta diretamente o trabalho dos HTML *parsers*. Portanto, a primeira etapa para a indexação dos objetos é preparar o documento para que o *parser* consiga percorrer suas *tags*.

3.3.2. Identificação e indexação do objeto

Nesta etapa, são aplicadas as heurísticas para identificação de objetos, bem como suas classes (definidas na Seção 3.2.1) em cada documento visitado pelo *crawler*. Valores como a URL do objeto, seu nome e classe são salvos na tabela de índice.

As colunas que compõem a tabela de índice são descritas conforme a Tabela 1.

Tabela 1 - Descrição dos campos da tabela de índice

Campo	Descrição
ID	Código identificador único
ID_PAI	Código que identifica o documento ao qual o atributo pertence
CLASSE	É preenchido somente quando a linha for um objeto. Representa a classe do objeto
NOME	Pode ser tanto o nome de uma página, quanto o nome de um atributo
VALOR	É preenchido somente quando a linha for um atributo, representa o valor de um atributo.
NOME_URL	No caso da indexação de objetos, é a URL do documento. Para a indexação de um atributo, o campo representa a URL a qual o atributo

[3] Não é objetivo do presente trabalho, aprofundar-se sobre a estrutura proposta pela *W3 Consortium*.

	faz referência, no caso de atributos contidos em *hyperlinks*
VALOR_URL	Representa a URL a qual o valor do atributo faz referência, caso a palavra esteja contida em um *hyperlink*

A página sobre o município de São Paulo, por exemplo, na *Wikipedia* teve sua classe identificada como *Cidade* através das heurísticas da Seção 3.2.1. Um exemplo de indexação deste objeto é representado através da primeira linha da Figura 11; onde seu código, nome, URL e classe são descritos. Os campos que se encontram em branco são utilizados somente para a indexação de atributos.

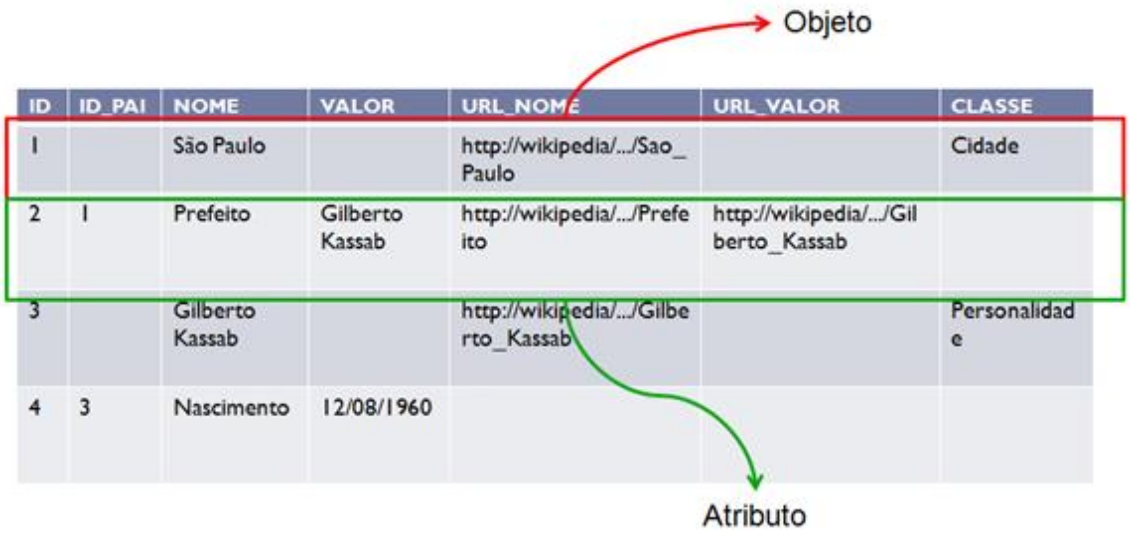


Figura 11 - Exemplo de índice

3.3.3. Identificação e indexação de atributos

Nesta etapa, a regras para identificação de atributos definidas na Seção 3.2.2 são aplicadas no documento indexado na etapa anterior. Como mencionado, há diferenças para a identificação de atributos atômicos e atributos que referenciam objetos. A identificação para cada tipo de atributo é feita conforme segue:

Atributos atômicos – Quando nem o nome do atributo nem seu valor estão contidos em *tags* "âncoras", ou seja, não possuem links para outro documento, a indexação ocorre conforme o exemplo na quarta linha da Figura 11. O campo ID_PAI é preenchido com o ID do objeto ao qual o atributo pertence. Por exemplo, o atributo "*Nascimento*" pertence ao objeto

"*Gilberto Kassab*". Portanto, o campo CD_PAI é preenchido com o código da linha onde o objeto "*Gilberto Kassab*" foi indexado. O campo NOME significa o nome do atributo, enquanto o campo VALOR deve estar preenchido com o valor correspondente a esta característica. Como a linha se refere a um atributo, o campo CLASSE não deve ser preenchido e por este atributo ser atômico, os campos URL_NOME e URL_VALOR também devem estar vazios.

Atributos/objetos – Nos casos em que o nome ou o valor de um atributo está contido em uma *tag* <A>, os campos utilizados para indexar atributos atômicos continuam sendo preenchidos da mesma forma. Porém, o que determina que o atributo seja também um objeto é a presença de alguma URL em seu nome ou valor.

Quando o identificador de índices encontra um atributo que também é um objeto, o documento representado pela URL do nome ou valor do atributo deve ser enviado para a visita do *crawler* que deve, então, iniciar um novo ciclo de indexação. Este caso é representado na Figura 11 na linha número 2, onde foi indexado o atributo de São Paulo: prefeito Gilberto Kassab. Na linha número 3 foi indexado o objeto Gilberto Kassab, tendo em vista que o atributo prefeito de São Paulo faz referência a este objeto.

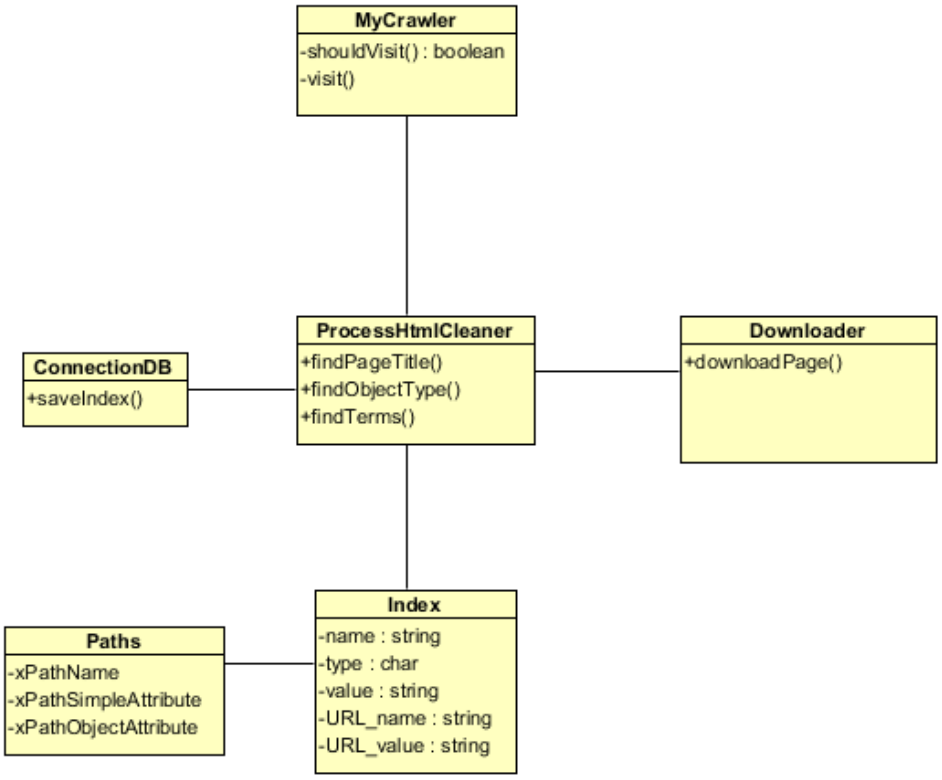
4. Implementação

Este capítulo apresenta as tecnologias utilizadas para o desenvolvimento da ferramenta *Find Me*, bem como a descrição do processo de implementação do projeto.

