UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GUSTAVO ZANINI KANTORSKI

Preenchimento Automático de Formulários na Web Oculta

Tese apresentada como requisito parcial
para a obtenção do grau de
Doutor em Ciência da Computação

Orientador: Prof. Dr. Carlos Alberto Heuser
Co-orientador: Prof. Dra. Viviane Pereira
Moreira

Porto Alegre
novembro de 2014

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Kantorski, Gustavo Zanini

Preenchimento Automático de Formulários na Web Oculta / Gustavo Zanini Kantorski. – Porto Alegre: PPGC da UFRGS, 2014.

87 f.: il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2014. Orientador: Carlos Alberto Heuser; Co-orientador: Viviane Pereira Moreira.

1. Preenchimento de formulários web. 2. Web oculta. 3. Web profunda. 4. Crawling. I. Heuser, Carlos Alberto. II. Moreira, Viviane Pereira. III. Título.

“The Internet is the first thing that humanity has built that humanity doesn’t understand, the largest experiment in anarchy that we have ever had.”

— Eric Schmidt

AGRADECIMENTOS

Gostaria de agradecer a minha família pelo incentivo e investimento na minha formação.

Aos meus colegas de mestrado e doutorado. Eles foram especiais, seja na contribuição científica, seja nos momentos de lazer. Em especial, o meu amigo Tiago Moraes, pela pesquisa compartilhada, contribuições científicas e discussões sobre a pesquisa.

Aos meus professores, em especial a Viviane Moreira e Carlos Heuser, pela orientação e experiência compartilhada.

Ao CPD da UFSM, que concedeu o meu afastamento para realização do curso e à CAPES, pelo apoio financeiro durante o curso.

Preenchimento Automático de Formulários na Web Oculta

RESUMO

Muitas informações disponíveis na Web estão armazenadas em bancos de dados *on-line* e são acessíveis somente após um usuário enviar uma consulta por meio de uma interface de busca. Essas informações estão localizadas em uma parte da Web conhecida como Web Oculta ou Web Profunda e, geralmente, são inacessíveis por máquinas de busca tradicionais. Uma vez que a forma de acessar os dados na Web Oculta se dá por intermédio de submissões de consultas, muitos trabalhos têm focado em como preencher automaticamente campos de formulários. Esta tese apresenta uma metodologia para o preenchimento de formulários na Web Oculta. Além disso, descreve uma categorização das técnicas de preenchimento de formulários existentes no estado da arte de coleta na Web Oculta, produzindo uma análise comparativa entre elas. A solução proposta descreve um método automático para seleção de valores para campos de formulários combinando heurísticas e técnicas de aprendizagem de máquina. Experimentos foram realizados em formulários reais da Web, de vários domínios, e os resultados indicam que a abordagem proposta apresenta desempenho comparável aos obtidos pelas técnicas do estado da arte, sendo inclusive significativamente diferente com base em avaliação estatística.

Palavras-chave: Preenchimento de formulários web, web oculta, web profunda, crawling.

Automatically Filling in Hidden Web Forms

ABSTRACT

A large portion of the information on the Web is stored inside online databases. Such information is accessible only after the users submit a query through a search interface. The Web portion in which that information is located is called Hidden Web or Deep Web, and generally this part is inaccessible by traditional search engines crawlers. Since the only way to access the Hidden Web pages is through the query submissions, many works have focused on how to fill in form fields automatically, aiming at enhancing the amount of distinct information hidden behind Web forms. This thesis presents an automatic solution to value selection for fields in Web forms. The solution combines heuristics and machine learning techniques for improving the selection of values. Furthermore, this proposal also describes a categorization of form filling techniques and a comparative analysis between works in the state of the art. Experiments were conducted on real Web sites and the results indicated that our approach significantly outperforms a baseline method in terms of coverage without additional computational cost.

Keywords: Crawling, Deep Web, Filling Web Forms, Hidden Web.

