UNIVERSIDADE FEDERAL DE SANTA CATARINA
INE - DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
SIN – SISTEMAS DE INFORMAÇÃO

UM COMPARATIVO ENTRE TÉCNICAS DE CASAMENTO (MATCHING) DE FORMULÁRIOS NA DEEP WEB

JOSÉ JULIO RESENES NETO

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Florianópolis 2012

JOSÉ JULIO RESENES NETO

UM COMPARATIVO ENTRE TÉCNICAS DE CASAMENTO (MATCHING) DE FORMULÁRIOS NA DEEP WEB

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Ronaldo dos Santos Mello, Dr.

Banca Examinadora

_____

Carina Friedrich Dorneles, Dra.

_____

Frank Augusto Siqueira, Dr.

Sumário

Lista de Figuras

Lista de Tabelas

RESUMO

Grande parte da informação relevante na Web só está acessível através de formulários. Essa informação é resultado de consultas realizadas em bancos de dados escondidos a partir de filtros definidos pelo usuário. Esta é a chamada *Deep Web*, *Hidden-Web* ou ainda Web oculta. Um dos principais objetivos de um ambiente de integração de dados ou de busca integrada é fornecer ao usuário uma visão única de diversas fontes de dados heterogêneas e distribuídas, criando a impressão no usuário de se estar utilizando um sistema centralizado e homogêneo. Nesse contexto de integração, técnicas de *matching* (casamento) de formulários são fundamentais para fornecer informação relevante ao domínio de interesse do usuário (revendas de automóveis, passagens aéreas, etc) no contexto da *Deep Web*. Inserido neste tema, o trabalho apresenta uma revisão da literatura sobre *matching* de formulários na *Deep Web*. Critérios são definidos e utilizados para uma análise entre as abordagens relacionadas e, por fim, são sugeridos tópicos para pesquisas futuras.

Palavras chave: *Deep web, Matching, web forms*

Abstract

Most of the relevant information on the Web is accessible only through Web forms. This information is gathered as the result of queries submitted through the form to hidden databases. This kind of information is known as the Deep Web, Invisible Web or Web-hidden. One of the main goals of a data integration system or search system over several data sources is to provide the user with a single view of various heterogeneous and distributed data sources, giving the notion of a centralized and homogeneous system. In this context, Web forms integration techniques are the key to provide relevant information to the user's interest (car dealers, airline tickets, etc.).This work presents a review of the literature about matching web forms approaches in the Deep Web context. Some comparison criteria are defined and used for analyzing the related approaches. Some hints for future research are also given.

1- INTRODUÇÃO

A Web é um ambiente complexo que contém diversas fontes de informação, em diferentes tipos e formatos. Atualmente, fazer buscas na Web é como arrastar uma rede em toda a superfície de um grande oceano. Grande quantidade de informações pode ser apanhada na rede, ou seja, na superfície do oceano (dita *Surface Web*), mas há um vasto volume de informações que estão nas profundezas e, portanto, não pode ser alcançada facilmente (BERGAMAN, 2007).

O "oceano profundo" citado por Bergman é conhecido como "*Deep Web*", uma parcela da Web que compõe um grande número de dados e informações que não estão ao alcance dos mecanismos tradicionais de busca, como o Google[1]. Estes dados são normalmente acessados através de formulários (*Web forms*) específicos para um domínio.

O último grande estudo feito sobre a *Deep Web* estima que ela contenha entre 7500 e 91850 *terabytes* de informações. A medida que o volume de informações na *Deep Web* cresce, também aumenta o interesse em técnicas e ferramentas que permitam usuários e aplicações explorarem estas informações (FREIRE e BARBOSA, 2005).

Como grande parte da *Deep Web* está contida em bancos de dados ocultos, a forma mais usual de acessar estes dados é através de formulários web que permitem, através da entrada de dados para fins de busca, a recuperação destas informações ocultas. Estes bancos de dados ocultos contêm informações importantes para diversos serviços, de diferentes domínios, como aluguéis e vendas de veículos e reservas de passagens áreas e hotéis. Além disso, nesses bancos de dados podemos encontrar uma grande quantidade de dados úteis para fins acadêmicos, como sequências de DNA e publicações científicas.

[1] www.google.com

Desta forma, a pesquisa em *Deep Web* visa principalmente desenvolver sistemas de busca a bancos de dados na Web baseados em seus dados/metadados, sistemas integrados de busca/prestação de serviços, entre outras aplicações, que irão facilitar, por exemplo, a busca por um quarto de hotel em todos os hotéis de uma cidade, ou localizar um determinado veículo para compra em qualquer lugar do País, dentre outras possibilidades.

Neste contexto, várias são as técnicas e ferramentas utilizadas para a localização e extração dos dados ocultos da *Deep Web* através de *Matchers*, *Crawlers* e técnicas de similaridade. Uma vez alcançado os dados ocultos, o grande desafio é filtrar os vários formulários disponíveis, extrair e integrar informações realmente pertinentes ao usuário ou aplicação de forma eficaz.

Como o universo de dados contido na *Deep Web* é imenso, as técnicas empregadas geralmente setorizam estes dados em domínios distintos. Desta forma, é mais fácil criar mecanismos para acessá-los. Um grande problema é que mesmo em um único domínio, as diferenças entre os formulários são enormes. Isto se deve principalmente a dois fatos. O primeiro é que nem sempre são utilizados os mesmos termos para se referenciar um objeto. Em um formulário de busca de veículos, um carro, por exemplo, pode ser chamado de “Carro” e em outro formulário de “Veículo”, “Automóvel”, etc. O segundo fato é que nem todos os formulários solicitam o mesmo tipo de informação. Neste caso, em um formulário pode ser necessário preencher o ano de fabricação de um veículo e em outro não.

Como se não bastassem os problemas mencionados, ainda existem empecilhos relacionados à própria usabilidade dos formulários, que variam muito. Em um formulário para a localização de veículos pode ser necessário digitar a marca de um veículo enquanto em outro pode ser feita a seleção em componente de lista. Ainda neste contexto, podemos ter problemas com questões como um determinado campo só aparecer no formulário quando outro

for preenchido. Como exemplo, os modelos disponíveis de veículos só aparecerem após a marca ser selecionada.

Até mesmo a forma como os formulários são “implementados” dificulta a exploração dos mesmos por mecanismos automatizados. Por exemplo, se um *Matcher* utiliza a descrição de um campo como fonte de informação para a sua identificação, ele pode ter problemas com formulários onde o campo para se escolher o veículo não é a descrição “Carro”, mas sim a imagem de um carro.

Motivado pelas dificuldades apresentadas, no que tange a exploração de dados da *Deep Web* através de formulários, este trabalho visa estudar e comparar as técnicas atualmente propostas e utilizadas para unificar formulários Web de um mesmo domínio, com o objetivo de identificar as principais técnicas de similaridade, abordagens e informações utilizadas. O objetivo deste trabalho é contribuir com as pesquisas que necessitam da execução de *Matching* de dados na *Deep Web*.

Durante a revisão bibliográfica, identificaram-se alguns *surveys* sobre *matching* de esquemas de dados, como por exemplo (DORNELES, 2011) e (BERNSTEIN, 2011). Entretanto este trabalho apresenta e compara artigos recentes e focados na problemática de *matching* de formulários Web.

Este trabalho está organizado em mais 5 capítulos. O próximo capítulo apresenta uma visão geral de *Deep Web*. O capítulo 3 apresenta os fundamentos de *Matching* e Similaridade de dados. O capitulo 4 revisa e comenta as principais características de alguns trabalhos relacionados à problemática. O capitulo 5 apresenta um comparativo entre eles e, por fim, o Capítulo 6 é dedicado à conclusão e trabalhos futuros.

2- DEEP WEB

2.1 SURFACE WEB

A *Web*, também conhecida como WWW, é um ambiente complexo que contém diversas fontes de informação, em diferentes tipos e formatos. O primeiro grande estudo do instituto *Lawrence and Giles da Research*, baseado no que eles chamam de objetos "*publicly indexable*" (publicamente indexáveis), foi um artigo publicado na revista *Science* em 1998. Através da análise de dados do mês de dezembro de 1997, estimou-se que o tamanho total da *Web* era de 320 milhões de documentos (INKROMI CORP, 2000).

Uma atualização para esse estudo empregando uma metodologia diferente foi publicado na revista *Nature* em 1999, usando a análise de fevereiro de 1999 (INKROMI CORP, 2000). Este estudo documentou 800 milhões de documentos dentro da Web publicamente indexável, com uma média de tamanho de página de 18,7 *kilobytes*, exclusivo de imagens e cabeçalhos HTTP.

Em parceria com a Inktomi, a *NEC* atualizou suas estimativas de paginas Web em um bilhão de documentos no início de 2000. Em julho de 2000 estimava-se que conteúdo de dados na “Web profunda” fosse 500 vezes maior que o de “superfície” (FREIRE e BARBOSA, 2004). A Web rapidamente "aprofundou-se" por grandes bases de dados online (GHANEM e AREF, 2004). Enquanto que a chamada “*Surface Web*” ligou bilhões de páginas estáticas em HTML, acredita-se que uma quantidade muito mais significativa de informação esteja “escondida”.

2.2 MOTORES DE BUSCA

Os motores de busca começaram a surgir quando o número de recursos na Web adquiriu proporções tais que impediam a sua coleta por meios manuais e a busca apenas através da navegação. A maioria deles derivou do trabalho de

estudantes de pós-graduação, professores, funcionários do departamento de sistemas de empresas ou outras pessoas interessadas na Web (BLATTMANN, FACHIN e RADOS, 1999).

Em um motor de busca, páginas são recuperadas por um *Web crawler* (também conhecido como *spider*), um *web browser* automatizado que segue *hiperlinks*. O conteúdo de cada página é analisado para determinar de que forma este conteúdo será indexado. Os dados sobre as páginas são armazenados em um banco de dados indexado para uso em pesquisas futuras. Alguns sistemas, como *Google*, armazenam o conteúdo completo ou parcial da página de origem (referenciado como um *cache*), assim como informações sobre as páginas. Já outros armazenam cada palavra de cada página encontrada, como o *Alta Vista*.

Existem ainda barreiras que não permitem que o conteúdo disponibilizado na Web seja indexado devidamente, como:

1. Por uma variedade de razões, o proprietário ou desenvolvedor do *web site* pode especificar quais partes do site deverão ou não ser indexadas. Existem algumas *meta tags* (tipo de instrução HTML) que podem ser usadas no cabeçalho de um site, como exemplo "*NOINDEX*", tornando-os, desta forma, inacessíveis;
2. Páginas podem necessitar autenticação, estar protegidas com um usuário e senha que são obtidas em bases pagas ou não. *Crawlers,* desta forma, não podem acessá-las. Embora pouco possa ser feito, é uma questão de segurança que estas páginas sejam desconhecidas (LAWRENCE e GILES, 1998);
3. Para serem descobertas, as páginas devem ser estáticas e ligadas umas as outras. Motores tradicionais não conseguem recuperar conteúdo dinâmico;
4. Muitos dos resultados apresentados ao usuário são modificados, devido ao seu perfil de utilização nos motores de busca e serviços associados, normalmente devido a prioridade dada em publicidade;

5. Muitos sites também são automaticamente ignorados na busca por apresentarem conteúdo ofensivo ou potencialmente perigoso.

Mike Bergman (BERGAMAN, 2007) representa graficamente, vide Figura 1, as limitações típicas de um motor de busca, onde o conteúdo identificado é bastante indiscriminado e de superfície.

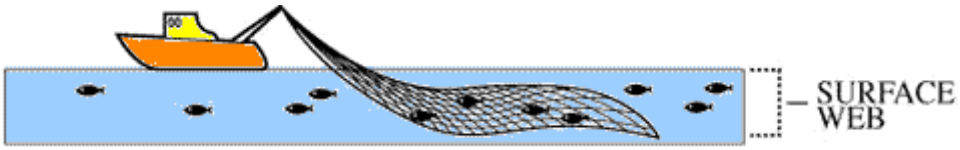

Figura 1 - Busca típica realizada por motores de busca (BERGAMAN, 2007)

Procurar informação na Web é como arrastar uma rede em toda a superfície do oceano. Uma grande quantidade pode ser apanhada na rede, mas há uma riqueza de informações que estão no “oceano” profundo, não podendo ser alcançada facilmente.

2.3 DEEP WEB

O termo *Deep Web*, também referenciado como *hidden* ou *invisible web*, é utilizado para definir o conteúdo oculto através de formulários HTML onde, para acessá-lo, o usuário necessita enviar uma requisição ao servidor com valores válidos de entrada. Como respostas às consultas submetidas ao banco de dados, páginas são construídas dinamicamente para apresentar o conteúdo (BERGAMAN, 2007). A Figura 2 mostra um exemplo hipotético de formulário Web para o domínio de carros. Através desta figura é possível ver que apenas alguns poucos atributos, com seus rótulos (nomes) e seus valores, do banco de dados escondido (aqueles usados para definir as buscas) são visíveis.

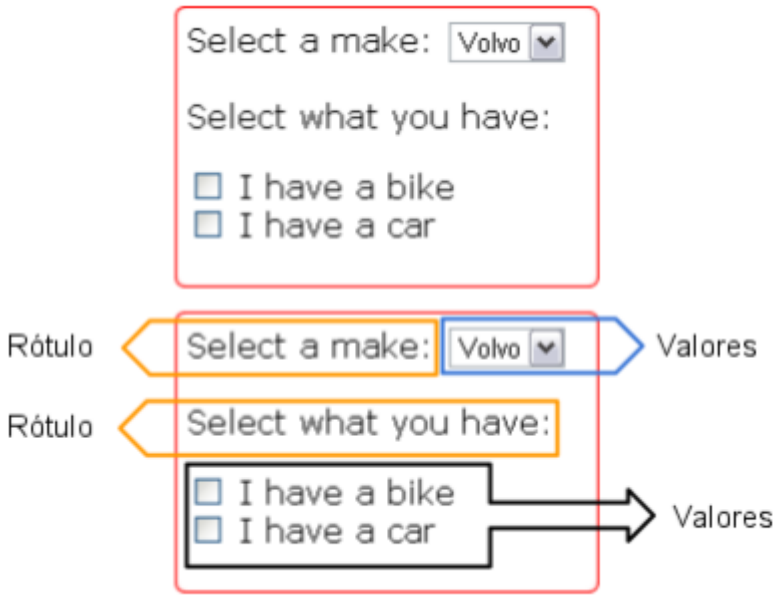

Figura 2 - Formulário web e seus componentes

Enquanto boa parte da informação na superfície da web é principalmente textos e HTML não estruturados, a *Deep Web* pode ser classificada em dois tipos (GOOGLE INC., 2009):

(i) Bases de dados não estruturadas que fornecem objetos não estruturados como mídias (textos, imagens, áudio e vídeo);

(ii) Base de dados estruturados que fornecem dados relacionais, ou seja, registros com pares (atributo, valor);

A *Deep Web* já é reconhecida como uma lacuna significativa na abrangência dos motores de busca atuais, pois os *Crawlers* empregados nos motores contam com os *hiperlinks* para descobrir novas paginas na web e, normalmente, não têm a capacidade de realizar envios para tais formulários. O resultado disso são milhões de sites que nunca são alcançados por buscadores comuns, um conteúdo secreto e invisível que sempre esteve ali, mas que dificilmente um dia será visualizado por um usuário ou aplicação, conforme pode ser visto na Figura 3.

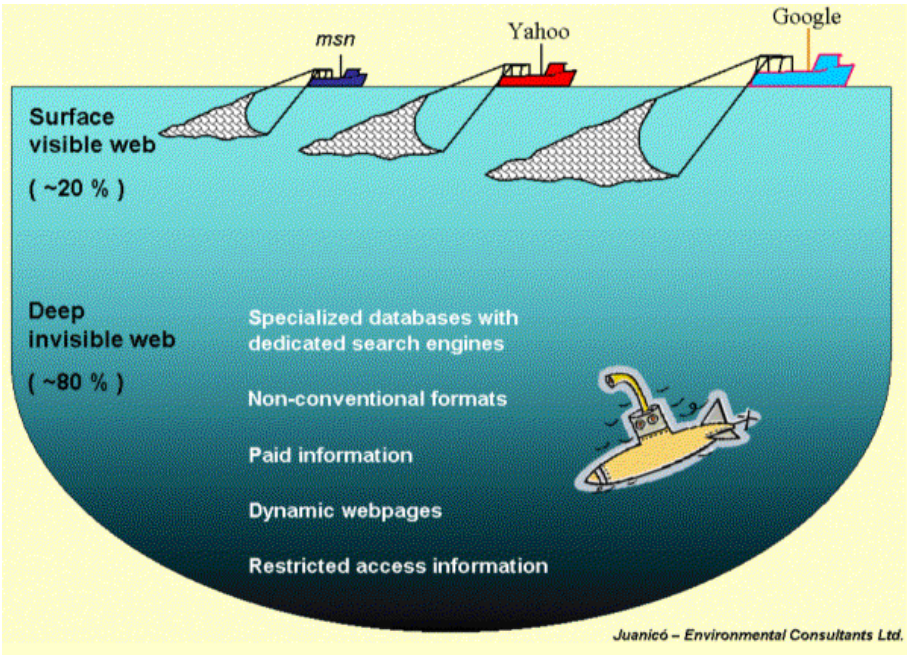

Figura 3 - Representação da DeepWeb (JUANICÓ – ENVIRONMENTAL CONSULTANTS LTD, 2006)

Por este, motivo a *Deep Web* tem sido um desafio de longa data para a comunidade de banco de dados, já que ela representa uma grande parte dos dados estruturados na Web.