4.1. Projeto

O *Find Me* foi desenvolvido em duas etapas. A primeira etapa desenvolvida engloba a indexação das páginas. Nesta etapa foi utilizado um *crawler* para percorrer páginas *Web* e o *parser HtmlCleaner* [4]para percorrer as páginas a fim de identificar os termos a serem indexados posteriormente. O desenvolvimento foi feito utilizando a linguagem Java SE, através da IDE Eclipse.

A Figura 12 apresenta o diagrama de classes da etapa de indexação. A classe *MyCrawler* se encarrega de verificar as regras para a visita em uma página, bem como a chamada do *ProcessHtmlCleaner* que controla o download das páginas, reconhecimento de objetos e atributos, além da indexação. A classe *Downloader* é encarregada de salvar o HTML no servidor, enquanto uma instância da classe *ConnectionDD* é chamada para salvar o índice no banco de dados.



[4] http://htmlcleaner.sourceforge.net/

Figura 12 - Diagrama de classe da etapa de indexação

A segunda etapa de desenvolvimento foi a interface ao usuário e consulta ao índice. A interface de consulta foi desenvolvida também em Java e JSPs. Esta etapa é a interface com o usuário que se encarrega de receber a consulta em *SQL-like,* interpretá-la e traduzir para um SQL real que será executado na tabela de índice (que foi criada em um banco de dados MySQL e é explicada a seguir).

O diagrama de classes desta etapa pode ser observado na Figura 13. A classe *ConsultaAtion* é chamada quando o formulário de consulta é submetido. Esta classe é encarregada de tratar os dados vindos da página e encaminhá-los para a classe que consulta os registros na tabela de índice, a *ConnectionDB*. O resultado da consulta é retornado para a ConsultaAction que novamente efetua os tratamentos nos dados retornados e retorna para o usuário.

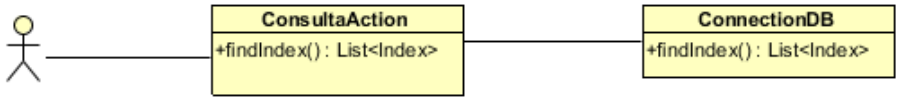


Figura 13 - Diagrama de classes da interface ao usuário

4.2. Índices

Um banco de dados relacional foi utilizado para o armazenamento do índice. Devido sua simplicidade, o índice foi desenvolvido em apenas uma tabela. Para suportar consultas com restrições através da cláusula *WHERE*, esta tabela ainda conta com um auto-relacionamento, onde o atributo referencia o objeto a que ele pertence. A tabela e seu relacionamento podem ser vistos na Figura 14.

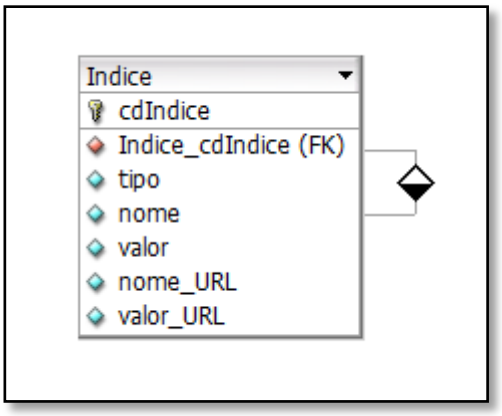


Figura 14 - Modelo lógico da tabela de índice

4.3. Interface de consulta

A interface final com o usuário é representada pela Figura 15 onde os campos necessários para a consulta são pré-definidos de acordo com o conteúdo da tabela de índice. O campo *SELECT* possui uma lista de todos os nomes dos atributos encontrados nos documentos indexados mais uma lista com todas as classes de documentos identificados. O campo FROM apresenta a classe de documento onde a consulta será efetuada, enquanto os valores possíveis para a cláusula WHERE são também os atributos indexados. Apenas o campo referente ao valor a ser comparado na cláusula *WHERE* é aberto para digitação livre do usuário.

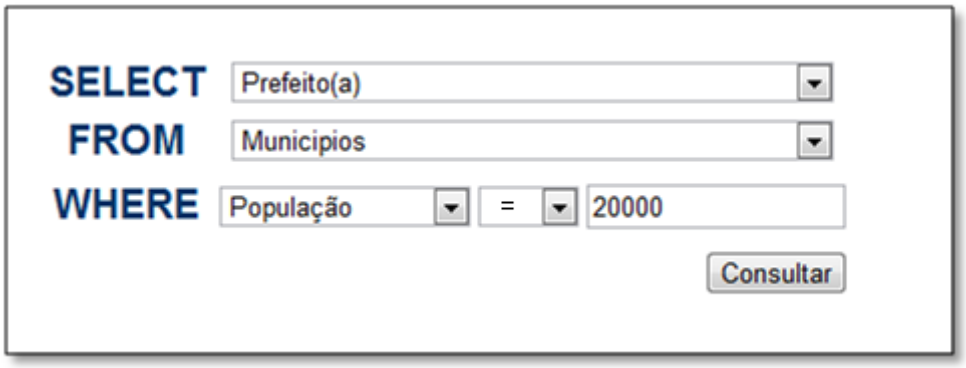


Figura 15 - Interface de consulta

A consulta informada nos campos da Figura 15 indica que serão recuperados todos os prefeitos de municípios que possuem população menor que 20.000 habitantes. Em outras palavras, serão recuperados todos os valores de atributos de nome "*Prefeito(a)*" pertencentes a um documento de classe igual a "Cidade" que possua o atributo "*População*" com um valor menor que 20.000.

4.4. Processo de consulta

A consulta informada pelo usuário no estilo do SQL não pode ser efetuada diretamente ao índice. Na verdade, todas as consultas informadas no *Find Me* precisam ser interpretadas e adaptadas para o contexto da tabela de índice. Esta etapa deve ser transparente ao usuário. Um exemplo de adaptação de consulta é representado na Figura 16.

