Relato sobre o Desenvolvimento de Modelos para Obtenção Automática do Conteúdo de Sites sobre Saúde

Ana Marilza Pernas Fleischmann (organizadora)1 2

José Palazzo Moreira de Oliveira (orientador) 1

Alessandro Huber dos Santos

Bruno Luis de Moura Donassolo

Carlos Eduardo Benevides Bezerra

Edimar Manica

Fahad Kalil

Liziane Santos Soares

Luiz Hermes Svoboda Junior

Marcos Paulo de Mesquita

Paulo Andre Reis Torres

Rafael de Lima Petry

Ricardo Luis dos Santos

Valderi Reis Quietinho Leithardt

Resumo: Este relato técnico descreve o desenvolvimento de modelos, técnicas e protótipos para localização, padronização e extração automática do conteúdo apresentado em *sites*/páginas *web* com assuntos relacionados à área

1 Instituto de Informática, UFRGS, Caixa Postal 15064 {ana.pernas, palazzo @inf.ufrgs.br}
2 Departamento de Informática, UFPel, Caixa Postal 354

da saúde, visando à estimativa da qualidade destes *sites*/páginas extraídos. As técnicas e propostas descritas neste documento foram desenvolvidas ao longo do primeiro semestre de 2009 pelos alunos da disciplina CMP112 – Sistemas de Informação Distribuídos do Programa de Pós-Graduação do Instituto de Informática da Universidade Federal do Rio Grande do Sul, ministrada pelo Professor Dr. José Palazzo Moreira de Oliveira. Cada uma das tarefas descritas aplicou técnicas e tecnologias diferentes para o seu desenvolvimento, apresentando resultados de diferentes naturezas, como tabelas, protótipos e modelos. Entretanto, todas foram desenvolvidas em busca do mesmo objetivo: a extração automática do conteúdo de *sites*/páginas que tratam sobre o tema "Doença de Alzheimer". Ao final to trabalho, obteve-se um conjunto de resultados, os quais serão utilizados para possibilitar a realização de estimativas a respeito da qualidade dos *sites*/páginas extraídos, de acordo com métricas de qualidade definidas.

Abstract: This report describes the development of models, techniques and prototypes to location, standardization and automatic extraction of content presented in web sites/pages with subject related to health, objecting estimate its quality. The techniques and proposals described here was performed during the first half of 2009 by students of the lecture CMP112 – Distributed Information Systems of Institute of Informatics of Federal University of Rio Grande do Sul, conducted by Professor Dr. José Palazzo Moreira de Oliveira. Each one of the tasks described in this report used different techniques and technologies for their development, presenting results of different natures, such as tables, prototypes and models. However, all tasks were developed looking for the same objective: the automatic extraction of content from web sites/pages related with the subject "Alzheimer's Disease". At the end of the work, we obtained a set of results, which will be used to enable the development of estimative concerning the quality of extracted web sites/pages, according with defined quality metrics.

1 Introdução

Este trabalho descreve o processo de desenvolvimento de modelos, técnicas e protótipos para a extração automática do conteúdo apresentado em *sites*/páginas que tratam sobre saúde, visando à entrega padronizada e normalizada de dados para realização de estimativas sobre a qualidade do *site*/página. De posse destes dados, será possível medir a qualidade do *site*/página em questão de acordo métricas específicas para estimativa de qualidade. O trabalho está relacionado ao Projeto SALUS – CYTED – Qualidade em Sites na Área de Saúde.

O Projeto SALUS tem por objetivo discutir e criar instrumentos para avaliação de *sites* da área de saúde visando fornecer a usuários com perfis distintos indicativos de qualidade dos *sites*.

O processo de desenvolvimento do modelo para obtenção automática de dados foi realizado ao longo do primeiro semestre de 2009 pelos alunos da disciplina CMP112 – Sistemas de Informação Distribuídos do Programa de Pós-Graduação do Instituto de Informática da Universidade Federal do Rio Grande do Sul, ministrada pelo Professor Dr. José Palazzo Moreira de Oliveira.

Na disciplina CMP112, o trabalho começou a ser encaminhado como um dos grandes desafios atuais, que se trata da necessidade de se lidar, de forma automática, com a grande quantidade de dados disponíveis na *web*, sendo a *web* apontada como um grande sistema distribuído. Em sistemas distribuídos convencionais, os dados são organizados desta forma visando: diminuição do tempo de processamento, confiabilidade e comunicação [1]. Entretanto, no que se trata da *web*, o caso é bem diferente. Apesar de ter surgido por motivos similares aos de um sistema distribuído convencional, seu rápido crescimento, não padronizado, atingiu proporções muito grandes para que este sistema seja eficientemente controlado.

Por este crescimento descontrolado, o problema relativo ao tratamento automático dos dados distribuídos na *web* se torna complexo. Um dos principais motivos atribuídos a esta complexidade é a não aplicação de um padrão amplamente conhecido para construção destas páginas, o que levaria, em conseqüência, a uma forma também padronizada de acessá-las.

Desta forma, chegou-se a conclusão de que, para que seja possível tratar a grande quantidade de dados da *web* de forma a se estimar sua qualidade de forma automática, seria necessário, a priori, automatizar adequadamente as seguintes etapas:

- Localização – trata-se, resumidamente, da solução para a seguinte pergunta: dado um item X, armazenado em algum conjunto dinâmico de nodos do sistema, como encontrá-lo? [2]. No trabalho, foram investigados os vários motores de busca existentes, com escopo limitado ou não a páginas sobre saúde.

- Padronização – trata-se da descoberta de conjuntos de padrões existentes nas páginas *web* que estão relacionados à apresentação dos dados e podem indicar formas de realizar o acesso (extração) a dados contidos nas paginas *web*.
- Extração – uma vez localizada a informação na *web*, trata-se do problema relacionado à como obter esta informação, para devido tratamento (armazenamento em uma base de dados, ontologia, etc.). Após poderá ser realizada a tarefa de análise de qualidade.
- Qualidade – trata-se da análise em si dos dados coletados, onde métricas são aplicadas de forma a definir o grau de qualidade presente para um determinado perfil pré-definido de usuários.

O trabalho foi desenvolvido tendo em vista as etapas apresentadas acima. O ponto de partida para o desenvolvimento do trabalho foram os resultados obtidos durante o desenvolvimento da disciplina CMP234 – Modelagem Conceitual e Ontologia do Programa de Pós-Graduação da UFRGS, também ministrada pelo Professor Dr. José Palazzo Moreira de Oliveira, mas que ocorreu durante o semestre 2008/2. Nesta disciplina foi desenvolvida uma ontologia para guiar a inferência da qualidade de *sites*/páginas sobre saúde. Esta ontologia foi simplificada de forma que apenas algumas de suas classes e propriedades foram consideradas para a realização dos trabalhos de extração. As classes e propriedades que foram consideradas são mostradas no modelo E-R (Entidade-Relacionamento) apresentado na Figura 1.

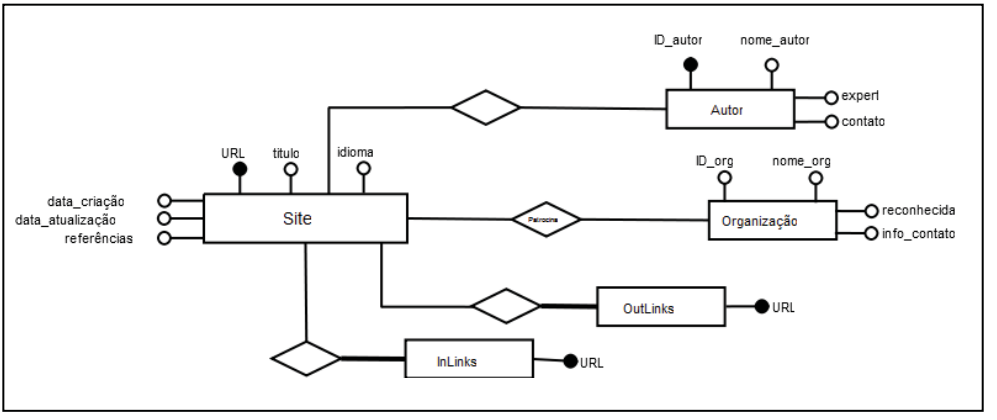

Figura 1. Modelo E-R proposto de acordo com a ontologia desenvolvida

Com relação à última etapa, de Qualidade, entende-se que no trabalho da disciplina anterior este problema já foi tratado, tendo resultado na ontologia utilizada como ponto de partida para o desenvolvimento do trabalho aqui relatado. Desta forma, definiu-se como

prioridade o tratamento da localização, padronização e extração automática dos dados vindos de *sites*/páginas sobre saúde. Além disso, definiu-se que o foco de estudo seria **Doença de Alzheimer** e que a língua utilizada para busca de *sites*/páginas que tratam sobre doença de Alzheimer seria o **inglês**.

Partindo-se, então, do trabalho desenvolvido na disciplina CMP234 e das etapas concluídas como necessárias para definição automática da qualidade de *sites*/páginas da *web*, a Figura 2 demonstra o modelo geral de desenvolvimento do trabalho descrito neste relatório. A etapa 4 não está descrita neste relatório pelos motivos já exposto. Para verificação das métricas de qualidade adotadas, recomenda-se leitura do Relatório de Pesquisa anterior do Projeto SALUS-CYTED508RT0361 (de fevereiro de 2009) e do trabalho desenvolvido em [3].

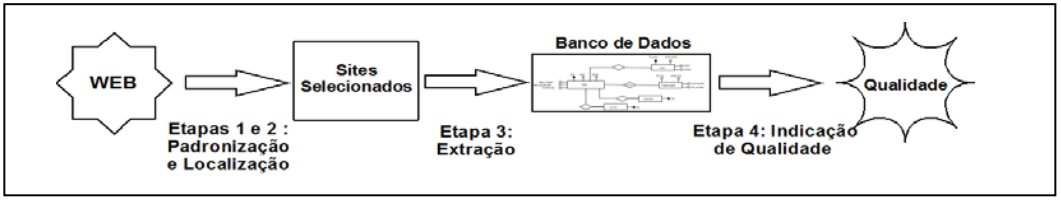

Figura 2. Etapas desenvolvidas para indicação automática de qualidade

Para descrição do trabalho realizado, o presente documento está organizado da seguinte forma. A seção 2 trata da primeira etapa do trabalho, referente à localização dos *sites*/páginas sobre saúde a serem analisados. A seção 3 fala sobre a etapa de padronização dos dados, necessária para efetivar a extração dos dados. A seção 4 trata sobre a extração em si dos dados. A seção 5 apresenta as considerações finais e também futuras atividades necessárias para prosseguimento do projeto.

2 Localização Automática de Dados

O objetivo da localização de dados na *web* é de, dado um critério de busca definido pelo usuário – ou por um sistema que esteja executando a busca de maneira automática –, encontrar páginas (que podem ser páginas pessoais, artigos científicos, notícias etc.) que satisfaçam aquele critério de busca definido. Além disso, espera-se que estes mesmos resultados estejam ordenados de acordo com sua relevância, ou seja, de acordo com o quanto satisfazem o critério de busca utilizado.

A questão reside no fato de que, na *web*, existem bilhões de páginas disponíveis [4] e é necessário um sofisticado algoritmo, aliado a um grande poder computacional, para realizar as buscas. Obviamente, as buscas são feitas sobre uma base de dados com páginas previamente indexadas e em constante atualização, ao invés de ser realizado um *string-matching* em cada *site* a cada busca.

No caso do motor de busca Google[3], são indexadas todas as palavras encontradas nas páginas, vinculando-as às páginas que as contêm e, para cada relação palavra-página, é atribuído um peso, que determina o quão relevante é aquele termo para aquela página. Esse peso irá depender de como aquela palavra aparece na página: por exemplo, palavras nos títulos e subtítulos ou repetidas diversas vezes recebem uma pontuação maior do que palavras que aparecem com pouca freqüência e/ou no corpo do texto.

Para realizar esse tipo de indexação, armazenamento em uma base de dados e recuperação posterior, os motores de busca, como o Google, utilizam agentes autônomos que buscam visitar o máximo possível de páginas. Dada uma página inicial, o *crawler* **Googlebot** entrega para o indexador do Google aquela página e busca nela referências (*links*) para outras páginas, que são enfileirados para serem visitados recursivamente. Através dessa recursão, tenta-se visitar o máximo possível de páginas da *web*, indexando seus termos. A página inicial – raiz da árvore de recursão do Googlebot – pode ter sido adicionada manualmente, ou por outra busca feita pelo próprio agente.

A partir de 2008, o Googlebot começou a indexar páginas da *web* profunda (*deep web*) [5], que são as páginas resultantes de consultas a formulários, e que representam mais de 99% das informações disponíveis na *web* [6]. Para gerar as páginas, o agente preenche o formulário e submete ao servidor. No caso de caixas de texto, são utilizadas palavras da própria página. No caso de caixas de seleção, botões de múltipla escolha ou listagens (*combo boxes*), o agente utiliza um dos valores disponíveis para esses campos. A página retornada pelo servidor é, então, indexada pelo mecanismo de indexação. O motor de busca Google alega que o agente não testa campos como nome de usuário e senha, por razões óbvias. Além disso, são consultados apenas formulários que utilizam o método GET, que não armazenam informações no servidor da página.

O presente trabalho optou por desenvolver o modelo a partir de motores de busca pré-existentes, disponíveis na *web*, ao invés de implementar um mecanismo para localização das páginas próprio, o que exigiria tempo e recursos computacionais. Desta forma, a tarefa específica de localização automática de *sites*/páginas na *web* foi dividida em três partes principais:

- Pesquisa e levantamento dos motores de busca existentes e análise das vantagens e desvantagens de cada um deles, verificando quais seriam suficientemente adequados para realização da pesquisa de dados sobre saúde.
- Definição dos critérios a serem utilizados para as buscas nos motores.
- Modelagem e implementação de um protótipo para recuperação e indexação dos resultados encontrados, de acordo com as buscas efetuadas, em uma base de dados.

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

Nas subseções seguintes, estes passos são descritos.

2.1 Levantamento dos motores de busca na w*eb*

A primeira etapa desenvolvida foi o levantamento de informações a respeito dos motores de busca mais apropriados para utilização no trabalho. Os motores de busca analisados foram:

- **Google Health**[4]: Através do Google Health, que requer *login* de usuário, é possível criar (e publicar) um perfil pessoal com informações a respeito do estado de saúde, assim como buscar informações a respeito de doenças e buscar médicos de áreas específicas. Ele também permite integração com bases de dados de hospitais e de médicos para atualizar as informações do paciente automaticamente. O serviço é interessante no sentido de que busca cruzar automaticamente os elementos do perfil do usuário para buscar informações a respeito de interações. Isso seria útil, por exemplo, se um usuário informasse que tem alergia a um determinado princípio ativo e, posteriormente, adicionasse um remédio que contém justamente esse princípio. No entanto, devido ao seu funcionamento exigir a entrada manual no *site* (*login*), o Google Health não é adequado para um sistema de busca automático.
- **Google Scholar**[5]: Permite busca de artigos científicos e pode ser usado para encontrar informações mais técnicas, voltadas a um público acadêmico. Por um lado, esta restrição parece interessante, pois apresenta resultado focado aos interesses deste público. No entanto, os resultados retornados são geralmente trabalhos acadêmicos com um enfoque muito específico (por exemplo, *Association of apolipoprotein E allele e4 with late-onset familial and sporadic Alzheimer's disease*). Por fim, grande parte dos resultados retornados por este motor são páginas de editoras que exigem pagamento para acesso aos textos completos dos trabalhos.
- **CurbSide.MD**[6]: É voltado para público da área de saúde. Tenta buscar seus resultados a partir de perguntas fornecidas pelo usuário, utilizando uma ontologia própria. Seus resultados já vêm classificados de acordo com o tipo de informação que trazem. Para a busca de informações sobre uma determinada doença, como Alzheimer, ele apresenta os resultados de forma organizada de acordo com, por exemplo: diagnóstico, tratamento, epidemiologia e relatos.
- **Medstory**[7]: Adquirido pela Microsoft em 2007 [7], e integrado ao **bing health** – que deixou de existir ao ser englobado pelo **bing** –, o Medstory faz uma busca na

[4] https://www.google.com/health
[5] http://scholar.google.com.br/
[6] http://www.curbside.md/
[7] http://www.medstory.com/

web de forma geral, classificando os resultados de acordo com diversas palavras-chave, indicando quais palavras-chave aparecem com maior freqüência nos resultados. Essas palavras-chave, que são todas da área de saúde, fazem parte de uma lista pré-definida e estática, sendo agrupadas por categorias, também pré-definidas, tais como: drogas, sintomas, procedimentos, entre outras.