3- *MATCHING* E SIMILARIDADE DE DADOS

3.1 *MATCHING* DE DADOS

Um importante conceito para se entender como é o funcionamento e as dificuldades enfrentadas no processo de localização, extração e integração de dados na *Deep Web* é o conceito de *Matching* de Dados. *Matching* é o processo de identificar a correlação semântica entre dois objetos, ou seja, identificar se dois objetos podem ser tratados como semelhantes.

Matching é um problema recorrente em vários problemas associados à gerência de dados, como integração de dados, *data warehouse* e processamento semântico (RAHM e BERNSTEIN, 2001). Juntamente com processos de mapeamentos de dados, processos de *Matching* formam os pilares da integração de dados. Desta forma, elaborar um processo automatizado de *Matching* é essencial para se obter um processo eficiente de integração. Entretanto, os desafios são vários, devido às diferenças encontradas nas definições dos dados. Essas diferenças podem ser: (i)*Sintáticas*: diferenças de linguagem utilizadas para representar os elementos; (ii) *Estruturais*: diferenças quanto ao tipo e a estrutura dos elementos; (iii) Diferenças de *modelos de representação*, que podem ser relacionais, orientados a objetos, etc; e (iv)*Semânticas*: situações onde a descrição de um dado pode ser representada por diferentes termos.

As dificuldades encontradas no processo de *Matching* podem ocorrer quando os objetos comparados estão em domínios distintos. Contudo, ainda que em um mesmo domínio, estas diferenças podem estar presentes, visto que os modelos de dados são desenvolvidos por diferentes pessoas. Assim, a primeira etapa para a integração de diferentes objetos é identificar as semelhanças entre eles. Uma vez identificadas, os objetos podem ser unificados em um terceiro

objeto que seja coerente com a representação dos dois objetos iniciais. Paralelo a este processo ou posteriormente a ele, podem ser utilizados mecanismos automáticos que permitam a tradução dos dados dos objetos originais para o objeto que representará a integração entre eles.

3.2 Similaridade de Dados

Outro importante conceito é o de Similaridade de Dados. Medir a similaridade ou a distância entre dois objetos distintos é o requisito principal para várias tarefas que necessitam realizar processos de *Matching* de dados. As técnicas têm uso frequente em áreas como mineração de dados e exploração de bases de conhecimento. Desta forma, a similaridade entre dois objetos nada mais é do que mostrar computacionalmente, através de um valor (grau de similaridade), o quanto dois objetos são semelhantes entre si. Por “Objeto” entende-se qualquer item de dado, como uma palavra, tupla, documento, estrutura de dados complexa, imagem, som, vídeo, etc.

Neste trabalho utiliza-se o termo função de similaridade de forma genérica, referindo-se também às métricas de similaridade. De forma simplificada, medidas de similaridade podem ser definidas como: sejam x e y objetos de um determinado universo U, a medida de similaridade é a função sim(x; y) →[0; 1] (STASIU, 2007). Embora os valores sejam dependentes da implementação da função de similaridade, tipicamente um resultado (escore) igual a 0 significa que os dois objetos analisados são totalmente diferentes, e um escore igual a 1 indica que são iguais. Existe um grande número de algoritmos, no entanto, todos eles têm em comum o fato de que o resultado dos cálculos de similaridade é representado por um único valor.

As funções de similaridade, assim como as funções de distância, têm por finalidade atribuir uma medida do grau de semelhança entre dois objetos. Para receber a denominação de métrica (BIMBO, 1999), uma função de distância precisa atender as seguintes condições:

- Identidade: *d(s1, s2) = d(s1,s2)= 0*;
- Minimalidade: *d(s1, s2) ≥ d(s1,s2) ≥ 0;*
- Simetria: *d(s1, s2) = d(s2,s1);*
- Desigualdade triangular: d(s1, s2) + d(s2,s3) ≥ d(s1,s3).

Quando os dados analisados pelo processo de *matching* são atômicos, como por exemplo (*strings*, valores numéricos, datas, etc.), o problema pode ser resolvido através do uso de uma função de similaridade apropriada para o tipo de dado. Em casos onde os valores são complexos, como tuplas em bancos de dados relacionais, formulários web e documentos XML, podem ser adotadas abordagens que combinam as funções de similaridade para dados atômicos, com outras técnicas mais sofisticadas, como por exemplo, aprendizado de máquina (DORNELES, 2011).

Existem várias funções de similaridade para *strings*, como Levenshtein (LEVENSHTEIN, 1966), Jaro-Winkler (JARO, 1989), N-Grams (ELGARAMID, 2007), entre outras. As diferentes funções de similaridade resultam valores diferentes para as mesmas entradas. A escolha do melhor método vai depender do domínio em que o cálculo de similaridade será aplicado (LEVIN, 2010).

4 - *MATCHING* DE FORMULÁRIOS NA DEEP WEB

Com o objetivo de contribuir com as pesquisas que necessitam da execução de *Matching* de dados na *Deep Web*, este capítulo revisa e comenta as principais características de alguns trabalhos relacionados a esta problemática.

4.1 PRUSM

PRUSM (*Prudent Schema Matching*) é um sistema que combina várias fontes de informação para efetuar o *Matching* de forma “cautelosa”, ou seja, com mínima propagação de erros (FREIRE, NGUYEN e NGUYEN, 2010). A visão em alto nível da arquitetura do sistema é representada na **Erro! Fonte de referência não encontrada.**.

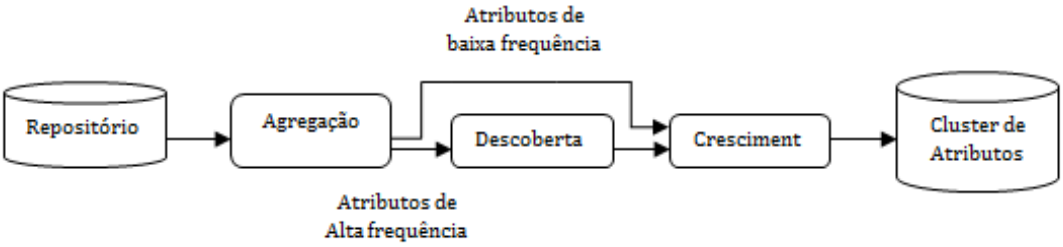

Figura 4- Processo de Extração do PRUSM, adaptado de (FREIRE, NGUYEN e NGUYEN, 2010)

PruSM recebe como entrada, através do módulo de Agregação, uma coleção de formulários previamente coletados de um repositório, sem necessidade de pré-processamento. Ele utiliza, então, técnicas de mineração de texto para a preparação dos dados, como por exemplo, a radicalização ou *stemming*, que reduz variações de uma mesma raiz vocabular com a finalidade de recuperar palavras correlatas. Outra forma de tratamento é a remoção de *stopwords*, que são palavras consideradas sem valor para a busca devido a sua natureza frequente ou semântica.

O módulo de agregação é responsável ainda por agrupar atributos similares, gerando como saída um conjunto de atributos frequentes *S1* para o módulo de Descoberta e também um conjunto de atributos infrequentes *S2* para o módulo de Crescimento. Dados dois atributos (*Ai, Aj*) recebidos como entrada pelo módulo de Descoberta, PrusM quantifica a semelhança entre eles por meio de três medidas: *(i)* similaridade do rótulo*, (ii)* similaridade do valor de domínio e *(iii)* correlação. Inicialmente é calculada uma medida de importância do termo dentro do conjunto baseado na sua frequência, utilizando a distância do cosseno entre os termos dos vetores de seus rótulos, como por exemplo, “Fabricação” e “Ano de Fabricação”, conforme Equação 1.

$$lsim(A_i, A_j) = cos(l_i, l_j)$$

Equação 1 - Similaridade do rótulo

Para calcular a similaridade entre os valores do domínio, primeiramente é necessário agregá-los em relação a cada atributo. Desta forma, dado o atributo *Ak*, é construído um vetor que possui todas as ocorrências de valores associadas ao rótulo *l* para *Ak* e sua frequência *Dk*. A distância do cosseno é então utilizada para calcular a similaridade entre os vetores, conforme Equação 2.

$$dsim(A_i, A_j) = cos(D_i, D_j)$$

Equação 2 - Similaridade valores

Posteriormente, é realizada a correlação dos valores dos atributos utilizando-se as medidas *X/Y* (SU, WANG e LOCHOVSKY, 2006), conforme mostra a Equação 3. As variáveis *Cp, Cq, Cpq* correspondem ao número de formulários que contêm os atributos *Ap, Aq* e ambos (*Ap, Aq*), respectivamente. A medida X é responsável por mensurar a correlação negativa, enquanto a medida Y captura a correlação positiva. A intuição vem do fato de que os atributos são sinônimos (semanticamente equivalentes) e raramente aparecem na mesma interface.

Com isso, é possível identificar casos como “Montadora” e “Fábrica”, que apesar de possuírem grafia diferente, possuem o mesmo significado no domínio em análise.

$$X(A_p, A_q) = \begin{cases} 0 & \text{if p,q} \subset \text{form F} \\ \frac{(C_p - C_{qp})(C_q - C_{qp})}{(C_p + C_q)} & \text{otherwise} \end{cases}$$

$$Y(A_p, A_q) = \frac{C_{pq}}{min(C_p, C_q)}$$

Equação 3 - Medida de correlação

Os relacionamentos identificados, junto aos atributos de baixa frequência, são utilizados pelo módulo de Crescimento para obter relacionamentos adicionais. Neste módulo, os atributos com baixa frequência têm seus pesos recalculados através de um algoritmo chamado *STF (Singular Token Frequence)* (FREIRE, NGUYEN e NGUYEN, 2008)**.** O *STF* aplica outro algoritmo conhecido como *1NN (1-Nearest-Neighbor Clustering)* que agrupa o atributo raro ao seu vizinho mais próximo de maior frequência.

Este último passo gera um novo cluster de atributos considerados representativos. Por fim, incorpora-se este novo cluster ao cluster confiável, utilizando um algoritmo denominado *HAC (Hierarchical Agglomerative Clustering)*. No *HAC*, esses atributos são representados em uma estrutura de árvore e são aglomerados aos mais próximos sucessivamente. Ocorre, então, a identificação de conjuntos de alta confiança, ou seja, *clusters* de atributos afins.

Uma avaliação experimental do PruSM demonstra que a abordagem é eficaz e capaz de corresponder com precisão a um grande número de formulários, principalmente em domínios mais heterogêneos, como por exemplo o de livros.

4.2 WF-SIM

WF-Sim é um método de busca para formulários Web, com base na similaridade de suas interfaces (GONÇALVES, D'AGOSTINI, *et al.*, 2011). Ele é composto por 3 módulos principais: clusterização, indexação e busca. O objetivo do método é retornar um conjunto de formulários semelhantes a partir de uma entrada composta por um conjunto de atributos (e os valores possíveis, caso existam).

Primeiramente, visando definir uma base de formulários para posterior busca, dado um conjunto de formulários, são extraídas as propriedades dos elementos. Cada elemento representa um atributo contendo um rótulo e opcionalmente um conjunto de valores. Essa divisão é exemplificada pela Figura 5 onde, no primeiro elemento, temos o rótulo "*Select a make"* e uma relação de valores iniciada por "Volvo".

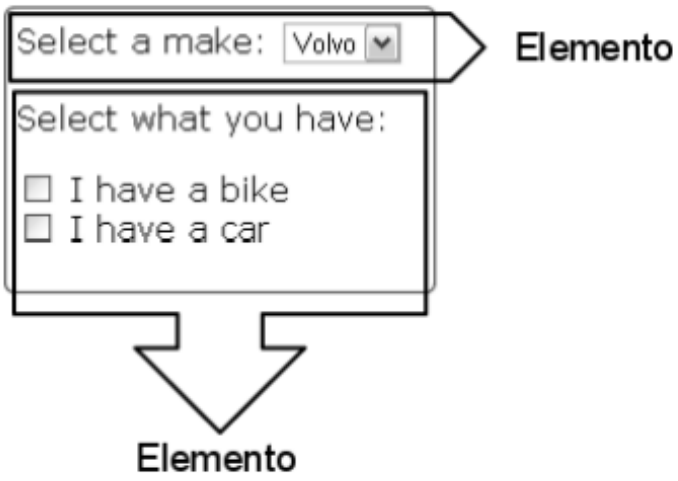

Figura 5 - Divisão de um formulário web em elementos

Na etapa seguinte, técnicas de agrupamento são utilizadas para reduzir o número de entradas no índice de elementos com base em uma métrica de

similaridade denominada *elementSim*. A partir do cálculo das similaridades dos rótulos e dos valores separadamente, é feita uma média ponderada desses valores, conforme mostra a Equação 4.

$$elementSim = labelSim(l_1, l_2) * labelWeight + valuesSim(vl_1, vl_2) * valuesWeight$$

Equação 4 - Cálculo da similaridade do elemento

Nesta Equação, *l1* e *l2* são os rótulos, *vl1* e *vl2* os valores, e *labelSim* e *valuesSim* são os cálculos de similaridade dos rótulos e valores, respectivamente. Por ser uma média ponderada, é possível aplicar pesos diferentes para cada atributo do elemento (*labelWeight* e *valuesWeight*), sendo a soma dos pesos igual a um (1). *LabelSim* representa a semelhança entre os rótulos utilizando a métrica *TF-IDF (Term Frequency Inverse Document Frequency)* (COHEN, RAVIKUMAR e FIENBERG, 2003)**.** A *TF-IDF* é uma estatística que reflete numericamente a importância de uma palavra em relação a uma coleção. A semelhança entre os valores é denotada por *ValuesSim*, calculada através da métrica *SubSetSimmetric* (DORNELES, 2004). Ela obtém bons resultados quando se compara múltiplos valores, porque embora os valores sejam bastante heterogêneos, essencialmente eles são o mesmo conjunto (um dos conjuntos é mais completo ou uma extensão do outro). Vale mencionar que, para elementos pertencentes a um mesmo formulário não é calculada a similaridade, visto que o objetivo é encontrar similaridades entre formulários diferentes.

O processo de agrupamento (clusterização) é realizado considerando um elemento por vez. Para cada elemento, um subconjunto é selecionado, onde $\forall$ *ei, E* $\in$ *SE, distance(ei, E)* $\leq$ *r*. Os resultados são organizados em uma matriz de distâncias como ilustrado na Tabela 1.

Elemento	Coordenadas do Vetor

	e1	e2	e3	e4	e5
e1	0	1	5	9	9
e2	1	0	5	9	9
e3	5	5	0	9	9
e4	9	9	9	0	9
e5	9	9	9	9	0

Tabela 1 - Matriz de distâncias

A tabela apresenta um exemplo simplificado, onde cada linha pode ser vista como um vetor n-dimensional. Cada vetor é considerado como uma coordenada no espaço vetorial correspondente à linha do elemento na matriz e, para todos os vetores, a mediana (*Median Graph*) é calculada como ilustrado na Tabela 2.

Elemento	Coordenadas do Vetor				
	e1	e2	e3	e4	e5
e1	0	1	5	~~9~~	~~9~~
e2	1	0	5	~~9~~	~~9~~
e3	5	5	0	~~9~~	~~9~~
~~e4~~	~~9~~	~~9~~	~~9~~	~~0~~	~~9~~
~~e5~~	~~9~~	~~9~~	~~9~~	~~9~~	~~0~~
Mediana	1	1	5	-	-

Tabela 2 - Matriz da distância para elementos dentro de um raio r <6 de e1 entre elementos

Entretanto, como não é possível converter uma coordenada de vetor espacial para um texto, é definido como elemento médio aquele que possui a menor distância para o vetor de mediana (neste caso, *e2* ou *e1*). O processo se repete até que o elemento médio retornado permaneça o mesmo. Quando isso

acontece, o elemento *E* é definido como centroide do cluster. Um raio *r* de clusterização é determinado pelo usuário. Assim, verifica-se, para cada elemento, quais outros elementos estão contidos dentro desse raio. Esse processo é chamado de cálculo de vizinhança.

A partir dessa clusterização, indexam-se apenas os elementos centroides. Os demais elementos no mesmo cluster são inseridos em uma lista de elementos associados à entrada do seu respectivo centroide. Uma vez indexados os elementos, é possível realizar buscas, ou seja, a partir de um formulário de entrada são encontrados outros semelhantes. A busca não compara elemento por elemento de todos os formulários. Ela compara somente os centroides, que são os elementos com maior afinidade com todos os demais elementos no cluster e que representam o cluster como um todo. Assim, a partir das comparações com os centroides, encontram-se os grupos de elementos semelhantes ao formulário de entrada.

O estudo apresenta experimentos realizados em um conjunto com aproximadamente 500 formulários Web, distribuídos em quatro domínios distintos. Os melhores resultados em relação à revocação (*recall)* foram atingidos no domínio de passagens aéreas, uma vez que os formulários são pequenos e possuem um conjunto mais homogêneo de atributos. Todavia, o domínio de filmes apresentou os piores valores, já que os formulários são maiores e não existe um padrão para os atributos. Em termos de precisão (*precission*), o domínio de carros obteve os melhores resultados devido ao grande número de formulários. Desta forma, a probabilidade de encontrar um subconjunto de atributos comuns é maior do que em outros domínios.

4.3 ESQUEMA MEDIADOR

A abordagem (LIANG, ZUO e REN, 2010) utiliza o termo visão integrada ou esquema mediador para designar uma interface, cujos elementos são obtidos a partir da integração de elementos de um ou mais esquemas de fontes de dados já especificados. Esta proposta emprega uma abordagem denominada LAV *(Local As View)* ou visão local. Isto requer que o esquema mediador seja especificado independente das fontes de dados. O que significa que, para adicionar uma nova fonte de dados, é necessário apenas adicionar uma nova declaração ao mapeamento. Uma visão simplificada do sistema pode ser observada na Figura 6.