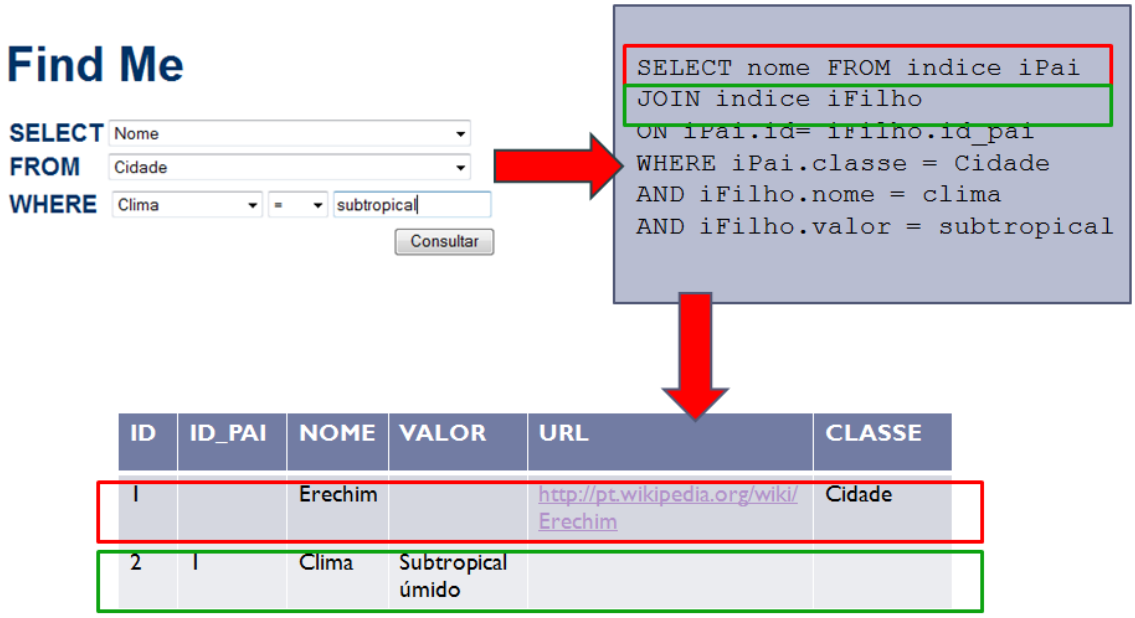


Figura 16 - Processo de consulta

No caso apresentado na Figura 16, é necessário consultar na tabela de índice linhas referentes aos objetos que pertençam à classe "Cidade". Além disso, para cada objeto encontrado, deve-se consultar linhas referentes a atributos deste objeto que tenham nome e valor igual a "clima" e "subtropical", respectivamente. Esta ligação é feita através de um auto-relacionamento na tabela de índice, no qual as linhas que representam atributos apresentam chave estrangeira para o objeto a quem pertencem. Portanto, para cada consulta é necessário efetuar um *join* entre o índice que representa o objeto e o índice que representa seu atributo.

5. Trabalhos relacionados

Este capítulo apresenta alguns trabalhos que propõem abordagens para a elaboração de consultas estruturadas na Web. A tentativa de identificar ou organizar os dados da Web é um tema que vem sendo bastante explorado nos últimos tempos [BAEZA-YATES e NETO, 1999], fato este que justifica trabalhos como o *DBPedia* [AUER et. al. 2007], o desenvolvimento da linguagem *Squeal* [SPERTUS e STEIN, 2000] e a pesquisa de novas técnicas de indexação. Os principais trabalhos relacionados a este projeto estão listados conforme segue.

5.1. Mesa: A Search Engine for Querying Web Tables

A presença de algum tipo de estrutura em documentos textuais permite a aplicação de técnicas mais sofisticadas de recuperação de informação, como a apresentada na ferramenta Mesa [MERGEN et al, 2008], que é um motor de busca que permite acesso aos dados estruturados em tabelas HTML. *Mesa* possibilita a consulta de dados previamente indexados tanto com palavras-chave quanto com *queries* formatadas em SQL. Ao efetuar a consulta `select title, gross from movie` são retornadas todas as tabelas HTML que podem satisfazer a relação entre os atributos *title* e *gross* com o objeto *movie*. Quando a busca é feita por palavras-chave, o *Mesa* simplesmente retorna as páginas que possuem tabelas Web com o termo consultado. Os dois tipos de consulta ainda podem ser agrupados, buscando tabelas que possuem os termos do SQL, onde a palavra-chave está presente.

O resultado da consulta é exibido em um sumário com todas as tabelas resultantes e a URL de onde esta tabela foi extraída. Ao disponibilizar o link para a página original do conteúdo, o sistema permite que o usuário navegue pelo real contexto da tabela podendo alcançar novas informações que possuem relação com a busca. A consulta também permite cláusulas de condição, como *ano > 1990*, onde a busca resulta em tabelas que possuam pelo menos uma linha com este valor. Neste caso, o *Mesa* utiliza os dados sumarizados para comparar com a condição, não se fazendo obrigatório o acesso a página original na Web.

Os autores utilizaram um *crawler* específico para acessar páginas que possuem tabelas no contexto de filmes. Como regra para a indexação das páginas, o documento deve possuir pelo menos uma tabela com as três colunas: *title*, *film* e *movie*. O *crawler* iniciou os acessos em uma página do *Wikipedia* sobre os melhores filmes americanos e parou após a recuperação de 400 páginas; ao todo 993 tabelas HTML foram extraídas. Após a recuperação das páginas, um *parser* é utilizado para extrair o esquema da tabela recuperando o nome das colunas (identificadas pela tag TH). *Mesa* indexa as tabelas HTML em uma tabela de índice, que é posteriormente utilizada para comparar com os atributos da consulta. O resultado da consulta

apresenta, em uma única página, todas as tabelas encontradas na tabela de índice, porém o usuário também possui a opção de ser direcionado para a página original desta tabela. A ferramenta também utiliza um documento de índice que auxilia a consulta por palavras-chaves. A necessidade de formulação de uma consulta em SQL foi apontada como limitação da ferramenta, visto que informar palavras-chave seria muito mais fácil para o usuário, porém, pouco expressiva. Como trabalhos futuros, os autores sugerem a pesquisa de novas interfaces de consulta que equilibrem expressividade e simplicidade de uso.

5.2. Squeal

O Squeal é uma linguagem de consulta desenvolvida para recuperar informações de páginas HTML. O foco da consulta efetuada através do Squeal é a estrutura do documento como listas e hiperlinks que fazem referência a outro documento. Esta linguagem é preparada para responder perguntas como "*Quais páginas estão sendo referenciadas tanto pelo Yahoo quanto pelo Netscape Netcenter?*". A linguagem é baseada em uma estrutura semelhante ao SQL, o que beneficia programadores que já conhecem a linguagem de consulta a banco de dados.

Apesar da utilização de uma linguagem similar ao SQL, o usuário não percebe que o processador da linguagem *Squeal* apenas trata a *Web* como um banco de dados relacional. Um esquema pré-definido, inspirado nos esquemas de bancos de dados relacionais, determina uma estrutura com tabelas e campos que poderão ser utilizados na consulta. Este esquema é composto por tabelas como *page*, *tag* e *link*. A tabela *page*, por exemplo, descreve uma página *Web. Esta tabela* possui os campos:

- *url*: URL da página
- *contents*: texto contido na página
- *bytes*: tamanho em bytes da página
- *when*: data de quando a página foi recuperada

Com base nesta informação, para consultar o texto de uma página que possui a URL http://www9.org, por exemplo, através do *Squeal*. A consulta a ser formada deverá ser a seguinte:

```
SELECT contents
FROM page
WHERE url="http://www9.org";
```

Os autores desta linguagem ressaltam que a consulta formatada através de um dialeto em SQL pode ser confuso para pessoas que não conhecem a sintaxe da linguagem. Por causa

dos campos descritos em inglês, usuários que não familiarizadas com a língua também encontrariam dificuldades.

5.3. SPARQL

RDF (Resource Description Framework) [BRICKLEY e GUHA, 2004] é um modelo de metadados destinado à representação de dados para *Web* semântica. A representação dos dados em RDF é feita através de grafos, onde seus nodos são tuplas (assunto – predicado – objeto) e suas arestas os relacionamentos entre as tuplas, como na Figura 17:

```
1  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
2      xmlns:ex="http://www.example.org/">
3      <rdf:Description rdf:about="http://www.example.org/vincent_donofrio">
4          <ex:starred_in>
5              <ex:tv_show rdf:about="http://www.example.org/law_and_order_ci" />
6          </ex:starred_in>
7      </rdf:Description>
8      <rdf:Description rdf:about="http://www.example.org/the_thirteenth_floor">
9          <ex:similar_plot_as rdf:resource="http://www.example.org/the_matrix" />
10     </rdf:Description>
11 </rdf:RDF>
```

Figura 17 - Tuplas descritas em RDF

O exemplo acima descreve que o ator Vincent Donofrio (linha 3) estrelou o programa de TV *Law and Order* (linhas 4 a 6) e que *Matrix* (linha 9) é similar ao *Thirteenth Floor* (linha 8).