- **PubMed**[8]: Uma ferramenta relativamente simples de busca por artigos científicos da área médica. Sua busca está restrita à MEDLINE, uma base de dados compilada a partir das publicações contidas na Biblioteca Nacional de Medicina dos Estados Unidos. Acredita-se que o Google Scholar, que teria uma abrangência maior, muito provavelmente englobaria os possíveis resultados dessa base.
- **SearchMedica**[9]: Este motor permite ao usuário escolher o espaço de busca onde irá pesquisar – toda a rede ou *sites* recomendados. Além disso, sua pesquisa pode ser feita visando uma área específica da medicina (cardiologia, geriatria, dermatologia etc.). A página de resultados inclui uma barra lateral com diversas pesquisas sugeridas, relacionadas com a pesquisa realizada pelo usuário: subconjuntos da pesquisa realizada (para uma pesquisa sobre "cardiopatia", seria sugerido "doença de chagas", por exemplo), super conjuntos (para a mesma pesquisa de "cardiopatia", seria sugerido "doenças cardiovasculares"), além de temas relacionados.
- Motores de uso geral, como **Google**[10] **(Web), Yahoo!**[11]**, bing**[12] e outros: Estes motores de busca permitem a realização de buscas em *sites*/páginas de toda *web*. Dessa forma, pode-se encontrar desde informações introdutórias com linguagem acessível a usuários leigos (não pertencentes à área de medicina) a notícias e a trabalhos científicos mais recentes. Contudo, a garantia de confiabilidade se torna mais difícil, devido justamente a não restrição das páginas a serem procuradas. Mesmo assim, outros motores de busca (exceto a PubMed, que consulta uma base de dados compilada por uma biblioteca específica) não garantem por completo que as páginas que retornam contenham informações fidedignas. Além disso, um resultado contendo maior quantidade de páginas pode possibilitar a localização de resultados mais relevantes. Nestes casos, algoritmos como PageRank™ (BRIN; PAGE, 1998), utilizado pelo motor de busca Google, procuram fazer com que esses motores retornem páginas com alto grau de confiabilidade logo nos primeiros resultados.

[8] http://www.ncbi.nlm.nih.gov/pubmed/
[9] http://www.searchmedica.com/
[10] http://www.google.com.br/
[11] http://br.yahoo.com/
[12] http://www.bing.com/

2.1.1 Motores de busca escolhidos

Deseja-se obter, basicamente, dois tipos de informações: textos em linguagem apropriada para o público em geral e textos técnicos/científicos com informações mais detalhadas a respeito do tema pesquisado.

Os motores de busca Yahoo! e bing não foram utilizados, pois seus resultados são bastante semelhantes aos apresentados pelo motor Google, o qual foi escolhido para utilização neste trabalho. O motor PubMed não será utilizado, pois seus resultados, apesar de estarem estruturados, são bastante semelhantes aos do Google Scholar, sendo que seu espaço de busca é consideravelmente menor. Com base nisto e nas vantagens e desvantagens encontradas para cada motor de busca pesquisado, optou-se pela seguinte categorização:

- Textos de toda *web*: Google (Web) e SearchMedica (utilizando a opção de buscar em toda a *web*).
- Textos da área de saúde ou técnicos/científicos: Medstory, SearchMedica (buscando em sites recomendados) e Curbside.MD.

Algumas questões a serem consideradas: para a integração dos diferentes motores em um único sistema de localização, é necessário definir as interfaces uma a uma, já que cada um possui padrões diferentes. Além disso, mudanças em seus padrões e/ou critérios podem exigir novo esforço de padronização e definição de critérios de busca. Por último, utilizar diferentes motores num mesmo sistema pode levar a incoerência dos tipos de resultados encontrados (por exemplo, artigos acadêmicos de alta relevância aparecendo juntamente a postagens de um *blog*).

2.2 Critérios de busca para localização dos dados

Após a seleção dos motores de busca que melhor se adaptam as necessidades do projeto, foi preciso realizar a definição dos critérios de busca a serem utilizados nessas consultas. Ou seja, quais serão as palavras chaves utilizadas nas pesquisas que podem, potencialmente, retornar os resultados mais relevantes no contexto da pesquisa de Alzheimer.

Foi dado enfoque ao usuário, a fim de selecionar os critérios de busca que possam retornar os resultados desejados por ele. Além disso, definiu-se que as buscas seriam realizadas a partir de palavras-chaves escritas em língua inglesa, de forma a homogeneizar os resultados apresentados. De acordo com estas definições, os critérios foram separados em categorias, conforme as possíveis buscas efetuadas pelos usuários. São elas:

- **Informações gerais**

  São informações de cunho introdutório, que visam dar uma visão geral ao assunto e que permitem aos usuários decidir se continuam ou não sua pesquisa. Geralmente são requisitadas por usuários leigos, tendo, portanto, a necessidade de apresentar uma linguagem mais simples e acessível.

  Palavras chaves usadas:

- *alzheimer*;
- *alzheimer introduction*.

- **Métodos de tratamento**

  Essa categoria possui dois enfoques principais. O primeiro seria de um profissional, interessado em buscar novos métodos de tratamentos que estejam sendo desenvolvidos e possam ser aplicados aos seus pacientes. O segundo, voltado ao usuário normal, seria de verificar quais as suas opções: confrontar o tratamento de seu médico ou se interar melhor no assunto.

  Palavra chave usada:

  - *alzheimer treatment*.

- **Diagnóstico**

  Informações de vital importância, pois permitem ao usuário verificar os primeiros sintomas da doença, sendo assim o primeiro passo para um paciente procurar ajuda profissional ou não. Por outro lado, permitem ao médico atualizar seus métodos de diagnóstico, seja através de novos exames, seja por inferência aos sintomas apresentados pelo seu paciente.

  Palavras chaves usadas:

  - *alzheimer's diagnostic*;
  - *alzheimer's symptoms*.

- **Remédios**

  Nessa categoria são descritas as informações específicas dos medicamentos, tais como: interações medicamentosas, reações alérgicas aos medicamentos, eficiência dos mesmos. Ela é destinada principalmente a um público especialista que possui a formação necessária para a compreensão dos dados ali apresentados.

  Palavras chaves usadas:

  - *alzheimer drugs interactions*;
  - *alzheimer drug treatment*.

- **Estudos de caso**

  Pesquisas abrangentes sobre a doença, onde um estudo complexo é feito em cima de pacientes, sendo todo o processo analisado e descrito com padrões científicos. É claro que nesses casos o nível técnico do leitor é importante, já que os resultados são descritos com linguagem complexa.

  Palavra chave usada:

- *alzheimer case study*.

- **Dicas**

  Esta seção tem como principal objetivo servir de apoio a pessoas que necessitam de auxílio na prestação de suporte a pessoas com a doença de Alzheimer. Como algumas questões típicas que esta categoria visa responder, podem ser citadas:

  - Como lidar com um paciente com Alzheimer?

  - Como ajudá-lo?

  - Quais os fatores de risco da doença?

  Palavra chave usada:

  - *alzheimer's pratical tips*.

- **Prevenção**

  Esta categoria visa satisfazer uma tendência cada vez mais crescente nos dias atuais: a curiosidade de usuários que buscam cuidar de sua saúde. Dessa forma, os resultados esperados devem estar escritos numa linguagem mais simples para a compreensão do público geral.

  Palavra chave usada:

  - *alzheimer's prevention*.

Nota-se que todas as categorias, apesar de algumas estarem mais voltadas para um tipo específico de usuário, possuem uma dualidade na complexidade da linguagem dos resultados esperados. Não seria eficaz retornar resultados muito complexos, com embasamento científico e linguagem complexa, para uma pessoa que não possui interesse em alto nível de detalhe. Da mesma forma, o especialista não deseja ter informações muito simplificadas.

O uso do motor de busca irá guiar os resultados. Por exemplo, ao buscar por "*alzheimer's symptoms*" no Google Scholar, a página de resultados será composta de artigos científicos com uma linguagem voltada para especialistas. Entretanto, ao fazer a mesma consulta no Google, os resultados serão mais voltados para leigos.

Durante o desenvolvimento das categorias colocadas acima, surgiram algumas questões que dificultaram a sua definição, como:

- Coerência e relevância das categorias utilizadas, isto é, até que ponto essas categorias expressam as necessidades do usuário e respondem as suas dúvidas.

- Cobertura das pesquisas utilizadas, ou seja, até que ponto os resultados recuperados são abrangentes.

Nesse sentido, é possível citar algumas melhorias que poderiam ser realizadas a fim de amenizar essas questões:

- Incrementar o número de categorias para melhor satisfazer aos usuários. Para isso, seria necessário fazer uma pesquisa junto aos diversos usuários para ver quais são as suas necessidades e como atendê-las.
- Aumentar e refinar as palavras chaves utilizadas nas consultas. Dessa forma, seria possível retornar resultados melhor selecionados.

2.3 Modelo proposto para localização automática

O modelo proposto para a realização da tarefa de localização compreende a proposta para implementação de uma aplicação. O modelo baseia-se na interação entre a aplicação desenvolvida, uma base de dados e a Internet. A Figura 3 apresenta o esquema de comunicação entre os componentes presentes no modelo.

Inicialmente, foram definidas as tabelas que seriam utilizadas pela aplicação. A tabela ***Engine(ID, Name)*** define quais serão os motores de busca que serão consultados pela aplicação. Além do cadastro no banco, a aplicação deve prover o modelo correto para submissão e extração dos dados resultantes em cada um dos motores. Na tabela ***URLs(ID, url)*** são armazenadas as *URLs (Uniform Resource Locator)* recuperadas pelo mecanismo.

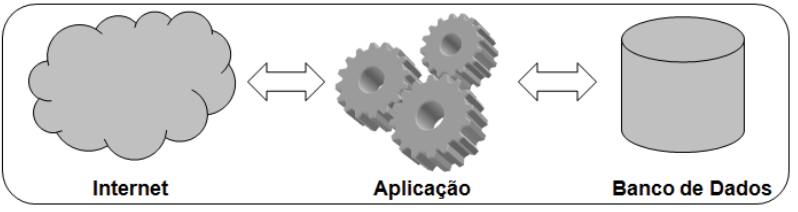

Figura 3: Modelo proposto para a realização da tarefa de localização**.**

Considerando a Figura 3, o processo completo de interação entre os componentes é descrito através da especificação das atividades presentes no processo e o fluxo no qual as atividades são executadas para o cumprimento da tarefa de localização. Para a realização desta especificação, adotou-se a notação BMPN (*Business Process Modeling Notation*), a qual fornece subsídios para a definição de modelos de processo.

O modelo de processo resultante pode ser verificado na Figura 4. As atividades são representadas pelos retângulos azuis enquanto que o fluxo de execução segue a ordem definida através das setas direcionais. A atividade destacada na Figura 4 representa o sub-processo no qual as seguintes atividades são executadas:

- Envio da requisição ao motor de busca;

- Armazenamento do resultado em uma estrutura de dados.

Este sub-processo é executado de acordo com o número de motores de busca a serem consultados (motores cadastrados na tabela ***Engines***).

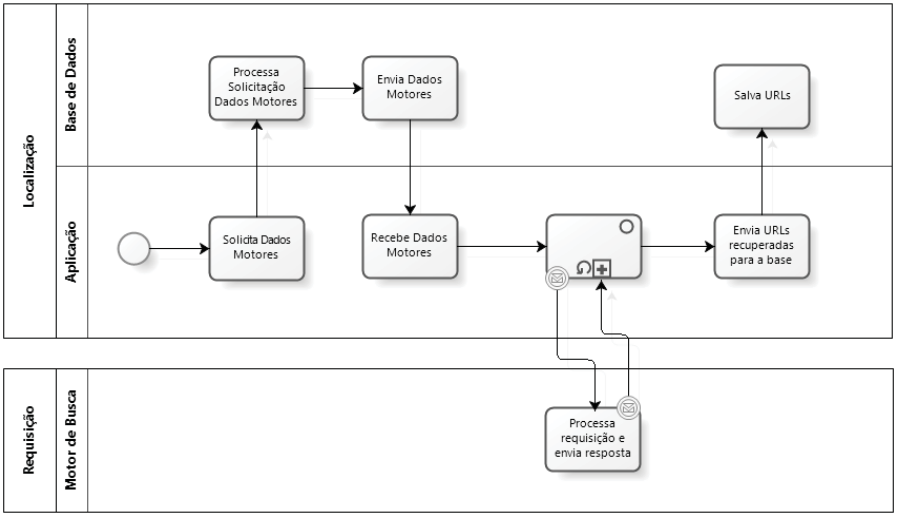

Figura 4: Modelo BMPN do processo de localização de URLs**.**

2.4 Protótipo do modelo

Como parte deste trabalho, foi desenvolvido um protótipo com o objetivo de validar o modelo proposto. Para a implementação do protótipo, adotou-se a linguagem de programação JAVA e o banco de dados embarcado Derby. A escolha de um banco de dados embarcado deu-se devido à facilidade de utilização destes mecanismos de persistência de dados e também à sua portabilidade.

De acordo com a especificação do modelo, os seguintes motores de busca foram cadastrados para utilização: **Google**, **SearchMedica** e **Medstory**. Além do cadastro, desenvolveu-se para cada motor uma classe que é responsável pelo acesso e, posteriormente, pela extração dos resultados. A implementação destas classes foi realizada de acordo com o diagrama de classes representado na Figura 5. A classe abstrata ***SearchEngine*** define o método de inicialização da conexão HTTP (*Hypertext Transfer Protocol)*, o qual é comum para todos os motores. Além disso, esta classe especifica dois métodos que a classe de cada motor deve implementar. São eles: ***buildSearchURL*** e ***getURL***. No método ***buildSearchURL*** a URL que será submetida ao motor é construída. Além de cada motor possuir sua URL específica, é necessário definir como os termos de busca devem ser concatenados para que a consulta gerada seja corretamente processada pelo motor. No método ***getURL*** são definidos os padrões para extração dos resultados.