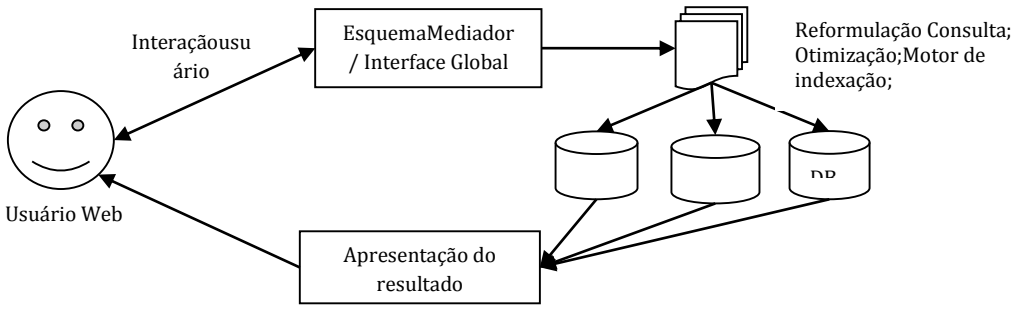

Figura 6 - Visão simplificada do esquema mediador, adaptado de (LIANG, ZUO e REN, 2010)

Na Ciência da Computação, ontologias representam o conhecimento adquirido a partir de dados semi-estruturados utilizando um conjunto de métodos, técnicas ou processos automáticos ou semi-automáticos. Neste sistema, o *Matching* entre o esquema mediador e as visões locais da *Deep Web* é gerado pela afirmação das relações dos grupos semânticos de acordo com uma ontologia, que é utilizada como apoio à determinação da importância do atributo.

Para a definição dos relacionamentos são utilizados os rótulos e os valores do formulário e para mensurar a semântica entre eles. Utiliza-se um

método que faz uso do dicionário WordNet (FELLBAUM, 2005). O dicionário representa a maior base de dados léxica da língua inglesa e relaciona verbos, advérbios, adjetivos e substantivos. Estes elementos estão agrupados em um conjunto de sinônimos cognitivos e relacionados sob a perspectiva léxica e semântica. O WordNet e outras taxonomias similares são vistas como uma estrutura de grafo. O relacionamento semântico, neste caso, pode ser obtido usando o tamanho do caminho entre os termos (nós do grafo). A Equação 5 representa o método de cálculo da distância semântica *S (w1, w2)* entre duas palavras diferentes *w1* e *w2*.

$$S(w_1, w_2) = \begin{cases} \dfrac{1}{\delta * len(w_1, w_2) + \varepsilon * turns(w_1, w_2)}, if \quad w_1 \\ and \quad w_2 \quad are \quad not \quad in \quad the \quad same \quad synset \\ 1, \qquad else \end{cases}$$

Equação 5 - Cálculo da distância semântica

Len(w1,w2) é o tamanho do menor caminho entre *w1* e *w2, turn(w1,w2)* o número de interações necessárias para o menor caminho entre as palavras. As variáveis δ e ε são relativas ao domínio. O algoritmo avalia duas cadeias de caracteres. Nesta avaliação, considera-se o número mínimo de operações necessárias para transformar uma cadeia de caracteres em outra. Por exemplo, dadas duas cadeias de caracteres "*computer*" e "*computing*", o resultado será igual a 5, pois para transformar a cadeia "*computer*" na cadeia "*computing*" são necessárias duas operações de remoção (respectivamente, dos caracteres "e" e "r") e três operações de inserção (dos caracteres "i", "n" e "g"). Para definir os grupos semânticos que serão estendidos pela ontologia, são aplicados 2 algoritmos. O primeiro é responsável por gerar uma matriz triangular superior,

denominada Matriz Semântica (SM), onde cada célula representa a distância semântica entre o par de atributos no domínio. O segundo algoritmo, chamado RUTMP, recebe como parâmetro a SM e gera os grupos candidatos a partir da sua inversa.

Finalmente, o relacionamento entre o esquema mediador e as visões locais é gerado pela afirmação das relações dos grupos semânticos através de hiponímia/hiperonímia, uma relação hierárquica fundamental na WordNet. Esta relação é inversa, assimétrica e transitiva, definida formalmente como: (i) *A* é um hipônimo de *B* se *A* é um tipo de *B, B* não é um tipo de *A*; (ii) *B* é um hiperônimo de *A*, se *A* é um hipônimo de *B*. Assim, hiperônimo é uma palavra que apresenta um significado mais abrangente do que o do seu hipônimo (de sentido mais especifico). Desta forma, é possível determinar, por exemplo, que “nome” é um hiperônimo de “título” e “título” é um hipônimo de “nome” no domínio de livros.

4.4 Teoria da evidência de Dempster-Shafer

A abordagem de (HONG, 2010) combina múltiplos *matchers* utilizando fundamentos da teoria da Evidência de Dempster-Shafer (*Dempster–Shafer theory of evidence*). A teoria da Evidência é um modelo matemático que permite a representação e a manipulação de informação com incerteza (SMETS e KENNES, 1994). A evidência diz respeito a um fato acontecido, ou uma informação disponível, a partir da qual se pode inferir outra informação, dependendo do grau de certeza (ou grau de crença) obtido sobre tal fato ou informação.

Quando existem evidências diversas, com diferentes graus de certeza, uma inferência final pode ser alcançada por meio de uma combinação consensual de todas as evidências disponíveis. Por exemplo, suponha que um paciente apresente manchas vermelhas pelo corpo e que isso seja sintoma de qualquer um dos seguintes problemas: alergia {a}, intoxicação {i}, sarampo {s},

rubéola {r}. Suponha agora que outro sintoma (evidência) considerado pelo médico aponte para um diagnóstico de “reação orgânica”, definida no exemplo como o conjunto {alergia, intoxicação}, ou então um diagnóstico de infecção, definida como o conjunto {sarampo, rubéola}. Se o médico observar uma evidência que confirma com um determinado grau o diagnóstico “reação orgânica”, ele irá atribuir uma quantidade de crença ao conjunto {alergia, intoxicação} proporcional ao grau observado de confirmação destes indícios. Uma nova evidência pode, por exemplo, excluir sarampo do diagnóstico. Uma evidência que desacredita ser sarampo pode ser tratada como uma evidência que confirme o resto do conjunto de hipóteses, ou seja, que confirma o conjunto {rubéola, alergia, intoxicação}.

Como fonte de coleta de evidências, esta abordagem utiliza quatro *matchers* individuais. Três deles analisam as características contidas no nome do atributo (semântica dos rótulos) e o quarto utiliza o tipo de dados como medida para o cálculo da similaridade. O primeiro *matcher* utiliza o dicionário WordNet, assim como a abordagem *LAV*, para computar a semelhança entre palavras com significados semanticamente equivalentes, como por exemplo “marca” e “fabricante”. O segundo *matcher* obtém a distância de edição (HALL, 1980) para mensurar a similaridade entre duas palavras. A partir de operações básicas de substituição, remoção e inserção é calculado o número de operações necessárias para transformar uma string em outra. Por exemplo, para transformar a *string* “PERDE” em “PODER” seriam necessárias às operações: PERDE → PRDE → PODE → PODER.

A medida de similaridade utilizada no terceiro *matcher* é a distância de Jaro (JARO, 1989). Ela se baseia na disposição e número total de caracteres comuns, conforme mostra a Equação 6.

$$Jaro(s,t) = \frac{1}{3} \times \left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s',t'}}{|s'|} \right)$$

Equação 6 - Função distância de Jaro

Dadas as *strings s = a1, ..., ak e t = b1, ..., bk,* um caractere ai em *s* é comum com t se existe um *bj = ai* em *t* tal que *i-H <= j <= i+H, onde H = min (*|s|, |*t|) /2*. Sejam *s' = a'1, ..., a'k* os caracteres em s que são comuns com *t* (na mesma ordem em que aparecem em *s*) e seja *t' = b'1, ..., b'k,* construído de forma análoga a *s'*. Uma transposição para *s'* e *t'* corresponde a uma posição *i* tal que *a'i ≠ b'i*. Assim, *Ts', t'* é definido como a metade do número de transposições para *s'* e *t'*. Por exemplo, dadas as *strings* “Thiago Silva” e “Tiago Silvio”, temos que: Jaro("Thiago Silva","Tiago Silvio") = (1/3) *(10/12+10/12+(10-0) /10) = 32/36 = 0.88888884.

O último dos *matchers* é responsável por mensurar a compatibilidade dos tipos de dados. Ele define se dois tipos de dados são iguais ou se algum deles está contido em outro, como por exemplo, uma data sendo comparada com valores de dia, mês e ano.

Ao final, a abordagem utiliza um algoritmo para resolver eventuais conflitos entre correspondências dos atributos com origens diversas (*Matchers* diferentes). Este algoritmo pode ser visualizado na Figura 7.

Algorithm 1 Resolving Conflicts

Input: A set of all the possible combinations of attribute correspondences for each source attribute $\Omega = \{C|C = \{< a_1, b'_1 > , < a_2, b'_2 > ... < a_m, b'_m >\}\}$,where $< a_i, b'_i > \in \{< a_i, b_{i1} > , < a_i, b_{i2} >, ..., < a_i, b_{ik} >\}$ (the top k correspondences of a_i)

Output: A collection of attribute correspondences with the maximum sum of the mass values of the correspondences for every source attribute

1: $Max \leftarrow 0; Best \leftarrow null.$
2: **for** each C$\in \Omega$ **do**
3: $Sum = \Sigma_{i=1}^{m} m(< a_i, b'_i >)$, where $m(< a_i, b'_i >)$ is the mass function value of $< a_i, b'_i >$
4: **if** $Sum > Max$ **then**
5: $Max \leftarrow Sum; Best \leftarrow C;$
6: **return** Best

Figura 7 - Algoritmo para resolução de conflitos

Como exemplo, suponha que o esquema de origem possui três atributos: *{autor, editora, Data de Publicação}*, e o esquema de destino tem três atributos*: {autor, palavras-chave, data de lançamento}*. Temos as melhores *k (k = 3)* correspondências de cada atributo de origem, como: *{m (<Autor; Autor>) = 0:88, m (<Autor;> null) = 0:11, m (<Autor; Palavras-chave>) = 0:01}, {m (<Editora; Autor>) = 0:47, m (<Editora;> null) = 0:40, m (<Editora; Palavras-chave>) = 0:13}, {m (<Data de Publicação; Data de Lançamento>) = 0:87, m (<Data de Publicação;> null) = 0:13, m (<Data de Publicação; Autor>) = 0:00}.* Os atributos "*Autor*" e "*Editora*" têm "*Autor*" como sua correspondência superior e, portanto, estão em conflito. Usando o Algoritmo 1, obtemos *{<Autor; Autor>, <Editora; null>, <Data de Publicação; Data de Lançamento>}*, os quais possuem os maiores valores associados.

Experimentos com um conjunto de aproximadamente 88 formulários distribuídos em 6 domínios obtiveram um valor de precisão de 96% no domínio de automóveis(*Auto*) e a sua menor percentagem foi no domínio de Empregos (*Jobs*) com 91,9%. Apesar dos bons resultados, esta abordagem possui a

limitação de comparar apenas duas *strings* por vez, devido ao sistema utilizar o recurso do dicionário semântico.

4.5 DYNABOT

Dynabot (DANIEL, CAVERLEE, *et al.*, 2005) é um sistema de descoberta de fontes de dados orientado à *Deep Web*. Ele possui uma arquitetura modular típica de coletores temáticos (*focused crawlers*), com ênfase em processos de similaridade, sondagem e ranqueamento de páginas. Entre as funcionalidades dos coletores temáticos estão localizar e coletar todos os tipos de páginas que estejam relacionadas a um determinado tópico de interesse.

A arquitetura deste sistema inclui componentes usuais de um *crawler*, como gerenciador de *URL*, módulos de interação de rede, armazenamento global, gerenciadores de dados e processadores de documentos. Seu diferencial está em um componente plugável denominado analisador semântico específico, o qual possui um Combinador de Classe de Serviço (*Service Class Matcher*) que analisa as fontes de dados candidatas. Uma visão da arquitetura do sistema pode ser vista na Figura 8.

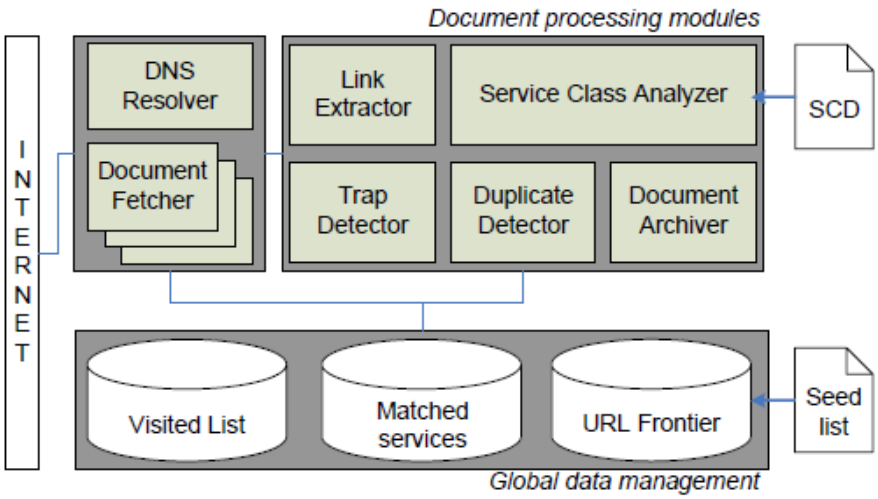

Figura 8– Arquitetura do Dynabot (DANIEL, CAVERLEE, *et al.*, 2005)

O ponto chave desta abordagem é o modelo de classe de serviço (*service class model*), que permite descobrir e relacionar fontes da *Deep Web*. A primeira definição deste modelo é a classe de serviço (*Service Class*) que fornece uma descrição geral dos dados de um conjunto de fontes da *Deep web* que apresentam a mesma funcionalidade. A segunda definição é a descrição da classe de serviço ou SCD (*Service Class Description*). Ela descreve de forma abstrata as funcionalidades mínimas que as fontes devem exportar para que sejam consideradas membro da classe de serviço.

Como exemplo, a abordagem considera membros de uma classe de serviço denominada nucleotídeos BLAST (*nucleotide blast*). Este serviço fornece operadores de busca (por similaridade) em bases de dados que possuem sequências genéticas, especialmente importantes para pesquisadores de bioinformática. As características de entrada relevantes para esta classe de serviço são *strings* que especificam sequências genéticas, uma base de dados contendo nucleotídeos e um mecanismo para submeter à consulta ao servidor apropriado. Esta descrição não contêm detalhes de implementação. Ela apenas define as funcionalidades mínimas para classificar o serviço como membro da classe de serviço nucleotídeos BLAST.

Uma descrição da classe de serviço é formalmente definida como *SCD = <T, G, P>,* onde *T* denota um conjunto de definição de tipos (t*ype definitions*), *G* representa um diagrama para controle de fluxo (*control flow graph*) e *P* um conjunto de modelos para sondagem (*probing template*). O primeiro componente apresenta os tipos de dados que serão utilizados como parâmetros de entrada e saída pelos membros da classe de serviço. Por exemplo, no caso dos nucleotídeos BLAST, este componente foi modelado em XML, conforme Figura 9.

```
<type name="DNASequence"
      type="string"
      pattern="[GCATgcat-]+" />

<type name="AlignmentSequenceFragment" >
    <element name="AlignmentName"
             type="string"
             pattern="[:alpha:]+:" />
    <element type="whitespace" />
    <element name="start-align-pos"
             type="integer" />
    <element type="whitespace" />
    <element name="Sequence"
             type="DNASequence" />
    <element type="whitespace" />
    <element name="end-align-pos"
             type="integer" />
</type>
```

Figura 9 - Nucleotídeos BLAST: Definição dos tipos (DANIEL, CAVERLEE, *et al.*, 2005)

Esta modelagem fornece basicamente duas possibilidades de tipos: os atômicos e os complexos. Na Figura 9, *DNASequence* é um exemplo de tipo atômico. Os tipos atômicos são elementos simples que possuem uma especificação base, como *strings* e inteiros, que ainda podem ser refinados utilizando um padrão de expressão regular, restringindo assim o intervalo de valores aceitáveis. A Figura 9 apresenta também a declaração *AlignmentSequenceFragment*, um tipo de dado complexo que é capaz de reconhecer um fragmento genético alinhado em uma sequência de nucleotídeos. Os tipos complexos podem ser definidos a partir de uma série de elementos, onde cada elemento de um tipo complexo pode ter uma referência a outro tipo atômico ou complexo. A *string* “280 TGGCAGGCGT CCT 292”, por exemplo, seria reconhecida como um tipo de dado complexo válido da classe de serviço Nucleotídeos BLAST.

O segundo componente, denominado diagrama de fluxo de controle, apresenta os possíveis caminhos de transições durante o fluxo de interação do serviço. Por exemplo, na Figura 10 está o diagrama da SDC Nucleotídeos BLAST.