O SPARQL [PRUD'HOMMEAUX e SEABORNE, 2005] é uma linguagem *SQL-like*, bastante expressiva, utilizada para efetuar consultas a dados representados em RDF. Assim como o próprio RDF, também é uma recomendação da W3C.

O exemplo abaixo mostra como é feita uma consulta SPARQL para encontrar o título de um livro através de um grafo de dados em RDF. A consulta consiste em duas partes: a cláusula SELECT identifica as variáveis que aparecerão no resultado da consulta e a cláusula WHERE prevê as restrições que serão comparadas com o grafo de dados.

Dados consultados:

```
<http://example.org/book/book1>  <http://purl.org/dc/elements/1.1/title>
"SPARQL Tutorial"
```

Consulta:

```
SELECT ?title
WHERE
```

```
{
<http://example.org/book/book1><http://purl.org/dc/elements
/1.1/title> ?title .
}
```

Resultado da consulta:

title
"SPARQL Tutorial"

5.4. DBPedia

DBpedia é um projeto que busca extrair os dados estruturados da *Wikipedia* e torná-los públicos e padronizados utilizando RDF. O obetivo do projeto é recuperar informações relevantes em diferentes documentos da *Wikipedia* e alimenta arquivos RDF que descrevem os objetos contidos nestes documentos. A Figura 18 é uma representação de um arquivo RDF criado pelo *DBPedia*. Várias informações recuperadas e pertencentes ao objeto "Rio de Janeiro" foram salvas no arquivo. Na Figura 18 podem ser observados os *labels* encontrados para este objeto, bem como o apelido e listas sobre pessoas que nasceram e morreram na cidade, por exemplo.

rdfs:label	▪ Rio de Janeiro ▪ Rio de Janeiro ▪ Río de Janeiro ▪ Rio de Janeiro ▪ Rio de Janeiro ▪ リオデジャネイロ ▪ Rio de Janeiro ▪ Rio de Janeiro (stad) ▪ Rio de Janeiro ▪ Rio de Janeiro (cidade) ▪ Рио-де-Жанейро ▪ Rio de Janeiro ▪ 里约热内卢
owl:sameAs	▪ http://linkedgeodata.org/triplify/node/34582422#id ▪ freebase:Rio de Janeiro (cidade) ▪ http://sws.geonames.org/3451190/ ▪ http://data.nytimes.com/N94093083678682840 61
foaf:homepage	▪ http://www.rio.rj.gov.br/
foaf:name	▪ Rio de Janeiro ▪ Município do Rio de Janeiro ▪ The Municipality of Rio de Janeiro
foaf:nick	▪ Cidade Maravilhosa ("The Marvelous City") or simply Rio
foaf:page	▪ http://en.wikipedia.org/wiki/Rio_de_Janeiro
is dbpedia-owl:almaMater of	▪ dbpedia:Milton_Thiago_de_Mello
is dbpedia-owl:binomialAuthority of	▪ dbpedia:White-thighed_Swallow
is dbpedia-owl:birthPlace of	▪ dbpedia:Augusto_Boal ▪ dbpedia:Amon_Tobin ▪ dbpedia:Peter_Medawar ▪ dbpedia:Heitor_Villa-Lobos ▪ dbpedia:Joaquim Maria Machado de Assis

Figura 18 - RDF sobre Rio de Janeiro criado pelo DBPedia. Fonte: http://dbpedia.org/page/Rio_de_Janeiro

Uma grande vantagem de recuperar estas informações e salvar os dados em RDF é a padronização e estruturação de dados na *Web*, desta forma contribuindo para a *Web semânt*ica. Outra vantagem desta proposta é a possibilidade da utilização da linguagem SPARQL para consultar estas informações. Como já citado na Seção 5.3, o SPARQL permite maior expressividade em consultas.

5.5. WikiQuery

A *WikiQuery* [NGUYEN et. al. 2010] é uma proposta que permite consultas simples à *Wikipedia,* porém mais expressivas, como a SPARQL e a SQL. O diferencial desta proposta para as demais ferramentas de consultas já existentes é a aplicação semântica na recuperação de informação de acordo com os termos da consulta informada. Além disso, o *Wikiquery* considera os relacionamentos através de links entre os documentos wiki, sendo possível retornar informações que estão presentes em dados espalhados em mais de um documento. Por exemplo, a consulta "*atores que atuaram em filmes que foram dirigidos por Steven Spielberg*".

O modelo usado pelo *WikiQuery* foi inspirado no modelo BANKS [BHALOTIA, 2002], uma proposta que modela banco de dados relacionais em um grafo, permitindo consultas a dados estruturados através de palavras-chave. O *WikiQuery* utiliza a modelagem em grafos para relacionar os documentos da *Wikipedia* entre si. Este trabalho utiliza um grafo, chamado *data graph*, para modelar todos os documentos com seus relacionamentos, onde cada nodo é um documento e suas arestas são os relacionamentos através de links com outros nodos. Outro grafo modelado, chamado de *schema graph*, possui tipos de documentos/entidades como nodos e as arestas são os relacionamentos possíveis entre cada tipo de documento.

Os nodos do *data graph* e *schema graph* possuem pesos cujos valores dependem de sua relevância. O peso de cada nodo do *data graph* é calculado de acordo com o seu tipo, ou seja, de acordo com o peso do nodo encontrado no *schema graph* referente ao seu tipo. Porém, uma entidade pode pertencer a mais de um tipo, ou seja, pode referenciar a mais de um nodo no *schema graph*. Por este motivo, os pesos dos nodos do *data graph* são recalculados de acordo com o escopo da consulta, pois depende do peso de seu tipo. Desta forma é garantida a semântica da consulta. Por exemplo, caso a consulta seja "*filmes que o diretor é Steven Spielberg*", e considerando que a entidade "Steven Spielberg" pertence aos tipos "Diretor", "Produtor" e "Roteirista", nesta consulta, o peso da entidade encontrado no *data graph* levará em consideração o peso do tipo "Diretor" do *schema graph.*

Não é objetivo do *WikiQuery* oferecer interface simples ao usuário. As consultas são efetuadas através de *c-queries* que obrigam o usuário a conhecer como a consulta deve ser escrita. Por exemplo, a *c-query* da consulta "*filmes que o diretor é Steven Spielberg*" é ("filme", Diretor(nome=Steven Spielberg)).

5.6. Análise comparativa

Trabalhos como o *Mesa* recuperam somente informações contidas em uma estrutura definida, como tabelas do HTML. O *Find Me* busca ir além de tabelas e consultar dados semi-estruturados.

Uma diferença entre o proposto trabalho e o *Squeal* é a recuperação contextualizada do conteúdo das páginas HTML, enquanto o *Squeal* preocupa-se em recuperar informações referentes à estrutura de um documento HTML, como suas *tags* e seus relacionamentos. A proposta deste presente trabalho é consultar também de forma expressiva o texto contido nos documentos. Ao contrário do *Squeal*,neste trabalho não há um esquema pré-definido, sendo necessário inferir em uma consulta semelhante ao SQL, tabelas e campos de acordo com o conteúdo encontrado na *Web*.

O SPARQL, assim como o *Squeal*, é uma linguagem de consulta que é efetuada a dados encontrados na *Web*, enquanto neste trabalho existe a proposta de um modelo de dados, onde documentos são objetos que possuem atributos e que se relacionam. O objetivo principal deste trabalho é possibilitar consultas mais expressivas a dados encontrados nos documentos *Web*. Esta expressividade é garantida através do SPARQL, porém, esta linguagem necessita que os dados estejam representados em um documento RDF.