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
AVG	Attribute Value Graph
BDO	Banco de Dados *On-line*
CEP	Código de Endereçamento Postal
CF	Collection Frequency
CF-IDF	Collection Frequency-Inverse Document Frequency
CFS	Correlation-based Feature Selection
DAV	Distinct Attribute Value
DB	Database
DF	Document Frequency
DOM	Document Object Model
DT	Domain Statistic Table
EE	Execution Efficiency
FTF	Filling Text Fields
HIWE	Hidden Web Exposer
HR	Harvest Rate
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDF	Inverse Document Frequency
ISIT	Incremental Search for Informative query Templates
kNN	k Nearest Neighbor
LVS	Label Values Set
MDR	Mining Data Records
MLP	MultiLayer Perceptron
OR	Overlapping Rate
RIDF	Residual IDF

RL	Reinforcement Learning
smartFTF	Smart Filling Text Fields
TF	Term Frequency
TF-IDF	Term Frequency - Inverse Document Frequency
TIP	Template Instance Pruning
TITP	Template Instance and Template Pruning
URL	Uniform Resource Locator

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Este capítulo descreve as motivações, os objetivos e uma visão geral das contribuições desta tese. O problema de pesquisa investigado é apresentado antes de discutir em detalhes as contribuições e suas consequências.

1.1 Motivação

A busca de informações na Web faz parte do cotidiano de, praticamente, todas as pessoas que utilizam a Internet, as quais buscam informações sobre assuntos variados de seu interesse. Segundo Lawrence and Giles (1998) 85% dos usuários da Web utilizam máquinas de busca para localizar informação. A Internet e a Web transformaram a sociedade, e as máquinas de busca têm importante participação nesse processo.

Atualmente as máquinas de busca incluem somente uma porção do conteúdo da Web em seus índices. Existe uma grande parte da Web que não está indexada e é desconhecida pelos motores de busca. Isso significa que uma pesquisa realizada por um usuário em uma máquina de busca retornará somente as páginas indexadas pelo buscador, enquanto que os usuários tratam tais resultados como um conjunto completo (ou quase completo) das páginas relevantes sobre o assunto pesquisado.

A porção da Web que não está indexada é conhecida como Web Invisível (SHERMAN; PRICE, 2001) e inclui toda parte da Web, estática e dinâmica, pública e privada. A Web Invisível consiste de documentos da Web que são fracamente indexados ou não são indexados por máquinas de busca. Um Web *crawler*, uma ferramenta automática usada pelas máquinas de busca para coletar páginas na Web, funciona por meio da busca de páginas na Web partindo de uma lista de URLs predefinidas, denominadas sementes. A busca é realizada nas páginas e os *hiperlinks* das páginas visitadas são adicionados a uma lista de URLs a visitar. Assim, as URLs são recursivamente visitadas de acordo com as regras definidas no *crawler*. Páginas que não são adicionadas na lista e que não estão entre as sementes nunca serão visitadas e, consequentemente, não serão indexadas. Além disso, páginas protegidas, por exemplo, por *login* e senha, e páginas dinâmicas, baseadas em formulários de consulta, também não são indexadas.

Essa parte da Web Invisível que está escondida por trás de formulários de consulta, na maioria das vezes, tem seu conteúdo gerado dinamicamente. Esse conteúdo é formado por páginas geradas como resultado de consultas enviadas por meio de interfaces de busca para bancos de dados *on-line* na Web. Essa parte da Web, incluída na Web Invisível, é conhecida como Web Oculta (FLORESCU; LEVY; MENDELZON, 1998) ou Web Profunda (BERGMAN, 2001).

A Figura 1.1 ilustra uma visão geral da Web. Os dados na Web podem ser divididos ortogonalmente de duas maneiras: quanto à forma de acesso, público ou privado e quanto

