UNIVERSIDADE FEDERAL DE SANTA CATARINA

CURSO DE GRADUÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GUILHERME DA SILVA BRITTO GOMES

EXTRATOR DE REGISTROS DE RESULTADOS DE PÁGINAS DE PESQUISA A FORMULÁRIOS DA WEB

FLORIANÓPOLIS, SC.

2013

GUILHERME DA SILVA BRITTO GOMES

EXTRATOR DE REGISTROS DE RESULTADOS DE PÁGINAS DE PESQUISA A FORMULÁRIOS DA WEB

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do título de Bacharel, do curso de Ciências da Computação na Universidade Federal de Santa Catarina.

Orientador: Ronaldo dos Santos Mello

FLORIANÓPOLIS, SC.

2013

EXTRATOR DE REGISTROS DE RESULTADOS DE PÁGINAS DE PESQUISA A FORMULÁRIOS DA WEB

Este trabalho de conclusão de curso foi julgado e aprovado para a obtenção do grau de **Bacharel em Ciência da Computação** no **Curso de Graduação em Ciência da Computação** da Universidade Federal de Santa Catarina.

_____

Prof. Dr. Ronaldo dos Santos Mello

Orientador

Banca Examinadora:

_____

Prof. Dr. Renato Fileto

_____

Prof. Dr. Roberto Willrich

Agradecimentos

Antes de tudo queria agradecer a todos da minha família que sempre me ajudaram muito nessa etapa, dando muito carinho e forças. À minha mãe que desde pequeno viu que eu já gostava dessa área e sempre me incentivou a fazer esse curso, seja compartilhando sua experiência de trabalho ou me inscrevendo em cursos que me ajudasse nisso. Ao meu pai, que mesmo sendo contra a idéia de eu morar longe da família, nunca me negou ajuda, o qual sem ela nunca conseguiria finalizar o curso. À minha irmã, que sempre esteve do meu lado durante a minha vida inteira e como sempre me apoiou durante todo o momento. Ao carinho recebido por minhas avós que tiveram que ficar sem seu único neto em Brasília.

Queria também agradecer a todos meus amigos sejam os de Brasília que infelizmente só os viam poucas vezes no ano, mas que mesmo assim sempre me faziam se sentir como se nunca tivesse saído de lá, e os que fiz aqui, que me ajudaram nos momentos de estudo e de festas, que tornaram minha estadia nessa ilha algo inesquecível. Em especial aos que tive a sorte de compartilhar a moradia, sendo primeiro o Renato que sempre me acompanhou nessa jornada, me ajudando muito nos estudos e, claro, na diversão. E ao João e ao Romão, que tive a sorte de encontrar no último semestre da UFSC e que tornaram esse período tão tenso, em um período tão agradável.

Gostaria de agradecer a Natália que foi a minha maior parceira durante essa estadia em Florianópolis. Ela, a minha namorada, com seu jeito carinhoso e paciente de ser que sempre me ajudou em todos os momentos de dificuldades que tive e que, com certeza, também é responsável pelos meus melhores momentos nessa ilha.

Por fim, gostaria de agradecer a todos meus professores que ajudaram nessa jornada acadêmica, compartilhando seus conhecimentos e experiências, principalmente ao meu orientador Prof. Dr. Ronaldo Mello pelos conselhos e orientações nesse trabalho, ao Augusto de Souza que me ajudou muito neste trabalho sempre dando sugestões e tirando dúvidas nas nossas reuniões, e aos membros da banca Prof Dr. Roberto Willrich e Prof. Dr. Renato Fileto pelas suas sugestões e por terem aceitado meu convite.

RESUMO

Como grande parte do conteúdo da Web se encontra escondido na Deep Web em bancos de dados cujos dados são importantes para consumo humano, se fazem necessárias soluções que tornem tais dados disponíveis (*"to surface the Deep Web"*). Para tanto, é preciso inicialmente descobrir Web sites que sejam "portas de acesso" a bancos de dados da Deep Web através de sistemas especializados, como *focused crawlers* e *meta searchers* Web. Porém, após a descoberta desses bancos de dados, a extração dos seus dados, presentes em páginas de resultado de consultas é também necessária e é uma tarefa complexa, uma vez que as páginas são em sua maioria fracamente estruturadas e repletas de informações não relevantes. Esta problemática, associada ao fato de que páginas de resultado são muito heterogêneas e dificultam a localização de informações relevantes faz com que as soluções propostas por trabalhos relacionados tenham limitações. Assim sendo, este trabalho propõe um novo algoritmo de extração que apresenta um melhor desempenho em relação aos principais trabalhos relacionados. Para isso ele busca combinar as abordagens existentes além de incluir uma base de conhecimento para o auxilio na definição do que é relevante e o que não relevante a ser extraído. O desempenho do algoritmo é avaliado através de experimentos.

Palavras Chave: *Deep Web, Extração, Formulário.*

ABSTRACT

As a great part of relevantcontent for human consumption on the Web is found hidden in Deep Web databases, solutions are necessary to became these data accessible ("to surface the Deep Web"). In order to achieve that, it is necessary to first discover Web sites that are "gateways" to databases on the Deep Web by using specific systems, like *focused crawlers* and *meta searchers*. After such a discovery, these data must be extracted from the pages returned from the hidden database search. This is a complex work because in most of the cases, data is weakly etructured and pages have a lot of not relevant information. This problem, associated to the fact that the returned Web pages are very heterogeneous, makes the proposed solutions limited. In order to deal more efficiently with this problem, this work proposes a new extraction algorithm which has a better performance compared to the main related work. To achieve that the extraction algorithm combines differents approaches and includes a knowlegdement base to help decided which data is relevant for extraction from the other thar are not. The performance of this algorithm is evaluated through some experiments.

Keywords: *Deep Web, Extraction, Web Forms.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

BC – Base de Conhecimento.

BD – Banco de Dados.

DOM - DocumentObjectModel.

HTML - HyperTextMarkupLanguage.

MDR - Mining Data Records.

AMBER - Adaptable Model-based Extraction of Result Pages.

XML - eXtensible Markup Language.

1. INTRODUÇÃO

1.1 Visão Geral

Uma grande motivação para a extração de dados da Web é a grande quantidade de informação contida nela. Como se sabe, atualmente a informação é algo bastante apreciado por todos, seja para sua empresa conseguir estar à frente dos seus rivais ou para um indivíduo qualquer que necessita de um conhecimento específico, como ofertas de emprego ou hotéis em uma cidade. Considerando que a maior porção disponível de dados na Web está escondida na chamada *Deep Web* (Halevy et al. 2009), é de interesse de todos que esses dados venham á tona para ser acessados com mais facilidade pelos usuários.

Grande parte da Deep Web compreende dados presentes em bancos de dados relacionais que não podem ser alcançados com uma simples busca na Web através de uma máquina de busca. Isto por quê esses dados estão "escondidos" e se tornam visíveis somente em páginas HTML dinamicamente geradas como resultado da submissão de uma consulta através do preenchimento de um formulário Web. Mesmo utilizando tecnologias para a descoberta e preenchimento de formulários Web, é necessária, posteriormente, a aplicação de técnicas de extração dos dados exibidos nas páginas, visando facilitar o acesso dos usuários a essas fontes de dados.

Essas páginas HTML, retornadas de uma consulta a um banco de dados escondido, são exibidas na maioria das vezes com diversas propagandas, menus e outros dados irrelevantes, além dos registros com dados de interesse advindos dos bancos de dados escondidos Web, como por exemplo, registros de veículos de um site de locação ou de venda de veículos. Assim sendo, a proposta desse trabalho é, dada uma página HTML com o resultado de uma consulta, aplicar um algoritmo que desconsidere dados irrelevantes e extraia apenas dados de interesse, ou seja registros do banco de dados escondido.

Devido às diferentes formatações e padrões utilizados para se criar uma página HTML, a extração somente de conteúdos relevantes não é uma tarefa simples. Os algoritmos mais utilizados atualmente para essa finalidade, o *Road Runner (*Crescenzi et al, 2001) e o *MDR* (Liu, B., Grossman, R., Zhai, Y., 2003), não possuem um alto desempenho. Desta forma, este trabalho se propõe

também a desenvolver um algoritmo com melhor desempenho, em termos de qualidade e velocidade da extração, que esses algoritmos.

Esse trabalho é parte integrante de uma dissertação de mestrado intitulada ”Extração e catalogação de informações sobre banco de dados escondidos na Web”, em desenvolvimento no PPGCC/UFSC. Esta dissertação propõe uma solução para realizar a extração e catalogação de dados relevantes em bancos de dados na Deep Web presentes em uma página HTML gerada através de uma consulta sobre formulários Web. A finalidade é obter o conhecimento sobre estes bancos de dados e, conseqüentemente, permitir buscas estruturadas e com melhor qualidade sobre estes dados.