O *DBPedia* é um trabalho com objetivos semelhantes a este trabalho. Porém, a proposta do *DBPedia* consiste em representar através de RDF os dados recuperados. Por este motivo, as consultas efetuadas a estes dados poderão ser bastante expressivas caso seja utilizada uma linguagem de consulta como o SPARQL. Porém, é também objetivo deste trabalho permitir ao usuário uma interface simples para consulta. Este objetivo é cumprido através de uma interface que auxilia o usuário a montar consultas no formato SQL-like.

Possibilitar a construção de consultas mais expressivas é também o objetivo principal do *WikiQuery* que utiliza um modelo de várias camadas de grafos para dar pesos à relevância dos documentos e seus relacionamentos. Não é objetivo do presente trabalho modelar dados da *Wikipedia* em grafos. A modelagem proposta é bastante simples, fato que pode sair em vantagem com relação a consultas efetuadas em grafos.

Um dos objetivos do presente trabalho é oferecer uma interface de consulta simples que auxilie o usuário. Além disso, não é objetivo deste trabalho a elaboração de uma linguagem de consulta, como feito no *WikiQuery* que obriga que suas consultas sejam efetuadas através de palavras-chave ou *c-queries.* A estrutura deste trabalho é mais simples, sem a necessidade de implementar um modelo em grafos, por exemplo.

6. Experimentos

Neste capítulo, são expostos os experimentos realizados após o desenvolvimento do *Find Me* de forma a comprovar a eficácia da proposta. Com base nos experimentos obtidos foi possível avaliar a confiabilidade dos resultados obtidos através de medidas de revocação, precisão e F-value [BAEZA-YATES e NETO, 1999].

6.1. Medidas utilizadas

As seguintes medidas, comuns em experimentos que avaliam recuperação de informação, foram utilizadas:

- Revocação: a medida de revocação calcula a proporção entre o número de objetos recuperados relevantes para a consulta informada, em comparação com o número total da coleção de objetos relevantes. Esta medida revela, em um intervalo de zero a um, se todos os objetos relevantes a uma dada consulta foram recuperados. O valor da medida de revocação, quando igual à 1, determina que todos os objetos relevantes da coleção foram recuperados pela ferramenta de busca.
- Precisão: a medida de precisão, também expressada em um intervalo de zero a um, calcula a proporção de objetos relevantes entre todos os objetos recuperados. Quando a precisão de um resultado é igual a 1, todos os objetos recuperados são relevantes, o que significa que nenhum objeto irrelevante foi retornado na consulta.
- F-Measure: média harmônica entre revocação e precisão, que indica se o resultado obtido através de alguma técnica de recuperação de informação possui o maior número possível de objetos relevantes, ao mesmo tempo em que os objetos que não são relevantes para a consulta não são recuperados.

A Tabela 3 apresenta a forma de cálculo de cada uma das medidas.

Tabela 2 - Medidas de revocação, precisão e F-value

Medidas	
Revocação (R)	Objetos recuperados / Total de itens da coleção
Precisão (P)	Objetos relevantes recuperados / Total de objetos recuperados
F- Measure	$\frac{2 \times P \times R}{P + R}$

6.2. Configuração dos testes

Como mencionado na Seção 3.3, os registros que compõem a tabela de índice foram recuperados através de um *crawler* que percorre páginas HTML e de um identificador de índices que identifica os objetos e atributos dos objetos, conforme as heurísticas definidas em 3.3.2 e 3.3.3. Os objetos recuperados foram classificados durante a etapa de indexação.

Para a etapa de testes foram selecionados três domínios distintos, referentes a três classes: *Personalidade*, *Cidade* e *Filme*. O critério definido para escolha dos domínios foi o maior número de objetos classificados para determinada classe.

Os testes foram realizados em uma base de dados com os seguintes registros, descritos na Tabela 3.

Tabela 3 - Dados utilizados nos experimentos

Classe	Número de objetos	Número de atributos	Total	Número de consultas
Filme	303	1.987	2.290	13
Personalidade	313	1.762	2.075	9
Cidade	628	10.439	11.067	19
Outras classes	918	6.290	7.208	-
Total	2.162	20.478	22.640	41

Para cada domínio, uma bateria de testes foi realizada. Cada teste equivale a uma consulta a determinada classe. Por exemplo, para o domínio *Cidade* um teste efetuado foi a execução da seguinte consulta: "*SELECT nome FROM Cidade WHERE fuso horário = UTC-4"*. A mesma consulta foi verificada manualmente em todos os objetos indexados na base de dados que correspondem à classe *Cidade*.

6.3. Resultados

Nesta seção, são apresentados os resultados obtidos através dos testes realizados. Os resultados estão divididos em três grupos de acordo com os três domínios escolhidos para o teste: *Filme*, *Personalidade* e *Cidade*. Para cada consulta testada, foram obtidos os valores de revocação, precisão e F-Measure a fim de validar as heurísticas definidas para definição de objeto, atributo e valor.

Tabela 4 - Domínio *Filme*: medidas de precisão, revocação e F-value

	Consulta	Recall	Precision	F-value
1	Diretor = Steven Spielberg	1	1	1
2	Distribuição = Universal Pictures	1	1	1
3	Elenco original = Leonardo DiCaprio	1	1	1
4	Elenco original = Tom Hanks	0.8	1	0.888888889
5	Estúdio = 20th Century Fox	1	1	1
6	Gênero = Comédia	0.891304348	0.976190476	0.931818182
7	Gênero = Drama	0.835978836	0.957575758	0.892655367
8	Gênero = Musical	0.592592593	1	0.744186047
9	Gênero = Romance	0.444444444	1	0.615384615
10	Idioma original = Inglês	0.996428571	0.985865724	0.991119005
11	Música = James Horner	1	1	1
12	Produção = Brian Grazer	1	1	1
13	Roteiro = James Cameron	1	1	1

A Tabela 4 lista as consultas executadas no *Find Me* com base no domínio *Filme*. Considera-se que cada consulta esteja selecionando o nome de determinado filme e cada linha da Tabela 4 é o conteúdo da cláusula *WHERE* da consulta. Por exemplo, a consulta completa para a linha 1 da Tabela 4 é representada da seguinte forma: "*SELECT nome FROM Filme WHERE Diretor = Steven Spielberg"*.

As medidas presentes na Tabela 4 estão representas na Figura 19.

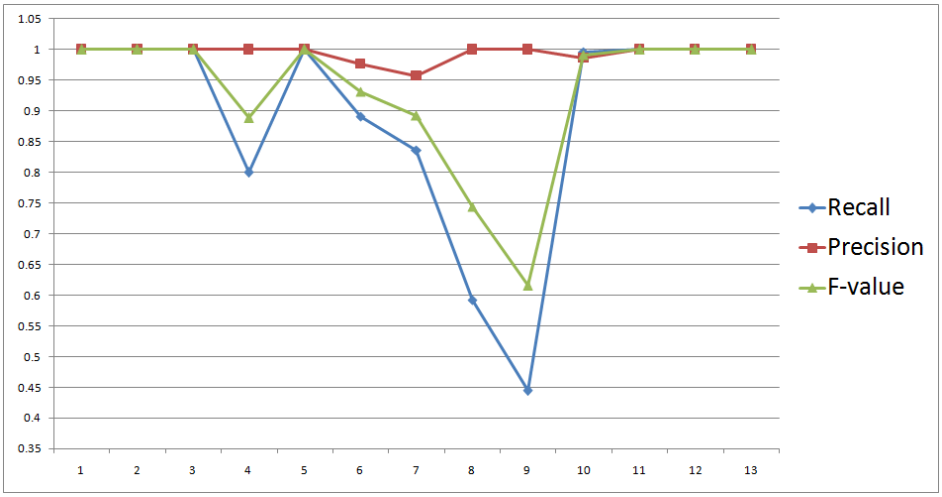


Figura 19 - Domínio *Filme*: medidas de precisão, revocação e F-value

Para o domínio de filmes, treze consultas foram efetuadas. A média total de precisão dos resultados foi 0.993817843 e a média total de revocação, 0.889288369. Com base na precisão dos resultados, pode-se concluir que na maior parte das consultas, os resultados obtidos foram de grande relevância para a consulta. Os resultados das consultas 4, 8 e 9, entretanto, apresentam revocação um pouco mais baixa. No caso da consulta 9, por exemplo, foi observado que muitos objetos que possuem o atributo "Gênero" igual a "Comédia" não foram recuperados.