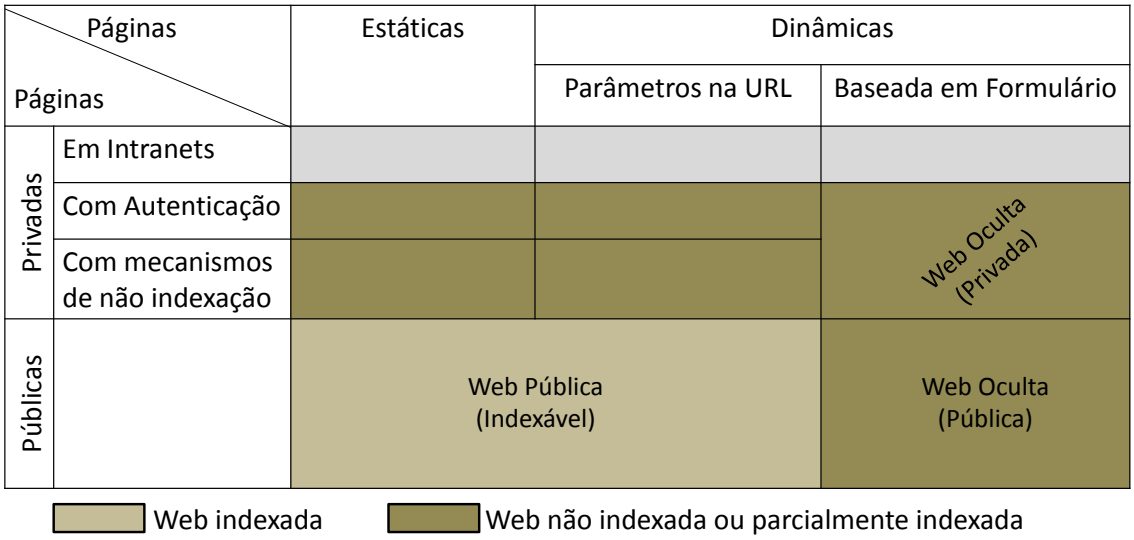


Figura 1.1 – Visão Geral da Web

à forma de geração do conteúdo, estático ou dinâmico (SHESTAKOV et al., 2008). A Web Privada, por sua vez, é dividida em intranets, páginas com necessidade de autenticação e páginas que possuem mecanismos que controlam qual informação presente no sítio deve ou não ser indexada pelo *crawler*. A Web Dinâmica é dividida em páginas com parâmetros especificados juntamente com a URL e em páginas em que os parâmetros são submetidos por meio de uma interface de busca. Assim, o termo Web Oculta é usado para definir as páginas geradas pelo resultado de consultas submetidas por meio de uma interface de busca em bancos de dados disponíveis *on-line*.

O trabalho de Chang et al. (2004) distingue três ideias relacionadas para acessar a Web Oculta — sítio, banco de dados e interface de busca. Um sítio na Web Oculta é um servidor Web que provê informação mantida em um ou mais bancos de dados, cada um dos quais acessíveis por intermédio de um ou mais formulários HTMLs, denominados interfaces de busca (por exemplo, busca simples ou avançada). O acesso a Web Oculta é realizado em bancos de dados *on-line* (BDO), isto é, os dados a serem apresentados como resultado de uma consulta são gerados dinamicamente e são recuperados a cada submissão. Formulários HTML que não são de consulta são excluídos da Web Oculta. Em particular, formulários com funcionalidades de *login* e cadastros não são considerados interfaces de busca na Web Oculta. De forma semelhante, são excluídas as pesquisas "*site search*"que permitem a busca em páginas estaticamente ligadas, pois elas não são dinamicamente geradas com base em um banco de dados. Esta tese está contextualizada na parte pública da Web e considera páginas baseadas em interfaces de busca (formulários HTML).

A Web Oculta é qualitativamente diferente da Web Tradicional. Fontes na Web Oculta armazenam seu conteúdo em BDOs que podem ser pesquisados por meio de uma interface de busca e que produzem dinamicamente os resultados para os usuários. A consulta e os dados retornados são de uma finalidade específica, diferentemente de uma busca em mecanismos tradicionais de busca na Web. A Web Oculta é caracterizada pelo seu crescimento, diversidade de assuntos e grande número de bancos de dados estruturados (LU et al., 2007). O trabalho de Bergman (2001) estimou que:

1. A informação pública disponível na Web Oculta é quatrocentas a quinhentas vezes maior que a disponível na Web tradicional.
2. A Web Oculta contém 7.500 terabytes de informação comparada com os 19 terabytes da Web tradicional.
3. A Web Oculta contém perto de 550 bilhões de documentos comparados com 1 bilhão da Web pública indexável.
4. Os sítios da Web Oculta tendem a retornar 10% mais documentos que a Web tradicional e com três vezes mais qualidade no resultado.
5. Existem mais de 200 mil sítios contidos na Web Oculta.