1.2 Objetivos

Esse trabalho tem como objetivo o desenvolvimento de um algoritmo de extração de resultados de consulta a formulários Web exibidos em páginas HTML, ou seja, extração de registros estruturados recuperados dos bancos de dados escondidos. Almeja-se um algoritmo que seja mais eficiente que os algoritmos clássicos Road Runner e MDR para extração de registros de páginas HTML.

1.3 Metodologia

Para o desenvolvimento deste trabalho foi realizado um levantamento bibliográfico para verificar quais métodos de extração estão sendo utilizados atualmente, de forma a coletar subsídios para o desenvolvimento da solução proposta. Experimentos iniciais mostraram que o algoritmo proposto teve um desempenho melhor do que os existentes. Como estratégia para esta melhoria de desempenho em termos de qualidade do resultado, o algoritmo proposto utiliza uma base de conhecimento para achar um provável conteúdo relevante na pagina HTML e conta com o auxilio de heurísticas que auxiliam a decidir se o conteúdo encontrado é relevante ou não.

1.4 Estrutura do Trabalho

Os demais capítulos desse trabalho são os seguintes:

1. Fundamentação Teórica: Apresentação dos conceitos da Deep Web, extração de dados e introdução à abordagem DeepEC;

2. Trabalhos Relacionados: Apresentação dos principais trabalhos relacionados disponíveis atualmente: MDR, Road Runner e AMBER;

3. Extrator Proposto: Detalhamento do algoritmo de extração proposto, incluindo suas heurísticas e a base de conhecimento utilizada;

4. Experimentos: Demonstração dos experimentos realizados utilizando o algoritmo proposto, explicando a origem dos dados, as métricas de avaliação utilizadas e os resultados;

5. Conclusão: Conclusão do trabalho realizado, mostrando as vantagens trazidas com o desenvolvimento do novo extrator, bem como sugestões de trabalhos futuros.

2. FUNDAMENTAÇÃO TÉORICA

2.1 Deep Web

A *Surface Web* corresponde aos dados que são indexados por motores de buscas como Google[1], Bing[2] e Yahoo[3]. Ela se refere a dados presentes em sites referenciados pelos buscadores, que são acessíveis comumente. A *Deep Web*, por outro lado, corresponde à porção de dados da Internet que são de difícil acesso, ou seja, não são acessíveis pelos buscadores comuns.

Comparando a Web com um iceberg (Figura 1) os dados que estão visíveis a qualquer um correspondem a ponta do iceberg, sendo o restante a Deep Web. De acordo com um estudo realizado pela Bright Planet (Berkman, 2011), a Deep Web possui cerca de 7,5 mil terabytes de informação, tendo a *surface Web* ou *visible Web* somente 19 terabytes. É de se imaginar que o crescimento da Deep Web aumente também a necessidade pelo acesso a esses dados por parte dos usuários.

O problema neste contexto é que esses dados escondidos são acessados de forma dinâmica, ou seja, páginas HTML com formulários para consulta precisam receber uma entrada a ser processada para que os dados possam ser buscados no banco de dados e retornados em uma nova página HTML. Este acesso é diferente das páginas estáticas da *Surface Web*, que sempre estão disponíveis e podem ser acessadas através de índices presentes nos motores de busca.

[1] http://www.google.com
[2] http://www.bing.com
[3] http://www.yahoo.com

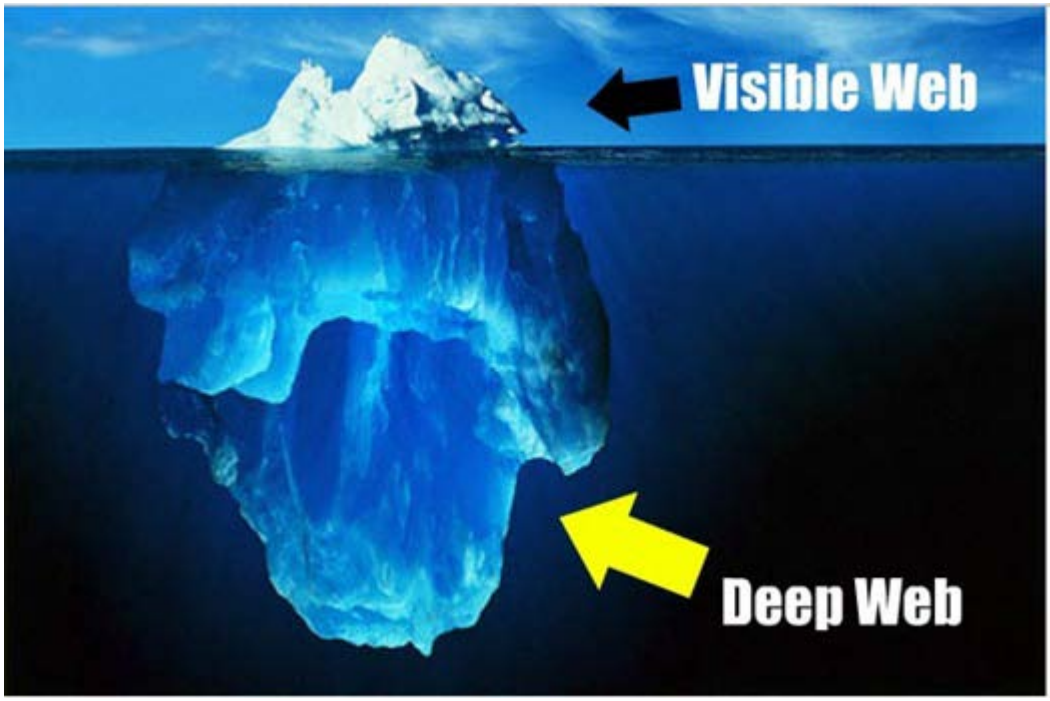

Figura 1. Correlação da Deep Web com um Iceberg (disponível em: http://www.revista.espiritolivre.org/cibercrime-clandestinidade-e-deep-web)

Conforme explicado, para o acesso a bancos de dados escondidos na Deep Web, é preciso preencher formulários (*Web forms*), que tem como função definir filtros de busca. Um *Web form* possui diversos campos a ser preenchidos como *check boxes* e caixas de texto, conforme mostra a Figura 2.

Figura 2. Exemplo de Web Form (disponível em: http://www.icarros.com.br)

Esses *Web forms* podem ser preenchidos de forma automática por *crawlers* específicos, que após determinar o tipo de formulário, o preenche e submete a pesquisa. Uma vez preenchidos os dados de filtro, eles são enviados ao servidor do banco de dados para serem processados. O servidor associa o que você selecionou ou escreveu com atributos de tabelas do banco de dados. Por exemplo, se você escreveu no campo “Modelo do carro:” o valor “Ford”, ele vai buscar dados de carros que tenham esse modelo, conforme exemplifica a Figura 3.