As consultas que apresentaram valores de revocação mais baixos indicam que muitos objetos também relevantes para a consulta não foram recuperados. Estes objetos não recuperados ajudam a identificar algumas limitações da ferramenta. Por exemplo, as consultas de números 7, 8 e 9 que correspondem a consultas sobre o elenco e gênero do filme, obtiveram revocação menor que as demais consultas. Esta medida foi mais baixa, pois atualmente a ferramenta não está totalmente preparada para indexar atributos multi-valores, conforme exposto na Figura 20. No caso de um objeto que representa o filme *Titanic,* o valor para os atributos "Elenco original" e "Gênero" possuem mais de um valor, que estão listados um abaixo do outro. No processo de indexação, o *Find Me* identificou apenas o primeiro item de cada lista, ou seja, indexou apenas o valor "Leonardo DiCaprio" para o atributo "Elenco original" e para o atributo "Gênero", apenas o valor "Romance". No caso de atributos multi-valorados onde os valores encontram-se lado a lado, geralmente separados por vírgula, o *Find Me* os indexa como um único valor.

Figura 20 – *Infobox* sobre o filme *Titanic*. Fonte: http://pt.wikipedia.org/wiki/Titanic_(1997)

Para os testes efetuados no domínio *Personalidade* foram utilizadas as consultas apresentadas na Tabela 5. Neste domínio, a revocação apresentou média de 0.823525619, enquanto a precisão chegou a média de 0.959987727. Portanto, neste escopo também foi identificado que grande parte dos objetos relevantes foi recuperada. Em contrapartida, alguns objetos relevantes não foram recuperados.

Tabela 5 - Domínio *Personalidade*: medidas de precisão, revocação e F-value

	Consulta	Recall	Precision	F-value
1	Campo(s) = Física	0,916666667	1	0,956521739
2	Campo(s) = Política	0,9	0,9	0,9
3	Gênero musical = Pop	0,882352941	0,9375	0,909090909
4	Morte = Setembro	0,909090909	0,909090909	0,909090909
5	Nacionalidade = Estadunidense	0,171428571	1	0,292682927
6	Nascimento = Fevereiro	0,869565217	1	0,930232558
7	Nascimento = Outubro	0,818181818	0,931034483	0,870967742
8	Ocupação = Ator	0,944444444	0,962264151	0,953271028
9	Olhos = castanhos	1	1	1

Conforme a Figura 21, pode-se observar que a consulta número 5 obteve revocação muito baixa. Este valor é um indício de mais um ponto na etapa de indexação que pode ser aprimorado: o atributo “Nacionalidade” geralmente possui uma imagem da bandeira do país de origem antes da descrição de seu valor, conforme ilustrado na Figura 22. Durante a etapa de indexação, o *Find Me* indexou apenas um valor vazio referente à *tag* da bandeira e ignorou a *tag* conseqüente, onde o real valor estaria presente.

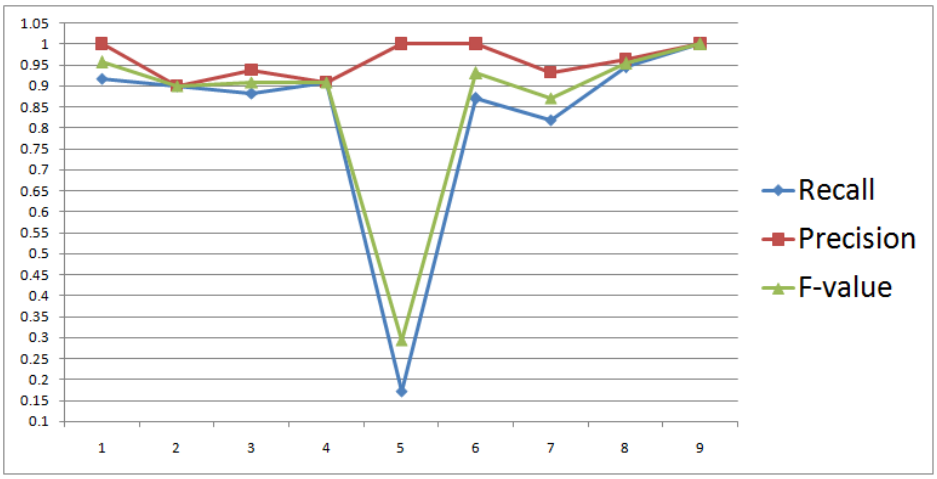


Figura 21 - Domínio *Personalidade*: medidas de precisão, revocação e F-value

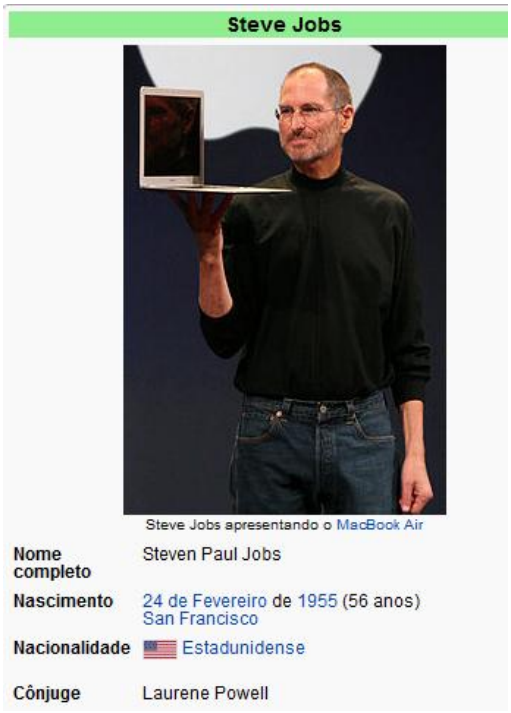


Figura 22 - *Infobox* sobre a personalidade Steve Jobs. Fonte: http://pt.wikipedia.org/wiki/Steve_jobs

Os resultados para os testes no domínio *Cidade* foram mais homogêneos que os testes realizados nos demais domínios, conforme dados expostos na Tabela 6. Com média de precisão igual a 0.95073984 e revocação igual a 0.987387541, foi possível identificar que quase todas as consultas efetuadas recuperaram objetos relevantes e não incluíram objetos irrelevantes no resultado.