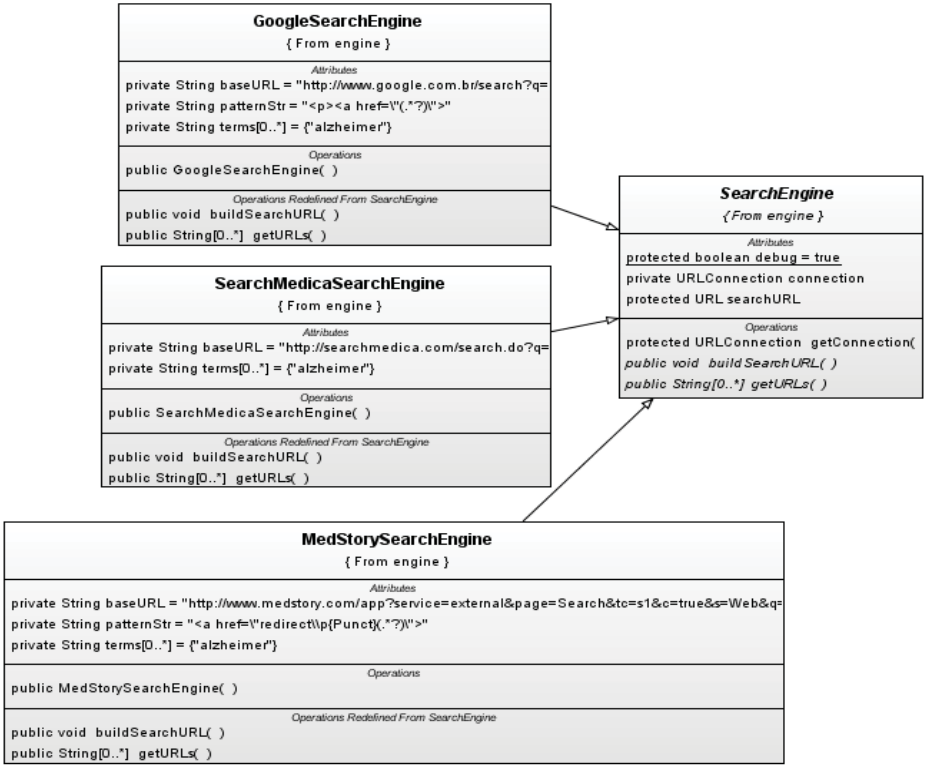

Figura 5: Diagrama de classes de acesso e extração de dados dos motores.

Em alguns casos (Google e Medstory) a definição de um padrão foi suficiente para a extração. Contudo, em outros (SearchMedica), foi necessário a definição de mais padrões para que fosse possível executar esta tarefa. Os padrões utilizados nesta etapa são listados abaixo de acordo com a sua utilização:

Google: <p><a href=\"(.*?)\">

MedStory: <a href=\"redirect\\p{Punct}(.*?)\">

SearchMedica: (1) <span class=\"rank\">\\p{Digit}+

(2) href=\"(.*?)\"

Como um dos objetivos no desenvolvimento do protótipo foi torná-lo flexível, foram utilizados recursos da linguagem JAVA, como reflexão. Mais detalhes sobre a implementação podem ser observados na documentação do código fonte principal (*main*) do protótipo, o qual está listado no Apêndice 1 deste documento.

2.5 Considerações finais sobre localização

Esta seção tratou a respeito do trabalho realizado durante a etapa de localização automática, visando à extração automática de dados contidos em *sites*/páginas sobre saúde, enfatizando o que foi realizado e os problemas encontrados durante essa etapa. Mostrou-se a importância da etapa da localização para o restante do projeto, uma vez que ela determina quais *sites*/páginas serão selecionados para analise de qualidade. Entretanto, essa tarefa não é simples, selecionar os melhores *sites* no grande número de páginas existentes na *web* é um desafio.

Primeiramente, foi realizada a seleção dos motores de busca que melhor se adaptavam a nossa necessidade. Assim, foram escolhidos alguns motores específicos da área médica, os quais, teoricamente, fornecem como resposta URLs de *sites*/páginas com conteúdo mais confiável, além de alguns motores de uso geral.

Após este passo, foram descritas as palavras chaves a serem utilizadas nas pesquisas aos motores de busca. As palavras chaves foram divididas em categorias, de forma a se aproximar das necessidades que podem ser apresentadas por diferentes tipos de usuários da *web*.

Em seguida, foi proposto um modelo para a resolução dos problemas da etapa de localização. Esse modelo tem como objetivo ser simples e expansível, de forma a ser adaptado para possíveis melhorias e também capaz de ser modificado para tratar outros temas, além da doença de Alzheimer.

Por fim, um protótipo foi implementado para verificar a funcionalidade do modelo proposto. Nem todas as funcionalidades foram implementadas, mas foi possível comprovar a viabilidade da solução.

3 Padronização dos Dados

A padronização considerada nesse trabalho refere-se à padronização de formato de páginas *web* que, posteriormente, será usada como fonte para extração de informações e análise de qualidade. Com relação à padronização relativa à qualidade de conteúdo das páginas, serão utilizados os resultados obtidos na disciplina CMP234 – Modelagem Conceitual e Ontologias, a qual gerou um modelo entidade-relacionamento com classes e atributos desejáveis para o conteúdo das páginas *web* na área da saúde. Esses padrões de qualidade definidos embasarão a procura e definição da padronização de formato. Assim, serão buscados padrões de formato que satisfaçam a busca pelos padrões de qualidade de conteúdo.

A estrutura e o conteúdo, tanto dos motores de busca na *web* quanto dos *sites*/páginas disponibilizados para acesso, são extremamente dinâmicos. Desta forma, os dados que compreendem esta padronização realizada podem sofrer alterações constantemente pelos desenvolvedores dos motores de busca e dos *sites*/paginas *web*. A coleta das informações

apresentadas nesta seção ocorreu no período de agosto de 2009, sendo válida neste período para pesquisa durante o trabalho realizado na disciplina.

Em função da carência de padrões existentes para o formato utilizado para as informações disponíveis em páginas *web*, optou-se por realizar o desenvolvimento do trabalho de padronização em duas fases: **(1)** padronização dos motores de busca, definidos pela etapa de localização, apresentada na seção 2; **(2)** padronização das páginas *web* retornadas como resultado de pesquisa efetuada nos motores de busca padronizados na fase 1.

Com relação ao desenvolvimento da fase 1, entende-se que o uso de resultados providos por motores de busca pode ser um primeiro passo em direção à definição dos padrões de formato. Mesmo que esses resultados apresentem um número pequeno de informações, as informações já são apresentadas com alguma padronização, onde os dados retornados e o formato desses dados são os mesmos para todas as páginas retornadas. Em alguns casos, esses dados são ainda enriquecidos por informações adicionais providas pelos motores de busca, como por exemplo: existência de *links* patrocinados, tópicos relacionados ao termo procurado, número de citações indexadas para o conteúdo do resultado, entre outras.

Já relacionado à fase 2, foram selecionadas para padronização as páginas retornadas na primeira página de resultado da pesquisa a cada motor de busca. Obviamente, as páginas que se repetiam foram excluídas.

Tanto para análise de padronização da estrutura da página apresentada inicialmente pelo motor de busca quanto da estrutura das páginas listadas no resultado da busca, foi necessário realizar a procura por dois tipos de padrão para o formato. Foi considerado o **código da página**, isto é, os dados definidos internamente às *tags* de dados (ou, quando houver, *tags* de metadados); e também foi considerada a estrutura de **informação textual** contida na página, isto é, posicionamento relativo de dados no texto apresentado pela página.

Nas subseções seguintes são apresentados o desenvolvimento e os resultados obtidos durante as duas fases de padronização, rapidamente explicadas acima.

3.1 Fase 1 – Padronização dos motores de busca

Para realização da primeira etapa de definição dos padrões, foram priorizados os motores de busca definidos pela etapa de localização como relevantes para o trabalho. Na etapa de localização foram utilizados critérios específicos para seleção dos motores de busca, apresentados na seção 2. De acordo com estes critérios, os motores selecionados foram os seguintes:

- Textos com linguagem acessível: Google (Web), Google Health e Medstory;
- Textos técnicos/científicos: Google Scholar, Curbside e SearchMedica;

Destes motores de busca selecionados, para a padronização foi excluído o motor de busca SearchMedica, pois este não permite a análise apropriada de código-fonte. Os motores restantes foram padronizados.

Além da seleção dos motores de busca a serem utilizados, a etapa de localização também definiu uma série de palavras-chaves a serem usadas quando da análise dos motores, as quais podem ser vistas na seção 2. Destas palavras-chaves definidas, a etapa de padronização aplicou, na maior parte dos casos, as palavras "*alzheimer*" e "*alzheimer introduction*", pois com estas palavras-chave já foi possível filtrar grande parte das páginas sobre a doença.

A análise dos dados foi realizada manualmente, basicamente salvando-se "amostras" dos resultados - o código fonte da página – dos motores de busca, direto do sítio ao qual se encontram publicados (ex.: *www.curbside.md*) através do navegador (*browser).* Inicialmente, uma amostra antes de executar a busca e uma após a execução da busca foi analisada, para identificar como o motor de busca, através do *browser,* captura e processa o termo buscado. Não foram utilizados os recursos de busca cruzada, pois não são oferecidos igualmente em todos os motores de busca analisados.

Para realização desta etapa, foram utilizados os seguintes recursos:

- Sistemas Operacionais Windows Vista Business, Windows XP Professional e GNU/Linux Ubuntu Ver. 9.04 Kernel 2.6.28-11.37.
- Navegador Mozilla Firefox 3.0.11 e Internet Explorer 8.
- Editor de texto simples, com função de realce do(s) termo(s) encontrado(s).
- Acesso à Internet sem "filtros" (*proxies*, *firewall's*)

Nas seções seguintes estão descritas as padronizações realizadas de acordo com cada um dos motores de busca selecionados. Os motores de busca descritos, por serem escolhidos de acordo com os estudos realizados na etapa de localização, encontram-se também descritos na seção 2. Estão aqui novamente por entender-se que, para padronização destes motores de busca, foram necessárias informações complementares as apresentadas na seção 2.

3.1.1 Motor de busca Google

O motor de busca Google apresenta grande quantidade de dados complementares para auxílio do usuário, como páginas semelhantes, possibilidade de pesquisa em páginas que estejam armazenadas em cachê, promover a página ou retirá-la dos resultados. Estas informações complementares não foram padronizadas nesta etapa porque se entende que não apresentam tanta relevância para padronização, sendo mais importante a análise da forma de apresentação das páginas de resultado propriamente ditas.

Este motor de busca permite também a busca avançada, possibilitando aos usuários a discriminação de uma série de itens relativos à pesquisa, como inserção de expressões relacionais a chave de pesquisa, escolha do idioma das páginas resultantes, formato de

arquivo, entre outros. A pesquisa para padronização do motor não levou em consideração a pesquisa avançada, visto que não é uma opção válida em outros motores de busca.

Com relação à página de resultados do motor de busca Google, são retornados para cada página: título da página; texto sobre o conteúdo da página (que parece consistir de um trecho inicial do conteúdo da própria página); endereço completo (URL) da página, por extenso; identificação da existência ou não de conteúdo da página armazenado em cachê; pesquisa por páginas semelhantes.

Sobre os padrões obtidos no motor de busca, o texto de apresentação na página não apresenta problemas, pois o texto é apresentado nesta ordem: número da página resultante, *link* para a página, texto sobre conteúdo, endereço completo, *link* para páginas semelhantes e conteúdo armazenado em cachê.

Com relação às *tags* de código, não apresentam padrão significativo, como mostra a Tabela 1.

Tabela 1: Padrões encontrados no motor de busca Google.

Descrição_Campo	(código) tag_inicio	(código) tag_fim
Título do site	<li class="g w0"> <h3class=r> <a href="	" class=l onmousedown= "return clk(this.href, '','','res',
Texto sobre conteúdo do site		 <cite>
URL do site por extenso	 <cite>	- </cite>

Da análise do motor de busca Google pode-se concluir que este não apresenta resultados satisfatórios para análise de padronização, visto que as *tags* definidas não são auto-descritivas e não prezam por seguir uma padronização específica de *meta tags*. O código-fonte do motor de busca é bastante confuso, tornando ainda mais difícil a sua interpretação.

A análise completa dos padrões apresentados pelo motor de busca Google pode ser obtida no Apêndice 2 deste documento.

3.1.2 Motor de busca Google Health

O Google Health é um serviço da empresa Google Inc, que disponibiliza aos usuários o armazenamento e gerenciamento de suas informações médicas em um local centralizado, permitindo também que médicos busquem tais informações. É um serviço que requer autenticação dos usuários para ser utilizado, limitando seu uso de forma automática, como constatado na etapa de localização descrita na seção 2.

Possibilita busca cruzada, com as informações sobre as doenças ou condições médicas do usuário e o termo a ser pesquisado. Geralmente retorna localização física de um serviço ou *site* (Google Maps), *link* para página e opção de gravar o serviço ou *site* retornado nos

"favoritos" do usuário. Uma observação interessante: O Curbside é um serviço integrado ao Google Health, porém não utilizam necessariamente o mesmo motor de busca. A Tabela 2 apresenta uma amostra do padrão encontrado na página de resultados.

Tabela 2: Exemplo de padrões encontrados em Google Health.

Descrição_Campo	(codigo) tag_inicio	(codigo) tag_fim
Título do site, com a URL do resultado		
Observação:	latlang → coordenadas latitude e longitude para o google maps (distintas para cada url retornada)	

O Google Health aponta para localização de serviços médicos e utiliza padrões comuns para motores de busca do Google (ex.: Google Maps). Tanto os resultados retornados na página de resultados do motor de busca quanto os próprios resultados (*sites* externos ao domínio do motor de busca) constituem um conjunto muito heterogêneo de padrões. Foram analisados os 5 primeiros resultados e os 5 primeiros "melhores" resultados, ou seja, que melhor se encaixam nos padrões estabelecidos na ontologia estabelecida para este trabalho.

Devido à extrema heterogeneidade dos padrões obtidos, bem como a necessidade de autenticação para usar o motor de busca, este foi desconsiderado para uso no trabalho proposto.

3.1.3 Motor de busca Google Scholar

O motor de busca Google Schoolar apresenta uma quantidade expressiva de campos relacionados a cada um dos seus resultados de busca. Para identificar esses campos, foram realizadas duas buscas sobre Alzheimer neste motor. Na primeira consulta foi utilizado o termo "*alzheimer*" e na segunda, o termo "*alzheimer introdution*". A partir do resultado dessas duas buscas, o padrão para exibição dos resultados foi coletado. Esse padrão é composto da descriminação dos campos referentes a cada um dos resultados e das *tags* do código fonte e/ou textos da página que os delimitam. Os campos listados para cada resultado retornado pela busca são:

- **URLTrabalho**: URL para o trabalho.
- **Título do trabalho**.
- **URLVersãoDisponibilizada**: aparece nos casos em que o trabalho já está disponível diretamente em uma URL.
- **DominioVersaoDisponibilizada**: domínio da página que contém o trabalho.