Além da estimativa de Bergman (2001), um outro estudo estima em três a sete vezes no volume da informação contida na Web Oculta no período de 2000 a 2004 (HE et al., 2007). Ainda, um estudo recente avalia a qualidade do conteúdo da Web Oculta indicando que a qualidade total do conteúdo da Web Oculta é, pelo menos, mil a 2 mil vezes maior que a Web tradicional (NOOR; RASHID; RAUF, 2011).

Finalmente, a Web Oculta cobre uma grande variedade de áreas, como governo, educação, entretenimento entre outras (MADHAVAN et al., 2008; BERGMAN, 2001). Para qualquer área de interesse existem centenas de bancos de dados. Uma pesquisa realizada por Chang et al. (2004) mostra uma distribuição de diferentes bancos de dados disponíveis na Web. O estudo apresenta várias categorias de assuntos assim classificadas: negócios e economia, computação e internet, notícias, entretenimento, recreação e esportes, saúde, governo, sociedade e cultura, educação, arte, ciência entre outras. Se considerarmos um assunto específico, por exemplo, entretenimento, vários bancos de dados podem ser acessados.

1.2 Objetivos

Esta tese é sobre preenchimento de formulários na Web Oculta. No decorrer do texto é apresentada uma pesquisa do estado da arte em preenchimento de formulários. A pesquisa descreve os métodos existentes e mostra suas características. Baseado nesta pesquisa, é proposto um novo método para preenchimento automático de formulários na Web.

O método proposto nesta tese preenche formulários de busca *on-line* disponíveis na Web por meio da combinação de heurísticas e aprendizagem de máquina. O preenchimento é realizado em formulários que possuem vários campos e em formulários que possuem somente um campo textual, por exemplo, um campo de palavra-chave.

A hipótese desta tese é que é possível combinar técnicas de preenchimento de formulários, tais como, heurísticas e aprendizagem de máquina, para selecionar valores para os campos de um formulário. A verificação dos objetivos foi alcançada por meio de observações empíricas em vários formulários presentes na Web. Como objetivos específicos têm-se:

1. Realizar um estudo sobre os métodos existentes para coleta na Web Oculta.
2. Identificar as técnicas de preenchimento de formulários utilizadas nos trabalhos de coleta na Web Oculta.
3. Apresentar uma taxonomia para categorização dos trabalhos relacionados.
4. Realizar uma análise comparativa entre os métodos de preenchimento de interfaces focados na consulta de um usuário.
5. Propor um novo método para preenchimento automático de formulários.

6. Realizar experimentos para uma avaliação qualitativa do método proposto.

1.3 Contribuições

A descoberta de conteúdo na Web Oculta pode ser dividida em três fases. A primeira é descobrir se uma página tem um ponto de entrada, ou seja, formulários caracterizados como de busca na Web Oculta (MADHAVAN et al., 2008; BARBOSA; FREIRE, 2004; LU et al., 2007; MORAES et al., 2012). Na próxima fase, os campos do formulário devem ser manipulados. A manipulação dos campos inclui o preenchimento e a submissão do formulário. Exemplos de trabalhos que tratam essas questões são os de Raghavan and Garcia-Molina (2001), Liddle et al. (2003), Barbosa and Freire (2004), Wu et al. (2006), Ntoulas, Zerfos and Cho (2005), Álvarez et al. (2007), Toda et al. (2010), Dragut et al. (2009), Lage et al. (2004). A terceira fase consta da extração das informações de páginas retornadas pela submissão dos formulários (SOUZA; MELLO, 2014; LAENDER et al., 2002; CAVERLEE; LIU; BUTTLER, 2004; CHANG et al., 2006; ZHAI; LIU, 2006; ZHAO et al., 2005; LU et al., 2007). Geralmente, as páginas resultantes de buscas na Web Oculta possuem algum tipo de estrutura nos dados. Assim, o *crawler* necessita identificar esses resultados e diferenciar resultados válidos e inválidos. A extração de informações das páginas resultantes de submissões envolvem limpeza (YI; LIU; LI, 2003) e mapeamento (CRESCENZI; MECCA; MERIALDO, 2002; WANG; LOCHOVSKY, 2003; LERMAN et al., 2004) de dados das páginas Web para algum modelo de dados com a finalidade de permitir consultas estruturadas.