Tabela 6 - Domínio *Cidade*: medidas de precisão, revocação e F-value

	Consulta	Recall	Precision	F-value
1	Aniversário = Março	1	1	1
2	Aniversário = Outubro	1	1	1
3	Clima = Tropical	0.543046358	0.543046358	0.703862661
4	Fundação = Agosto	0.923076923	0.923076923	0.96
5	Fundação = Dezembro	0.989361702	0.989361702	0.994652406
6	Fundação = Janeiro	0.953488372	0.953488372	0.976190476
7	Fuso horário = UTC-3	0.990619137	0.977657765	0.988764045
8	Fuso horário = UTC-4	1	0.977777778	0.988764045
9	Prefeito = Antônio	0.916666667	0.916666667	0.956521739
10	Prefeito = José	0.983606557	0.983606557	0.991735537
11	Prefeito = Silva	0.964285714	0.897783251	0.947368421
12	Região metropolitana = Porto Alegre	0.909090909	0.909090909	0.952380952
13	Região metropolitana = São Paulo	1	0.956521739	0.977777778
14	UF= Bahia	1	1	1
15	UF = Minas Gerais	0.97	0.95040404	0.974874372
16	UF = Paraná	1	1	1
17	UF = Rio Grande do Sul	0.968253968	0.952636969	0.976
18	UF = Santa Catarina	0.971428571	0.917460317	0.957746479
19	UF = São Paulo	0.981132075	0.981132075	0.99047619

Conforme a Figura 23, pode-se observar que apenas uma consulta apresentou valores muito distantes da média. A consulta número 3 recuperou muitos objetos irrelevantes e revelou uma limitação na consulta aos objetos indexados. Isto aconteceu ao consultar cidades que possuem o valor do atributo "Clima" igual a "Tropical". Manualmente, foram encontradas 164 cidades que possuem clima tropical, porém, o *Find Me* recuperou quase o triplo de objetos: 302. O resultado deve-se ao fato de que os objetos irrelevantes recuperados são referentes a cidades que possuem clima como sub-tropical, conforme exposto na Figura 24 onde nenhum registro possui clima tropical. Atualmente o *Find Me* não possui uma técnica de similaridade de consultas suficientemente boa para distinguir estes valores.

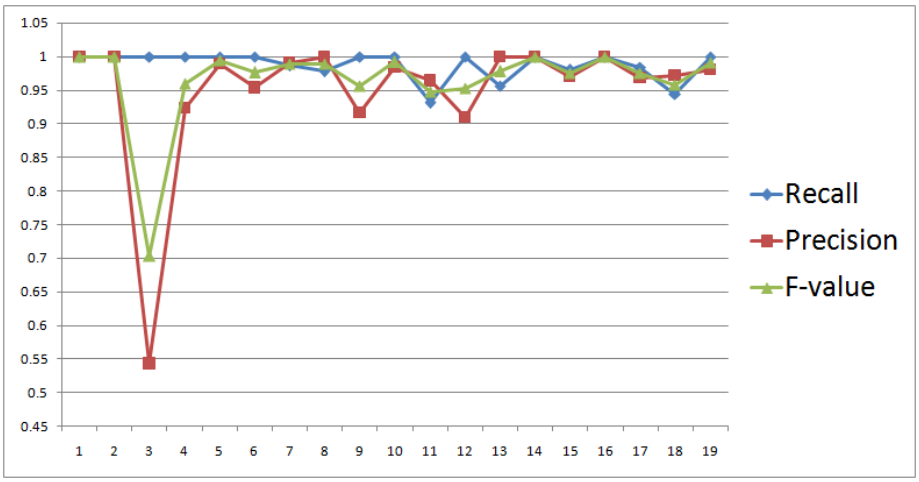


Figura 23 - Domínio *Cidade*: medidas de precisão, revocação e F-value

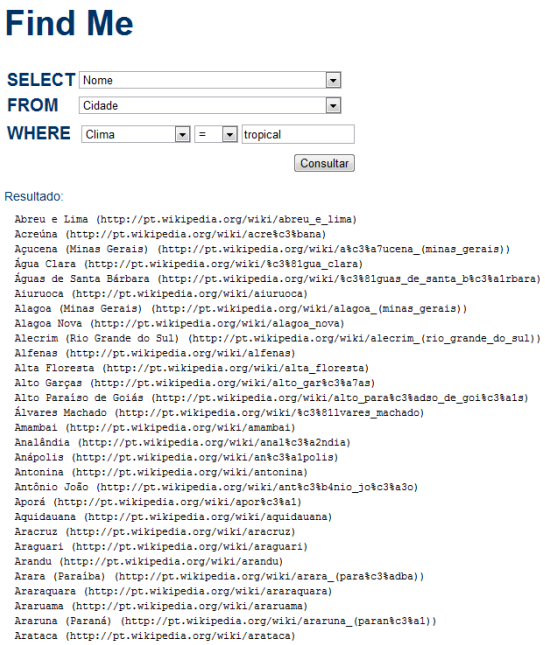


Figura 24 - Resultado da consulta "*SELECT nome FROM Cidade WHERE clima = tropical*"

O resultado dos experimentos realizados sob os domínios *Filme*, *Personalidade* e *Cidade* indica que para os três domínios testados, o *Find Me* recupera grande parte dos documentos que respondem as consultas, de acordo com a média de F-value aproximadamente igual a 0.917033177. Porém, também com base dos dados obtidos, foram identificadas algumas falhas de indexação, como visto em algumas consultas do domínio *Filme* e na consulta número 5 do domínio de *Personalidade*. Também foi identificado que a consulta por similaridade pode ser aprimorada, de acordo com a consulta de número 3 no domínio de *Cidade*.

7. Conclusões e Trabalhos futuros

Atualmente os mecanismos de consultas utilizados na *Web* utilizam palavras-chave para recuperar registros relevantes. Existe a necessidade do estudo de novas técnicas de recuperação de informação, mais sofisticadas ou mais expressivas, que levem em consideração o contexto dos dados encontrados nos documentos *Web*.

Este trabalho propõe um método para consultas *Web* que possa varrer documentos HTML e indexar dados semi-estruturados, tratando-os como objeto, atributo e valor de atributo dentro de um documento HTML. Como vantagem às demais técnicas de recuperação de informação atuais, foi também proposto ao trabalho, uma interface de consulta amigável, em que usuários mais leigos possam descrever suas consultas de forma mais expressivas. Esta interface deve possibilitar acesso a dados semi-estruturados através de consultas tipicamente formatadas para banco de dados relacionais, bem como a SQL. A idéia principal da proposta é considerar que todo documento HTML é um objeto, todo objeto pode possuir atributos e que todo atributo deve possuir algum valor. Além disso, os relacionamentos entre objetos são identificados através de links encontrados em seus atributos.

Para testar a eficácia da proposta, foi desenvolvida uma ferramenta para consultas *Web* chamada *Find Me*. O *Find Me* é composto por quatro componentes principais, conforme exposto no Capítulo 3: (i) um *crawler* que percorre documentos *Web* pertencentes ao domínio *Wikipedia*; (ii) um identificador de índices que recebe os documentos encontrados pelo *crawler* e identifica os termos que deverão ser indexados; (iii) um índice, representado por uma estrutura de tabela, atualmente armazenado em banco de dados relacional e (iv) um interpretador de consultas. A etapa de *parsing* considera algumas heurísticas para identificar objetos e seus relacionamentos, atributos e valores. Estas heurísticas foram definidas na Seção 3.2.1 e 3.2.2.

Como resultado final, alguns experimentos foram efetuados a fim de verificar a confiabilidade do resultado obtido através da proposta implementada no *Find Me*. Com base nas medidas de precisão dos resultados, foi possível verificar que a média de objetos retornados através da ferramenta foi superior a 95%, porém, a média de revocação dos resultados foi pouco mais que 90%, isto significa que quase 10% dos objetos recuperados não foram relevantes à consulta. Algumas limitações foram identificadas a partir deste resultado e incluídas aos trabalhos futuros.

Os principais trabalhos futuros identificados neste projeto são listados conforme segue:

***Joins* entre objetos -** um usuário pode desejar efetuar uma consulta como "*Personalidade de nacionalidade estaduniense que more na cidade de São Paulo*", em que seria necessário efetuar a junção dos objetos personalidade e cidade. Portanto, um trabalho futuro de grande relevância para o *Find Me* é o desenvolvimento de novos operadores de consulta que identifiquem diferentes objetos em uma mesma consulta e executem a junção dos dados, baseada em uma condição de junção.