- **AutorFonteAnoIntituicao**: exemplo de fonte: periódico, exemplo de instituição: editora.
- **FragmentoTexto**: fragmento do texto contido na página.
- **NumCitacoes**: é um dado complementar apresentado pelo motor. Designa o número de outros trabalham (também indexados por este motor), que citam o trabalho em questão.
- **ArtigosQueCitam**: um dado complementar apresentado pelo motor. Oferece uma URL para uma página de resultados de busca do motor que contenham apenas trabalhos que citem o trabalho em questão.
- **ArtigosRelacionados**: um dado complementar apresentado pelo motor. Oferece uma URL para uma página de resultados de busca do motor que contenham apenas trabalhos que estejam relacionados com o trabalho em questão.
- **PesquisaWeb**: URL para uma página de resultados de busca do termo no motor Google.
- **NumVersoesDoTrabalho**: número versões encontradas em diferentes URLs e indexadas por esse motor.
- **VersoesDoTrabalho**: *link* para uma página de resultados com *links* para as versões do trabalho encontradas em diferentes URLs e que sejam indexadas por esse motor.

Alguns resultados apresentam ainda uma informação a mais antes do título do trabalho como: `[citação], [livro]`, etc.

Com relação ao motor Google Scholar, pode-se dizer que é de fácil utilização, apresentando como uma vantagem o fato de que os resultados são complementados com dados fornecidos pelo motor como: artigos que citam, número de artigos que citam, artigos relacionados e número de versões.

O motor não apresenta *meta tags* significativas que possam contribuir para o trabalho de padronização. A seguir, na Tabela 3, é apresentada a padronização extraída do motor de busca Google Schoolar.

Tabela 3: Padrão extraído do motor de busca Google Scholar.

Campo	(codigo) tag_inicio	(codigo) tag_fim
URLTrabalho	`<p class=g><h3 class="r">`	`"`
Titulo	`;">`	`</a></h3>`
URLVersaoDisponibilizada	`<span class=a>&#x25ba;</span><b><a class=fl href="`	`"`
DominioVersao Disponibilizada	`;">`	`</a>`

AutorFonteAnoIntituicao		
FragmentoTexto	 	<a class=fl href="/
NumCitacoes		
ArtigosQueCitam	 <a class=fl href	">Citado por
ArtigosRelacionados	<a class=fl href="/scholar?hl=pt-BR&lr=&q=related:	">Artigos relacionados
PesquisaWeb	<a class=fl href="http://www.google.com.br/search	">Pesquisa na web
NumVersoesDoTrabalho	">Todas as	versões
VersoesDoTrabalho	<a class=fl href="/scholar?hl=pt-BR&lr=&cluster=	">Todas as

A partir dos resultados de busca sobre "a*lzheimer*" e "*alzheimer introduction*", foram coletados manualmente os dados dos primeiros resultados listados nessas duas busca utilizando-se o padrão extraído mostrado na Tabela 3.

A tabela completa, incluindo observações sobre o padrão de cada campo, pode ser visualizada no Apêndice 2.

3.1.4 Motor de busca Medstory

O motor de busca Medstory apresenta pesquisa sendo informada apenas em um campo, não sendo possível a realização de pesquisas avançadas. Com relação a este fato, o Medstory acrescenta uma ajuda interessante ao usuário – após a realização da busca a partir de uma palavra-chave, na página de retorno dos resultados, o Medstory apresenta uma série de categorias de dados complementares relacionados a busca efetivada pelo usuário. Estas categorias, no caso de busca pela palavra "*alzheimer*" são as seguintes: "*Drugs and Substances*", que permite um refinamento da busca a respeito de remédios relacionados a Alzheimer; "*Conditions*", refere-se a condições para a doença, como diagnóstico; "*Procedures*", relativo a procedimentos a serem realizados em caso da doença; "*In Clinical Studies*", relativo a estudos mais técnicos a respeito; "*Complementary Medicine*", dados sobre medicina complementar; "*Personal Health*", refinamentos sobre questões de saúde, como sintomas da doença; "*Nutrition*", refinamentos sobre nutrição; "*People*", médicos e especialistas no assunto.

Todas estas possibilidades de refinamento da consulta, apresentadas acima, podem auxiliar muito a pesquisa do usuário. No trabalho, foi analisada a codificação destes dados, e percebeu-se que se trata de informação pré-processada, armazenada em um banco de dados específico da Medstory. No momento da consulta do usuário, parece que o sistema consulta no seu banco de dados interno a existência da chave de pesquisa, e apresenta sua informação complementar para refinamento. Por se tratar de uma informação interna da Medstory, não sendo possível acesso por meio automático, não se buscou a padronização destes dados de refinamento.

Quanto à listagem em si das páginas de resultados à busca por palavra-chave, apresenta: título da página, com *link* para URL; texto sobre conteúdo da página (pré-definido); URL da página, por extenso. Os padrões obtidos a partir da análise do código-fonte do motor de busca Medstory estão apresentados na Tabela 4.

Tabela 4: Padrões encontrados no motor de busca Medstory.

Descrição_Campo	(codigo) tag_inicio	(codigo) tag_fim
Título da página, com a URL do resultado retornado	`<p class="result-title"> <a href="redirect?`	`</a></p> <p class="result-text">`
Texto sobre conteúdo da página	`<p class="result-text">`	`<p class="result-source">`
URL da página por extenso	`<p class="result-source">`	`<p class="result-tags"> </p>`

Como se pode perceber na Tabela 4, as *tags* contidas no código-fonte do motor de busca Medstory já possuem um formato mais fácil de ser padronizado, uma vez que os campos *"result-title*", "*result-text*" e "*result-source*" nos levam, respectivamente, para o título da página, ao seu conteúdo e a sua URL.

A padronização completa do motor de busca Medstory pode ser visualizada no Apêndice 2.

3.1.5 Motor de busca Curbside

O motor de busca Curbside oferece respostas baseadas em evidências para responder perguntas médicas de fato (especialista). Utiliza uma sofisticada API (*Application Programming Interface*) patenteada (Praxeon's) que utiliza semântica e *fingerprinting* inteligente com um sofisticado modelo de terminologia médica para extrair o significado de palavras e relações dentro de qualquer texto (estruturado ou não).

Basicamente, retorna uma página com os resultados divididos em categorias: *Best Hits, Visual diagnosis, Quick Consult, The best Evidence e Clinical Trials*. O Código fonte da página retornada possui padrões definidos, com identificadores específicos para cada campo (ver Tabela 5).

É de fácil navegação, sendo considerada para este trabalho a categoria "*Best Hits*" que apresenta os resultados mais relevantes em todas as buscas de categorias e fontes. Podem ser feitos refinamentos nos resultados utilizando os recursos oferecidos na pagina de resultados. Não foram feitos refinamentos para a execução desta análise, pois o foco é a análise do código retornado (e padrões usados) e não a análise aprofundada do *site* em si. Vale ressaltar que, quanto mais detalhes forem informados na busca, melhor serão os resultados apresentados.

Tabela 5: Padrões encontrados em Curbside.md

Descrição_Campo	(codigo) tag_inicio	(codigo) tag_fim
Título do site, com a URL do resultado	<a href="	
Titulo do site, com a URL interna	<title>	</title>
Texto sobre conteúdo do site		
URL do site por extenso	<a class="aHit" href="#"	
Autores		
Data		
"Go to full Paper"(p/ resultado)		Go to Full Paper

Foram considerados os atributos/padrões mais relevantes de acordo com a ontologia definida para o trabalho. Este motor de busca apresenta atributos e padrões bem identificados, com resultados ricos para popular o banco de dados, o que caracteriza este motor como bem organizado e definido, um ponto favorável para busca com qualidade.

3.2 Fase 2 – Padronização das páginas sobre saúde

Na segunda etapa da pesquisa, foram analisados os padrões de formato das páginas propriamente ditas, retornadas pelas buscas nos motores. Para isso, adotou-se como estratégia a análise das URLs que estivessem na primeira página de resultado de cada busca. Ainda, foram levados em consideração os padrões de qualidade definidos no modelo E-R, apresentado na Figura 1 da seção 1, para buscar padrões de formato presentes em *tags* de meta dados das páginas.

Para análise das páginas, não foi discriminado o motor de busca responsável pela obtenção da página porque, em geral, este tipo de informação não foi relevante para a padronização. Além disso, muitas páginas ocorrem repetidamente como resultado de mais de um motor de busca.

Com relação a alguns padrões encontrados, a página da Medline[13], que possui bastante abrangência na área médica e apresenta conteúdo, a principio, confiável, apresenta algum tipo de padrão interessante no seu código-fonte:

- data_atualização - <td class="updated">
- nome_autor - <p class="attribution">

[13] http://www.nlm.nih.gov/medlineplus/alzheimersdisease.html

Outro exemplo é a página do Dr. Jonny Bowden[14], que apresenta um formato interessante em suas tags de dados:

- título - <h3 class="post-title"> ... </h3>
- data_atualização - <h2 class="date-header"> ... </h2>
- nome_autor - <p class="post-footer“>

Usando como exemplo a página do Dr. Jonny Bowden, observou-se também que várias páginas utilizam as *tags* de destaque para títulos, como <h1>, <h2> ou <h3> para identificação de títulos, ou datas. Assim, analisando as páginas, puderam-se determinar alguns padrões simples e não totalmente decisivos, como:

- Padrão para título: <title> ou <meta name=“Title”>
- Padrão para datas: “updated” ou © ou “date”

Dentre as páginas analisadas individualmente, algumas delas eram páginas de periódicos acadêmicos. Apesar de essas páginas apresentarem carência de padrões bem estabelecidos, foi observado apresentam diversos metadados no código-fonte, referentes ao trabalho sendo mostrado pela página do periódico. Esses metadados contém informações como título, palavras-chaves do trabalho, colaboradores e outros detalhes relativos ao trabalho. Diversas páginas de periódicos diferentes apresentaram os mesmos metadados. Vale ressaltar que esses metadados referem-se ao trabalho em si e não a pagina do periódico. Ainda assim, a presença desses metadados pode ser útil nas buscas motivadas por perfis de usuários classificados como “especialistas”.

3.3 Análise de aplicação dos padrões obtidos

Apesar de não terem sido encontrados padrões bem definidos para apresentação dos dados nas páginas, entende-se que a definição de alguns padrões pode estar associada a **termos específicos**. Desta forma, sugere-se que o mesmo padrão, associado a um **sinônimo** do termo, possa implicar na coleta do mesmo campo em outras páginas. A questão está em como definir o grupo de termos sinônimos.

Uma sugestão é o uso de termos definidos na Wordnet[15], objetivando encontrar sinônimos para palavras-chave procuradas nas páginas. Exemplo: *author*, sinônimos na Wordnet: *writer*, *generator*, *source*. Já por *source*, é possível encontrar muitos outros sinônimos. Também poder ser feito o uso de conceitos – aplicação de **ontologias**. Em [8], é sugerido o uso de ontologias para apoiar a determinação de tópicos da página, podendo estas ser enriquecidas a partir de termos da ***Wordnet.***

[14] http://www.jonnybowden.com/2009/03/fast-food-diet-may-raise-alzheimers.html
[15] http://wordnet.princeton.edu/

Outra sugestão é o possível uso de técnicas de ***Information Storage and Retrieval***, a serem aplicadas na identificação de determinado tópico pertinente a pagina, visto que é importante para padronização a definição do tópico pertinente (fase de *information filtering*) [9]. Algumas ***tags html,*** por si só podem ser boas indicadoras para a definição/busca de padrões, por exemplo, as *tags* <*h1*>, <*h2*>, <*h3*>, <*h4*> podem conduzir a padrões de campos, entre eles: título, nome do autor e contato.

Outra possibilidade é a **concatenação de *links***, o que geralmente representa uma descrição simples e objetiva do conteúdo da página para a qual o *link* faz referência [10]. No caso do trabalho, isso pode ser usado para preparar o extrator. No momento em que é feita a análise por concatenação de *links,* dada a página resultante apresentada pelo motor de busca, a concatenação dos *links* pode ser usada para se concluir se cada uma das páginas *web,* referentes ao resultado da busca, é interessante para pesquisa ou não. Essa análise da concatenação de *links* pode ser usada combinada à análise do valor de reputação de documentos, o que pode ser obtido pelo próprio motor de busca usado (como ***PageRank***^{TM} [11]). Técnicas para **concatenação de textos de *hiperlinks*** podem ser usadas para identificação do conteúdo das páginas e, conseqüentemente, de título.

Estas técnicas, usadas em conjunto, podem ser usadas para **validação** de padrões definidos para título, por exemplo. Em função da natureza heterogênea da *web*, não só a definição dos padrões, mas também a validação destes deve ser considerada.

3.4 Considerações Finais sobre Padronização

Foram analisados um total de 5 motores de busca. Estes padrões foram testados e propiciaram a coleta manual dos valores esperados.

Com relação aos **motores**, foram possíveis as seguintes considerações:

- Na padronização do Medstory, Google e Curbside não foram enfrentados muitos problemas.
- O Google Health aponta para localização de serviços médicos, utiliza padrões comuns para motores de busca do Google (ex.: Google Maps).
- Motores que utilizam APIs semântica para textos (como o Curbside), estruturados ou não, podem oferecer melhor qualidade nos resultados.
- No caso do motor Curbside, com realização de buscas manuais, encontrou-se diferença no código-fonte apresentado pelo *browser* Internet Explorer 8, Firefox e Google Chrome. Essa observação leva a considerar que *browsers* diferentes podem apresentar diferentes códigos-fonte, influenciando na coleta de dados.
- Motores como Google e Google Scholar evoluem com alta freqüência, apresentando diferenças em relação aos padrões observados.

Na análise, concluiu-se que alguns motores de busca apresentam características específicas. Alguns padrões utilizados por eles adotam tecnologias de busca específicas e distintas entre si (ou combinadas) oferecendo, em alguns casos, resultados diferenciados de acordo com os padrões adotados. Nesses casos, a tecnologia empregada no motor de busca determina o padrão do código fonte retornado e a precisão da busca. A análise completa dos padrões dos motores de busca encontra-se no Apêndice 2.

Com relação às páginas sobre saúde, mais especificamente sobre Doença de Alzheimer, foram analisadas um total de 16 páginas. A descoberta de padrões entre elas se tornou difícil, pois cada uma apresenta os dados de forma própria. Além disto, muitas delas não informam dados necessários para determinação de qualidade, como nome do autor e data de atualização da página.

Algumas peculiaridades foram encontradas durante a pesquisa e análise dos padrões. Alguns navegadores compilam o código fonte das páginas, adicionando informações ao seu conteúdo (código fonte, não na exibição ou conteúdo da página a ser exibida), ou até omitindo informações relevantes como algumas *meta tags* importantes. Isso ocorre porque alguns navegadores compilam o código fonte das páginas apresentadas, suprimindo ou enriquecendo o código de informações.

A análise dos motores de busca foi realizada com os navegadores Internet Explorer 8, Mozilla Firefox e o Google Chrome versão 2.0, rodando no sistema operacional Windows Vista, Windows XP e GNU/Linux (neste último, somente Firefox).

O Google Health aponta para localização de serviços médicos, e busca combinada com os dados do cadastro do usuário, porém com foco na localização de serviços médicos. Este serviço de busca, para sua área de negócio, se assemelha a um sistema de integração de informações, porém não se aplica ao projeto aqui proposto, pois não são disponibilizadas informações específicas sobre a doença, e sim sobre serviços e profissionais da área.

4 Extração dos Dados

Esta seção do relatório compreende uma análise e estudo da arte sobre os problemas e conceitos encontrados para extração de dados na busca de informações confiáveis na *web*. Esses conceitos foram adquiridos através dos artigos estudados.