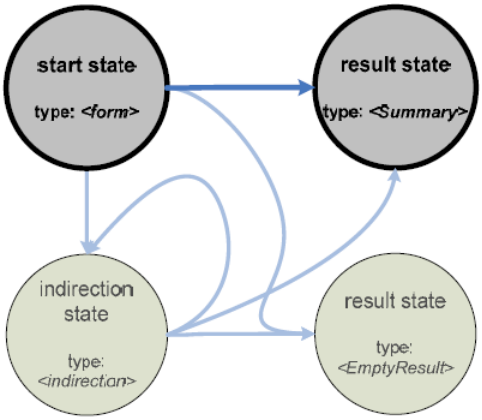

Figura 10 - Nucleotídeos BLAST: Diagrama de fluxo (DANIEL, CAVERLEE, *et al.*, 2005)

O diagrama possui quatro estados que são constituídos basicamente de um rótulo e um tipo de dados. Este diagrama tem um estado inicial único que define o tipo de entrada que um membro da classe deve conter (*Start State*). Para ser considerado um candidato Nucleotídeo BLAST, o serviço deve produzir uma ou mais transições até o estado de resultado (*Result State*), conforme indicado na Figura 10. Isto é importante, pois assim como no Nucleotídeo BLAST, muitos serviços produzem resultados intermediários conforme a pesquisa é executada. O sistema usa este diagrama para determinar que um candidato é membro de uma classe de serviço específica e para guiar a escolha do analisador semântico em análises adicionais. Assim, quando ele encontra um serviço BLAST relativo a proteínas, diferente de Nucleotídeos, por exemplo, ele pode usar o fluxo de controle para catalogar adequadamente este serviço, invocando os analisadores corretos.

O terceiro componente da descrição da classe de serviço é o conjunto de modelos de sondagem. Cada modelo contém um conjunto de parâmetros de entrada que pode ser usado para realizar o *Matching* de um candidato contra a descrição da classe de serviço e determinar se ele é uma instância da classe de serviço. Por exemplo, a Figura 11 mostra dois exemplos de formulários de pesquisa BLAST.

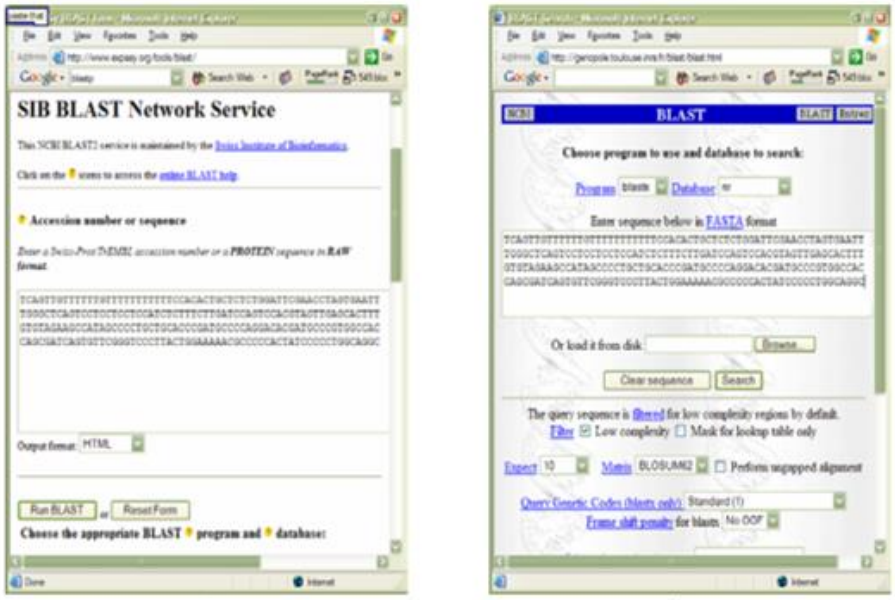

Figura 11 - Exemplos de formulários (a) simples e complexos (b) (DANIEL, CAVERLEE, *et al.*, 2005)

O exemplo da esquerda tem uma interface relativamente simples, incluindo um caixa de texto para a inserção de uma sequência de DNA e um único botão “Enviar”. O exemplo da direita é mais complexo, com parâmetros adicionais para seleção da base de dados, opções para filtragem de resultados da consulta, e assim por diante.

O objetivo é definir um modelo de sondagem para esta interface de busca que seja genérico o suficiente para corresponder a ambas as interfaces do

exemplo, mas não tão genérica a ponto de corresponder a formulários que não sejam BLAST. Desta forma, ao cruzar as informações da interface de consulta, o analisador de classe de serviço é chamado para identificar se a fonte é candidata à classe. Caso seja, esta é classificada como um membro da classe de serviço baseada na descrição de classe de serviço. Uma visão do funcionamento do sistema pode ser observada na Figura 12.

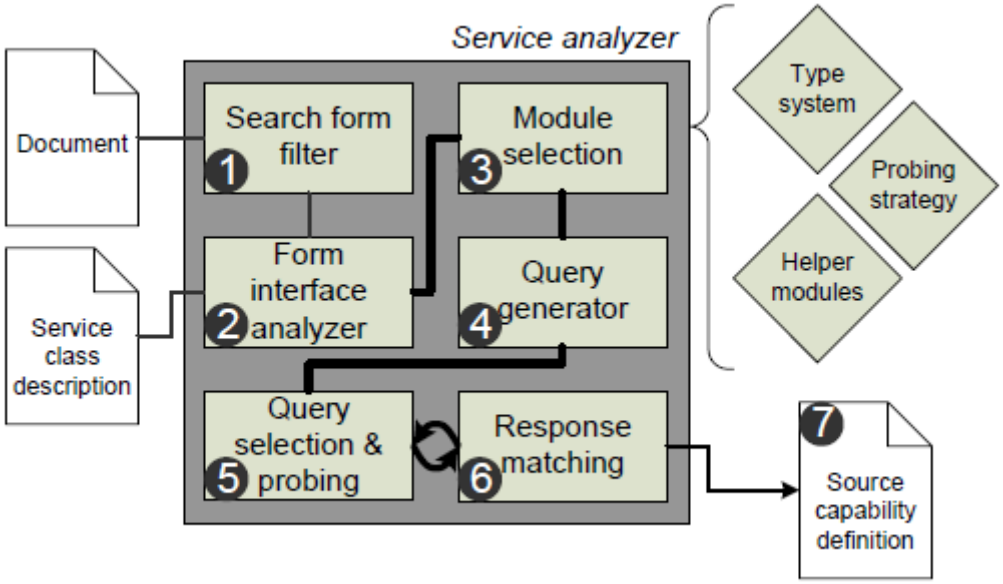

Figura 12 - Serviço de análise Dynabot (DANIEL, CAVERLEE, *et al.*, 2005)

O *crawler* inicia o processo de *Matching* com a passagem das URL candidatas ao módulo de análise. Neste módulo, a primeira tarefa é invocar o filtro de formulário (1), o qual garante que a URL possui ao menos um formulário como interface. O segundo passo (2) é extrair o conjunto de formulários, carregar a descrição da classe e seus módulos auxiliares (3). Nesta análise, cada formulário *F* é composto de um conjunto de atributos *F(P,B). "P"* representa os parâmetros *P(t,i,v)*, onde t é o tipo, como *Checkbox* ou *List*, *i* é o identificador e v é o valor do parâmetro. O formulário também contém um conjunto de botões

B, responsável por disparar ações para o servidor, como encaminhar a consulta ou limpar os formulários.

O ponto chave do processo é determinado nos passos (4), (5) e no *loop* presente no passo (6). O processo de sondagem consiste na manipulação dos formulários com o preenchimento automatizado dos parâmetros de acordo com um conjunto de modelos *E* definidos como parte da classe de serviço. Cada modelo possui um conjunto de argumentos *A = (r,t,v),* onde r indica se o parâmetro do exemplo é opcional ou obrigatório, t é o tipo do parâmetro e v o valor do parâmetro (4). Uma vez preenchido o formulário candidato, a consulta é encaminhada ao servidor (5) e seu retorno analisado conforme descrição da classe de serviço, caso seja reconhecido, ele é identificado como um resultado válido do processo, caso negativo, novas consultas são geradas até que todos os exemplos sejam utilizados (6).

4.6 ONTOMACH

A abordagem (BHATTACHARJEE, 2009) utiliza uma série de combinadores simples (*matchers*) com auxílio de ontologias. Ela é baseada no princípio de que um processo que utiliza diferentes *matchers*, através de uma função combinatória estática, leva a um *matching* de baixa qualidade. Desta forma, dados dois termos *t1* e *t2*, este sistema é capaz de selecionar dinamicamente um conjunto de *matchers* adequados para se efetuar a análise. Estabelecendo assim, de forma ordenada, uma melhora na qualidade do mapeamento entre os termos.

O sistema considera 5 *matchers* durante o processo de casamento: *EditDistanceMatcher* (EDM), *AbbreviationMatcher* (AM), *SynonymMatcher* (SynM), *ConstraintMatcher* (CM) e o *StructureMatcher* (SM). Durante o processo são utilizadas técnicas de normalização de texto, como por exemplo, a eliminação de espaços em brancos, remoção da pontuação e caracteres

especiais. No EDM é obtida a distância de edição, através do número de operações necessárias para que dois termos se tornem idênticos. O AM compara dois termos abreviando-os. Já o SynM é responsável por identificar se dois valores são sinônimos entre si. O SM é utilizado para estruturas aninhadas e mede a similaridade entre as sub-árvores enraizadas nos nós correspondentes aos valores. Quanto mais duas sub-árvores são parecidas, mais próximos os valores são. O CM verifica se dois termos são relativos, definindo uma matriz de distância, através de regras que envolvem o tipo de dado e suas características de chaves (primária ou estrangeira).

O processo de *matching* pode ser caracterizado em três fases distintas:

1. Seleção dos combinadores adequados;
2. Utilização dos combinadores selecionados para o cálculo da similaridade entre termos;
3. Aplicação de um algoritmo combinatório para obter o mapeamento final com base nos escores do combinadores*, threshold* e a matriz de distância.

Um combinador é selecionado com base em um conjunto de propriedades. Algumas propriedades dependem do tamanho do termo, enquanto outras dependem de uma função de distância presente no *ConstraintMatcher*.

$$
\begin{aligned}
p_1: &\quad \text{if } l_1 \geq avg_l \; s_1 = 1 \text{ else } s_1 = 0 \\
p_2: &\quad \text{if } l_2 \geq avg_l \; s_2 = 1 \text{ else } s_2 = 0 \\
p_3: &\quad s_3 = |l_1 - l_2| / max(l_1, l_2) \\
p_4: &\quad s_4 = 1 - s_3 \\
p_5: &\quad s_5 = 1 - (|card_1 - card_2| / max(card_1, card_2)) \\
p_6: &\quad s_6 = dist(t_1, t_2) \text{ from constraint matcher}
\end{aligned}
$$

Tabela 3 - Propriedades utilizadas para seleção do *Matcher* (BHATTACHARJEE, 2009)

A Tabela 3 mostra o conjunto de propriedades *p*, onde *l1; l2* são tamanhos dos termos *t1 e t2*; *avgl* é a média do tamanho de todos os termos, e *card1; card2* são cardinalidades de dois esquemas. Diferentes propriedades são utilizadas para selecionar diferentes *matchers*. Para a seleção do EDM e *SynM* é utilizado o conjunto EDM = *SynM* = *{p1; p2; p4},* para o AM é utilizada AM = *{p3}*, para PSM = *{p5; p6}* e para o PCM = *{p6}.* Assim o algoritmo pode ser definido conforme mostra a Figura 13.

Algorithm 1 Algorithm to select the best applicable matcher

Require: $\mathscr{M}$, where $\mathscr{M} \subseteq M$, $bestMatcher = \emptyset$, $bestScore = 0$
Returns: The most suitable matcher

```
1:  for each matcher m ∈ M do
2:      score = 0 ;
3:      for each property p_i ∈ P_m do
4:          score = score + s_i ;
5:      end for
6:      if score ≥ τ_m and score > bestScore then
7:          bestMatcher = m, bestScore = score;
8:      end if
9:  end for
10: return bestMatcher ;
```

Figura 13 – Algoritmo para seleção do *Matcher* (BHATTACHARJEE, 2009)

Uma vez que o combinador mais adequado tenha sido identificado, o próximo passo é aplicá-los aos termos, possibilitando gerar um novo par de termos modificados. Por exemplo, o *AbreviationMatcher* (AM) pode transformar termos em abreviações, ex: (‘*mobilenumber’*, ‘*cellnumber’*) em (‘*mobnum’*,‘*cellnum’*). De forma similar, o *SynM* pode transformar os termos em sinônimos, ex:(‘*mobilenum’*, ‘*cellnumber’*) em (‘*cellnum’*, ‘*cellnumber’*). EDM, SM,

e CM não efetuam transformações nos termos. Este processo continua até que se obtenha uma distância que seja menor que o *threshold* para todas as correspondências. Assim, é obtida uma matriz dimensional entre dois esquemas. Um segundo algoritmo é utilizado para encontrar a distância entre os termos *t1,t2,* vide Figura 14. Com as medidas adquiridas, a próxima tarefa é selecionar uma lista de correspondências.

Algorithm 2 Algorithm to calculate the distance matrix

Require: a pair of terms (t_1, t_2), and a set of simple matchers M

1: $score = 0$;
2: matcher m =get best matchers from M by algorithm 1;
3: terminate the algorithm and return *score* if infinite loop occurs;
4: get distance *dist* by applying matcher m to t_1, t_2;
5: $score = (score + 1 - dist)/2$;
6: **if** $score \leq \tau$ **then**
7: return *score* ;
8: **end if**
9: **if** $m =$ SynM or $m =$ AM **then**
10: transform (t_1, t_2) to (t_1', t_2') by applying m;
11: **else**
12: $(t_1', t_2') = (t_1, t_2)$
13: **end if**
14: call this algorithm with t_1', t_2', M ;
15: return *score* ;

Figura 14 - Algoritmo para Calcular Matriz de distância (BHATTACHARJEE, 2009)

O algoritmo de casamento estável (*stable marriage*) é utilizado para a seleção do resultado final do Ontomatch. Dada uma matriz de distância |*m x n*|, onde *(m ≤n),* o algoritmo pode selecionar *m* elementos correspondentes de *n* elementos de tal modo que, pelo menos, o primeiro par de elementos distantes

é selecionado e removido da matriz. Por fim, a correspondência com a distância ≤*T* é então filtrada a partir das correspondências de *m*, finalizando o processo.

As avaliações do sistema foram realizadas com dados do repositório UIUC[2].Enquanto em outros sistemas os resultados variam de forma imprevisível, no OntoMatch é provável a melhoria de precisão e revocação.

4.7 MOTOR DE BUSCA POR ENTIDADE

(CHANG, 2007) propõe o conceito de um motor de busca por entidades e sua codificação inicial com baseado em um framework que utiliza o modelo *Best-Effort*. O objetivo deste estudo é realizar integração na web em larga escala de informação de forma holística (o sistema como um todo determina como se comportam as partes). No contexto desta abordagem, uma entidade representa determinados tipos de dados, como por exemplo, um número de telefone, um nome ou uma data.

A Figura 15 apresenta a arquitetura geral do sistema. Entre os principais componentes do sistema está o *DataCollector* (Coletor de dados). O coletor funciona como um *crawler*, podendo recuperar páginas da web ou conteúdo especifico da *deep web*.

[2]http://metaquerier.cs.uiuc.edu/repository/.

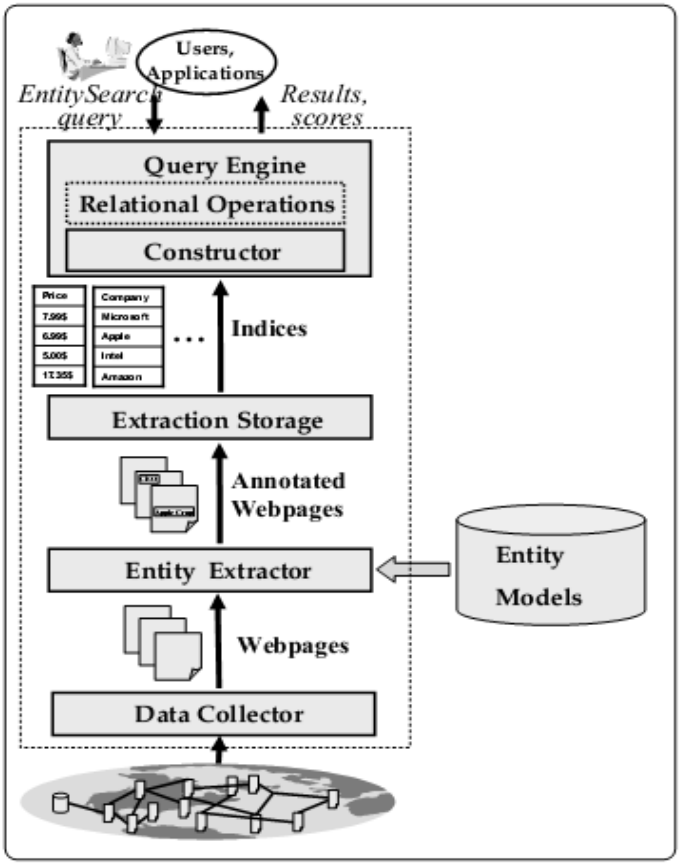

Figura 15 - Visão simplificada do sistema (CHANG, 2007).

O *Entity Extractor* (Extrator de Entidades) é responsável por descobrir a semântica do domínio. Neste componente, cada entidade é extraída de forma independente, através de um modelo que descreve como identificar instâncias de uma entidade nas páginas web. Já o *Query Engine* (Motor de consulta) é responsável pela execução do *Matching*. Ele também permite a construção de uma maneira natural de utilizar operadores relacionais. Muitas dessas operações podem ser realizadas a partir das tuplas, geradas como resultado. Como entrada, um usuário ou uma aplicação formula uma consulta que descreve quais os tipos de dados serão pesquisados. Pode-se simplesmente especificar quais as entidades alvos e quais palavras-chaves podem aparecer no contexto de uma resposta correta. O prefixo *"#"* é utilizado para distinguir as entidades e as

palavras-chaves. Nas consultas existem basicamente dois elementos, que podem ser visualizados na Figura 16.

```
Q1:    (amazon customer service #phone)
Q2:    (#professor #university #research='database')
Q3: ow(sigmod 2006 #pdf_file #ppt_file)
Q4:    (#title='hamlet' #image #price)
```

Figura 16 - Exemplos de consultas (CHANG, 2007).