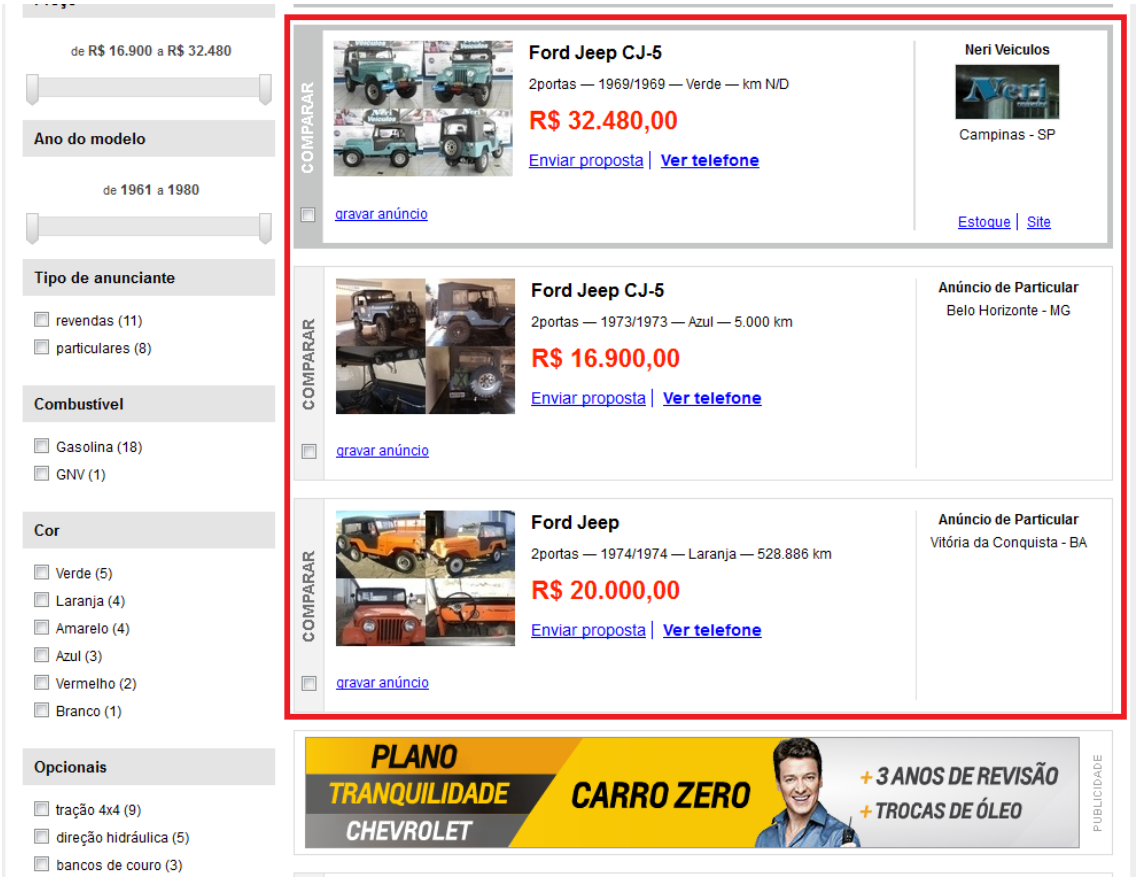

Figura 3. Resultado de uma pesquisa a um banco de dados escondido sobre veículos.

Por conter a maior parte dos dados da Internet, a Deep Web se torna interessante para consumo humano. Com esta necessidade de explorar esse universo de difícil acesso, existe a necessidade de desenvolvimento de ferramentas para extrair dados da Deep Web. Porém, como se pode ver na Figura 3, parte da informação retornada não é relevante, como a parte fora do quadrado vermelho. O algoritmo proposto neste trabalho se preocupa com este problema. Ele extrai apenas os dados considerados relevantes, neste caso, os

dados que estão dentro do quadrado vermelho e que representam registros de veículos.

2.2 Extração de Dados e a Abordagem DeepEc

A extração de dados da Web consiste em coletar dados considerados relevantes, pelo extrator, geralmente de páginas HTML. Como no caso da Deep Web esses dados estão escondidos, é preciso antes da etapa de extração preencher os formulários para que se possa extrair os dados do resultado dessa pesquisa.

As páginas que exibem resultados de buscas a bancos de dados escondidos vêm repleto de informações inúteis como anúncios, menus, etc. Por isso, é importante a utilização de técnicas de extração de dados, de modo que o usuário receba só aquilo que procura. Existem dois algoritmos populares na literatura que realizam tal extração de dados: o Road Runner e o MDR. Além destes, existe a abordagem *DeepEc* que propõe uma solução para a extração e catalogação desses dados da Deep Web. Essa abordagem foi desenvolvida como uma dissertação de mestrado no Grupo de Banco de Dados da UFSC (GBD/UFSC) (De Souza e Mello, 2012). Ela tem sua arquitetura mostrada na Figura 4.

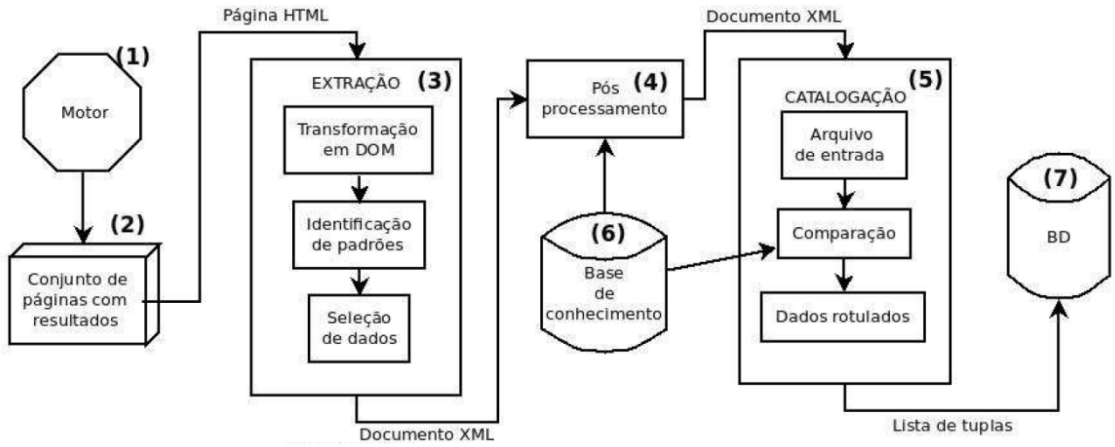

Figura 4. Arquitetura da DeepEc

O processo executado pela DeepEC inicia pelo componente externo Motor (1), que encapsula o processo de descoberta de bancos de dados escondidos na Web e, após a descoberta, o preenchimento de formulários Web, a submissão de consultas e a geração de páginas HTML com os resultados obtidos. A abordagem recebe, então, um conjunto de páginas (2) que apresentam dados de um banco de dados escondido oriundos de uma pesquisa formulada através de Web forms.

Cada página deste conjunto serve de entrada para o módulo de extração (3). Este trabalho de conclusão de curso contribui com esta dissertação através do desenvolvimento de um algoritmo de extração que compreende este módulo. O algoritmo analisar a página de modo a retornar, no formato XML, somente os dados relevantes. Este algoritmo é detalhado no Capítulo 4.

Na seqüência, a etapa de pós-processamento (4) analisa os arquivos gerados pelo módulo de extração. Caso seja necessário, pode-se, através da utilização de uma base de conhecimento (conforme explicado a seguir) (6), remover possíveis dados que não são úteis no domínio do banco de dados em questão, como por exemplo, nomes de menus.

A catalogação dos dados (5) aplica um algoritmo baseado na análise e caracterização de segmentos de texto (se são rótulos ou se são valores), utilizando como apoio também a base de conhecimento (6) que inclui amostras de dados nos domínios dos bancos de dados escondidos cujos dados são extraídos.

Para finalizar o processo, os registros de dados, representados pelos rótulos e seus valores são armazenados em um banco de dados relacional (7). Maiores detalhes sobre o processo de catalogação estão fora do escopo deste trabalho.

3. TRABALHOS RELACIONADOS

Este capítulo descreve os dois principais algoritmos de extração de registros de dados de páginas na Web. Estes algoritmos serviram de base para este trabalho e são comparados com ele. Além deles, um terceiro trabalho relacionado também é apresentado, porém, não é comparado por ter uma qualidade baixa em termos de recuperação de dados relevantes.

3.1 Road Runner

O Road Runner (Crescenzi, Mecca e Merialdo, 2001) recebe várias páginas HTML como entrada e cria uma expressão regular a partir delas, comparando o código HTML das páginas para encontrar as similaridades e diferenças entre elas. Quando ele encontra registros similares, ele simplesmente anota na expressão regular. Já quando são diferentes, ele separa as diferenças (quando um atributo não se assemelha ao outro) em dois tipos: o de *string* e o de *tag*. No caso da string, ele é detectado quando duas páginas que estão na mesma classe possuem valores diferentes no campo de dados. Quando isso ocorre, ele coloca #PCDATA no lugar da string e interpreta isso como informação adicional, pois ela não será usada pra gerar a expressão, como é mostrado na Figura 5. No caso da diferença de tag, ele define se é um item opcional da página HTML ou não baseado na quantidade de padrões repetidos. Na figura 5 podemos vemos a diferença de tags nas linhas 06 e 19 e as diferenças de string nas linhas 04, 11 e 17