Inicialmente seria realizada somente a extração a partir de uma tabela contendo uma série de padrões encontrados nas páginas pesquisadas. Entretanto, no decorrer dos estudos, observou-se que havia dependências entre as diferentes etapas descritas neste relatório. A solução adotada foi realizar uma separação entre os atributos do modelo E-R, apresentado na seção 1. Os atributos tratados especificamente nesta etapa de extração foram os seguintes:

- Idioma da página;
- *Inlinks*;

- *Outlinks*;
- E-mail do autor;
- Autor é especialista (verifica se o autor é especialista em determinada área).

4.1 Ferramentas de Apoio à Extração

Para o desenvolvimento da solução proposta foram utilizadas algumas ferramentas de apoio para auxiliar na busca de dados em *sites* da *web*, são elas: *Web Harvest* e *XPather*.

4.1.1 Web Harvest

A ferramenta *Web-Harvest*[16] fornece uma API que permite consultar servidores *web*, obter a página HTML de resposta, transformá-la para XHTML (*Extensible Hypertext Markup Language*) e aplicar tecnologias de manipulação de documentos XML (*Extensible Markup Language*) tais como *XPath*, *XQuery*, XSLT (*Extensible Stylesheet Language Transformations*) para extrair as informações desejadas.

Todo procedimento de extração no *Web-Harvest* é definido pelo programador através de arquivos de configuração, no formato XML. Cada arquivo de configuração descreve uma seqüência de processadores executando determinadas tarefas para atingir um objetivo final. Estes processadores rodam em forma de *pipeline*, ou seja, a saída de um processador é utilizada como entrada para outro. A Figura 6 mostra a ordem no qual as funções presentes na *Web-Harvest* estão organizadas.

Figura 6: Visão geral da arquitetura da aplicação *Web-Harvest.*

Essa ferramenta segue a idéia de *pipelines*, onde um fluxo serve de entrada para outro. Dessa forma, na Figura 7 é apresentado um exemplo de arquivo de configuração em XML

[16] http://web-harvest.sourceforge.net/

que exemplifica a obtenção automática de todas as imagens presentes no site http://news.bbc.co.uk.


```
<?xml version="1.0" encoding="UTF-8"?>

<config charset="UTF-8">
    <var-def name="urlList">
        <xpath expression="//img/@src">
            <html-to-xml>
                <http url="http://news.bbc.co.uk"/>
            </html-to-xml>
        </xpath>
    </var-def>

    <loop item="link" index="i" filter="unique">
        <list>
            <var name="urlList"/>
        </list>
        <body>
            <file action="write" type="binary" path="images/${i}.gif">
                <http url="${sys.fullUrl('http://news.bbc.co.uk', link)}"/>
            </file>
        </body>
    </loop>
</config>
```

Figura 7: Exemplo de arquivo de configuração para obter imagens de um *site*

No primeiro fluxo visto na Figura 7, são obtidos todas as URLs que representam imagens, usando a expressão *XPath "//img/@src"* e essa lista de URLs é armazenada em uma variável. No segundo fluxo descrito, é executado um laço pela lista de URLs e são salvas em disco todas as imagens apontadas pelo *link* presente na lista de URLs a cada iteração. Além disso, na definição do laço (*tag <loop>*) também é passado um atributo *unique* para que sejam removidas possíveis duplicatas, que nesse caso seriam URLs iguais.

4.1.2 XPather

O *XPather* é uma ferramenta semi-automática para a localização de padrões em documentos *web*, é uma extensão do *plugin* do *Web Browser Firefox* que pode ser utilizada gratuitamente. Permite avaliar uma expressão *XPath*. O modo de funcionamento da ferramenta é simples, clica-se em um *site* após o *plugin* ser instalado e seleciona a opção *XPather*, abre-se então uma janela com a ferramenta e todos os possíveis endereços a ela associados, como mostrado na Figura 8.

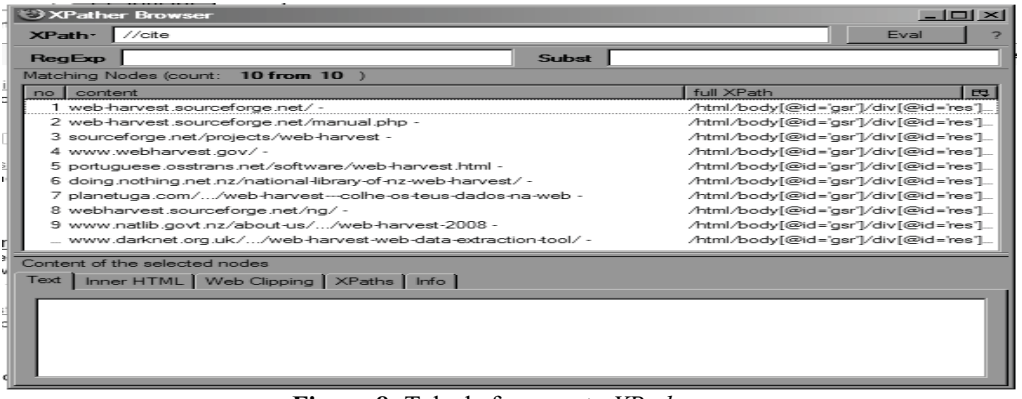

Figura 8: Tela da ferramenta *XPather*.

4.2 Identificação do idioma

A forma de pesquisa pelo idioma procura identificar a língua estrangeira em que o *site* foi desenvolvido. O processo para identificação do idioma é apresentado na Figura 9, onde se tem como entrada a URL da página a qual se deseja descobrir o idioma. A partir desta URL obtém-se o HTML desta página. Em seguida, é verificado se este HTML possui uma *meta tag* que represente o idioma segundo o padrão Dublin Core[17], ou seja, *meta tag language*.

Em caso afirmativo, a extração é feita via *meta tag,* produzindo como saída o idioma da página. Em caso negativo são utilizados serviços *web* identificadores de linguagem[18], os quais, dado um texto como entrada, informam o idioma deste texto. Para isto, extrai-se apenas uma passagem do texto da página, o qual deve conter a palavra *"alzheimer"* e ter no mínimo 20 palavras. Não é utilizado todo o texto da página, para sobrecarregar o menos possível o serviço utilizado, como ocorreu, em um dos testes, em que o texto que falava sobre a doença de Alzheimer estava em um idioma e o resto da página em outro.

Após obter a passagem de texto, ela é submetida a um dos serviços *web* escolhidos aleatoriamente para não sobrecarregar um determinado serviço. Após a submissão é realizada a extração via identificador. Para se obter maior precisão, deve-se submeter a passagem a *n* identificadores até que um idioma consiga 2 votações. Esta votação é importante, pois cada identificador tem uma construção própria, logo, alguns apresentam melhores resultados para determinados idiomas enquanto que outros apresentam melhores resultados para outros idiomas. Este esquema de lista de identificadores de idiomas permite também que se tenham outras opções quando um *site* está indisponível. Além disso, os

[17] http://dublincore.org/

[18] Estes serviços *web* identificadores de linguagem são *sites* que permitem ao usuário informar em um campo *textarea* um texto e submeter o formulário, a fim de apresentar o idioma do texto informado

proprietários destes identificadores estão constantemente aperfeiçoando seus algoritmos, o que se torna uma vantagem sobre usar um algoritmo local, embora este seja mais rápido.

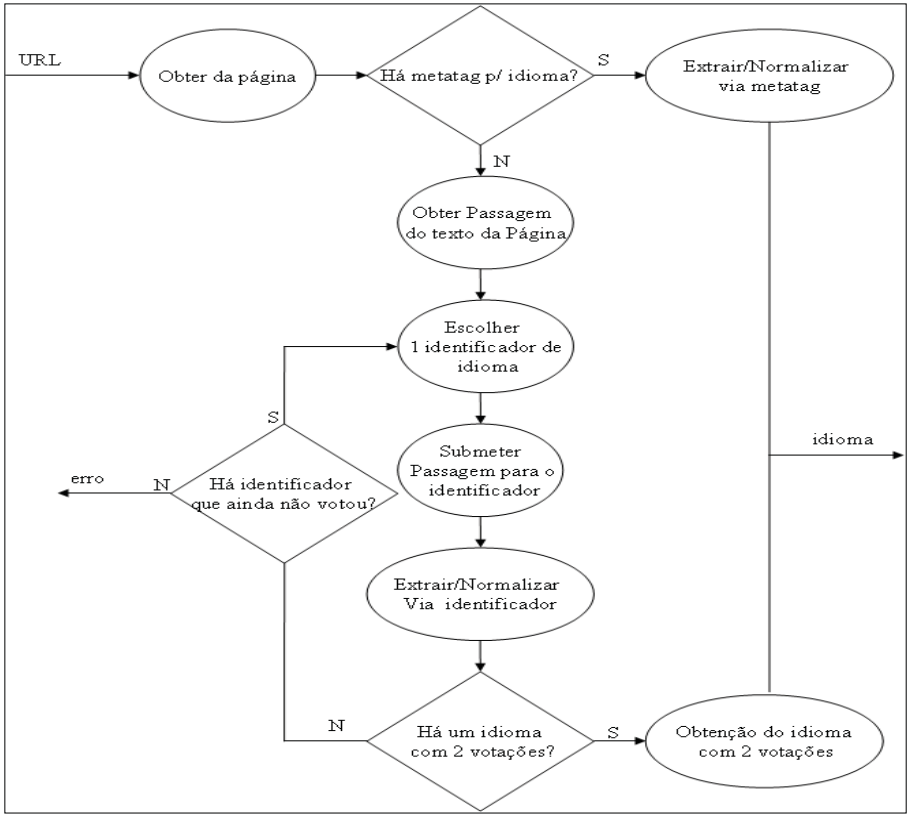

Figura 9: Processo de identificação do idioma.

4.2.1 Extração do idioma via *meta tag*

O formato de uma *meta tag* de idioma no padrão Dublin Core é apresentado na Figura 10, onde o atributo *name* da *meta tag* identifica o metadado especificado (no caso, *language*) e o atributo *content* especifica o valor do metadado (no caso, *english*). Assim, caso a página possua uma *meta tag* que represente o idioma, a extração do mesmo é realizada através da ferramenta *Web-Harvest* com a expressão *XQuery* ilustrada na Figura 11, a qual retorna o valor do atributo *content* de uma *meta tag* que possua como valor para o atributo *name* o valor *language*.

```
<meta name="language" content="english" />
```

Figura 10: Exemplo de *meta tag*.

```
//meta[name= 'language']/@content
```

Figura 11: Código *XQuery* para extração do idioma de uma *meta tag.*

Os formatos para o atributo idioma extraídos por meio de *meta tag* são apresentados na Tabela 6, onde se percebe a existência de formatos diferentes, que precisam ser normalizados. Para normalização dos idiomas extraídos por extenso, basta traduzi-los do idioma origem para o português através de um dicionário, por exemplo. Para normalização dos idiomas extraídos como sigla basta utilizar um dicionário que tenha o idioma por extenso, sua sigla e a tradução para português. Finalmente, para normalização dos idiomas extraídos como sigla e seguidos por hífen e pela sigla do país, basta eliminar os caracteres após o hífen e inclusive o hífen, após seguindo o mesmo processo da normalização da extração de sigla de idioma.

Tabela 6: Formatos de idioma extraídos através de *meta tag*

Extração	Descrição
english	Idioma por extenso
EN	Sigla do idioma
EN-GB	Sigla do idioma hífen sigla do país

4.2.2 Extração do idioma via identificador

Nem sempre uma página possui declarada de forma explícita, por meio de *meta tags,* o idioma no qual está descrito o seu conteúdo. Neste caso, podem ser utilizados programas que realizam esta identificação por meio da análise do texto. Os identificadores de idioma encontrados e utilizados são apresentados na Tabela 7.

Tabela 7: Identificadores de idioma e suas URLS

Identificador	URL
Xérox	http://www.xrce.xerox.com/competencies/content-analysis/tools/guesser-ISO-8859-1.en.html
Fuzzums	http://www.fuzzums.nl/~joost/talenknobbel/index.php
Applied Language	http://www.appliedlanguage.com/translation_resources/language_identifier.aspx?
Translated Labs	http://labs.translated.net/language-identifier/
Google Language Detection	http://www.google.com/uds/samples/language/detect.html

Estes identificadores podem ser utilizados em conjunto com a ferramenta *Web-Harvest*. As configurações da ferramenta *Web-Harvest* específicas para cada identificador estão apresentadas na Tabela 8, onde `action` corresponde a URL a ser submetida pela ferramenta *Web-Harvest*, `método` corresponde ao método de submissão do formulário, `textArea` e `button` são parâmetros a serem submetidos.

Tabela 8: Configurações dos identificadores para extração do idioma.

Identificador	Action	Método	TextArea	Button
Xérox	http://www.xrce.xerox.com/cgi-bin/mltt/LanguageGuesser	Post	Text	Não precisa
Fuzzums	http://www.fuzzums.nl/~joost/talenknobbel/index.php	Post	Input	**name**="action" **value**= "Guess"
Applied Language	http://www.appliedlanguage.com/translation_resources/language_identifier.aspx?	Post	ctl00$ctl00$ctl00$ctl00$ContentPlaceHolderDefault$baseMasterContentPlaceHolder$MenuMasterContentPlaceHolder$ctl00$Page_language_identifier_3$txtTextToIdentify	Name="ctl00$ctl00$ctl00$ctl00$ContentPlaceHolderDefault$baseMasterContentPlaceHolder$MenuMasterContentPlaceHolder$ctl00$Page_language_identifier_3$btnIdentifyText" value="Guess the language"
Translated labs	http://labs.translated.net/language-identifier/	Post	Text	Não precisa
Google Language Detection	http://www.google.com/uds/samples/language/detect.html	Post	Source	Não precisa

Cada identificador retorna o idioma em um formato específico, como observado na Tabela 9, logo, são necessárias algumas etapas para normalização:

- Eliminar caracteres extras (caracteres após o "_", inclusive o "_" e caracteres após o "(" e inclusive o "(");
- Traduzir o idioma de inglês para português.

Tabela 9: Formato dos idiomas extraídos.

Identificador	Extração	Exemplo
Xérox	Idioma (em inglês)_codificação	Portuguese_iso1
Fuzzums	Idioma (em inglês)	Portuguese
Applied Language	Idioma (em inglês)_codificação	Portuguese_UTF8
Translated Labs	Indisponível	Indisponível
Google Language Detection	Idioma (em inglês) (...)	ENGLISH (realiable : 0,325898)

Para avaliação do modelo definido, foram executados testes com 12 páginas. Destas, 9 eram em inglês e 3 em português. Como resultado, observou-se que 4 destas páginas obtiveram as duas primeiras votações iguais, correspondendo ao idioma correto. Já 8 páginas necessitaram de três votações para chegar ao idioma correto, uma vez que as duas primeiras apresentaram resultados distintos. Em ambos os casos o identificador que apresentou a votação errada foi o Fuzzums. Conclui-se que com o esquema de votação, consegue-se uma boa precisão, pois um identificador ataca os pontos fracos de outros.

Com relação à implementação, foi desenvolvida uma aplicação *web* usando a linguagem Java, a ferramenta *Web-Harvest* e a tecnologia *Struts2*[19]. O objetivo desta implementação foi mostrar a viabilidade da implementação do modelo, uma vez que a principal preocupação foi pesquisar formas para solução dos problemas.