O primeiro, denominado *Context pattern* (Contexto padrão), define como as entidades se "parecem". A consulta Q1, por exemplo, determina que o *#phone* irá aparecer com as palavras-chaves *(Amazon, customer, service*). O segundo componente, *Content restriction* (Restrição de conteúdo), restringe o tipo do conteúdo de pesquisa. Por exemplo, em *Q3 "#pdf file" e "#ppt file"* caracterizam onde as buscas irão acontecer, ou seja, a pesquisa irá acontecer em arquivos do tipo explicitado. Alguns exemplos de resultado para as consultas *Q1* e *Q3* podem ser vistos na Tabela 4.

rank	phone number	score	urls
1	800-201-7575	0.9	amazon.com/support.htm myblog.org/shopping
2	800-988-0886	0.8	Dell.com/supportors
3	800-342-5283	0.6	xyz.com
4	206-346-2992	0.2	hp.com
...	...	...	...

rank	PDF	PPT	score	urls
1	sigmod6.pdf	sigmod6.ppt	0.8	db.com,sigmod.com
2	surajit21.pdf	surajit21.ppt	0.7	ms.com
...	...	...	...	...

Tabela 4 - Exemplos de resultados (CHANG, 2007).

Cada resultado do sistema corresponde a uma tupla. Uma tupla pode conter uma ou mais instâncias, cada uma delas associada a um tipo de entidade. Por exemplo, (*David, david@hotmail.com*) é uma tupla do tipo *(#name, #email*). A ordem de classificação é caracteriza por um escore (valor). Este é determinado a partir das métricas presentes na Tabela 5, onde é calculada a frequência da tupla, frequência da instância e a incerteza da instância da entidade.

β	*formula*
tf	$f(e_1,\ldots,e_m)=\sum_{[\tilde{e}_1,\ldots,\tilde{e}_m]\in\alpha(x)}1$
dtf	$\sum_{[\tilde{e}_1,\ldots,\tilde{e}_m]\in\alpha(x)}\frac{1}{D([\tilde{e}_1,\ldots,\tilde{e}_m])}$, where $D([\tilde{e}_1,\ldots,\tilde{e}_m])=\frac{\sqrt{\sum_1^{m-1}(\tilde{e}_i.pos-\tilde{e}_{i+1}.pos)^2}}{m-1}$
mi	$\log\frac{f(e_1,\ldots,e_m)}{\prod_{i=1}^{m}f(e_i)}$
tscore	$\frac{f(e_1,\ldots,e_m)-E}{\sqrt{f(e_1,\ldots,e_m)}}$, where with D as corpus size: $E=f(e_1)\prod_{i=2}^{m}\frac{windowSize\cdot f(e_i)}{D}\prod_{j=1}^{l}\frac{windowSize\cdot f(k_j)}{D}$
cprod	$\prod_{i=1}^{m}conf(e_i)$

Tabela 5 - Métricas de similaridade (CHANG, 2007).

Na sua essência, o sistema assume a "integração em um sentido probabilístico". Dadas entidades independentes com probabilidades, a pesquisa encontra casamentos com outras entidades.

5 - COMPARATIVO

Este capítulo apresenta um comparativo dos trabalhos apresentados no capítulo 4. A

Tabela 6 exibe algumas características consideradas relevantes para a análise. Cada uma das características apresentadas na

Tabela 6 é analisada a seguir.

Trabalho	Abordagem	Elementos de entrada	Métricas	Intervenção Manual
PRUSM	Correlação de atributos	Rótulos e Valores	Distância do cosseno, medidas X/Y, STF	Não
WF-Sim	Grupos semânticos	Rótulos e Valores	SubSetSimmetric e ElementSim	Não
Esquema Mediador	Grupos semânticos, Ontologia e Dicionário Semântico (WordNet)	Rótulos e valores	Distância de edição e Matriz semântica	Não
Dempster–Shafer	Teoria da Evidência e Dicionário Semântico (WordNet)	Rótulos e tipos de dados	Distância de edição e Distância de Jaro	Não
Dynabot	Descrições de Classes de Serviço	Tipo de dados e atributos	Não disponível	Opcional
Ontomatch	Ontologia	Valores	Distância de edição e Matriz semântica	Não
Motor de busca por entidade	Holística	Tipos de dados	TF,DTF,MI, TSCORE,CPROD	Não

Tabela 6 - Comparativo das Abordagens

5.1 ABORDAGENS

Não existe unanimidade entre os autores, ou seja, observa-se estratégias de pesquisa bem diversificada. Cada abordagem apresenta suas vantagens e desvantagens, abrangendo análise sintática, semântica ou ambas durante a execução do *matching*.

A utilização de ontologia em não é plenamente aceita devido ao esforço necessário para construí-las e mantê-las, visto que elas são grandes e possuem uma vasta gama de conceitos relacionados ao domínio. Cabe observar a utilização de uma ontologia como apoio à determinação da importância de um atributo nas abordages (LIANG, ZUO e REN, 2010) e (BHATTACHARJEE, 2009)

Recursos externos, como o dicionário WordNet, também são utilizados como ferramenta auxiliar no processo de *matching*, como por exemplo, em Dempster–Shafer. Este dicionário possui informações relacionais entre os termos. Como consequência, essas abordagens estão geralmente fundamentadas em estruturas de grafos, utilizando propriedades de caminho (tamanho) para calcular o grau de similaridade semântica ou a distância semântica. Usualmente, esse tipo de abordagem se utiliza de relacionamentos para definir relações de subclasses e superclasses entre os conceitos presentes na hierarquia. Dentre os relacionamentos, citam-se a hiperonímia (generalização), hiponímia (especialização), meronímia (parte de), holônímia (inverso do “parte de”), antônimo (oposto) e o sinônimo (equivalência).

Há abordagens que confiam no fato de que a similaridade entre dois termos pode ser calculada através do grau de informações que eles têm em comum, ou seja, o grau de informações que elas compartilham. Nestes casos, um conjunto de classes de palavras hierárquicas pode ser extraído através de

distribuição e agrupamento, como na abordagem (FREIRE, NGUYEN e NGUYEN, 2010).

5.2 ELEMENTOS DE ENTRADA

Existe uma tendência na utilização das informações dos atributos (rótulos e valores) ao invés de somente os rótulos ou apenas valores. Abordagens que utilizam os atributos possuem a vantagem de ter mais informação para a confirmação de uma alta similaridade.

Algumas abordagens, como (FREIRE, NGUYEN e NGUYEN, 2010) e (BHATTACHARJEE, 2009) fazem uso de técnicas de pré-processamento para efetuar normalização linguística nos parâmetros de entrada. O processo de normalização é baseado na língua padrão definida pelo usuário para ser usada na concepção do esquema (inglês, português, francês, etc.) e consiste em eliminar alguns termos e caracteres que não trazem significado (como vírgulas, hífens, pontos, etc.), análises morfológicas, como supressão de gêneros e verificação de possíveis flexões e derivações da raiz de uma determinada *string*, além da eventual tradução para a língua padrão. Além disso, esta abordagem possui uma maior amplitude em termos de análise de dados para fins de determinação de similaridade, pois considera correlações de atributos, além das informações dos atributos.

5.3 MÉTRICAS

As funções apresentadas para cálculo de semelhança entre os termos têm como base a utilização de algum recurso. Entre os principais analisados estão o uso do espaço vetorial, a comparação entre *strings e* o uso da frequência de valores.

O espaço vetorial é empregado principalmente quando existe a necessidade de comparar propriedades de atributos, como o caso do WF-Sim. Para comparação entre *strings* é comumente utilizada a distância de

edição, onde operações como remoção e inserção de caracteres são utilizadas com o objetivo de converter uma *string* e outra e através das propriedades das transformações (número, tempo, etc) mensurar a semelhança.

A frequência de ocorrência de um termo pode também ser utilizada para agregar valor ao cálculo de similaridade. Este tipo de técnica pode ser vista no (FREIRE, NGUYEN e NGUYEN, 2010), através da *IDF* (*Inverse Document Frequency*).

As diferentes funções de similaridade resultam valores diferentes para as mesmas entradas. A escolha do melhor método vai depender do domínio em que o cálculo de similaridade será aplicado (LEVIN e HEUSER, 2010).

5.4 Intervenção Manual

Outro aspecto opcional, presente apenas no DynaBot, é a intervenção manual. Neste sistema, durante algumas etapas do processo, o usuário pode confirmar ou rejeitar correspondências que são automaticamente propostas pelo sistema. Apesar da intervenção manual não ser um aspecto positivo, ela pode ser uma alternativa plausível no sentido de confirmar a relevância dos resultados no processo de *Matching*.

Por outro lado, observa-se que grande parte das abordagens é totalmente automática. A vantagem é que as abordagens não atuam apenas como um sistema de apoio na determinação das similaridades, mas sim como um sistema que decide quais dados são similares, retirando o fardo desta decisão do usuário especialista.

6 - CONCLUSÃO

Processos de *Matching* possuem diversas aplicações, como a geração de visões centralizadas de esquemas de dados, a distribuição de consultas e a possibilidade de integrar resultados advindos de diversas fontes de dados. Isso torna possível para o usuário encontrar de forma mais eficaz os dados e informações relevantes para as suas necessidades.

A *deep web* é um conjunto de fontes de dados que cresce em ritmo acelerado. A quantidade de informações distribuídas na Web que ainda se encontram inacessíveis é muito grande, o que vêm motivar ainda mais trabalhos nessa área. Com o objetivo de contribuir com as pesquisas que necessitam da execução de *Matching* de dados na Web oculta, este trabalho apresenta uma revisão a respeito de algumas abordagens.

Não existe unanimidade entre os autores, ou seja, observa-se estratégias de pesquisa bem diversificada. Cada uma apresenta suas vantagens e desvantagens, abrangendo análise sintática, semântica ou ambas durante a execução do matching. Grande parte das abordagens é totalmente automática. É possível identificar uma tendência na utilização das informações dos atributos (rótulos e valores) ao invés de somente os rótulos ou apenas valores. Entre os principais recursos utilizados para cálculo da semelhança estão o uso do espaço vetorial, a comparação entre *strings* e o uso da frequência de valores.

Durante o estudo, foi possível identificar alguns pontos que estão em aberto e que podem ser alvo de trabalhos futuros. Primeiramente, uma exploração mais ampla das informações disponíveis nos *Web sites* é interessante a fim de encontrar semelhanças que contribuam para a relevância do *Matching*, como por exemplo, a nome da página e termos próximos aos formulários. Outro aspecto relevante é a descoberta do domínio na *Deep Web* ao qual a página pertence, visando auxiliar melhor a determinação da similaridade semântica.

Outro ponto é a adoção de técnicas de pré-processamento, pois não são todas as abordagens que efetuam uma limpeza dos dados, ou restringem os valores dos atributos dos formulários coletados a fim de melhorar a precisão do resultado. A utilização de técnicas de inteligência artificial poderia ser mais bem explorada para determinação de agrupamentos e a classificação dos atributos.

Outro aspecto pouco comentado nos trabalhos é o custo de processamento das técnicas utilizadas. Nenhuma das abordagens mensura o tempo necessário para execução da comparação e casamento entre os dados. Nesse aspecto, como sugestão de trabalhos futuros, um *benchmark* entre as abordagens contribuiria para se avaliar a viabilidade e a escalabilidade das mesmas.

Bibliografia

BERGAMAN, M. K. The Deep Web: Surfacing Hidden Value. [S.l.]: [s.n.]. 2007.

BERNSTEIN, P. A. . M. J. . R. E. Generic Schema Matching, Ten Years Later. Proceedings Very Large Data Base (VLDB). [S.l.]: [s.n.]. 2011.

BHATTACHARJEE, A. J. H. OntoMatch: A Monotonically Improving Schema Matching System for Autonomous Data Integration. IEEE International Conference on Information Reuse and Integration, 2009.

BIMBO, A. D. Visual Information Retrieval. San Francisco: Morgan Kaufmann, 1999.

BLATTMANN, U.; FACHIN, G.; RADOS, G. Recuperar a informação eletrônica pela internet. Revista da ACB: Biblioteconomia em Santa Catarina, v. 4,n. 4, p. 9-27, 1999., , v. 4, n. 4, p. 9-27, 1999. Disponivel em: <http://www.ced.ufsc.br/~ursula/papers/buscanet.html>. Acesso em: 08 nov. 2011.

CHANG, K. C.-C. A. C. T. Entity Search Engine: Towards Agile Best-Effort Information Integration over the Web. Proceedings of the Second Conference on Innovative Data Systems Research, 2007.

COHEN, W.; RAVIKUMAR, P.; FIENBERG, S. A Comparison of String Distance Metrics for Name-Matching Tasks. Workshop on Information Integration on the Web (IJCAI). [S.l.]: [s.n.]. 2003. p. 73-78.

DANIEL, R. et al. Proceedings of the 14th international conference on World Wide Web. Chiba: [s.n.]. 2005. p. 1174-1175.

DORNELES, C. . F. . G. R. E. M. R. . S. Approximate data instance matching: a survey. Knowledge and Information Systems, v. 27, 2011.

DORNELES, C. Measuring Similarity between Collection of Values. Proceedings of 6th ACM CIKM International Workshop on Web Information and Data Management. [S.l.]: [s.n.]. 2004. p. 56-63.

DORNELES, C. F. . G. R. . M. R. S. Approximate data instance matching: a survey. Knowledge and Information Systems, v. 27, n. 1, p. 1-21, 2011.

ELGARAMID, A. . K. . I. P. . G. E. V. V. . S. Duplicate Record Detection: A Survey. IEEE Transactions on Knowledge and Data Engineering, v. 19, p. 1-16, 2007.

FELLBAUM, C. WordNet and wordnets. In: BROWN, K. Encyclopedia of Language and Linguistics. [S.l.]: Oxford, 2005. p. 665-670. Disponivel em: <http://wordnet.princeton.edu/>. Acesso em: 26 Outubro 2012.

FREIRE, J.; BARBOSA, L. Siphoning Hidden web data throught keyword based interfaces. SBBD. [S.l.]: [s.n.]. 2004.

FREIRE, J.; BARBOSA, L. Searching for Hidden-Web Databases, Baltimore, 2005.

FREIRE, J.; NGUYEN, H.; NGUYEN, T. PruSM: A Prudent Schema Matching Approach for Web Forms. Information and Knowledge Management (CIKM). [S.l.]: [s.n.]. 2010. p. 1385-1388.

FREIRE, J.; NGUYEN, T.; NGUYEN, H. Learning to extract form labels. VLDB Endow. [S.l.]: [s.n.]. 2008.

GHANEM, T. M.; AREF, W. G. Databases deepen the web. IEE Computer. [S.l.]: [s.n.]. 2004.

GONÇALVES, R. et al. A Similarity Search Method for Web Forms. IADIS International Conference IADIS WWW/Internet. Rio de Janeiro: [s.n.]. 2011.

GOOGLE INC. Harnessing the Deep Web: Present and Future. [S.l.]. 2009.

HALL, P. . D. G. Approximate string matching. In: Computing Surveys. [S.l.]: [s.n.], 1980. p. 381–402.

HONG, J. . H. Z. . A. B. D. A. An evidential approach to query interface matching on the deepWeb. Journal Information Systems, 2010. 140-148.

INKROMI CORP. Web Surpasses One Billion Documents, 18 Janeiro 2000.

JARO, A. M. Advances in record-linkage methodology as applied to Matching. Journal of the American Statistical Association, Florida, 1989. 414–420.

JUANICÓ – ENVIRONMENTAL CONSULTANTS LTD. What is an information system on environmental information ?, 2006. Disponivel em: <http://www.juanico.co.il/main%20frame%20-%20english/issues/information%20systems.htm>. Acesso em: 10 nov. 2012.

LAWRENCE, S.; GILES, L. Searching the World Wide Web. Science, n. 280, p. 98-100, 1998.

LEVENSHTEIN, V. I. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady, v. 10, Feb 1966.

LEVIN, F. E. H. A. . C. Using Genetic Programming to Evaluate the Impact of Social Network Analysis in Author Name Disambiguation. AMW, 2010.

LEVIN, F.; HEUSER, A. . C. Using Genetic Programming to Evaluate the Impact of in Author Name Disambiguation. AMW. [S.l.]: [s.n.]. 2010.

LIANG, H.; ZUO, W.; REN, F. Describing the Semantic Relation of the Deep Web Query Interfaces Using Ontology Extended LAV. Journal of Software, v. 5, n. 1, p. 89-98, Janeiro 2010.

RAHM, E.; BERNSTEIN, A. A Survey of approaches to automatic schema matching. VLDN Journal, p. 334-350, 2001.

SMETS, P.; KENNES, R. The Transferable Belief Model. In: Classic Works of the Dempster-Shafer Theory of Belief Functions. [S.l.]: [s.n.], 1994. p. 693-736.

STASIU, R. K. Avaliação da qualidade de funções de similaridade no contexto de consultas por abrangência. [S.l.]: [s.n.]. 2007.

SU, W.; WANG, J.; LOCHOVSKY, F. Holistic query interface matching using parallel schema matching. Advances in Database Technology (EDBT). [S.l.]: [s.n.]. 2006. p. 77-94.