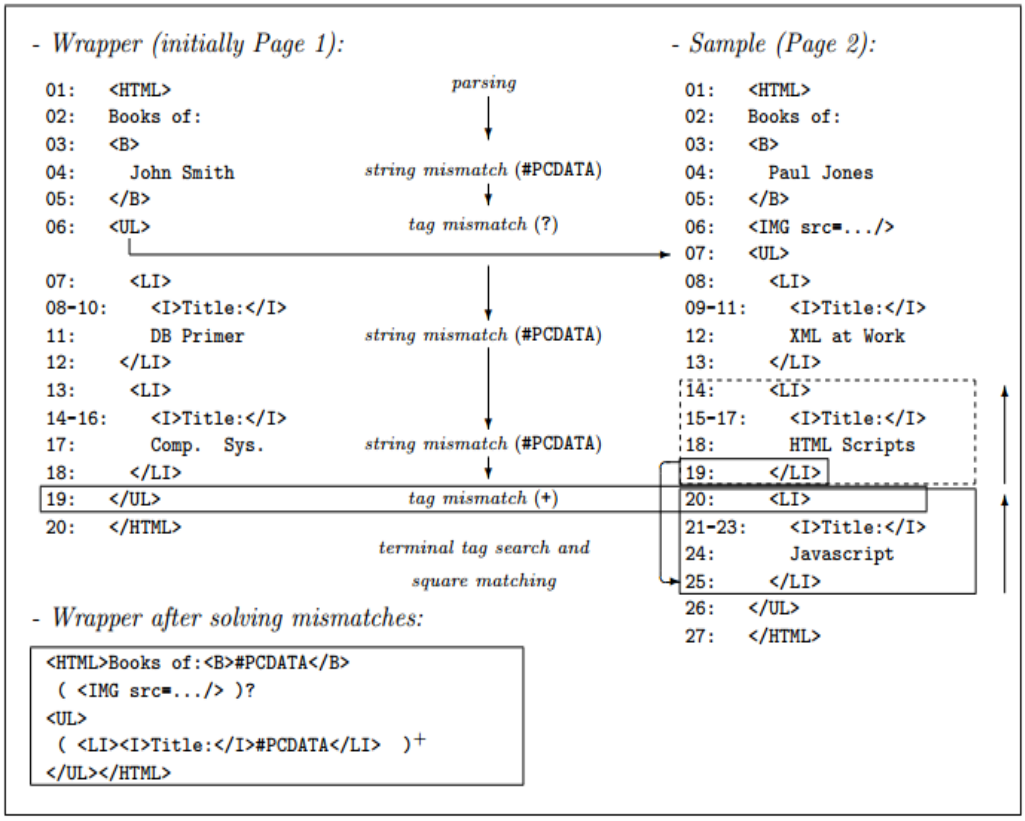

Figura 5. Função de Similaridade

A cada nova interação, essa expressão regular é refeita com base na nova página analisada. Após isso, ele usa essa expressão pra extrair informações úteis em outras páginas do conjunto. O resultado disso é uma página tabelada com os registros e o tempo de execução, como mostra a Figura 6. O problema deste algoritmo é que nem sempre se consegue construir uma expressão regular genérica, o que prejudica a qualidade do resultado.

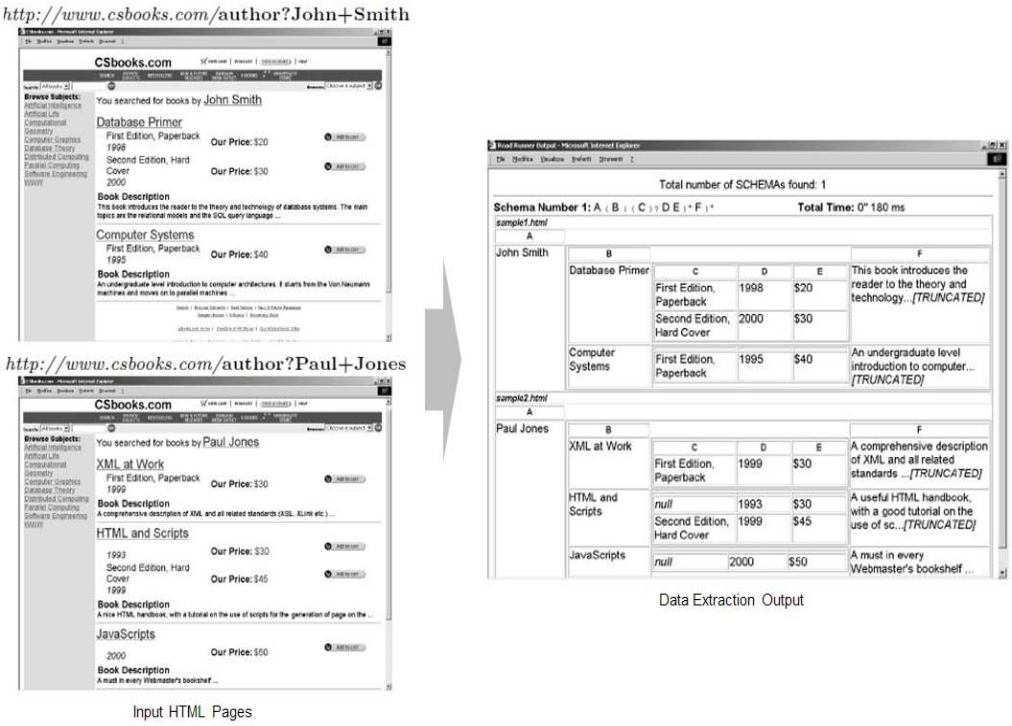

Figura 6. Resultado do Road Runner

3.2 MDR

O algoritmo MDR (Liu, Grossman e Zhai, 2003) também extrai registros de dados de uma página HTML e inicia criando uma árvore DOM (W3C DOM, 2013) da página HTML. Uma árvore DOM utiliza as tags de uma página HTML para criar seus nodos. Um nodo é filho de outro se a tag de um estiver dentro da tag do outro. A Figura 7 ilustra essa árvore sendo o nodo HTML a raiz dela.

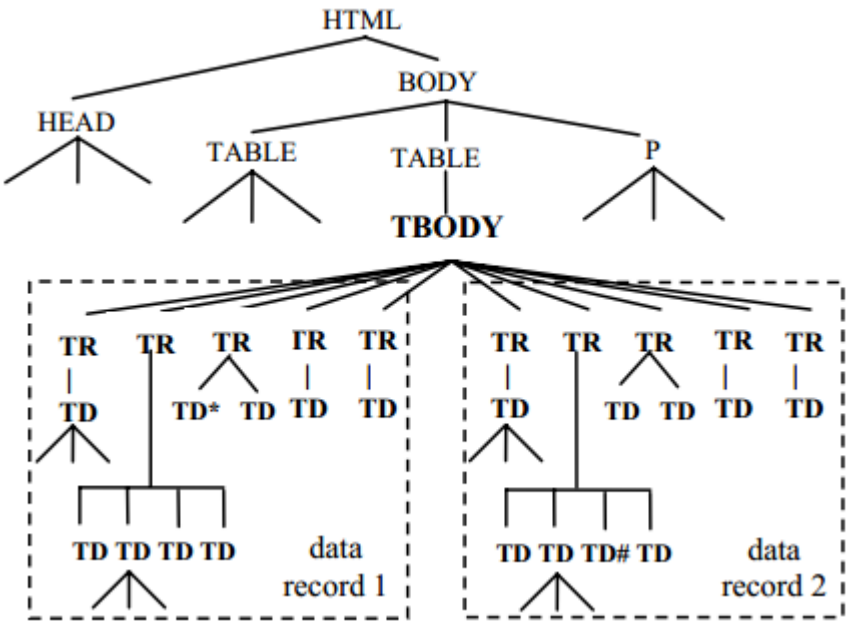

Figura 7. Árvore DOM

O algoritmo MDR percorre essa árvore de cima para baixo, verificando a similaridade dos nodos adjacentes. Para isso, ele usa o algoritmo STM *(Simple Tree Matching)* e a métrica *Edit Distance Function* para comparar as *strings*. Após isso, ele coloca os nodos similares em um conjunto, sendo esse conjunto chamado de região de dado, como ilustrado na Figura 8. Os nodos dessa região devem ter o mesmo nodo pai, ter a mesma quantidade de filhos, devem ser todos adjacentes e ter o resultado da *Edit Distance Function* menor que o limiar escolhido pelo usuário.

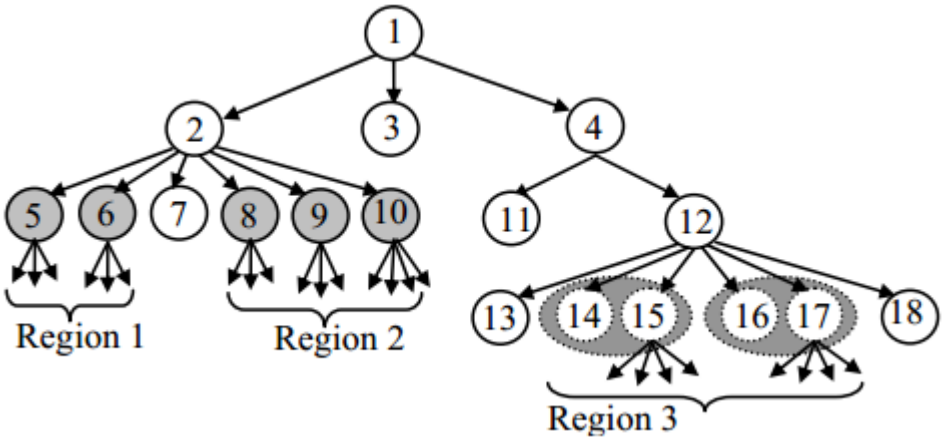

Figura 8. Árvore com Região de Dados

Por fim, ele procura campos de dados em cada um desses conjuntos para extrair somente o conteúdo que interessa, baseando-se na restrição que

se um nodo contém dois ou mais registros de dados, esses registros devem ser similares em termos da string da tag. A principal limitação deste algoritmo é que ele pode detectar diversos padrões, sendo difícil saber qual padrão é o correto, além de não conseguir coletar dados aninhados.