De forma simples, a implementação realizada consiste em: dada uma determinada URL, extraí-la via *Web-Harvest* por meio de uma passagem do texto. Esta passagem objetiva encontrar o conteúdo de qualquer *tag* do texto da página que possua a palavra *"alzheimer"* e que tenha mais de 20 palavras. Em seguida, usando-se a ferramenta *Web-Harvest*, submete-se esta passagem a dois identificadores (Xerox e Fuzzums) para que o idioma seja extraído e apresentado na tela. O código-fonte referente ao procedimento principal para extração do idioma é apresentado no Apêndice 3.

4.3 Extração de *inLinks* e *outLinks*

Os *Inlinks* e *Outlinks* estão associados à perspectiva da URL que recebe os dados e também à URL que fornece o *link*. Os *inLinks* se referem aos *links* que apontam para um determinado *site*. São usados por muitos motores de busca como um atributo no processo de classificação e para determinação do *ranking* de uma determinada página. Já os *outLinks* são *links* de determinado *site* que apontam para outros *sites,* hospedados fora de seu domínio de origem. A Figura 12 apresenta um esboço sobre o funcionamento. No contexto da pesquisa sobre extração de dados, esses atributos são considerados para posteriormente determinar-se a qualidade de *sites*/páginas sobre saúde.

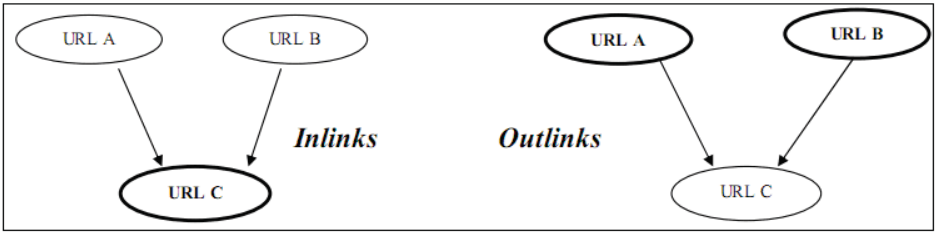

Figura 12: Funcionamento de *Inlink's* e *Outlink's*. Fonte: [12].

[19] http://struts.apache.org/2.x/

Para realização da extração desses atributos, são utilizadas expressões da linguagem *XPath*, por meio da ferramenta *Web-Harvest*. Nesse trabalho, foi definido que as informações sobre os atributos *inLinks* e *outLinks* são obtidas através do Google, e que este processo segue uma seqüência de passos.

- Para a obtenção dos *inLinks*, devem ser seguidos os seguintes passos:
- Submete-se uma consulta ao *Browser Google Web* (usando atributo *link*): `http://www.google.com.br/search?q=link%3A+`**`<url_página>`**;
- Após a obtenção da página de resultados, para cada *n* páginas listadas, executar a expressão `XPath: //cite`;
- Armazenar cada lista de URLs obtida por página.

Para a obtenção dos *outLinks*, devem ser seguidos os seguintes passos:

- Obtém-se a página a que se deseja consultar *os outLinks*;
- Após a obtenção da página, executar a expressão XPath: **`//a[not(contains(@href = 'DOMINIO'))]/@href`**, onde DOMINIO representa o domínio da página desejada;
- Através da expressão citada no passo anterior, são excluídos *links* dentro do mesmo domínio da página consultada;
- Ao contabilizar o número de *outLinks*, são excluídas as duplicatas e *links* que fazem parte de um mesmo domínio são contabilizados apenas como uma entrada.

4.4 Extração do *e-mail* do autor

A tarefa de extração do atributo *e-mail* do autor descrita neste ponto do relatório se trata da localização, para extração, do *e-mail* do autor em fontes externas à página a ser analisada. A primeira intenção na realização desta tarefa foi a localização de páginas que contivessem o nome do autor e seu *e-mail* e, destas páginas, extrair os *e-mails* existentes. Entretanto, percebeu-se que seria mais simples realizar a busca, por meio de um motor de busca, onde o *e-mail* do autor fosse retornado no resumo dos resultados do motor de busca.

O processo completo de identificação do *e-mail* do autor é apresentado na Figura 13, onde se submete uma consulta no padrão **Scielo** ao motor de busca. Scielo é um *site* que possui vários artigos e, para cada primeiro autor, tem-se o *e-mail*, com isso, se o autor da página que está sendo avaliada tiver um artigo como primeiro autor no *site Scielo,* consegue-se extrair o *e-mail* com precisão.

Uma vez retornados os resultados pelo motor de busca, extrai-se todos os *e-mails* da primeira página de resultados, no caso, extrai-se todos os termos da página que satisfazem a expressão regular apresentada na Figura 14. Caso nenhum *e-mail* seja extraído, tenta-se submeter uma consulta padrão de obtenção de *e-mail* em qualquer *site* e, novamente, os *e-*

mails da primeira página do motor de busca são extraídos. Caso em ambas as consultas não sejam extraídos *e-mails*, então o *e-mail* não será extraído por este processo.

Porém, se foi possível, em alguma das consultas, a extração de algum endereço de e-mail, então se obtém o *hit count* (número de resultados para a busca) do *e-mail* com o nome do autor, sendo que o *e-mail* que obter maior *hit count* é considerado como correto para a busca. Caso ocorra empate, utiliza-se a função de similaridade Levenshtein [13] entre o nome do autor e o *e-mail*. Ressalta-se que, para comparar a similaridade entre o nome do autor e o *e-mail*, é necessário manter o domínio do *e-mail*, pois parte do nome do autor pode estar no *e-mail*.

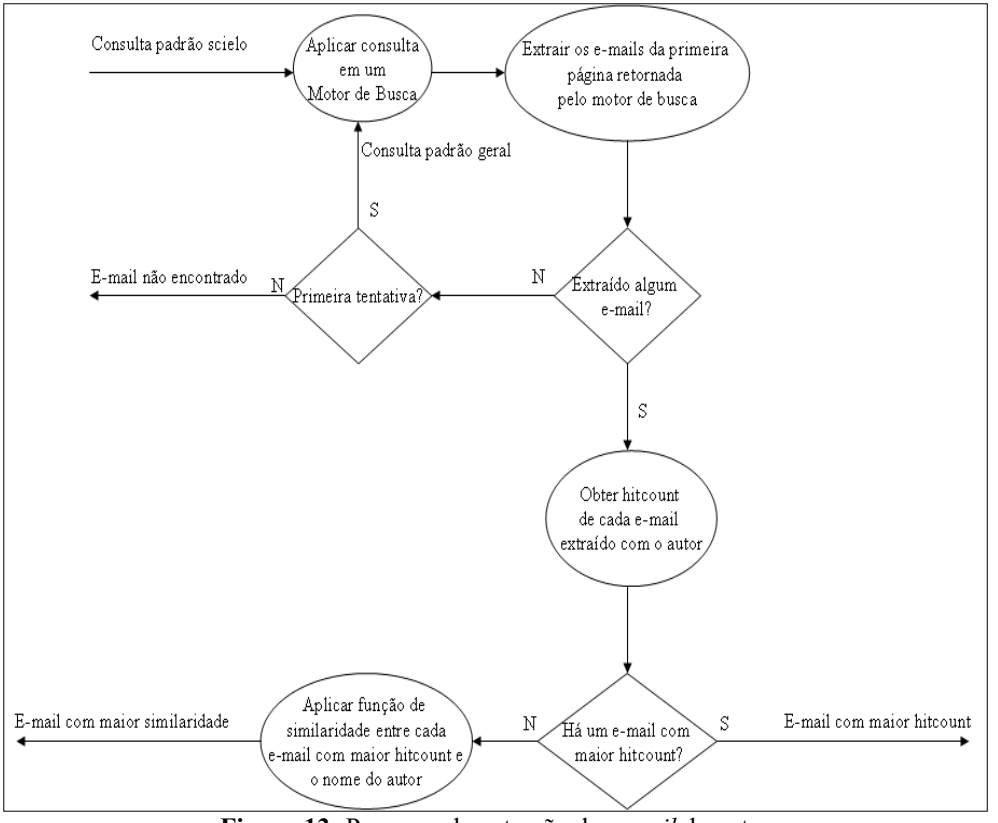

Figura 13: Processo de extração do *e-mail* do autor

```
(?i)([a-z][a-z0-9._+-]+@([a-z][a-z0-9-]+\\.)+[a-z]{2,4})
```

Figura 14: Expressão regular para o *e-mail*.

Na Figura 15 é apresentada a consulta no padrão Scielo, a qual tem por objetivo retornar, nos resultados do motor de busca, o *e-mail* do autor que está no *site* Scielo. Os termos "alzheimer", "*neurol*", "saúde" e "*health*" servem para desambiguação de nomes, eliminando, por exemplo, o *e-mail* de uma pessoa que escreve sobre teologia.

O símbolo "+" seguido por um termo, ambos entre aspas, identifica que aquele termo deve aparecer no documento. O símbolo "*" representa qualquer caractere, enquanto que o operador "OR", com operandos compostos pelo símbolo "+" e por um termo, representa que pelo menos um dos termos deve ocorrer nos documentos. Estes padrões são relativos a consultas ao motor de busca Google e isto foi descoberto por tentativa e erro, uma vez que não foi encontrado nenhum manual no motor de busca Google para a realização de consultas booleanas mais sofisticadas.

```
"+Endereço para correspondência:*nomeAutor" "+E-mail:"
"+alzheimer OR +neurol OR +saúde OR +health"
```

Figura 15: Consulta com padrão Scielo.

A consulta padrão **Geral** é apresentada na Figura 16, onde se observa que os resultados devem conter o nome do autor, a palavra "*e-mail*" ou a palavra "*email*" e uma das seguintes palavras "*health*", "alzheimer", "*neurol*", "saúde".

```
"+nomeAutor" "+e-mail OR +email" "+health OR +alzheimer OR
+neurol OR +saúde"
```

Figura 16: Consulta utilizando um padrão geral.

A consulta para obtenção do *hit count* de um *e-mail* extraído com o nome do autor é apresentada na Figura 17.

Na Figura 18 é apresentada uma parte do código XML de configuração do *Web-Harvest* para extração do *hit count*, onde está destacada a URL a ser submetida, bem como o parâmetro *q*, passado pelo método GET, o qual corresponde a consulta a ser realizada. Além disso, destaca-se a expressão *Xpath* de extração do *hit count* na página.

```
"+nomeAutor" "+email_extraído"
```