Muitos trabalhos já foram desenvolvidos para coleta na Web Oculta (MADHAVAN et al., 2008; ÁLVAREZ et al., 2007; WU et al., 2006; DRAGUT et al., 2009; TODA et al., 2010; RAGHAVAN; GARCIA-MOLINA, 2001) e todos tratam a questão de recuperar a informação que os mecanismos de buscas tradicionais não alcançam. Geralmente, os trabalhos estão preocupados em rastrear toda a informação e indexá-la, utilizando os mecanismos tradicionais ou criando uma abordagem focada em determinado domínio de conhecimento (domínio de formulário). Embora existam vários trabalhos que tratam a questão de coleta na Web Oculta, existem poucos trabalhos (MADHAVAN et al., 2008; ÁLVAREZ et al., 2007; RAGHAVAN; GARCIA-MOLINA, 2001; JIANG et al., 2009) que se preocupam em analisar como preencher as interfaces de busca de maneira eficiente.

Ainda, considerando os trabalhos existentes na literatura, verifica-se que são poucos métodos que trabalham com formulários complexos e formulários com entrada textual livre (MADHAVAN et al., 2008). O método proposto por Madhavan et al. (2008) apresenta uma solução para ambos os tipos de formulário e utiliza heurísticas para preencher campos de formulários. O trabalho de Madhavan et al. (2008) apresenta deficiências relativas ao método de submissão, fixação do número de consultas e não trata a dependência de valores entre campos de formulários complexos. Os métodos descritos por Jiang et al. (2009) e Zheng et al. (2013) utilizam técnicas de aprendizagem de máquina, mas consideram o preenchimento de somente campos textuais. A principal desvantagem das soluções que utilizam algoritmos de aprendizagem de máquina é a necessidade de possuir uma base de treinamento previamente classificada. Não existem trabalhos que combinem as técnicas de preenchimento, heurísticas e aprendizagem de máquina, para selecionar valores para campos em ambos os tipos de formulários.

Neste contexto, um dos desafios é como preencher automaticamente os campos de um formulário, sobretudo campos texto. O preenchimento de formulários na Web não é uma

tarefa trivial e exige muito conhecimento de uma aplicação para determinar os valores que serão selecionados para os campos e que formarão os domínios dos atributos. Isso ocorre porque os formulários foram projetados para serem utilizados por seres humanos e uma diversidade de formas de projeto é possível. Assim, um dos desafios do preenchimento automático de formulários é selecionar valores apropriados para os campos. Quando se questiona o preenchimento de formulários na Web, duas questões devem ser respondidas para que o resultado de uma consulta seja satisfatório:

1. Quais os campos do formulário devem ser preenchidos?
2. Quais os valores que devem ser colocados nesses campos?

Esta tese de Doutorado é focada na segunda fase de descoberta de conteúdo na Web Oculta. A principal contribuição desta tese é a proposta de uma solução automática para preenchimento de campos de formulários, especificamente para campos textuais. No decorrer da tese é proposta uma abordagem que combina regras heurísticas e técnicas de aprendizagem de máquina para seleção de valores para campos de formulários. Técnicas de aprendizagem de máquina não são muito exploradas no preenchimento de campos de formulários. A maioria das propostas utiliza heurísticas e trabalham com formulários com somente um campo textual. A ideia é utilizar a combinação de heurísticas e aprendizagem de máquina tanto para formulários com somente um campo textual, assim como, para formulários com múltiplos campos de preenchimento. Resumindo, esta tese apresenta como problema de pesquisa a seguinte hipótese:

É possível combinar as técnicas de preenchimento de formulários para automatizar o processo de preenchimento de interfaces de busca textuais livres e complexas na Web Oculta de maneira a minimizar o custo e maximizar a cobertura dos dados escondidos por trás dos formulários.