3.3 AMBER

O sistema AMBER (Furche et al.,2011), diferentemente dos algoritmos anteriores, utiliza uma ontologia de domínio. Ele analisa a página HTML em 3 fases, como mostra a Figura 9. Na primeira fase, ele representa a página em DOM. Na fase seguinte, essa modelagem é utilizada para derivar o modelo de atributo que contém os potenciais atributos de referência para os registros a serem extraídos. Na terceira fase, ele coleta a página e o modelo de atributo para achar as áreas onde o registro a ser extraído possa estar. Ele realiza todo este processamento de acordo com domínio da página a ser analisada.

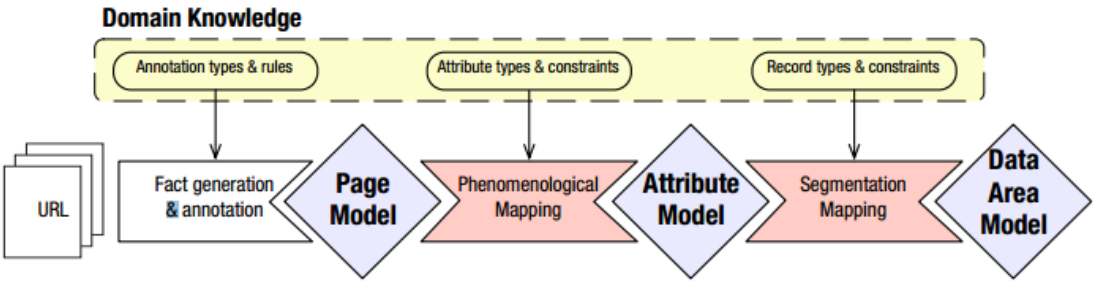

Figura 9. Arquitetura AMBER (Furche et al.,2011)

Esse sistema, no entanto, só foi testado para o domínio imobiliário do Reino Unido, apesar de afirmar que pode ser aplicado a qualquer domínio. Além disso, ele se preocupa apenas em extrair dados sem levar em conta a quantidade de dados considerados não relevantes.

4. EXTRATOR PROPOSTO

Este capítulo descreve o extrator de registros de resultados de consultas provenientes de bancos de dados escondidos e apresentados em páginas HTML desenvolvido neste trabalho.

Inicialmente, procurou-se desenvolver um extrator baseado no proposto por (Lan Yi, Bing Liu e Xiaoli Li, 2003), que busca, através da análise de uma série de páginas, uma estrutura recorrente na qual provavelmente estariam os dados relevantes. Porém, tal abordagem se mostrou complexa, pois ela utiliza diversos métodos para determinar a relevância de um atributo da página HTML baseados somente em probabilidades e o cálculo dessas probabilidades não se mostrou claro. Outro fator negativo foi a necessidade de diversas páginas para determinar o que eliminar. Após diversas pesquisas, foi visto que atualmente há alguns trabalhos que utilizam ontologias ou bases de conhecimento como forma de extração de dados, sendo o AMBER um dos mais recentes. Com isso, buscou-se combinar as duas abordagens para a solução proposta neste trabalho: a do MDR, que utiliza a estrutura da página e a AMBER, que utiliza uma ontologia de domínio.

O extrator foi implementado em Java usando como apoio a biblioteca JTIDY, responsável por transformar a página HTML em uma estrutura de árvore DOM.

A base da técnica proposta são um conjunto de heurísticas e o suporte de uma base de conhecimento. Eles são descritos a seguir.

4.1 Heurísticas

O algoritmo do extrator proposto utiliza algumas heurísticas, dessas heurísticas somente a primeira não foi criada na implementação deste trabalho. Elas são as seguintes:

1. HER (Heurística da Estrutura Relevante): a estrutura que mais se repetir na página HTML provavelmente é a estrutura onde estão os registros relevantes, ou seja, os resultados das consultas do banco de dados;
2. HEI (Heurística da Estrutura Irrelevante): a estrutura HTML que tiver em seu caminho a tag “script” ou as tags “select” e “option” devem

ser descartadas, pois provavelmente indicam uma função sem dado relevante ou um campo de formulário sem dado relevante, respectivamente.

3. HAM (Heurística do Atributo Mandatório): um atributo mandatório é um atributo significativo, ou seja, uma propriedade especifica de um determinado domínio, servindo para caracterizar, de forma mais evidente, este domínio.

Essas heurísticas foram definidas a partir da análise de resultados de experimentos preliminares aplicados sobre uma amostra de páginas HTML

A heurística HER é a mais importante para o bom funcionamento do extrator, pois é ela que decide o que é considerado relevante e o que não é. Ela foi criada com base em trabalhos já propostos como o de (Hong, 2010), que afirma que a estrutura com mais repetição tem a maior probabilidade de ser a estrutura buscada. Mesmo assim, o diferencial deste trabalho é o auxilio de uma base de conhecimento para confirmar os valores relevantes. Com a consideração da heurística HEI, o extrator conseguiu desconsiderar dados irrelevantes, ou seja, ele deixou de passar por sub-estruturas HTML que não trariam dados úteis.

A heurística HAM permite uma otimização no algoritmo proposto, pois dados os atributos mandatórios do domínio em questão, o algoritmo só utiliza os mesmos para comparar com os termos do registro analisado, visando confirmar se o registro é relevante ou não. Esta definição de atributos mandatórios é feita de maneira manual na própria base de conhecimento. Um exemplo de atributo mandatório é o atributo *make* que é um atributo bastante relevante no domínio de Veículos, pois é uma propriedade inerente a qualquer veiculo.

4.2 Base de Conhecimento

Base de Conhecimento (BC) é um repositório de informação ou conhecimento sobre um determinado assunto ou domínio. Uma base de conhecimento costuma ser bastante usada em sistemas especialistas, que tem a função de simular o raciocínio de um profissional sobre um determinado

assunto especifico, bem como em *help desks* que buscam resolver problemas dos usuários.

Neste trabalho, uma base de conhecimento foi projetada com o intuito de ajudar o algoritmo de extração proposto a escolher as estruturas relevantes com dados pertencentes ao domínio em questão. Essa base de conhecimento é a mesma utilizada pela abordagem DeepEC. O ciclo de vida da metodologia utilizada na construção da base de conhecimento para este trabalho pode ser visto na Figura 10 abaixo. A metodologia usada na construção da BC foi uma adaptação do modelo de cinco fases do processo de criação do conhecimento apresentado por (Nonaka e Takeuchi, 1997) onde primeiro ocorre a aquisição do conhecimento, após é feito a modelagem do conhecimento adquirido, a implementação (onde é construída a base de conhecimento em .XML) e, por último o refinamento caso seja necessário.

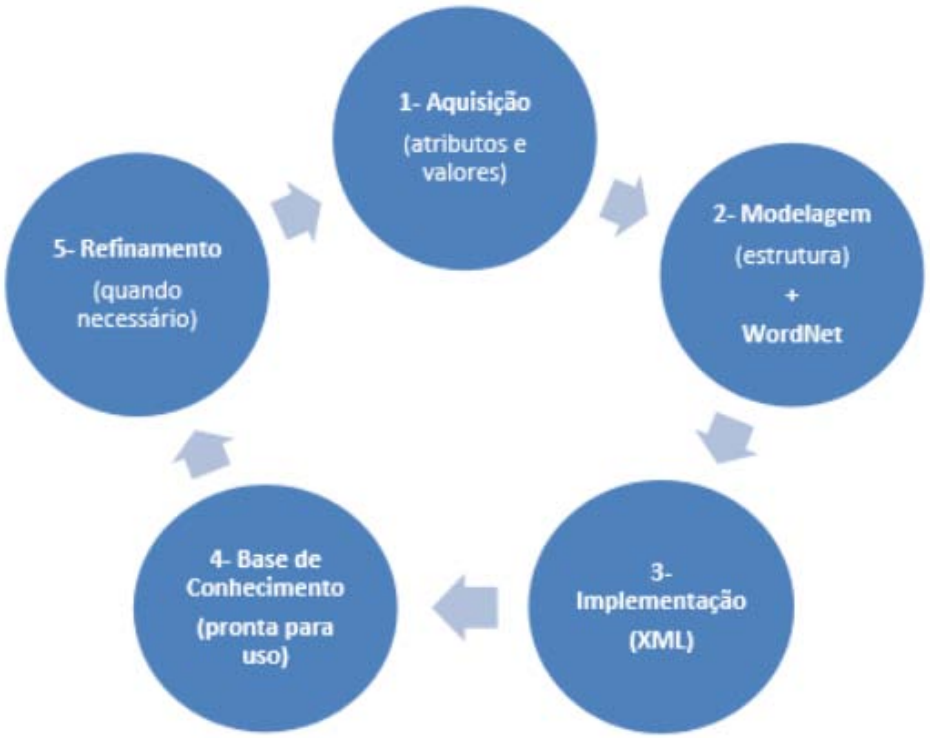

Figura 10. Ciclo de vida da BC desenvolvida.