Figura 17: Consulta do *hit count* do nome do autor com *e-mail* extraído.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <config charset="UTF-8">
    <!-- ${salvarEm}   -->
  - <file action="write" path="${salvarEm}">
    - <template>
        <![CDATA[  <hitcount>   ]]>
      </template>
    - <loop item="row">
      + <!--   -->
      - <list>
        - <xpath expression="//body[@id='gsr']/div[@id='ssb']/p/b[3]">
          - <html-to-xml>
            + <!--   -->
              <http url="http://www.google.com/search?q=${consulta}" />
            </html-to-xml>
          </xpath>
        </list>
      + <!--   -->
      + <body>
      </loop>
      <![CDATA[  </hitcount>   ]]>
    </file>
  </config>
```

Figura 18: Configuração da ferramenta *Web-Harvest* para extração dos *hit counts*.

Para avaliação do modelo descrito, foram utilizadas as mesmas páginas usadas nos testes do atributo idioma, sendo que em 7 páginas foi identificado manualmente o autor da página, onde, exceto em uma página que possuía dois autores, as demais apresentavam apenas um autor. Dos 8 autores, 4 estavam em páginas em português e 4 em páginas em inglês. A partir do modelo definido, foi possível extrair corretamente o *e-mail* de 6 autores. Os dois extraídos erroneamente tiveram um *hit count* muito baixo. Em 5 páginas não foi possível identificar manualmente o autor, mas encontrou-se a organização proprietária da página. Assim, testou-se o modelo definido para obtenção dos *e-mails* destas organizações, sendo obtidos 2 acertos e 3 erros. Um dos erros se tratou da obtenção do *e-mail* de uma organização cujo nome continha, além de outras palavras, o nome da organização a qual se estava realmente buscado o *e-mail*. Os outros dois erros retornaram o *e-mail* de pessoas que publicaram em eventos da organização.

Uma característica não considerada neste modelo é a comparação do *hit count* do *e-mail* com o *hit count* do autor e obter apenas o *hit count* do *e-mail*. Quando esta diferença é significativa, pode implicar em o *e-mail* não ser deste autor, mesmo possuindo o maior *hit count* para este autor. Acredita-se que esta comparação poderia ser feita como trabalho futuro.

Para testar a viabilidade da implementação deste modelo, desenvolveu-se uma aplicação *web* relativa à etapa considerada mais importante do processo completo. Na

aplicação, submete-se um nome de autor já normalizado. A aplicação faz a consulta no padrão **Scielo***,* através da ferramenta *Web-Harvest*, e extrai os *e-mails* (palavras que batem com a expressão regular definida na Figura 14) da primeira página de resultados do motor de busca Google. Caso nenhum *e-mail* seja extraído, submete-se a consulta padrão **Geral** e extraem-se os *e-mails*. Depois de extraídos os e-mails obtêm-se seus *hit counts* com o autor, também por meio da ferramenta *Web-Harvest,* e apresenta na tela os *e-mails* extraídos e seus *hit counts* com o autor, em ordem decrescente de *hit count*. Na implementação desta aplicação, utilizou-se a linguagem Java, a ferramenta *Web-harvest* e a tecnologia Struts2.

O código da classe responsável pela obtenção do *e-mail* do autor é apresentado no Apêndice 3.

4.5 Tratamento da informação se autor é especialista

Essa funcionalidade visa apontar para a referida página que possuir autores com artigos científicos e/ou citações, a fim de obter maior credibilidade nos dados extraídos da referida página.

Para a extração dos atributos é utilizada a página *web* da PubMed como referência aos itens a serem pesquisados para o tópico em questão, no caso, "Alzheimer"[20]. A partir da definição dos parâmetros *n* e *m*, necessários para determinação de "autor é especialista*"*, a extração dos atributos foi realizada de acordo com as seguintes regras:

- O autor possui, pelo menos, *n* artigos sobre Alzheimer publicados na PubMed nos últimos *m* anos?;
- O autor deve ter, pelo menos, *n hit counts* apontados no motor de busca Google Scholar para a pesquisa contendo o nome do autor e a palavra Alzheimer;
- O autor deve ter, pelo menos, *n hit counts* apontados no motor de busca Google para a pesquisa contendo o nome do autor e a palavra Alzheimer (posteriormente, esta fase foi eliminada).

A Figura 19 apresenta uma pesquisa por autor denominado ***Burns JM Alzheimer*** onde pode ser observado 9 resultados na busca por esse autor. *M* anos se refere ao ano inicial em que a pesquisa buscará por publicações desse autor, neste caso, a pesquisa irá buscar desde 01/01/2004.

[20] http://www.ncbi.nlm.nih.gov/pubmed/advanced?term=Alzheimer

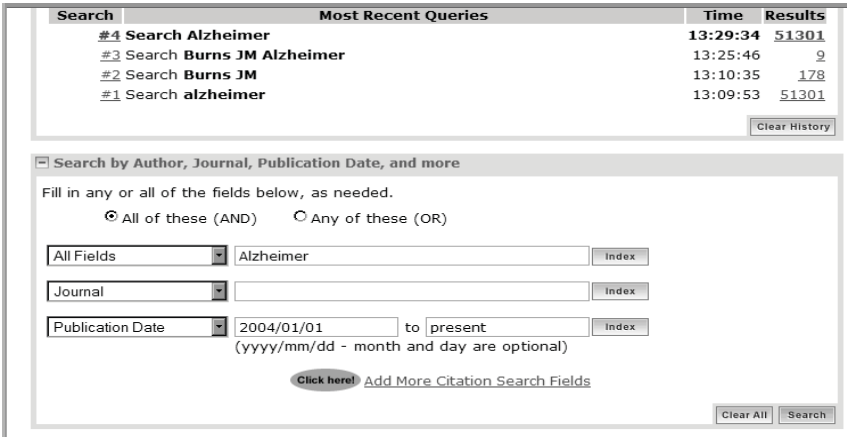

Figura 19: 1ª Etapa – Verificação do número de artigos publicados em *m* anos.

A Figura 20 apresenta o resultado da pesquisa, sendo gerado o código em XML.

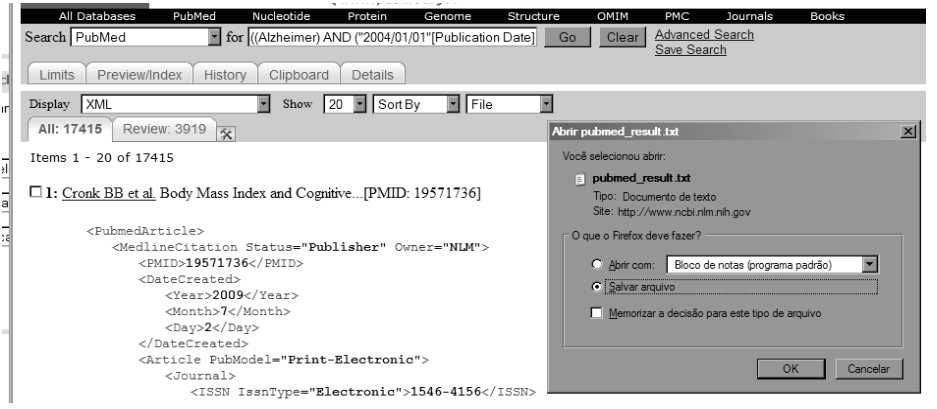

Figura 20: 2ª Etapa - Geração dos resultados em um arquivo formato XML.

A Figura 21 apresenta a 3ª Etapa, que contém o código XML gerado com as informações sobre o referido autor, bem como o número de artigos publicados. Essas informações, entre outras, podem ser consultadas através de uma consulta escrita em linguagem XQuery.

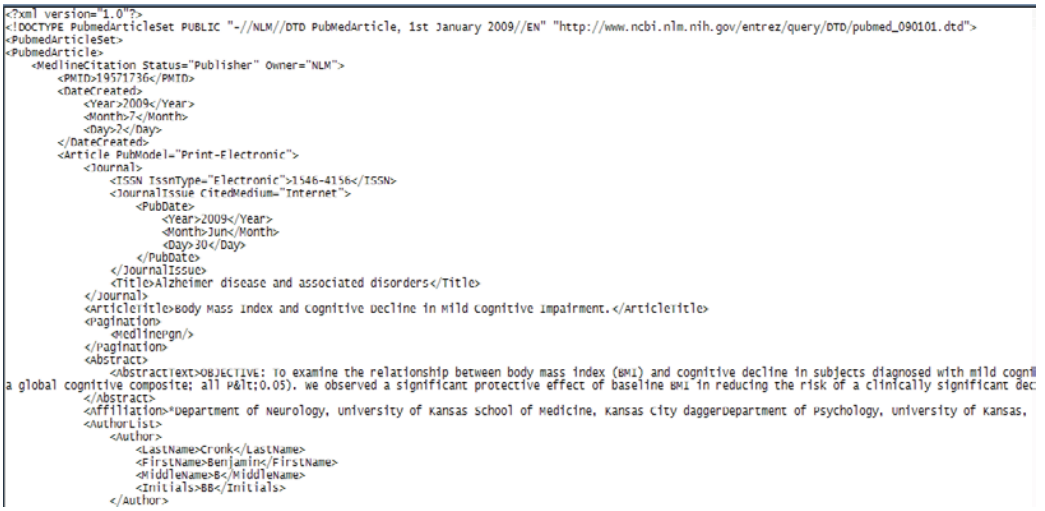

Figura 21: 3ª Etapa - Contagem do número de artigos para o autor avaliado.

A Figura 22 apresenta a consulta de um autor extraído do código XML gerado pelo motor de busca Google Scholar, onde se observa os *hit counts* para o mesmo autor, neste caso, 5.

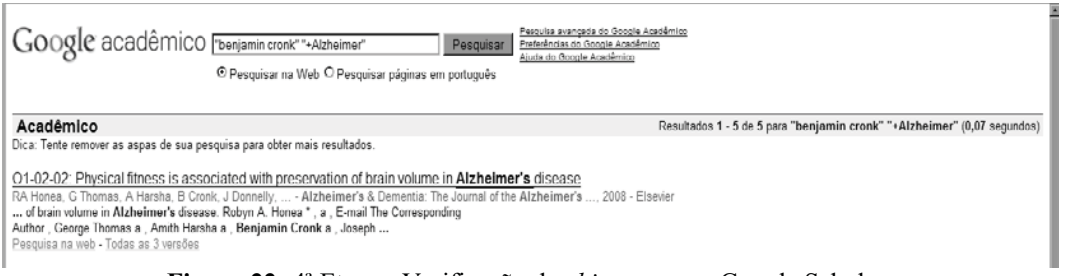

Figura 22: 4ª Etapa - Verificação dos *hit counts* no Google Scholar.

4.6 Consideração sobre o tipo de usuário

Para informar para qual tipo de usuário uma determinada página destina-se (por exemplo: leigos, técnicos ou médicos) foi discutida possibilidade de utilização do idioma e do índice que o editor de texto Microsoft Word utiliza para dizer a escolaridade que o leitor deve possuir para ler o texto.

Por fim, a sugestão proposta é a utilização de classificação baseada em centróides [14]. Este consiste de um algoritmo de descoberta de conhecimento supervisionado que tem como objetivo classificar documentos, onde dada uma coleção de treinamento com

documentos previamente classificados, cria-se um modelo para classificar novos documentos.

Para mostrar o funcionamento deste algoritmo para o contexto do trabalho é apresentada a Figura 23, onde são colocados 6 vetores, para representar a utilização de 6 páginas, cada uma sendo classificada manualmente de maneira a informar para qual tipo de usuário ela se destina (leigos, técnicos e médicos). Obviamente, foram utilizadas apenas 6 páginas para facilitar a visualização, mas o ideal é que o conjunto de treinamento seja maior.

O vetor que representa estas páginas é composto pelo TF-IDF (*Term Frequency – Inverse Document Frequency*) dos termos destas páginas após ter sido realizada a remoção de *stopwords* e *stemming*. O TF-IDF é uma forma de atribuir pesos aos termos de uma coleção, privilegiando termos que ocorrem várias vezes no documento e poucas vezes na coleção. Remoção de *stopwords* é uma etapa para remover termos que ocorrem freqüentemente e que, por isso, não são bons discriminadores. *Stemming* consiste em eliminar os prefixos e sufixos das palavras. Uma vez criados os vetores que representam as páginas, calculam-se os centróides para cada uma das classes fazendo a média do TF-IDF de cada termo dos vetores da respectiva classe.

Com isso, para classificar uma nova página, basta criar o vetor com o TF-IDF de seus termos e verificar qual o menor co-seno do ângulo entre o vetor da página e os vetores dos centróides. Ressalta-se que esta forma considera a dimensão dos vetores e não seu comprimento.

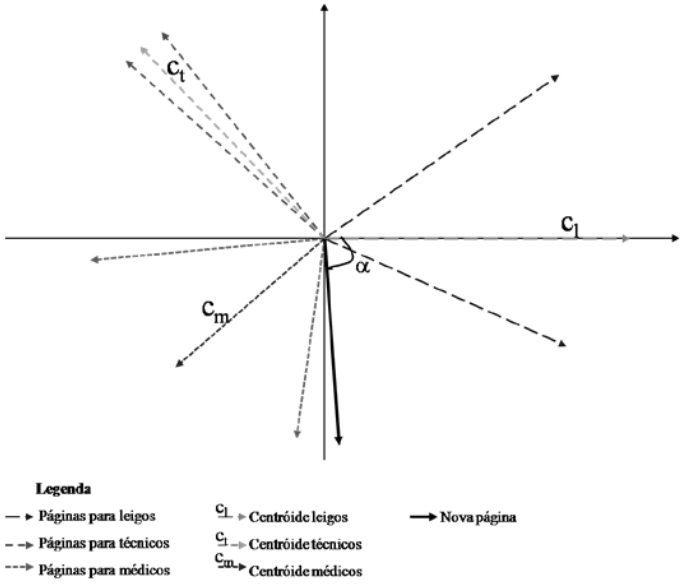

Figura 23: Exemplo de classificação de documentos baseada em centróides.

A vantagem desta forma sobre a criação de um dicionário de termos para cada um dos tipos de usuário é que é possível descobrir termos que são bons discriminadores de páginas, os quais não eram conhecidos previamente. Além disso, esta técnica permite que uma página pertença a apenas uma classe, ou seja, se destine a apenas um tipo de usuário. Ressalta-se que esta técnica não foi implementada para execução de testes.

4.7 Considerações finais sobre extração

Esta seção apresentou formas de extração de alguns atributos utilizados na ontologia desenvolvida para inferência sobre a qualidade de *sites*/páginas sobre saúde. Os atributos considerados foram: idioma, *e-mail* do autor, *inLinks*, *outLinks* e se o autor é especialista.

Para idioma, se utiliza *meta tags* ou *sites* identificadores de idioma. Para *e-mail* do autor é feita a consulta em motores de busca. Os *outLinks* são obtidos na própria página. Para extrair *inLinks* utiliza-se uma opção avançada do Google. Para determinar se autor é especialista, foi verificado se este possui publicações na PubMed ou tem *hit counts* no Google Scholar. Ainda, foi apresentada uma técnica para determinação do tipo de usuário para o qual uma página se destina, usando classificação de documentos baseado em centróides.

Ao longo do desenvolvimento da pesquisa, verificou-se que raramente as páginas seguem padrões. Por isso, para a grande maioria dos atributos, usou-se um conjunto limitado de *sites*/páginas *web* através dos quais foram extraídos os atributos dos demais *sites*/páginas, tendo assim que conhecer apenas os padrões destes *sites*/páginas e não de todos.

Portanto, foram utilizados serviços distribuídos, tais como: Google, PubMed, Google Scholar e demais *sites* de identificação de idiomas. Percebe-se que esta não é uma tarefa trivial, exigindo um grande esforço e sendo necessário definir um modelo específico para um domínio, pois um modelo genérico é extremamente complexo.

5 Considerações finais

Este relato técnico descreveu o desenvolvimento de modelos, técnicas e protótipos visando à extração automática de conteúdo apresentado em *sites*/páginas que tratam sobre saúde. Após os dados destas paginas serem localizados, padronizados e extraídos, de forma automática, objetiva-se à entrega padronizada e normalizada de dados para realização de estimativas sobre a qualidade do *site*/página.

O trabalho desenvolvido relaciona-se aos objetivos do projeto ao Projeto *SALUS – CYTED – Qualidade em Sites na Área de Saúde*, uma vez que este objetiva a discussão e criação de instrumentos para avaliação de *sites* da área de saúde.

O processo de desenvolvimento do modelo para obtenção automática de dados previu uma serie de desafios e etapas para serem vencidas, sendo o processo encaminhado de acordo com estas etapas. Para a localização de *sites*/páginas, optou-se pela utilização de

motores de busca, ao invés da criação de uma ferramenta própria para localização. Entende-se que foi a melhor alternativa, visto que motores de busca conhecidos utilizam algoritmos bastante eficazes para busca na *web*. Um fator importante com relação à utilização de motores de busca é a sua freqüente mudança de formato adotado para apresentação dos resultados aos usuários, visando fornecer um portal de busca cada vez mais interessante e eficiente. Isto se torna um problema, pois dificulta o desenvolvimento de uma ferramenta automatizada para realização de buscas automáticas nestes motores. Para o protótipo desenvolvido, são necessárias vistorias periódicas dos motores de busca utilizados, para averiguar a necessidade de atualizações no protótipo.

A busca por padrões para apresentação dos dados nos *sites*/páginas *web* levou a conclusão de que, apesar da existência de padrões para o desenvolvimento de páginas *web*, tais como XML ou Dublin Core, na prática esta padronização não é seguida a risca pela grande maioria das paginas, ou é seguida parcialmente. Após analisar as páginas e os motores de busca existentes atualmente (que não utilizam abertamente padrões como Dublin Core) constatou-se a falta de adoção de um padrão comum – tanto em nível de código-fonte quanto de apresentação do conteúdo – para paginas *web*. Desta forma, fica clara a carência de utilização de algum tipo de padronização para o desenvolvimento destas páginas.

Com relação à extração propriamente dita dos dados, existem ferramentas desenvolvidas para este fim, mas elas em geral exigem ou configuração antecipada do *site* a ser extraído ou são voltadas para um domínio específico. A extração de dados automática no domínio completo da *web* é uma tarefa realmente complexa, pois não existe ainda uma forma de se predeterminar a formatação apresentada pelos dados presentes em um domínio tão vasto.

Relativo à continuidade dos trabalhos descritos neste relatório, são colocadas abaixo algumas sugestões para trabalhos futuros:

- Com relação à **tarefa de localização** automática de *sites*/páginas sobre saúde é necessária a realização de busca e análise de novos motores de busca, de maneira a aumentar a abrangência das páginas resultantes. Além disso, pode-se melhorar o trabalho feito na etapa dos critérios de busca, principalmente junto aos usuários do sistema. Ainda com relação aos critérios de busca, é possível aumentar a complexidade das consultas utilizadas, através do uso de conectivos lógicos ou palavras técnicas utilizadas pelos especialistas.

- Com relação à **tarefa de padronização**, poucos padrões realmente significativos foram encontrados, já que cada site/página segue uma padronização própria. Desta forma, seria interessante pesquisar a padronização de bibliotecas e conjuntos de sites específicos, onde é mais provável a existência de padrões bem definidos para apresentação dos dados.

- Com relação à **tarefa de extração**, é necessário pesquisar formas de extração de atributos que geralmente são apresentados na forma de imagens (como os patrocinadores em alguns dos *sites*/páginas analisados). Além de outros campos

importantes, como data de criação, data de atualização, título, entre outros. Ainda, é necessário reunir os vários protótipos simples desenvolvidos para armazenamento de resultados em um único local (por exemplo, uma base de dados), para que este conteúdo extraído seja de fácil acesso á tarefa de determinação de qualidade.

Referências

[1] SILBERSCHATZ, A., GALVIN, P. **Sistemas Operacionais: conceitos**. São Paulo: Prentice Hall, 2000.

[2] BALAKRISHNAN, H., KAASHOEK, M. F., KARGER, D., MORRIS, R., STOICA, Ion. **Looking up data in P2P systems**. In Communications of the ACM, 2003. http://doi.acm.org/10.1145/606272.606299.

[3] LICHTNOW, D. **Um Estudo sobre Avaliação da Qualidade do Conteúdo de Sites com Recursos da Web Semântica**. Trabalho Individual I, Universidade Federal do Rio Grande do Sul, 2009, 56 pag.

[4] NEIMARK, G., HURFORD, M.O. DiGIACOMO, J. **The internet as collateral informant**. In: American Journal of Psychiatry. v. 163, n. 10, pp. 1842. 2003.

[5] MADHAVAN, J., HALEVY, A. **Crawling through HTML forms**. 11 Abr. 2008. Disponível em <http://googlewebmastercentral.blogspot.com/2008/04/crawling-through-html-forms.html>. Último acesso em 2 jul. 2009.

[6] BERGMAN, M.K**. The deep web: Surfacing hidden value**. In: Journal of Electronic Publishing. v. 7, n. 1, pp. 01-07. 2001.

[7] LOHR, S. **Google and Microsoft look to change health care**. In: New York Times. 2007. Disponível em: < http://www.nytimes.com/2007/08/14/technology/14healthnet .html?_r=1&pagewanted=2>.

[8] TIUN, S., ABDULLAH, R., KONG, T.E. **Automatic Topic Identification Using Ontology Hierarchy**. In Proceedings of the 2nd International Conference on Computational Linguistics and Intelligent Text Processing - CICLing '01. Springer-Verlag, London, UK. 2001.

[9] LIU, B., CHIN, C., NG, H. **Mining Topic-Specific Concepts and Definitions on the Web**. In The 12th International World Wide Web Conference - WWW 2003, Budapest, Hungary, May 20-24, 2003.

[10] LIMA, M. S. **Identificando o Tópico de uma Página Web**. Dissertação (Mestrado em Ciência da Computação), Programa de Pós-Graduação em Informática – PPGI/UFAM, Manaus, Amazonas. 2009. 71 f.

[11] BRIN, S., PAGE. L. **The Anatomy of a Large-Scale Hypertextual Web Search Engine**. In Computer Networks and ISDN Systems. Elsevier Science Publishers, Amsterdam, The Netherlands. 1998.

[12] GOUVEIA, F. C. **Webometria, Aplicações e Desafios**. In 1º. Encontro Brasileiro de Bibliometria e Cienciometria, MV-COC-FIORUCRUZ, Setembro 2008.

[13] CHARRAS, C., LECROQ, T. **Levenshtein Distance**. 1998. Disponível em http://www-igm.univ-mlv.fr/~lecroq/seqcomp/node2.html.

[14] HAN, E. H., KARYPIS, G. **Centroid-based document classfication: Analysis and experimental results**. In PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, pages 424–431, London, UK. Springer-Verlag. 2000.

Apêndice 1

Listagem do código-fonte para coleta automática de URLs a partir dos padrões pré-definidos dos motores de busca selecionados.