Técnicas de usabilidade - Um dos objetivos do trabalho é o desenvolvimento de uma interface amigável ao usuário final, não especialista em construção de consultas SQL. Este objetivo foi cumprido através da formatação fixa da consulta (campos *SELECT*, *FROM* e *WHERE* obrigatórios) e de campos pré-definidos que auxiliam o usuário a selecionar os valores apropriados para a consulta através de um campo combo. Porém, o formato da consulta é um dialeto de SQL, o que limita a utilização da ferramenta apenas a pessoas que tenham alguma noção sobre consultas estruturadas. Como trabalho futuro, seria interessante a aplicação de avaliações de usabilidade com usuários reais, visando identificação de pontos que podem ser aperfeiçoados, como por exemplo, a possibilidade de aproximar o formato da consulta com linguagem natural.

Técnicas de classificação próprias para Wiki - A classificação dos objetos no momento da indexação pode ser aprimorada. Foi verificado que artigos da *Wikipedia* possuem listas para classificação de documentos, conforme Figura 25. Estas classes podem ser consideradas em um algoritmo de *cluster* que agrupa objetos por suas classes. Outra proposta para classificar os objetos indexados seria identificar os atributos deste objeto e aplicar técnicas de data *matching* [DOAN e HALEVY, 2005] com objetos que foram anteriormente classificados.

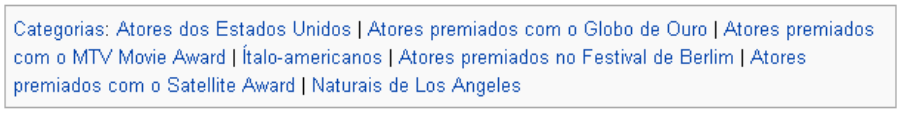


Figura 25 - Lista de categorias sobre o artigo "Leonardo DiCaprio". Fonte: http://pt.wikipedia.org/wiki/Leonardo_DiCaprio

Expansão de escopo da indexação – Os *infoboxes*, de onde os dados são extraídos para posterior indexação, são bastante heterogêneos. Conforme o guia de estilo determinado pela *Wikipedia*, o conteúdo e *layout* da caixa de informação muda de acordo com a classificação do artigo. Portanto, as heurísticas para a categorização de um documento

precisam ser aprimoradas para abranger um escopo maior de classes, considerando além de padrões genéricos, padrões definidos especificamente para uma categoria em si. Além disso, o escopo da indexação pode ser expandido para o conteúdo encontrado fora da caixa informativa, como por exemplo, nas demais tabelas que podem ser encontradas nos artigos e textos não estruturados.

Inclusão de novos operadores de comparação – Atualmente o *Find Me* permite apenas o operador de igualdade. No caso de valores numéricos, poderão ser implementados novos operadores como *maior* e *menor*. Além destes operadores, funções de similaridade para comparação de valores incertos também devem ser implementados. Foi identificado, por exemplo, na fase de experimentos, descritos na Seção 6.3 que a técnica de similaridade utilizada atualmente identifica muito mais objetos do que realmente deveria. Para melhoria dos resultados, deve ser implementadas funções de similaridade individuais para cada atributo dos objetos, a fim de os resultados tenham maior eficácia.

Identificação e conversão de unidades de medidas – Para garantir a confiabilidade do sistema, devem ser implementados métodos de conversão de valores dos atributos em unidades de medida padrão. Por exemplo, em diferentes artigos, o valor para o atributo *área* pode ser representado tanto por *25 km²* ou *25.000 m²*. Atualmente o sistema apenas captura o valor independente de sua unidade de medida.

Armazenar objetos em arquivos RDF - Conforme citado na Seção 5.3, o algoritmo de indexação do *Find Me* pode ser utilizado para encontrar termos do *Wikipedia* e armazenar estes dados em documentos RDF. Desta forma, seria possível utilizar a linguagem de consulta SPARQL, assim como feito no DBPedia.

8. Referências

ABITEBOUL, Serge; BUNEMAN, Peter; SUCIU, Dan. **Data on the Web: From Relations to Semistructured Data and** XML. Morgan Kaufman, 258 páginas, 1999.

AUER, Sören; BIZER, Christian; KOBILAROV, Georgi; LEHMANN, Jens; CYGANIAK, Richard; IVES, Zachary. **Dbpedia: A nucleus for a web of open data**. ISWC/ASWC, páginas 722–735, 2007.

BAEZA-YATES, Ricardo B.; NETO, Berthier R. **Modern Information Retrieval**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

BHALOTIA, Gaurav; HULGERI, Arvind; NAKHE, Charuta; CHAKRABARTI, Soumen; SUDARSHAN, . **Keyword searching and browsing in databases using banks**. In ICDE, page 431, 2002.

BRICKLEY, Dan; GUHA, R.V. **RDF Vocabulary Description Language 1.0: RDF Schema.** Recomendação W3C, 10 de Fevereiro de 2004. Disponível em: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/. Último acesso em 18 de maio de 2011.

BRIN, Sergey; PAGE, Larry. **Google search engine**. 1999. Disponível em: http://google.stanford.edu. Último acesso em 18 de maio de 2011.

CAFARELLA, Michael J.; RE, Christopher; SUCIU, Dan; ETZIONI Oren. **Structured Querying of Web Text Data: A Technical Challenge**. Conference on Innovative Data Systems Research - CIDR, páginas 225-234, 2007.

CHAMBERLIN, Donald D.; BOYCE, Raymond F. **ANSI. American National Standard for Information Systems - Database Language - SQL**, November, 1992

DOAN, AnHai; HALEVY, Alon Y. **Semantic Integration Research in the Database Community: A Brief Survey**. AI Magazine 26(1): 83-94, 2005.

HSIEH, Wilson; MADHAVAN, Jayant; PIKE, Rob. **Data management projects at Google**. International Conference on Management of Data - ICMD, páginas 725 – 726, 2006.

LI, Guoliang; FENG, Jianhua; ZHOU, Xiaofang; WANG, Jianyong. **Providing built-in keyword search capabilities in RDBMS**. The International Journal on Very Large Data Bases – VLDB Journal, vol. 20, ed. 1, páginas 1 -19, 2011

MADHAVAN, Jayant; AFANASIEV, Loredana; ANTOVA, Lyublena; HALEVY, Alon. **Harnessing the Deep Web: Present and Future**. Biennial Conference on Innovative Data Systems Research - CIDR, 2009. 6 p.

MELLO, Ronaldo dos S.; DORNELES, Carina F.; KADE, Adrovane; BRAGANHOLO, Vanessa de P.; HEUSER, Carlos A. **Dados Semi-Estruturados**. Simpósio Brasileiro de Banco de Dados - SBBD, 2009.

MERGEN, Sergio; FREIRE, Juliana; HEUSER, Carlos A. **MESA: A Search Engine for QueryingWeb Tables**. Salt Lake City - U.S, 2008. 6 p.

NGUYEN Huong; NGUYEN Thanh; NGUYEN Hoa; FREIRE, Juliana. **Querying wikipedia documents and relationships**. 13th WebDB, Indianápolis, EUA., 2010

PRUD'HOMMEAUX, E; SEABORNE, A. **SPARQL query language for RDF**. Recomendação W3C , 2005. Disponível em: http://www.w3.org/TR/rdf-sparql-query/. Último acesso em 18 de maio de 2011.

SPERTUS, Ellen; STEIN, Lynn A. **Squeal: A structured query language for the web**. Proceedings of the 9th International World Wide Web Conference (WWW9), páginas 95– 103, Amsterdã, Holanda, 2000.

WIKIPEDIA. **Wikipédia: Livro de estilo.** Disponível em: http://pt.wikipedia.org/wiki/Wikip%C3%A9dia:Livro_de_estilo. Último acesso em 10 de maio de 2011.