ANEXO 1 – ARTIGO

Um Comparativo entre Técnicas de Casamento (*Matching*) de Formulários na Deep Web

José Julio Resenes Neto, Ronaldo Santos dos Santos Mello

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina
Caixa Postal 476 – 88.040-900 – Florianópolis –SC – Brasil

jose.resenes@gmail.com, ronaldo@inf.ufsc.br

Abstract. *This paper presents a review of the literature about matching web forms approaches in the Deep Web context. Some comparison criteria are defined and used for analyzing the related approaches. Some hints for future research are also given.*

Resumo. *Este artigo apresenta uma revisão da literatura sobre matching de formulários Web no contexto da Deep Web. Critérios são definidos e utilizados para uma análise entre as abordagens relacionadas e, por fim, são sugeridos tópicos para pesquisas futuras.*

1. Introdução

Atualmente, fazer buscas na *Web* é como arrastar uma rede em toda a superfície de um grande oceano. Grande quantidade de informações pode ser apanhada na rede, ou seja, na superfície do oceano (dita *Surface Web*), mas há uma riqueza de informações que estão nas profundezas e, portanto, não pode ser alcançada facilmente (BERGAMAN, 2007).

O "oceano profundo" citado por Bergman é conhecido como "*Deep Web*", uma parcela da *Web* que compõe um grande número de dados e informações que não estão ao alcance dos mecanismos tradicionais de busca, como o Google[3]. Estes dados são normalmente acessados através de formulários *web* (*Web forms*). Estes bancos de dados ocultos contêm informações importantes para diversos serviços, de diferentes domínios, como aluguéis, vendas de veículos, reservas de passagens áreas e hotéis. Além disso, nesses bancos podemos encontrar uma grande quantidade de dados úteis para fins acadêmicos, como sequências de DNA e publicações científicas. Neste contexto, várias são as técnicas e ferramentas utilizadas para a localização e extração dos dados ocultos da *Deep Web*, como através de *Matchers, Crawlers* e diversas técnicas de similaridade. Uma vez alcançado os dados ocultos, o grande desafio é filtrar os vários formulários disponíveis, extrair e integrar informações realmente pertinentes ao usuário ou aplicação de forma eficaz.

Um grande desafio é que mesmo em um único domínio, as diferenças entre os formulários são enormes. Isto se deve principalmente a dois fatos. O primeiro é que nem sempre são utilizados os mesmos termos para se referenciar um objeto. Em um formulário de busca de veículos, um carro, por exemplo, pode ser chamado de "Carro" e em outro formulário de "Veículo", "Automóvel", etc. O segundo fato é que nem todos os formulários solicitam o mesmo tipo de informação. Neste caso, em um formulário pode ser necessário preencher o ano de fabricação de um veículo e em outro não. Se não bastassem os problemas mencionados, ainda existem empecilhos relacionados à própria usabilidade dos formulários, que variam muito. Em um formulário para a localização de veículos pode ser necessário digitar a marca de um veículo enquanto em outro pode ser

[1] www.google.com

feita a seleção em componente de lista. Ainda neste contexto, podemos ter problemas com questões como um determinado campo só aparecer no formulário quando outro for preenchido. Como exemplo, os modelos disponíveis de veículos só aparecerem após a marca ser selecionada. Até mesmo a forma como os formulários são "implementados" dificulta a exploração dos mesmos por mecanismos automatizados. Por exemplo, se um *Matcher* utiliza a descrição de um campo como fonte de informação para a sua identificação, pode ter problemas em formulários onde o campo para se escolher o veículo não é a descrição "Carro", mas sim a imagem de um carro.

Motivado pelas dificuldades apresentadas, este artigo visa revisar e comparar técnicas atualmente propostas e utilizadas para unificar formulários Web de um mesmo domínio. Assim, o objetivo deste trabalho é contribuir com as pesquisas que necessitam da execução de *Matching* de dados na *Deep Web*. Durante a revisão bibliográfica, identificaram-se alguns *surveys* sobre *matching* de esquemas de dados, como por exemplo, (DORNELES, 2011) e (BERNSTEIN, 2011). Entretanto este trabalho apresenta artigos recentes focados em formulários Web.

O artigo está organizado em mais 5 capítulos. O próximo capítulo apresenta uma visão geral de *Deep Web*. O capítulo 3 apresenta os fundamentos de *Matching* e Similaridade de dados. O capítulo 4 revisa e comenta as principais características de alguns trabalhos relacionados à problemática. O capitulo 5 apresenta um comparativo entre eles e, por fim, o Capítulo 6 apresenta a conclusão e sugestões para trabalhos futuros.

2. Deep Web

O termo *Deep Web*, também referenciado como *hidden* ou *invisible* web, é utilizado para definir o conteúdo oculto através de formulários HTML onde, para acessá-lo, o usuário necessita enviar uma requisição ao servidor com valores válidos de entrada. Como respostas às consultas submetidas ao banco de dados, páginas são construídas dinamicamente para apresentar o conteúdo (BERGAMAN, 2007). A Figura 1 mostra um exemplo de formulário Web para o domínio de carros. Através desta figura é possível ver que apenas alguns poucos atributos, com seus rótulos (nomes) e seus valores, do banco de dados escondido (aqueles usados para definir as buscas) são visíveis.

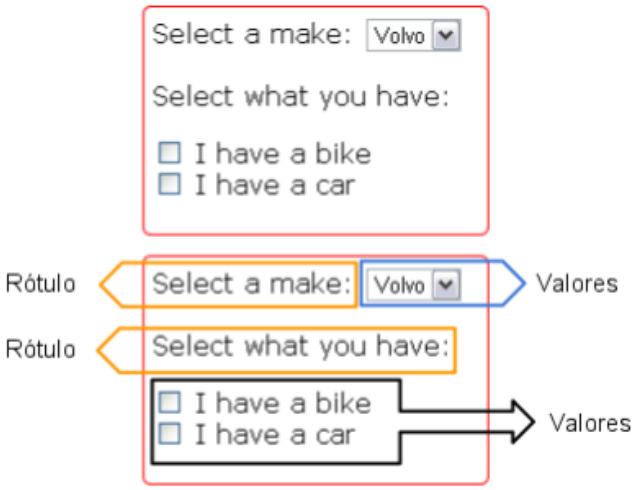

Figura 1. Formulário web e seus elementos

A *Deep web* representa uma grande parte dos dados estruturados na web, enquanto boa parte da informação na superfície da web é principalmente textos e HTML não estruturados (GOOGLE INC.). Ela já é reconhecida como uma lacuna significativa na abrangência dos motores de busca atuais, pois os *Crawlers* empregados nos motores contam com os *hiperlinks* para descobrir novas paginas na web e, normalmente, não têm a capacidade de realizar envios para tais formulários. O resultado disso são milhões de sites que nunca são alcançados por buscadores comuns, um conteúdo secreto e invisível que sempre esteve ali, mas que dificilmente um dia será visualizado por um usuário ou aplicação, conforme pode ser visto na Figura 2.

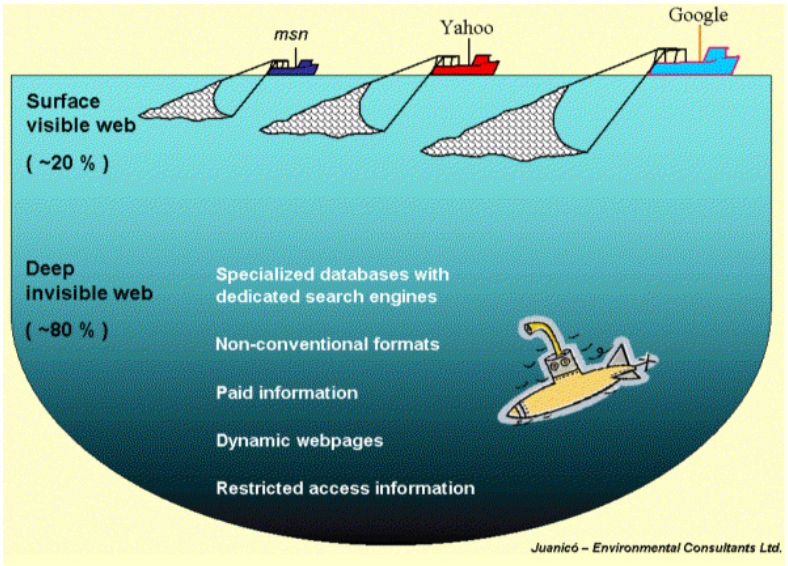

Figura 2. Representação da Deep Web (JUANICÓ – ENVIRONMENTAL CONSULTANTS LTD, 2006)

3. *Matching* e Similaridade de Dados

Matching é o processo de identificar a correlação semântica entre dois objetos, ou seja, identificar se dois objetos podem ser tratados como semelhantes. Esta é uma dificuldade recorrente em vários problemas associados à gerência de dados, como integração de dados, *data warehouse* e processamento semântico (RAHM e BERNSTEIN, 2001).

Juntamente com processos de mapeamentos de dados, processos de *Matching* formam os pilares da integração de dados. Desta forma, elaborar um processo automatizado de *Matching* é essencial para se obter um processo eficiente de integração. Entretanto, os desafios são vários, devido às diferenças encontradas nas definições dos dados. Essas diferenças podem ser: *(i)* Sintáticas: diferenças de linguagem utilizadas para representar os elementos; *(ii)* Estruturais: diferenças quanto ao tipo e a estrutura dos elementos; *(iii)* Diferenças de modelos de representação, que podem ser relacionais, orientados a objetos, etc; e *(iv)* Semânticas: situações onde a descrição de um dado pode ser representada por diferentes termos.

As dificuldades encontradas no processo de *Matching* podem ocorrer quando os objetos comparados estão em domínios distintos. Contudo, ainda que em um mesmo domínio, estas diferenças podem estar presentes, visto que os modelos de dados são desenvolvidos por diferentes pessoas. Assim, a primeira etapa para a integração de diferentes objetos é identificar as semelhanças entre eles. Uma vez identificadas, os objetos podem ser unificados em um terceiro objeto, que seja coerente com a representação dos dois objetos iniciais. Paralelo a este processo ou posteriormente a ele, podem ser utilizados mecanismos automáticos que permitam a tradução dos dados dos objetos originais para o objeto que representará a integração entre eles.

Outro importante conceito é o de Similaridade de Dados. Medir a similaridade ou a distância entre dois objetos distintos é o requisito principal para várias tarefas que necessitam realizar processos de *Matching* de dados. As técnicas têm uso frequente em áreas como mineração de dados e exploração de bases de conhecimento. Desta forma, a similaridade entre dois objetos nada mais é do que mostrar computacionalmente, através de um valor (grau de similaridade), o quanto dois objetos são semelhantes entre si. Por "Objeto" entende-se qualquer item de dado, como uma palavra, tupla, documento, estrutura de dados complexa, imagem, som, vídeo, etc.

Neste trabalho utiliza-se o termo de função de similaridade de forma genérica, referindo-se também às métricas de similaridade. De forma simplificada, medidas de similaridade podem ser definidas como: sejam x e y objetos de um determinado universo U, a medida de similaridade é a função $sim(x; y) \rightarrow [0; 1]$ (STASIU, 2007). Embora os valores sejam dependente da implementação da função de similaridade, tipicamente um resultado (*escore*) igual a 0 significa que os dois objetos analisados são totalmente diferentes, e um escore igual a 1 indica que são iguais. Existe um grande número de algoritmos, no entanto, todos eles têm em comum o fato de que o resultado dos cálculos de similaridade é representado por um único valor.

Existem várias funções de similaridade para *strings*, como Levenshtein (LEVENSHTEIN, 1966), Jaro-Winkler (JARO, 1989), N-Grams (ELGARAMID, 2007), entre outras. As diferentes funções de similaridade resultam valores diferentes para as mesmas entradas. A escolha do melhor método vai depender do domínio em que o cálculo de similaridade será aplicado (LEVIN, 2010).

4. *Matching* de Formulários na Deep Web

Com o objetivo de contribuir com as pesquisas que necessitam da execução de *Matching* de dados, este capítulo revisa e comenta as principais características de alguns trabalhos relacionados à problemática que pode ocorrer na *Deep Web*.

4.1. PRUSM

PRUSM (*Prudent Schema Matching*) é um sistema que combina várias fontes de informação para efetuar o *Matching* de forma "cautelosa", ou seja, com mínima propagação de erros (FREIRE, NGUYEN e NGUYEN, 2010).

PruSM recebe como entrada, através do módulo de Agregação, uma coleção de formulários previamente coletados de um repositório, sem necessidade de pré-processamento. Ele utiliza, então, técnicas de mineração de texto para a preparação dos dados. O módulo de agregação é responsável ainda por agrupar atributos similares, gerando como saída um conjunto de atributos frequentes *S1* para o módulo de Descoberta e também um conjunto de atributos infrequentes *S2* para o módulo de Crescimento. Dados dois atributos *(Ai, Aj)* recebidos como entrada pelo módulo de Descoberta, PrusM quantifica a semelhança entre eles por meio de três medidas: *(i)* similaridade do rótulo, *(ii)* similaridade do valor de domínio e *(iii)* correlação. Conforme pode ser visto na Tabela 1.

Tabela 1. Medidas Similaridade PRUSM

Similaridade do rótulo	$lsim(A_i, A_j) = cos(l_i, l_j)$
Similaridade do valor	$dsim(A_i, A_j) = cos(D_i, D_j)$
Medida de correlação	$X(A_p, A_q) = \begin{cases} 0 & \text{if p,q} \subset \text{form F} \\ \frac{(C_p - C_{qp})(C_q - C_{qp})}{(C_p + C_q)} & \text{otherwise} \end{cases}$ $Y(A_p, A_q) = \frac{C_{pq}}{min(C_p, C_q)}$

Inicialmente é calculada uma medida de importância do termo dentro do conjunto baseado na sua frequência, utilizando a distância do cosseno entre os termos dos vetores

de seus rótulos. Posteriormente, é realizada a correlação dos valores dos atributos utilizando-se as medidas *X/Y* (SU, WANG e LOCHOVSKY, 2006).

Os relacionamentos identificados, junto aos atributos de baixa frequência, são utilizados pelo módulo de Crescimento para obter relacionamentos adicionais. Neste módulo, os atributos com baixa frequência têm seus pesos recalculados através de um algoritmo chamado STF (*Singular Token Frequence*) (FREIRE, NGUYEN e NGUYEN, 2008). O STF aplica outro algoritmo conhecido como 1NN (*1-Nearest-Neighbor Clustering*) que agrupa o atributo raro ao seu vizinho mais próximo de maior frequência.

Este último passo gera um novo cluster de atributos considerados representativos. Por fim, incorpora-se este novo cluster ao cluster confiável, utilizando um algoritmo denominado HAC (*Hierarchical Agglomerative Clustering*). No HAC, esses atributos são representados em uma estrutura de árvore e são aglomerados aos mais próximos sucessivamente. Ocorre, então, a identificação de conjuntos de alta confiança, ou seja, clusters de atributos afins.

Uma avaliação experimental do PruSM demonstra que a abordagem é eficaz e capaz de corresponder com precisão a um grande número de formulários, principalmente em domínios mais heterogêneos, como por exemplo o de livros.

4.2. WF-SIM

WF-Sim é um método de busca para formulários Web, com base na similaridade de suas interfaces (GONÇALVES, D'AGOSTINI, et al., 2011). Ele é composto por 3 módulos principais: clusterização, indexação e busca. O objetivo do método é retornar um conjunto de formulários semelhantes a partir de uma entrada composta por um conjunto de atributos (e os valores possíveis, caso existam).

Primeiramente, visando definir uma base de formulários para posterior busca, dado um conjunto de formulários, são extraídas as propriedades dos elementos. Cada elemento representa um atributo contendo um rótulo e opcionalmente um conjunto de valores. Na etapa seguinte, técnicas de agrupamento são utilizadas para reduzir o número de entradas no índice de elementos com base em uma métrica de similaridade denominada *elementSim*. A partir do cálculo das similaridades dos rótulos e dos valores separadamente, é feita uma média ponderada desses valores, conforme Figura 3.

$$elementSim = labelSim(l_1, l_2) * labelWeight + valuesSim(vl_1, vl_2) * valuesWeight$$

Figura 3. Cálculo da similaridade do elemento

LabelSim representa a semelhança entre os rótulos utilizando a métrica TF-IDF (*Term Frequency Inverse Document Frequency*) (COHEN, RAVIKUMAR e FIENBERG, 2003). A TF-IDF é uma estatística que reflete numericamente a importância de uma palavra em relação a uma coleção. A semelhança entre os valores é denotada por *ValuesSim*, calculada através da métrica *SubSetSimmetric* (DORNELES, 2004). Ela obtém bons resultados quando se compara múltiplos valores, porque embora os valores sejam bastante heterogêneos, essencialmente eles são o mesmo conjunto (um dos

conjuntos é mais completo ou uma extensão do outro). Vale mencionar que, para elementos pertencentes a um mesmo formulário não é calculada a similaridade, visto que o objetivo é encontrar similaridades entre formulários diferentes.

O processo de agrupamento (clusterização) é realizado considerando um elemento por vez. Cada vetor é considerado como uma coordenada no espaço vetorial correspondente à linha do elemento na matriz e, para todos os vetores, a mediana (*Median Graph*) é calculada. Entretanto, como não é possível converter uma coordenada de vetor espacial para um texto, é definido como elemento médio aquele que possui a menor distância para o vetor de mediana. O processo se repete até que o elemento médio retornado permaneça o mesmo. Quando isso acontece, o elemento *E* é definido como centroide do cluster. Um raio *r* de clusterização é determinado pelo usuário. Assim, verifica-se, para cada elemento, quais outros elementos estão contidos dentro desse raio. Esse processo é chamado de cálculo de vizinhança.