```
1. package com.collector.main;
2. import com.collector.engine.SearchEngine;
3. import java.sql.Connection;
4. import java.sql.DriverManager;
5. import java.sql.PreparedStatement;
6. import java.sql.ResultSet;
7. import java.sql.SQLException;
8. import java.sql.Statement;
9. import java.util.ArrayList;
10. import java.util.List;
11. public class URLCollector {
12.     private static String driver =
    "org.apache.derby.jdbc.EmbeddedDriver";
13.     private static String connectionURL =
    "jdbc:derby:../database/URLInformation";
14.     private static String SELECT_SEARCH_ENGINES =  "SELECT * FROM
    Engine";
15.     private static String INSERT_URLS = "INSERT INTO Url values
    (?, ?)";
16.     private Connection conn;
17.     public static void main(String[] args) {
18.         new URLCollector().run();
19.     }
20.     private void run() {
21.         this.loadDriver();
22.         this.executeEngines();
23.     }
24.     private void executeEngines() {
25.         List<String> urlList = new ArrayList<String>();
```

```
26.         try {
27.             conn = DriverManager.getConnection(connectionURL);
28.             Statement stmt = conn.createStatement();
29.             System.out.println("\nRecuperaÃ§Ã£o dos motores de
    busca que serÃ£o utilizados.");
30.             ResultSet rs =
    stmt.executeQuery(SELECT_SEARCH_ENGINES);
31.             while (rs.next()) {
    urlList.addAll(this.executeEngine(rs.getString("NAME")));
32.             }
33.         } catch (SQLException e) {
34.             e.printStackTrace();
35.         }
36.         this.saveURLs(urlList);
37.     }
38.     /**
39.      * Executa o método de busca da engine passada como
    parâmetro.
40.      * @param engineName string indicando o nome da engine a ser
    carregada.
41.      * @return lista contendo as urls recuperadas.
42.      */
43.     private List<String> executeEngine(String engineName) {
44.         try {
45.             Class c = Class.forName("com.collector.engine." +
    engineName + "SearchEngine");
46.             SearchEngine engine = (SearchEngine) c.newInstance();
47.             System.out.println("\nCriaÃ§Ã£o da consulta a ser
    submetida ao motor " + engineName);
48.             engine.buildSearchURL();
49.             System.out.println("\nInÃcio do processo de
    recuperaÃ§Ã£o de URLs no motor " + engineName);
50.             return engine.getURLs();
51.         } catch (ClassNotFoundException e) {
```

```
52.             System.out.println("Engine " + engineName + " not
    supported.");
53.         } catch (InstantiationException e) {
54.             System.out.println("Engine " + engineName + " not
    supported.");
55.         } catch (IllegalAccessException e) {
56.             System.out.println("Engine " + engineName + " not
    supported.");
57.         }
58.         return new ArrayList<String>();
59.     }
60.     /**
61.      * Salva as URLs recuperadas no banco de dados.
62.      * @param urlList
63.      */
64.     private void saveURLs(List<String> urlList) {
65.         int index = 0;
66.         try {
67.             PreparedStatement pstmt =
    conn.prepareStatement(INSERT_URLS);
68.             for (String url : urlList) {
69.                 pstmt.setInt(1, index);
70.                 pstmt.setString(2, url);
71.                 pstmt.executeUpdate();
72.                 pstmt.clearParameters();
73.                 index++;
74.             }
75.             pstmt.close();
76.         } catch (SQLException e) {
77.             e.printStackTrace();
78.         }
79.     }
80.     /**
```

```
81.      * Método que carrega o driver do banco Derby.
82.      */
83.     private void loadDriver() {
84.         try {
85.             System.out.println("\nLoad do driver para acesso ao
    banco de dados embarcado.");
86.             Class.forName(driver);
87.         } catch (ClassNotFoundException e) {
88.             e.printStackTrace();
89.         }
90.     }
91. }
```

Apêndice 2

Tabelas contendo os padrões completos extraídos dos motores de busca analisados no Capítulo 2 deste relatório. Os padrões apresentados foram coletados no período de 01 a 10 de agosto de 2009, dado o caráter dinâmico da estrutura das páginas destes motores de busca, aconselha-se a busca por atualizações dos padrões apresentados.

Tabela A.1: Padrão extraído do motor de busca Google Scholar.

Campo	(codigo) tag_inicio	(codigo) tag_fim	Observações
URLTrabalho	<p class=g><h3 class="r">	"	Para os casos onde trata se de trabalhos sem links como **[livro]** ou **[citação],** esse campo não existe. Pode ser usado apenar o padrão **<p class=g><h3 class="r">** para extrair o título do trabalho. Se o valor extraido não começar por *http* e sim por um sinal de exclamação, a *url* do trabalho original deve ser obtida da seguinte forma: **[www.schoolar.com.br+URLTrabalho]**
Titulo	;">	</h3>	
URLVersao Disponibilizada	► <b><a class=fl href="	"	Esses dois campos são apresentados em um número pequeno de resultados, mas quando o são, são apresentados os dois ou nenhum.
DominioVersao Disponibilizada	;">		
AutorFonte AnoIntituicao			Este campo está estruturado da seguinte forma **[autores - Fonte, ano InstituiçãoResponsável pela Fonte] .**
FragmentoTexto	 	<a class=fl href="/	
NumCitacoes			
ArtigosQueCitam	 <a class=fl href	">Citado por	O que esse padrão extrai deve ser adicionado ao final da url schoolar.com.br [**schoolar.com.br+ArtigosQueCitam**] para formar a URL completa para a busca dos artigos que citam o trabalho.
ArtigosRelacionados	<a class=fl href="/scholar?hl=pt-BR&lr=&q=related:	">Artigos relacionados	Padrão depende to motor estar sendo usado em português(.br). O que esse padrão extrai deve ser complementado da seguinte forma**[schoolar.com.br+/scholar?hl=pt-BR&lr=&q=related:+ArtigosRelacionados]** para se obter a url da busca pelos artigos relacionados
PesquisaWeb	<a class=fl href="http://www.google.	">Pesquisa na web	Padrão depende do motor estar sendo usado em português(.br).

	com.br/search		O que esse padrão extrai deve ser complementado da seguinte forma**[http://www.google.com.br/search+PesquisaWeb]** para se obter a url da busca pela Web
NumVersoesDoTrabalho	">Todas as	versões	
VersoesDoTrabalho	<a class=fl href="/scholar?hl=ptBR&lr=&cluster=	">Todas as	Padrão depende to motor estar sendo usado em português(.br). O que esse padrão extrai deve ser complementado da seguinte forma**[schoolar.com.br+/scholar?hl=ptBR&lr=&cluster=+VersoesDoTrabalho]** para se obter a url da busca pelos artigos relacionados

Tabela A.2: Padrão extraído do motor de busca Google.

Motor debusca	Google				
URL do motor	http://www.google.com.br/				
Termo utilizado na busca	Alzheimer				
padrão (tag_inicio) para termo buscado	<input type=text name=q size=41 maxlength=2048 value="				
padrão (tag_fim) para termo buscado	title=Pesquisar">				
URL resultante da busca	http://www.google.com.br/search?hl=pt-BR&q=Alzheimer&meta=&aq=f&oq=				
Descrição_Campo	**Conteúdo_Campo**	**(codigo) tag_inicio**	**(codigo) tag_fim**	**(texto) depois de**	**(texto) antes de**
Título do site, com a URL do resultado retornado	Alzheimer - Mal de Alzheimer - Doença de Alzheimer	<li class="g w0"><h3 class=r><a href="	" class=l onmousedown="return clk(this.href,'','res',		botões promover e remover
Titulo do site, com a URL do resultado retornado, interna ao resultado anterior	Alzheimer - Wikipédia, a enciclopédia livre	<li class="g w0" style="margin-left:3em"><h3 class=r><a href="	" class=l onmousedown="return clk(this.href,'','res',		botões promover e remover
Texto sobre conteúdo do site	Tratamento do mal de Alzheimer. Conteúdo de alto nível científico sobre mal de Alzheimer ...		 <cite>	botões promover e remover	URL do site por extenso
URL do site por extenso	www.alzheimermed.com.br/	 <cite>	- </cite>	texto sobre conteúdo do	- Em cache

				site	
Conteúdo do site armazenado em cache	Em cache - endereço interno para conteúdo armazenado em cachê	<a href="	" onmousedown="ret urn clk(this.href,'','clnk ',	URL do site por extenso	- Páginas Semelhant es
Páginas semelhantes	Páginas Semelhantes	Em cache - <a href="/	">Páginas Semelhantes -	- Em cache	- Comentar (inserir comentário)

Tabela A.3: Padrão extraído do motor de busca Medstory.

Motor debusca	MedStory				
URL do motor	http://www.medstory.com				
Termo utilizado na busca	Alzheimer				
padrão (tag_inicio) para termo buscado	<td align="left"><nobr><input maxlength="200" name="q" value="				
padrão (tag_fim) para termo buscado	type=text" size="50"></input>				
URL resultante da busca	http://www.medstory.com/app?service=external&page=Search&c=true&s=Web&tc=h1&q=Alzheimer				
Descrição_Campo	**Conteúdo_Campo**	**(codigo) tag_inicio**	**(codigo) tag_fim**	**(texto) depois de**	**(texto) antes de**
Título do site, com a URL do resultado retornado	MedlinePlus: Alzheimer's Disease	<a href="redirect?		numero. (ex. 1.)	texto sobre conteúdo do site
Texto sobre conteúdo do site	Alzheimer's Disease ... Alzheimer's disease (AD) is the most …			titulo do site	url do site por extenso
URL do site por extenso	http://www.nlm.nih.gov/medlineplus/alzheimersdisease.html			texto sobre conteúdo do site	numero. (ex. 1.) com proximo resultado

Apêndice 3

Neste apêndice estão contidos os códigos-fonte gerados durante a etapa de extração de dados, para obtenção do idioma e do *e-mail* do autor.

Abaixo é apresentada a listagem do código-fonte para extração do idioma:

```
1. public class Main {
2.  public static void main(String args[]) {
3.     ObtemPassagem op = new ObtemPassagem();
4.     String texto = "";
5.     try {texto=op.obtemPassagem("http://www.jonnybowden.
   com/2009/03/fast-food-diet-may-raise-alzheimers.html");
6.      } catch (Exception e) {
7.         System.out.println("Erro: "+e.getMessage());
8.         }
9.    Xerox x = new Xerox();
10.  try {
11.    System.out.println(texto);
12.    System.out.println(x.obtemIdioma(texto));
13.   } catch (FileNotFoundException ex) {
14.      Logger.getLogger(Main.class.getName()).log(Level.  SEVERE,
   null, ex);
15.  }
16.  } catch (JDOMException ex) {
17.     Logger.getLogger(Main.class.getName()).log(Level. SEVERE,
   null, ex);
18.  } catch (IOException ex)
19.     Logger.getLogger(Main.class.getName()).log(Level. SEVERE,
   null, ex);
20.        }
21. Fuzzums y = new Fuzzums();
22.  try {
23.    System.out.println(y.obtemIdioma(texto));
24.   } catch (FileNotFoundException ex) {
```

```
25.  Logger.getLogger(Main.class.getName()).log(Level. SEVERE, null,
   ex);
26.    } catch (JDOMException ex) {
27.   Logger.getLogger(Main.class.getName()).log(Level.
28.SEVERE, null, ex);
29.  } catch (IOException ex) {
30.Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null,
   ex);
31.   }
32.}
33.}
```

Abaixo é apresentada a listagem do código-fonte da classe responsável pela obtenção do *e-mail* do autor:

```
1.  package email;
2.  import config.Consulta;
3.  import java.io.IOException;
4.  import java.util.ArrayList;
5.  import java.util.List;
6.  import org.jdom.JDOMException;
7.
8.  public class ObtemEmails {
9.    public ObtemEmails() {
10.}
11.  public List<String> obtemEmails(String pagina) throws
   IOException, IOException, IOException {
12.     String[] sts = pagina.split(" ");
13.     List<String> emails = new ArrayList<String>();
14.        for (int i = 0; i < sts.length; i++) {
15.             if (sts[i].trim().matches("(?i)([a-z][a-z0-9._+-
   ]+@([a-z][a-z0-9-]+\\.)+[a-z]{2,4})")) {
16.                 emails.add(sts[i].trim());
17.             }
```

```
18.         }
19.    return emails;
20.    }
21. public List<String> obtemEmailsPessoa(String nmPessoa) throws
    IOException, JDOMException{
22.     List<String> emails = new ArrayList<String>();
23.     int i = 1;
24.     do {
25.         emails =
    obtemEmails(ObtemPagina.obtemPagina(Consulta.getConsultaObtemEmai
    l(nmPessoa, i)));
26.         i++;
27.    } while (i <= 2 && emails.size() == 0);
28.    return emails;
29.   }
30.}
```