A fase de Aquisição foi feita acessando alguns sites da Web relativos a aos domínios escolhidos neste trabalho para os experimentos (Veículos e

Livros) e extraindo amostras de rótulos de atributos em formulários e seus valores associados de forma manual.

Na segunda fase, a da modelagem, foi-se utilizada o conteúdo disponibilizado pela *Freebase* para popular a base de conhecimento. A *Freebase* é um repositório colaborativo de dados estruturados que está disponível para pesquisa. Ela é referenciada em alguns trabalhos ((Zheng et al., 2012) (Serra et al., 2011)). A estrutura da base de conhecimento foi definida e construída hierarquicamente no formato XML. O formato XML foi escolhido devido à tecnologia disponível para a manipulação de dados neste formato.

Na raiz dessa hierarquia se encontra a divisão em domínios e, para o domínio de Veículos, considerou-se os atributos mais frequentes encontrados em formulários Web, que são: “make”, “model”, “year”, “color”, “door”, “mileage” e “price”. Para o domínio de Livros, foram considerados os seguintes atributos: “title”, “author”, “price”, “isbn”, “year” e “publisher” como se pode ver na Figura 11. Além disso, o esquema da BC considera ainda dependências de valor presentes entre atributos e que são muito comuns em formulários Web. Um exemplo é um valor de atributo *“make”*, que determina um ou mais valores de um atributo *“model”*. Se o valor de “make” for Ford por exemplo, os valores de “model” ficará restrito somente aos modelos fabricados pela Ford. Monstrando assim uma dependência entre “make” e “model”.

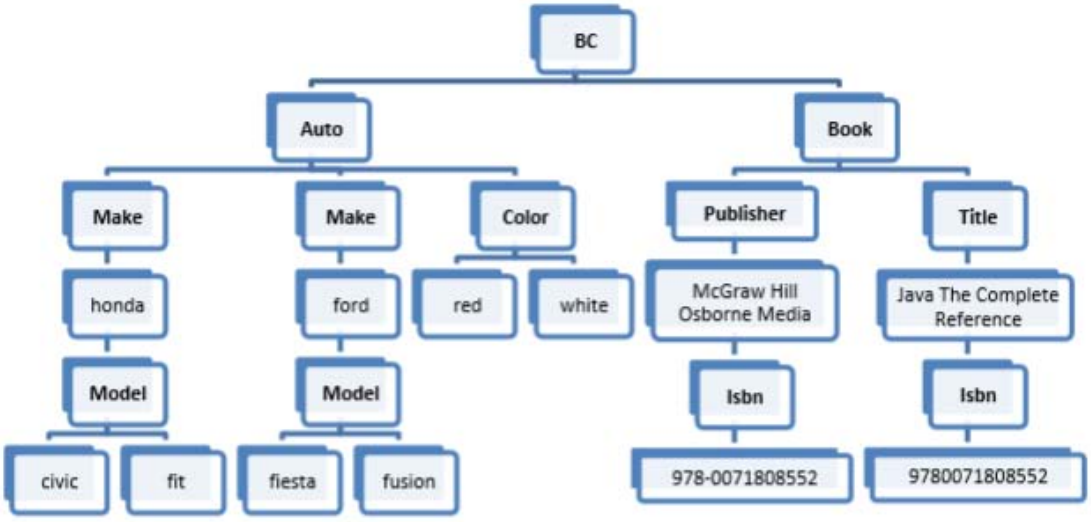

Figura 11. Esquema da BC.

A etapa de modelagem leva em conta ainda o enriquecimento semântico da BC através do suporte do *WordNet* (Miller, 1995). O WordNet é um banco de dados lexical da língua inglesa onde substantivos, verbos, adjetivos e advérbios são agrupados em conjuntos de sinônimos cognitivos (synsets), cada um expressando um conceito distinto.

Essa integração foi realizada de forma automática, utilizando-se uma API do WordNet para a linguagem Java chamada Java Wordnet Interface(JWI), comparando os conceitos já existentes na BC com os da base de dados do WordNet. Os novos termos reconhecidos pelo WordNet foram adicionados à BC. Um exemplo são os termos “brand” e “make”, que são classificados com baixo grau de similaridade pelos algoritmos de similaridade de texto, porém, nesse contexto, possuem o mesmo significado.

A última fase, de refinamento, só e realizada quando se é necessário adicionar ou atualizar informações à BC previamente criada, refazendo todas as etapas para isso.

4.3 Processo de Extração

O processo seguido pelo extrator proposto é sumarizado no algoritmo mostrado na Figura 12. O extrator recebe como entrada a base de conhecimento no formato .xml e uma página HTML. Inicialmente, ele gera uma árvore DOM da página HTML como mostrado na linha 6 do algoritmo, utilizando um *parser*. Após, ele busca nos nodos que não contêm tags irrelevantes (identificadas pela heurística HEI (linha 7)), atributos mandatórios de um determinado domínio com a ajuda da BC (linha 8), identificando os nodos que contêm atributos mandatórios (linha 10). A informação de quais são os atributos mandatórios fica na BC, como se pode ver na Figura 14, que ilustra um fragmento da BC codificada em XML. A identificação dos nodos que contêm atributos mandatórios (*matching*) é feita comparando um valor de atributo mandatório contido na BC com o conteúdo do nodo candidato na árvore DOM. Se no nodo da DOM está o valor, por exemplo, “Ford” (valor de atributo mandatório(“*model*”) para carro) ele irá fazer o *matching* com o mesmo valor encontrado na BC. Para uma comparação com melhor desempenho, a BC é separada por domínios, de forma que cada domínio possui seus próprios

atributos mandatórios, não sendo necessário assim varrer toda a base para achar os atributos mandatórios.

Com o nodo já identificado, ele guarda o caminho percorrido, desde a raiz da árvore até o nodo, em uma lista como mostrado na linha 11 do algoritmo. Na sequência, utilizando-se da heurística HER (heurística que assume que a estrutura que mais se repete provavelmente é onde está o resultado buscado), ele compara os caminhos que estão guardados na lista, em busca do caminho que apareça mais vezes. Por fim, após identificar esses caminhos, ele percorre de novo a árvore HTML buscando padrões de caminho semelhantes e realizando a extração dos dados presentes nestes caminhos (linhas 16 e 17).

Algoritmo 1: Método de Extração.

```
1:  Entrada: Base de Conhecimento BC;
2:  Entrada: página HTML;
3:  Início
4:  CaminhosCandidatos ← ( )
5:  CaminhosRelevantes ← ( )
6:  PDOM ← DOM Parser (pagina)
7:  para cada termo t de um nodo folha n ∈ PDOM ∉ HEI faça
8:   para cada valor de HAM v ∈ BC faça
9:     para cada termo t de um nodo folha n ∈ PDOM faça
10:       se matching (v, t) então
11:          CaminhosCandidatos ← CaminhosCandidatos + Caminho (n)
12:       fim se
13:    fim para
14: fim para
15: CaminhosRelevantes ← HER (CaminhosCandidatos)
16: para ∀ termo t de um nodo folha n ∈ CaminhosRelevantes faça
17:       Extrai(t)
18: fim para
19: Fim
```

Figura 12. Algoritmo Proposto

Um exemplo parcial de página HTML representada no formato de árvore DOM é mostrado na Figura 13. Ele mostra os dados no domínio de veículos. O algoritmo de extração, com o suporte da base de conhecimento exemplificada na Figura 14, identifica inicialmente a palavra *civic (modelo de carro), Honda* (marca de carro), 4 *portas* ou mp3 (atributos de um carro). Na sequência, ele guarda o caminho percorrido, no caso, *html->body->tr->td (no caso de honda,* um *civic achado, mp3 e 4 portas*) e *html->body* (no caso do outro *civic achado*).

Após, ele conta quantos registros possuem o mesmo caminho. Nesse caso, o primeiro caminho tem mais registros que o segundo e, portanto, provavelmente é o caminho para os registros relevantes. Assim sendo, ao final ele percorre novamente a árvore coletando os dados que contém o caminho *html->body->tr->td*.