A partir dessa clusterização, indexam-se apenas os elementos centroides. Os demais elementos no mesmo cluster são inseridos em uma lista de elementos associados à entrada do seu respectivo centroide. Uma vez indexados os elementos, é possível realizar buscas, ou seja, a partir de um formulário de entrada são encontrados outros semelhantes. A busca não compara elemento por elemento de todos os formulários. Ela compara somente os centroides, que são os elementos com maior afinidade com todos os demais elementos no cluster e que representam o cluster como um todo. Assim, a partir das comparações com os centroides, encontram-se os grupos de elementos semelhantes ao formulário de entrada.

O estudo apresenta experimentos realizados em um conjunto com aproximadamente 500 formulários *Web*, distribuídos em quatro domínios distintos. Os melhores resultados em relação à revocação (*recall*) foram atingidos no domínio de passagens aéreas, uma vez que os formulários são pequenos e possuem um conjunto mais homogêneo de atributos. Todavia, o domínio de filmes apresentou os piores valores, já que os formulários são maiores e não existe um padrão para os atributos. Em termos de precisão (*precission*), o domínio de carros obteve os melhores resultados devido ao grande número de formulários. Desta forma, a probabilidade de encontrar um subconjunto de atributos comuns é maior do que em outros domínios.

4.3. Esquema mediador

A abordagem (LIANG, ZUO e REN, 2010) utiliza o termo visão integrada ou esquema mediador para designar uma interface, cujos elementos são obtidos a partir da integração de elementos de um ou mais esquemas de fontes de dados já especificados. Esta proposta emprega uma abordagem denominada LAV (*Local As View*) ou visão local. Isto requer que o esquema mediador seja especificado independente das fontes de dados. O que significa que, para adicionar uma nova fonte de dados, é necessário apenas adicionar uma nova declaração ao mapeamento.

Na Ciência da Computação, ontologias representam o conhecimento adquirido a partir de dados semi-estruturados utilizando um conjunto de métodos, técnicas ou

processos automáticos ou semi-automáticos. Neste sistema, o *Matching* entre o esquema mediador e as visões locais da *Deep Web* é gerado pela afirmação das relações dos grupos semânticos de acordo com uma ontologia, que é utilizada como apoio à determinação da importância do atributo.

Para a definição dos relacionamentos são utilizados os rótulos e os valores do formulário e para mensurar a semântica entre eles. Utiliza-se um método que faz uso do dicionário WordNet (FELLBAUM, 2005). O dicionário representa a maior base de dados léxica da língua inglesa e relaciona verbos, advérbios, adjetivos e substantivos. Estes elementos estão agrupados em um conjunto de sinônimos cognitivos e relacionados sob a perspectiva léxica e semântica. O WordNet e outras taxonomias similares são vistas como uma estrutura de grafo. O relacionamento semântico, neste caso, pode ser obtido usando o tamanho do caminho entre os termos (nós do grafo). A Figura 4 representa o método de cálculo da distância semântica *S(w1, w2)* entre duas palavras diferentes *w1* e *w2*.

$$S(w_1, w_2) = \begin{cases} \dfrac{1}{\delta * len(w_1, w_2) + \varepsilon * turns(w_1, w_2)}, & if \quad w_1 \\ & and \quad w_2 \quad are \quad not \quad in \quad the \quad same \quad synset \\ 1, & else \end{cases}$$

Figura 4. Cálculo da distância semântica

O algoritmo avalia duas cadeias de caracteres. Nesta avaliação, considera-se o número mínimo de operações necessárias para transformar uma cadeia de caracteres em outra. Para definir os grupos semânticos que serão estendidos pela ontologia, são aplicados 2 algoritmos. O primeiro é responsável por gerar uma matriz triangular superior, denominada Matriz Semântica (SM), onde cada célula representa a distância semântica entre o par de atributos no domínio. O segundo algoritmo, chamado RUTMP, recebe como parâmetro a SM e gera os grupos candidatos a partir da sua inversa.

Finalmente, o relacionamento entre o esquema mediador e as visões locais é gerado pela afirmação das relações dos grupos semânticos através de hiponímia/hiperonímia, uma relação hierárquica fundamental na WordNet. Esta relação é inversa, assimétrica e transitiva, definida formalmente como: *(i) A* é um hipônimo de *B* se *A* é um tipo de *B*, *B* não é um tipo de *A*; *(ii) B* é um hiperônimo de *A*, se *A* é um hipônimo de *B*. Assim, hiperônimo é uma palavra que apresenta um significado mais abrangente do que o do seu hipônimo (de sentido mais especifico). Desta forma, é possível determinar, por exemplo, que "nome" é um hiperônimo de "título" e "título" é um hipônimo de "nome" no domínio de livros.

4.4. Teoria da evidência de Dempser-Shafer

A abordagem de (HONG, 2010) combina múltiplos *matchers* utilizando fundamentos da teoria da Evidência de Dempster-Shafer (*Dempster–Shafer theory of evidence*). A teoria

da Evidência é um modelo matemático que permite a representação e a manipulação de informação com incerteza (SMETS e KENNES, 1994). A evidência diz respeito a um fato acontecido, ou uma informação disponível, a partir da qual se pode inferir outra informação, dependendo do grau de certeza (ou grau de crença) obtido sobre tal fato ou informação.

Quando existem evidências diversas, com diferentes graus de certeza, uma inferência final pode ser alcançada por meio de uma combinação consensual de todas as evidências disponíveis. Por exemplo, suponha que um paciente apresente manchas vermelhas pelo corpo e que isso seja sintoma de qualquer um dos seguintes problemas: alergia *{a}*, intoxicação *{i},* sarampo *{s}*, rubéola *{r}*. Suponha agora que outro sintoma (evidência) considerado pelo médico aponte para um diagnóstico de "reação orgânica", definida no exemplo como o conjunto *{alergia, intoxicação},* ou então um diagnóstico de infecção, definida como o conjunto *{sarampo, rubéola}*. Se o médico observar uma evidência que confirma com um determinado grau o diagnóstico "reação orgânica", ele irá atribuir uma quantidade de crença ao conjunto *{alergia, intoxicação}* proporcional ao grau observado de confirmação destes indícios. Uma nova evidência pode, por exemplo, excluir sarampo do diagnóstico. Uma evidência que desacredita ser sarampo pode ser tratada como uma evidência que confirme o resto do conjunto de hipóteses, ou seja, que confirma o conjunto *{rubéola, alergia, intoxicação}*.

Como fonte de coleta de evidências, esta abordagem utiliza quatro *matchers* individuais. Três deles analisam as características contidas no nome do atributo (semântica dos rótulos) e o quarto utiliza o tipo de dados como medida para o cálculo da similaridade. O primeiro *matcher* utiliza o dicionário WordNet, assim como a abordagem LAV, para computar a semelhança entre palavras com significados semanticamente equivalentes, como por exemplo "marca" e "fabricante". O segundo *matcher* obtém a distância de edição (HALL, 1980) para mensurar a similaridade entre duas palavras. A partir de operações básicas de substituição, remoção e inserção é calculado o número de operações necessárias para transformar uma *string* em outra. A medida de similaridade utilizada no terceiro *matcher* é a distância de Jaro (JARO, 1989). Ela se baseia na disposição e número total de caracteres comuns, conforme mostra a Figura 5.

$$Jaro(s,t) = \frac{1}{3} \times \left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s',t'}}{|s'|} \right)$$

Figura 5. Função distância de Jaro

O último dos *matchers* é responsável por mensurar a compatibilidade dos tipos de dados. Ele define se dois tipos de dados são iguais ou se algum deles está contido em outro, como por exemplo, uma data sendo comparada com valores de dia, mês e ano. Ao final, a abordagem utiliza um algoritmo para resolver eventuais conflitos entre correspondências dos atributos com origens diversas (*Matchers* diferentes).

Experimentos com um conjunto de aproximadamente 88 formulários distribuídos em 6 domínios obtiveram um valor de precisão de 96% no domínio de automóveis *(Auto)* e a sua menor percentagem foi no domínio de Empregos *(Jobs)* com 91,9%. Apesar dos bons resultados, esta abordagem possui a limitação de comparar apenas duas *strings* por vez, devido ao sistema utilizar o recurso do dicionário semântico.

4.5.Dynabot

Dynabot (DANIEL, CAVERLEE, et al., 2005) é um sistema de descoberta de fontes de dados orientado à *Deep Web*. Ele possui uma arquitetura modular típica de coletores temáticos (*focused crawlers*), com ênfase em processos de similaridade, sondagem e ranqueamento de páginas. Entre as funcionalidades dos coletores temáticos estão localizar e coletar todos os tipos de páginas que estejam relacionadas a um determinado tópico de interesse.

A arquitetura deste sistema inclui componentes usuais de um *crawler*, como gerenciador de URL, módulos de interação de rede, armazenamento global, gerenciadores de dados e processadores de documentos. Seu diferencial está em um componente plugável denominado analisador semântico específico, o qual possui um Combinador de Classe de Serviço (*Service Class Matcher*) que analisa as fontes de dados candidatas.

O ponto chave desta abordagem é o modelo de classe de serviço *(service class model)*, que permite descobrir e relacionar fontes da *Deep Web*. A primeira definição deste modelo é a classe de serviço *(Service Class)* que fornece uma descrição geral dos dados de um conjunto de fontes da *Deep web* que apresentam a mesma funcionalidade. A segunda definição é a descrição da classe de serviço ou SCD *(Service Class Description)*. Ela descreve de forma abstrata as funcionalidades mínimas que as fontes devem exportar para que sejam consideradas membro da classe de serviço.

Uma descrição da classe de serviço é formalmente definida como SCD = $<T, G, P>$, onde T denota um conjunto de definição de tipos (*type definitions*), G representa um diagrama para controle de fluxo *(control flow graph*) e P um conjunto de modelos para sondagem (*probing template*). O primeiro componente apresenta os tipos de dados que serão utilizados como parâmetros de entrada e saída pelos membros da classe de serviço. Esta modelagem fornece basicamente duas possibilidades de tipos: os atômicos e os complexos. Os tipos atômicos são elementos simples que possuem uma especificação base, como *strings* e inteiros, que ainda podem ser refinados utilizando um padrão de expressão regular, restringindo assim o intervalo de valores aceitáveis. Os tipos complexos podem ser definidos a partir de uma série de elementos, onde cada elemento de um tipo complexo pode ter uma referência a outro tipo atômico ou complexo.

O segundo componente, denominado diagrama de fluxo de controle, apresenta os possíveis caminhos de transições durante o fluxo de interação do serviço. O sistema usa este diagrama para determinar que um candidato é membro de uma classe de serviço específica e para guiar a escolha do analisador semântico em análises adicionais.

O terceiro componente da descrição da classe de serviço é o conjunto de modelos de sondagem. Cada modelo contém um conjunto de parâmetros de entrada que pode ser usado para realizar o *Matching* de um candidato contra a descrição da classe de serviço e determinar se ele é uma instância da classe de serviço. Desta forma, ao cruzar as informações da interface de consulta, o analisador de classe de serviço é chamado para identificar se a fonte é candidata à classe. Caso seja, esta é classificada como um membro da classe de serviço baseada na descrição de classe de serviço.

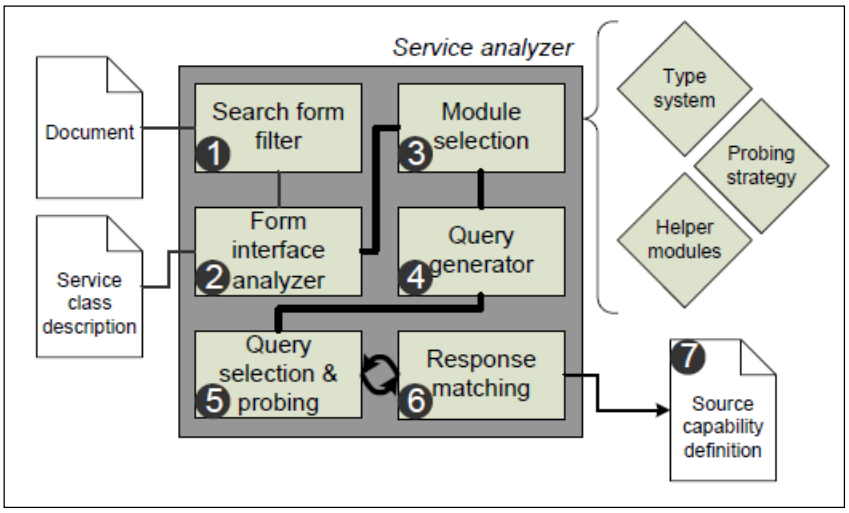

Figura 6. Serviço de análise Dynabot (DANIEL, CAVERLEE, et al., 2005)

O *crawler* inicia o processo de *Matching* com a passagem das URL candidatas ao módulo de análise, Figura 6. Neste módulo, a primeira tarefa é invocar o filtro de formulário (*1*), o qual garante que a URL possui ao menos um formulário como interface. O segundo passo (*2*) é extrair o conjunto de formulários, carregar a descrição da classe e seus módulos auxiliares (*3*). Nesta análise, cada formulário *F* é composto de um conjunto de atributos *F(P,B)*. "*P*" representa os parâmetros *P(t,i,v)*, onde *t* é o tipo, como *Checkbox* ou *List*, *i* é o identificador e *v* é o valor do parâmetro. O formulário também contém um conjunto de botões *B*, responsável por disparar ações para o servidor, como encaminhar a consulta ou limpar os formulários.

O ponto chave do processo é determinado nos passos (*4*), (*5*) e no *loop* presente no passo (*6*). O processo de sondagem consiste na manipulação dos formulários com o preenchimento automatizado dos parâmetros de acordo com um conjunto de modelos *E* definidos como parte da classe de serviço. Cada modelo possui um conjunto de argumentos *A = (r,t,v),* onde r indica se o parâmetro do exemplo é opcional ou obrigatório, *t* é o tipo do parâmetro e v o valor do parâmetro (*4*). Uma vez preenchido o formulário candidato, a consulta é encaminhada ao servidor (*5*) e seu retorno analisado conforme descrição da classe de serviço, caso seja reconhecido, ele é identificado como um resultado válido do processo, caso negativo, novas consultas são geradas até que todos os exemplos sejam utilizados (*6*).

4.6. OntoMatch

A abordagem (BHATTACHARJEE, 2009) utiliza uma série de combinadores simples (*matchers*) com auxílio de ontologias. Ela é baseada no princípio de que um processo que utiliza diferentes *matchers*, através de uma função combinatória estática, leva a um *matching* de baixa qualidade. Desta forma, dados dois termos *t1* e *t2*, este sistema é capaz de selecionar dinamicamente um conjunto de *matchers* adequados para se efetuar a análise. Estabelecendo assim, de forma ordenada, uma melhora na qualidade do mapeamento entre os termos.

O sistema considera *5 matchers* durante o processo de casamento: *EditDistanceMatcher* (EDM), *AbbreviationMatcher* (AM), *SynonymMatcher* (SynM), *ConstraintMatcher* (CM) e o *StructureMatcher* (SM). Durante o processo são utilizadas técnicas de normalização de texto, como por exemplo, a eliminação de espaços em brancos, remoção da pontuação e caracteres especiais. No EDM é obtida a distância de edição, através do número de operações necessárias para que dois termos se tornem idênticos. O AM compara dois termos abreviando-os. Já o *SynM* é responsável por identificar se dois valores são sinônimos entre si. O SM é utilizado para estruturas aninhadas e mede a similaridade entre as sub-árvores enraizadas nos nós correspondentes aos valores. Quanto mais duas sub-árvores são parecidas, mais próximos os valores são. O CM verifica se dois termos são relativos, definindo uma matriz de distância, através de regras que envolvem o tipo de dado e suas características de chaves (primária ou estrangeira).

O processo de *matching* pode ser caracterizado em três fases distintas: *(i)* Seleção dos combinadores adequados; *(ii)*. Utilização dos combinadores selecionados para o cálculo da similaridade entre termos; *(iii)*. Aplicação de um algoritmo combinatório para obter o mapeamento final com base nos escores do combinadores, *threshold* e a matriz de distância.

Um combinador é selecionado com base em um conjunto de propriedades. Algumas propriedades dependem do tamanho do termo, enquanto outras dependem de uma função de distância presente no *ConstraintMatcher*. Diferentes propriedades são utilizadas para selecionar diferentes *matchers*. Uma vez que o combinador mais adequado tenha sido identificado, o próximo passo é aplicá-los aos termos, possibilitando gerar um novo par de termos modificados. Por exemplo, o *AbreviationMatcher* (AM) pode transformar termos em abreviações, ex: *('mobilenumber', 'cellnumber')* em *('mobnum','cellnum')*. De forma similar, o SynM pode transformar os termos em sinônimos, ex:('*mobilenum*', '*cellnumber*') em ('*cellnum*', '*cellnumber*'). EDM, SM, e CM não efetuam transformações nos termos. Este processo continua até que se obtenha uma distância que seja menor que o *threshold* para todas as correspondências. Assim, é obtida uma matriz dimensional entre dois esquemas. Um segundo algoritmo é utilizado para encontrar a distância entre os termos *t1,t2*. Com as medidas adquiridas, a próxima tarefa é selecionar uma lista de correspondências.

O algoritmo de casamento estável *(stable marriage)* é utilizado para a seleção do resultado final do Ontomatch. Dada uma matriz de distância $|m \times n|$, onde $(m \leq n)$, o

algoritmo pode selecionar m elementos correspondentes de n elementos de tal modo que, pelo menos, o primeiro par de elementos distantes é selecionado e removido da matriz. Por fim, a correspondência com a distância $\leq T$ é então filtrada a partir das correspondências de *m,* finalizando o processo.