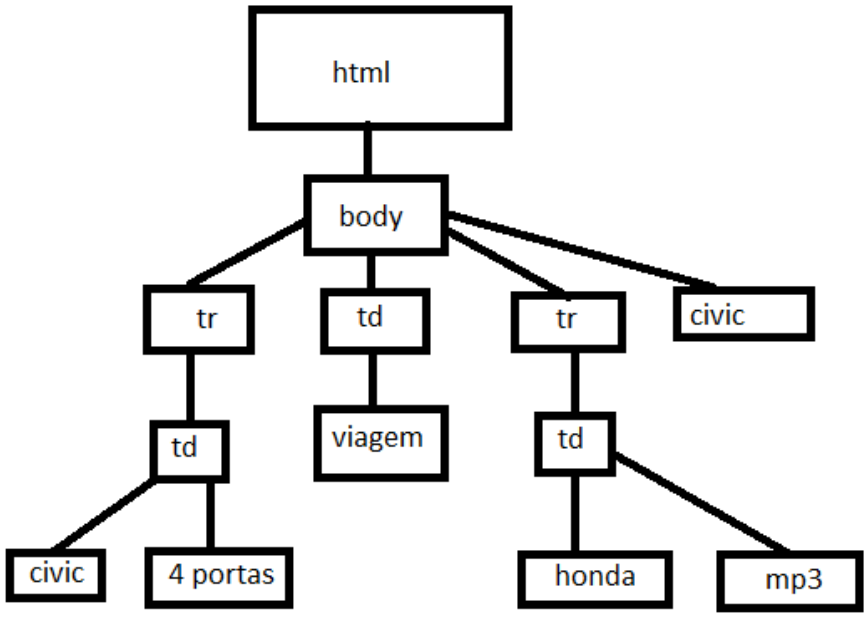

Figura 13. Exemplo de Árvore DOM

```
<?xml version="1.0"?>
- <KB>
  - <domain>
    - <auto>
        <mandatoryCar value="make,model"/>
      - <make value="HONDA">
          <model value="CIVIC,FIT,CRV,ACCORD"/>
        </make>
      - <make value="FORD">
          <model value="ESCORT,FIESTA,FUSION"/>
        </make>
        <color value="RED,WHITE,BLUE,BLACK,SILVER"/>
        ...
      </auto>
      <book>...</book>
    </domain>
</KB>
```

Figura 14. Base de Conhecimento de Veículos

A base de conhecimento utilizada, como foi dito anteriormente, é um arquivo .xml contendo informações de diversos domínios. Ele usa tags para descrever os conteúdos contextualizados. Na Figura 14, por exemplo, temos os domínios de livros e de veículos. No segundo caso, existem informações sobre o que compõe um carro, como a marca, o modelo, a cor, etc. Com isso é possível realizar buscas ou validações de registros de um domínio apenas com atributos e valores de atributos dentro deste contexto, como por exemplo, veículos com marca *Ford* e modelo *Fiesta* presentes na Deep Web.

A saída do algoritmo é um arquivo .txt com os dados extraídos por ele. Esse arquivo é dado como parâmetro na entrada do algoritmo, assim como a base de conhecimento a ser utilizada e a página HTML a ser extraída. Um exemplo do arquivo de saída pode ser visto na Figura 15. Note que o número 2013 na saída é extraído porém não se sabe se esse número é proveniente do ano ou do modelo do veículo, ambos atributos de veículos.Porém como o extrator proposto está interessado em somente extrair o registro, a origem dele na BC acaba não sendo relevante.

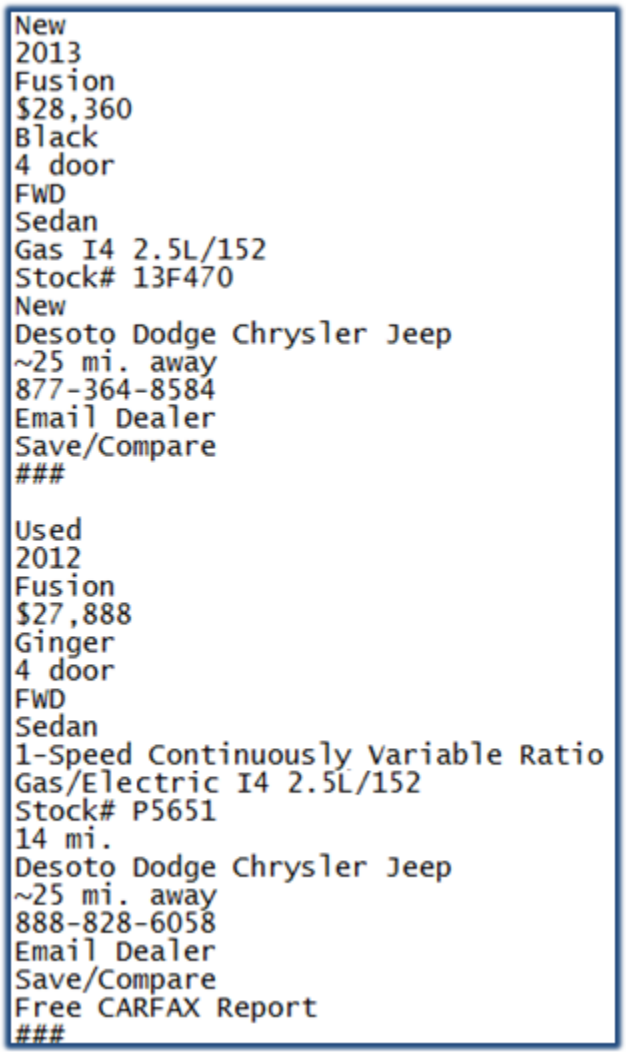

Figura 15. Saída do extrator proposto.

4.4 Pós-Processamento

Com o intuito de facilitar o módulo de catalogação na abordagem DeepEc, foi definido um pós-processamento nos dados retornados pelo extrator. Esse pós-processamento utiliza a base de conhecimento para descriminar os dados retornados pelo algoritmo proposto, de forma que ele retorne uma página onde cada valor possua o seu atributo correspondente identificado. Caso não se ache essa relação do valor do atributo com o atributo na BC, o pós processamento indica isso na página retornada. Por exemplo, se ele coletar um registro contendo “Fiat Uno 2012 Vermelho R$21000 4 portas” e compondo que estejam na base de conhecimento todos os dados menos o 4 portas, ele retornará:

Marca: Fiat
Modelo: Uno
Ano: 2012
Cor: Vermelho
Preço: R$21000
Não achado: 4 portas

Uma observação é que mesmo o algoritmo não contendo em sua base de conhecimento o conteúdo “4 portas”, como foi explicado anteriormente, estando esse dado junto, estruturalmente falando, com algum outro dado identificado pelo algoritmo, ele será extraído também.

5. EXPERIMENTOS

Experimentos foram realizados em dois domínios (Veículos e Livros) com o intuito de avaliar o algoritmo proposto. Foi escolhido o MDR e o Road Runner, dois algoritmos já consolidados para extração de registros em páginas Web, para fins de comparação. Eles foram escolhidos por quê, apesar de serem extratores sintáticos, diferentemente do proposto aqui, que considera semântica com o suporte de uma base de conhecimento, eles se enquadram dentre as poucos que realizam o processo de extração de forma automática, assim como o proposto neste trabalho, e possuem implementações disponíveis para uso.

5.1 Origem dos Dados

Devido a parceria entre os desenvolvedores da máquina de busca para formulários Web *DeepPeep* (Barbosa 31T. Al, 2010) e o Grupo de Banco de Dados da UFSC (GBD/UFSC), as URLs das páginas Web pertencentes aos dois domínios foram gentilmente cedidos pelos seus mantenedores, pois estes domínios eram os domínios com maior volume de dados indexados pela DeepPeepWeb. Parte dos dados concedidos foi também utilizada para popular a base de conhecimento.

Buscas aos formulários existentes nessas páginas foram realizadas e o retorno dessas buscas foi salvo para ser utilizado como entrada para o extrator proposto. Essas páginas geraram 1376 registros no domínio de Veículos e 352 registros no domínio de Livros.