As avaliações do sistema foram realizadas com dados do repositório UIUC[4]. Enquanto em outros sistemas os resultados variam de forma imprevisível, no OntoMatch é provável a melhoria de precisão e revocação.

4.7. Motor de busca por Entidade

(CHANG, 2007) propõe o conceito de um motor de busca por entidades e sua codificação inicial com baseado em um framework que utiliza o modelo *Best-Effort*. O objetivo deste estudo é realizar integração na web em larga escala de informação de forma holística (o sistema como um todo determina como se comportam as partes). No contexto desta abordagem, uma entidade representa determinados tipos de dados, como por exemplo, um número de telefone, um nome ou uma data.

Entre os principais componentes do sistema está o *DataCollector* (Coletor de dados). O coletor funciona como um *crawler*, podendo recuperar páginas da web ou conteúdo especifico da *deep web*. O *Entity Extractor* (Extrator de Entidades) é responsável por descobrir a semântica do domínio. Neste componente, cada entidade é extraída de forma independente, através de um modelo que descreve como identificar instâncias de uma entidade nas páginas *web*. Já o *Query Engine* (Motor de consulta) é responsável pela execução do *Matching*. Ele também permite a construção de uma maneira natural de utilizar operadores relacionais. Muitas dessas operações podem ser realizadas a partir das tuplas, geradas como resultado.

Como entrada, um usuário ou uma aplicação formula uma consulta que descreve quais os tipos de dados serão pesquisados. Pode-se simplesmente especificar quais as entidades alvos e quais palavras-chaves podem aparecer no contexto de uma resposta correta. O prefixo "#" é utilizado para distinguir as entidades e as palavras-chaves. Nas consultas existem basicamente dois elementos. O primeiro, denominado *Context pattern* (Contexto padrão), define como as entidades se "parecem". O segundo componente, *Content restriction* (Restrição de conteúdo), restringe o tipo do conteúdo de pesquisa.

Cada resultado do sistema corresponde a uma tupla. Uma tupla pode conter uma ou mais instâncias, cada uma delas associada a um tipo de entidade. Por exemplo, (David, david@hotmail.com) é uma tupla do tipo (*#name, #email*). A ordem de classificação é caracteriza por um *escore* (valor). Este é determinado a partir das métricas presentes na Tabela 2, onde é calculada a frequência da tupla, frequência da instância e a incerteza da instância da entidade.

[2] http://metaquerier.cs.uiuc.edu/repository/.

Tabela 2. Métricas de similaridade (CHANG, 2007).

β	*formula*
tf	$f(e_1,\ldots,e_m)=\sum_{[\tilde{e}_1,\ldots,\tilde{e}_m]\in\alpha(x)}1$
dtf	$\sum_{[\tilde{e}_1,\ldots,\tilde{e}_m]\in\alpha(x)}\frac{1}{D([\tilde{e}_1,\ldots,\tilde{e}_m])}$, where $D([\tilde{e}_1,\ldots,\tilde{e}_m])=\frac{\sqrt{\sum_1^{m-1}(\tilde{e}_i.pos-\tilde{e}_{i+1}.pos)^2}}{m-1}$
mi	$\log\frac{f(e_1,\ldots,e_m)}{\prod_{i=1}^{m}f(e_i)}$
tscore	$\frac{f(e_1,\ldots,e_m)-E}{\sqrt{f(e_1,\ldots,e_m)}}$, where with D as corpus size: $E=f(e_1)\prod_{i=2}^{m}\frac{windowSize\cdot f(e_i)}{D}\prod_{j=1}^{l}\frac{windowSize\cdot f(k_j)}{D}$
cprod	$\prod_{i=1}^{m}conf(e_i)$

Na sua essência, o sistema assume a "integração em um sentido probabilístico". Dadas entidades independentes com probabilidades, a pesquisa encontra casamentos com outras entidades.

5. Comparativo

Este capítulo apresenta um comparativo dos trabalhos apresentados no capítulo 4. A Tabela 3 exibe algumas características consideradas relevantes para a análise.

Table 3. Comparativo das Abordagens

Trabalho	**Abordagem**	**Elementos de entrada**	**Métricas**	**Intervenção Manual**
PRUSM	Correlação de atributos	Rótulos e Valores	Distância do cosseno, medidas X/Y, STF	Não
WF-Sim	Grupos semânticos	Rótulos e Valores	SubSetSimmetric e ElementSim	Não
Esquema Mediador	Grupos semânticos, Ontologia e Dicionário Semântico (WordNet)	Rótulos e valores	Distância de edição e Matriz semântica	Não

Dempster–Shafer	Teoria da Evidência e Dicionário Semântico (WordNet)	Rótulos e tipos de dados	Distância de edição e Distância de Jaro	Não
Dynabot	Descrições de Classes de Serviço	Tipo de dados e atributos	Não disponível	Opcional
Ontomatch	Ontologia	Valores	Distância de edição e Matriz semântica	Não
Motor de busca por entidade	Holística	Tipos de dados	TF,DTF,MI, TSCORE,CPROD	Não

5.1 Abordagens

Não existe unanimidade entre os autores, ou seja, observa-se estratégias de pesquisa bem diversificada. Cada abordagem apresenta suas vantagens e desvantagens, abrangendo análise sintática, semântica ou ambas durante a execução do *matching*.

A utilização de ontologia em não é plenamente aceita devido ao esforço necessário para construí-las e mantê-las, visto que elas são grandes e possuem uma vasta gama de conceitos relacionados ao domínio. Cabe observar a utilização de uma ontologia como apoio à determinação da importância de um atributo nas abordages (LIANG, ZUO e REN, 2010) e (BHATTACHARJEE, 2009)

Recursos externos, como o dicionário WordNet, também são utilizados como ferramenta auxiliar no processo de *matching*, como por exemplo, em *Dempster–Shafer*. Este dicionário possui informações relacionais entre os termos. Como consequência, essas abordagens estão geralmente fundamentadas em estruturas de grafos, utilizando propriedades de caminho (tamanho) para calcular o grau de similaridade semântica ou a distância semântica. Usualmente, esse tipo de abordagem se utiliza de relacionamentos para definir relações de subclasses e superclasses entre os conceitos presentes na hierarquia. Dentre os relacionamentos, citam-se a hiperonímia (generalização), hiponímia (especialização), meronímia (parte de), holônimia (inverso do "parte de"), antônimo (oposto) e o sinônimo (equivalência).

Há abordagens que confiam no fato de que a similaridade entre dois termos pode ser calculada através do grau de informações que eles têm em comum, ou seja, o grau de informações que elas compartilham. Nestes casos, um conjunto de classes de palavras

hierárquicas pode ser extraído através de distribuição e agrupamento, como na abordagem (FREIRE, NGUYEN e NGUYEN, 2010).

5.2 Elementos de Entrada

Existe uma tendência na utilização das informações dos atributos (rótulos e valores) ao invés de somente os rótulos ou apenas valores. Abordagens que utilizam os atributos possuem a vantagem de ter mais informação para a confirmação de uma alta similaridade.

Algumas abordagens, como (FREIRE, NGUYEN e NGUYEN, 2010) e (BHATTACHARJEE, 2009) fazem uso de técnicas de pré-processamento para efetuar normalização linguística nos parâmetros de entrada. O processo de normalização é baseado na língua padrão definida pelo usuário para ser usada na concepção do esquema (inglês, português, francês, etc.) e consiste em eliminar alguns termos e caracteres que não trazem significado (como vírgulas, hífens, pontos, etc.), análises morfológicas, como supressão de gêneros e verificação de possíveis flexões e derivações da raiz de uma determinada *string*, além da eventual tradução para a língua padrão. Além disso, esta abordagem possui uma maior amplitude em termos de análise de dados para fins de determinação de similaridade, pois considera correlações de atributos, além das informações dos atributos.

5.3 Métricas

As funções apresentadas para cálculo de semelhança entre os termos têm como base a utilização de algum recurso. Entre os principais analisados estão o uso do espaço vetorial, a comparação entre *strings* e o uso da frequência de valores.

O espaço vetorial é empregado principalmente quando existe a necessidade de comparar propriedades de atributos, como o caso do WF-Sim. Para comparação entre *strings* é comumente utilizada a distância de edição, onde operações como remoção e inserção de caracteres são utilizadas com o objetivo de converter uma *string* e outra e através das propriedades das transformações (número, tempo, etc) mensurar a semelhança.

A frequência de ocorrência de um termo pode também ser utilizada para agregar valor ao cálculo de similaridade. Este tipo de técnica pode ser vista no (FREIRE, NGUYEN e NGUYEN, 2010), através da IDF (*Inverse Document Frequency*).

As diferentes funções de similaridade resultam valores diferentes para as mesmas entradas. A escolha do melhor método vai depender do domínio em que o cálculo de similaridade será aplicado (LEVIN e HEUSER, 2010).

5.4 Intervenção Manual

Outro aspecto opcional, presente apenas no DynaBot, é a intervenção manual. Neste sistema, durante algumas etapas do processo, o usuário pode confirmar ou rejeitar correspondências que são automaticamente propostas pelo sistema. Apesar da

intervenção manual não ser um aspecto positivo, ela pode ser uma alternativa plausível no sentido de confirmar a relevância dos resultados no processo de *Matching*.

Por outro lado, observa-se que grande parte das abordagens é totalmente automática. A vantagem é que as abordagens não atuam apenas como um sistema de apoio na determinação das similaridades, mas sim como um sistema que decide quais dados são similares, retirando o fardo desta decisão do usuário especialista.

7. Conclusão

Processos de *Matching* possuem diversas aplicações, como a geração de visões centralizadas de esquemas de dados, a distribuição de consultas e a possibilidade de integrar resultados advindos de diversas fontes de dados. Isso torna possível para o usuário encontrar de forma mais eficaz os dados e informações relevantes para as suas necessidades.

A *deep web* é um conjunto de fontes de dados que cresce em ritmo acelerado. A quantidade de informações distribuídas na Web que ainda se encontram inacessíveis é muito grande, o que vêm motivar ainda mais trabalhos nessa área. Com o objetivo de contribuir com as pesquisas que necessitam da execução de *Matching* de dados na Web oculta, este trabalho apresenta uma revisão a respeito de algumas abordagens.

Não existe unanimidade entre os autores, ou seja, observa-se estratégias de pesquisa bem diversificada. Cada uma apresenta suas vantagens e desvantagens, abrangendo análise sintática, semântica ou ambas durante a execução do *matching*. Grande parte das abordagens é totalmente automática. É possível identificar uma tendência na utilização das informações dos atributos (rótulos e valores) ao invés de somente os rótulos ou apenas valores. Entre os principais recursos utilizados para cálculo da semelhança estão o uso do espaço vetorial, a comparação entre *strings* e o uso da frequência de valores.

Durante o estudo, foi possível identificar alguns pontos que estão em aberto e que podem ser alvo de trabalhos futuros. Primeiramente, uma exploração mais ampla das informações disponíveis nos Web sites é interessante a fim de encontrar semelhanças que contribuam para a relevância do *Matching*, como por exemplo, a nome da página e termos próximos aos formulários. Outro aspecto relevante é a descoberta do domínio na *Deep Web* ao qual a página pertence, visando auxiliar melhor a determinação da similaridade semântica.

Outro ponto é a adoção de técnicas de pré-processamento, pois não são todas as abordagens que efetuam uma limpeza dos dados, ou restringem os valores dos atributos dos formulários coletados a fim de melhorar a precisão do resultado. A utilização de técnicas de inteligência artificial poderia ser mais bem explorada para determinação de agrupamentos e a classificação dos atributos.

Outro aspecto pouco comentado nos trabalhos é o custo de processamento das técnicas utilizadas. Nenhuma das abordagens mensura o tempo necessário para execução da comparação e casamento entre os dados. Nesse aspecto, como sugestão de trabalhos

futuros, um benchmark entre as abordagens contribuiria para se avaliar a viabilidade e a escalabilidade das mesmas.

Referências:

BERGAMAN, M. K. The Deep Web: Surfacing Hidden Value. [S.l.]: [s.n.]. 2007.

BERNSTEIN, P. A. . M. J. . R. E. Generic Schema Matching, Ten Years Later. Proceedings Very Large Data Base (VLDB). [S.l.]: [s.n.]. 2011.

BHATTACHARJEE, A. J. H. OntoMatch: A Monotonically Improving Schema Matching System for Autonomous Data Integration. IEEE International Conference on Information Reuse and Integration, 2009.

BIMBO, A. D. Visual Information Retrieval. San Francisco: Morgan Kaufmann, 1999.

BLATTMANN, U.; FACHIN, G.; RADOS, G. Recuperar a informação eletrônica pela internet. Revista da ACB: Biblioteconomia em Santa Catarina, v. 4,n. 4, p. 9-27, 1999., , v. 4, n. 4, p. 9-27, 1999. Disponivel em: <http://www.ced.ufsc.br/~ursula/papers/buscanet.html>. Acesso em: 08 nov. 2011.

CHANG, K. C.-C. A. C. T. Entity Search Engine: Towards Agile Best-Effort Information Integration over the Web. Proceedings of the Second Conference on Innovative Data Systems Research, 2007.

COHEN, W.; RAVIKUMAR, P.; FIENBERG, S. A Comparison of String Distance Metrics for Name-Matching Tasks. Workshop on Information Integration on the Web (IJCAI). [S.l.]: [s.n.]. 2003. p. 73-78.

DANIEL, R. et al. Proceedings of the 14th international conference on World Wide Web. Chiba: [s.n.]. 2005. p. 1174-1175.

DORNELES, C. . F. . G. R. E. M. R. . S. Approximate data instance matching: a survey. Knowledge and Information Systems, v. 27, 2011.

DORNELES, C. Measuring Similarity between Collection of Values. Proceedings of 6th ACM CIKM International Workshop on Web Information and Data Management. [S.l.]: [s.n.]. 2004. p. 56-63.

DORNELES, C. F. . G. R. . M. R. S. Approximate data instance matching: a survey. Knowledge and Information Systems, v. 27, n. 1, p. 1-21, 2011.

ELGARAMID, A. . K. . I. P. . G. E. V. V. . S. Duplicate Record Detection: A Survey. IEEE Transactions on Knowledge and Data Engineering, v. 19, p. 1-16, 2007.

FELLBAUM, C. WordNet and wordnets. In: BROWN, K. Encyclopedia of Language and Linguistics. [S.l.]: Oxford, 2005. p. 665-670. Disponível em: <http://wordnet.princeton.edu/>. Acesso em: 26 Outubro 2012.

FREIRE, J.; BARBOSA, L. Siphoning Hidden web data throught keyword based interfaces. SBBD. [S.l.]: [s.n.]. 2004.

FREIRE, J.; BARBOSA, L. Searching for Hidden-Web Databases, Baltimore, 2005.

FREIRE, J.; NGUYEN, H.; NGUYEN, T. PruSM: A Prudent Schema Matching Approach for Web Forms. Information and Knowledge Management (CIKM). [S.l.]: [s.n.]. 2010. p. 1385-1388.

FREIRE, J.; NGUYEN, T.; NGUYEN, H. Learning to extract form labels. VLDB Endow. [S.l.]: [s.n.]. 2008.

GHANEM, T. M.; AREF, W. G. Databases deepen the web. IEE Computer. [S.l.]: [s.n.]. 2004.

GONÇALVES, R. et al. A Similarity Search Method for Web Forms. IADIS International Conference IADIS WWW/Internet. Rio de Janeiro: [s.n.]. 2011.

GOOGLE INC. Harnessing the Deep Web: Present and Future. [S.l.]. 2009.

HALL, P. . D. G. Approximate string matching. In: Computing Surveys. [S.l.]: [s.n.], 1980. p. 381–402.

HONG, J. . H. Z. . A. B. D. A. An evidential approach to query interface matching on the deepWeb. Journal Information Systems, 2010. 140-148.

INKROMI CORP. Web Surpasses One Billion Documents, 18 Janeiro 2000.

JARO, A. M. Advances in record-linkage methodology as applied to Matching. Journal of the American Statistical Association, Florida, 1989. 414–420.

JUANICÓ – ENVIRONMENTAL CONSULTANTS LTD. What is an information system on environmental information ?, 2006. Disponível em: <http://www.juanico.co.il/main%20frame%20-%20english/issues/information%20systems.htm>. Acesso em: 10 nov. 2012.

LAWRENCE, S.; GILES, L. Searching the World Wide Web. Science, n. 280, p. 98-100, 1998.

LEVENSHTEIN, V. I. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady, v. 10, Feb 1966.

LEVIN, F. E. H. A. . C. Using Genetic Programming to Evaluate the Impact of Social Network Analysis in Author Name Disambiguation. AMW, 2010.

LEVIN, F.; HEUSER, A. . C. Using Genetic Programming to Evaluate the Impact of in Author Name Disambiguation. AMW. [S.l.]: [s.n.]. 2010.

LIANG, H.; ZUO, W.; REN, F. Describing the Semantic Relation of the Deep Web Query Interfaces Using Ontology Extended LAV. Journal of Software, v. 5, n. 1, p. 89-98, Janeiro 2010.

RAHM, E.; BERNSTEIN, A. A Survey of approaches to automatic schema matching. VLDN Journal, p. 334-350, 2001.

SMETS, P.; KENNES, R. The Transferable Belief Model. In: Classic Works of the Dempster-Shafer Theory of Belief Functions. [S.l.]: [s.n.], 1994. p. 693-736.

STASIU, R. K. Avaliação da qualidade de funções de similaridade no contexto de consultas por abrangência. [S.l.]: [s.n.]. 2007.

SU, W.; WANG, J.; LOCHOVSKY, F. Holistic query interface matching using parallel schema matching. Advances in Database Technology (EDBT). [S.l.]: [s.n.]. 2006. p. 77-94.