5.2 Métricas utilizadas

Para a medição dos resultados obtidos foram usadas a precisão, a revocação e a medida F, por serem métricas consagradas na área de recuperação de informação (Yates e Neto, 1999). Precisão é a razão entre a quantidade de registros recuperados que são relevantes e o total de registros extraídos. Revocação é a razão entre a quantidade de registros extraídos que são relevantes e o total de registros relevantes existentes. A medida F é a média harmônica das métricas anteriores. As fórmulas das métricas podem ser vista na Figura 16, sendo A o número de registros relevantes recuperados, B o

número de registros não recuperados e C o número de registros recuperados, porém, irrelevantes. Além disso, foi comparado o tempo de processamento utilizado por cada algoritmo.

$$R = \frac{A}{A+B}, P = \frac{A}{A+C} \text{ and } F = \frac{2(R*P)}{(R+P)}$$

Figura 16. Fórmulas para Calculo de Revocação, Precisão e Medida F

5.3 Resultados

A Tabela 3 apresenta os resultados da extração realizada pelo algoritmo proposto (DeepEC - Extração) em comparação com os algoritmos Road Runner e MDR baixados de seus respectivos sites. O item Atual corresponde ao total de registros presentes nas páginas; Extraídos indica o número de registros extraídos pelos algoritmos; Corretos indica a quantidade de registros extraídos corretamente; Incorretos indica a quantidade de registros extraídos incorretamente. P é a precisão, R a revocação, F é a medida F e Tempo (s) o tempo em segundos utilizados por cada algoritmo. Esses experimentos obtiveram melhores resultados utilizando make (Marca) e model (Modelo) como atributos mandatórios para o domínio de Veículos, bem como publisher (Editora) e author (Autor) para o domínio de Livros.

Algoritmo	Domínio	Atual	Extraídos	Corretos	Incorretos	P	R	F	Tempo (s)
Road Runner	Auto	1376	1393	1320	73	0,94	0,95	0,95	18,10
MDR			1422	1312	110	0,92	0,95	0,93	13,56
DeepEC Extração			1410	1326	84	0,94	0,96	0,95	10,09
Road Runner	Livros	352	381	323	58	0,84	0,91	0,88	4,95
MDR			392	318	74	0,81	0,90	0,85	3,62
DeepEC Extração			364	320	44	0,87	0,90	0,89	2,88

Tabela 1. Resultado dos Experimentos

Como pode-se observar, o algoritmo proposto apresentou resultados similares aos demais, porém, no domínio de Livros conseguiu superar tanto na

precisão quanto na medida F, que era esperado uma vez que o algoritmo proposto tem uma base de conhecimento que o auxilia a encontrar os registros a serem extraídos. Já no domínio de Veículos, o algoritmo proposto conseguiu ter o melhor percentual de acertos na recuperação dos registros presentes nas páginas de resultados.

Além disso, um ponto bastante positivo a ressaltar foi a diferença em termos de tempo de processamento. O algoritmo proposto conseguiu um desempenho bem melhor que os outros em ambos os domínios. No domínio de Livros conseguiu ser 42% mais rápido que o Road Runner e 21% mais rápido que o MDR. No domínio de Veículos foi 44% mais rápido que o Road Runner e 26% mais rápido que o MDR.

6.CONCLUSÃO

A extração de dados da Internet sempre foi algo bastante relevante para diversas finalidades, principalmente para fins de consulta ou integração de dados. Com isso vem a necessidade de encontrar esses dados nessa grande rede global.

O que poucos sabem é que o conteúdo acessível através de motores de busca indexa e acessa é uma pequena parte do conteúdo disponível na Web. O resto desse conteúdo fica escondido no que chamamos de Deep Web. Este conteúdo só pode ser acessado dinamicamente, ou seja, você precisa preencher primeiro um formulário para que retornem os resultados que lhe interessam baseados no filtro que você informou no formulário. Para isso já existem algumas abordagens que preenchem esses formulários de forma automática. Porém, esses resultados retornados são exibidos em páginas cheias de informações que não são relevantes, como menus, propagandas, links externos, etc. Desta forma, se faz necessária uma solução que realize a extração somente dos dados relevantes para facilitar a catalogação e posterior acesso.

Esse trabalho contribui com essa problemática apresentando um novo algoritmo para extração de dados relevantes em páginas de resultados de pesquisas a formulários Web e o valida através dos experimentos realizados. O algoritmo usa para isso uma base de conhecimento e a estrutura HTML da página, sendo assim um algoritmo misto que usa tanto a semântica quanto a estrutura. Além disso, foram criadas heurísticas com o intuito de otimizar o processo de extração. O algoritmo tomou como base os algoritmos mais conhecidos na atualidade que são o Road Runner e o MDC que também foram usados como parâmetros de comparação na avaliação do algoritmo, além da abordagem AMBER que inspirou o uso de uma base de conhecimento.

O algoritmo proposto é parte integrante de uma abordagem denominada DeepEC, que visa extrair os dados recuperados da Deep Web, catalogá-los e armazená-los em um banco de dados relacional. O algoritmo proposto recebe os resultados das pesquisas aos formulários, usa a base de conhecimento do DeepEC para auxiliar na extração dos dados relevantes e produz como saída um arquivo texto com os dados extraídos para posterior catalogação.

Os resultados apresentados mostraram um bom desempenho e confirmaram que o algoritmo proposto não deixou a desejar em relação aos algoritmos clássicos em nenhum aspecto. Pelo contrário, na maioria dos casos ele teve um desempenho melhor, principalmente por considerar a semântica do domínio. Além disso, o algoritmo proposto tem a vantagem de não utilizar diversas páginas como amostragem, como é feito no Road Runner, nem considerar apenas a estrutura HTML, como é o caso do MDC.

Com o intuito de melhorar as funcionalidades e o desempenho do algoritmo proposto, os seguintes trabalhos futuros podem ser realizados:

- Uma realimentação automática da base de conhecimento, de forma que cada registro extraído que contenha informação nova sobre o domínio seja inserido na base de conhecimento;
- Uma personalização na filtragem dos registros extraídos, pois apesar do algoritmo extrair dados úteis que a página HTML possui com alta taxa de acerto, o resultado pode conter dados que o usuário não tenha interesse. Isso requer uma análise dos dados irrelevantes retornados e modificação do algoritmo para reduzir essa quantidade.

8. REFERÊNCIAS

BARBOSA, L., NGUYEN, H., NGUYEN, T.,PINNAMANENI , R., and FREIRE, J. (2010). **Creating and Exploring Web Form Repositories**. Em: Proceedings of the ACM SIGMOD International Conference on Management of data, p. 1175-1178.

BERKMAN. K. Michael. **The DeepWeb: Surfacing Hidden Value**. Disponível em:http://brightplanet.com/images/uploads/12550176481-deepWebwhitepaper.pdf Data de acesso: 27/11/2011.

CRESCENZI, V., MECCA, G., MERIALDO, P. (2001). "**RoadRunner: Towards Automatic DataExtraction from Large Web Sites**", Em: Very Large Data Base (VLDB).

DE SOUZA, A., MELLO, R. (2012). "**Extração e catalogação de informações sobre banco de dados escondidos na Web**"

Furche, T., Gottlob, G., Grasso, G., Orsi, G., Schallhart, C. and Wang, C. **Little Knowledge Rules The Web: Domain-Centric Result Page Extraction**. In *International Conference on Web Reasoning and Rule Systems (RR'11)*, pages 61-76, 2011.

HALEVY, A., MADHAVAN, J., AFANASIEV, L. and ANTOVA, L. (2009).**Harnessing the *Deep Web*: Present and Future**.In Conference on Innovative Data Systems Research (CIDR).

HONG, Jun; HE, Zhongtian and BELL, David A. (2010). **An Evidential Approach to Query Interface Matching on The Deep Web**. Em: Journal Information Systems, p. 140-148.

LAN Yi, BING Liu, XIAOLI Li. "**Eliminanting Noisy Information in Web Pages for Data Mining"**. Disponivel em: http://www.cs.uic.edu/~liub/publications/kdd2003-WebNoise.pdf

LIU, B. "**Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data**"p.362-364

LIU, B., GROSSMAN,R., ZHAI, Y. (2003). "**Mining Data Records in Web Pages.**"Proceedings of the ACM SIGKDD International Conference on KnowledgeDiscovery & Data Mining.

MILLER, George A. (1995). **WordNet: A Lexical Database for English**. In Communications of the ACM Vol. 38, N°. 11, p. 39-41.

NONAKA, I., TAKEUCHI, H. (1997). **Criação de Conhecimento na Empresa: como as empresas japonesas geram a dinâmica da inovação**. Rio de Janeiro: Campus, p. 97.

SERRA, E.,CORTEZ, E. SILVA, A. S. and MOURA, E. S. (2011). **On Using Wikipedia to Build Knowledge Bases for Information Extraction by Text Segmentation**.In Journal of Information and Data Management, Vol. 2, No. 3, p. 259–272.

W3C DOM. **W3C Document Object Model.** Disponível em: <http://www.w3.org/DOM/#what>. Acesso em: 08 nov. 2013.

YATES, R. B. and NETO, B. R. (1999). **Modern Information Retrieval.** First Edition. Addison Wesley Longman Limited, England.

ZHENG, Z., SI, X., LI, F., CHANG, E. Y. and ZHU, X. (2012).**Entity Disambiguation with Freebase**.In International Conferences on *Web* Intelligence and Intelligent Agent Technology (IEEE/WIC/ACM), p. 82-89.