Além disso, uma revisão dos trabalhos no estado da arte é realizada e uma taxonomia sobre os trabalhos que abordam o preenchimento de interfaces de busca na Web Oculta é apresentada. Como contribuições específicas, pode-se citar:

- Uma arquitetura para coleta na Web Oculta.
- A proposta de uma taxonomia para categorizar o estado da arte.
- A descrição de um novo método para preenchimento automático de campos em formulários Web.
- Uma avaliação experimental abrangente da abordagem proposta em vários sítios reais da Web.

1.4 Organização

Este trabalho está organizado da seguinte forma:

- O Capítulo 2 descreve os conceitos e definições que estão no escopo desta tese. Este capítulo inclui definições sobre formulários Web, tipos de campos em formulários e como realizar a submissão de formulários. Além disso, o Capítulo 2 apresenta dois novos conceitos que são utilizados no método proposto nesta tese.
- O Capítulo 3 apresenta um estudo sobre os trabalhos relacionados. Uma nova taxonomia para classificação dos métodos existentes é apresentada e são analisados 15 trabalhos relacionados organizados em quatro categorias.

- O Capítulo 4 descreve o método proposto nesta tese para preenchimento de formulários. As duas fases que compõem o método são detalhadas: a fase heurística e a fase de aprendizagem de máquina. Além do método, este capítulo apresenta como a contribuição desta tese está posicionada na atual literatura considerando a taxononomia definida no Capítulo 3.
- O Capítulo 5 mostra uma série de experimentos para avaliação do método proposto. Os experimentos avaliam cinquenta formulários reais na Web distribuídos em seis domínios de formulários, tais como, livros, empregos, receitas, eventos, pessoas e filmes. Uma comparação com vários algoritmos de aprendizagem de máquina é realizada para verificar qual algoritmo se comporta melhor para o problema de preenchimento de campos de formulários.
- Finalmente, no Capítulo 6, são apresentadas as conclusões alcançadas com esta tese, assim como as publicações decorrentes, e são discutidas algumas direções futuras no preenchimento automático de formulários na Web.

2 CONCEITOS

Este capítulo apresenta os principais conceitos sobre formulários Web. Esses conceitos são fundamentais para o entendimento do método de preenchimento de formulários apresentado nesta tese. O método proposto seleciona valores para preenchimento de campos presentes em formulários Web.

2.1 Formulários Web

Geralmente, os dados disponíveis na Web Oculta são acessados por meio de formulários Web. Autores utilizam os termos formulários Web e interfaces de busca, que são pontos de entrada na Web Oculta, como sinônimos. As interfaces de busca mais utilizadas são os formulários HTML. Um exemplo de formulário é mostrado na Figura 2.1 e o código HTML correspondente é apresentado na Figura 2.2. Um formulário HTML está embutido dentro de uma página Web por meio de um rótulo `form`. O atributo `action` identifica que o servidor vai processar a consulta em resposta à submissão de valores para o formulário.

Formulários podem ter vários controles, cada um definido por um rótulo `input`, `select`, `button` ou `textarea`. Os blocos de construção básica para os formulários são as caixas de texto (*text boxes*), listas de seleção (*selection lists*), *check boxes*, *radio buttons* e *submit buttons*. Geralmente, esses blocos são chamados de campos, entradas, elementos ou atributos. Uma caixa de texto é representada como uma caixa vazia com ou sem um valor *default*. Na Figura 2.1, os campos `Pub Name`, `Street` e `Town/County` são exemplos de caixa de texto. Um campo do tipo *selection list* mostra para o usuário uma lista de opções. Existem dois tipos de listas de seleção, listas de seleção única (*combo box*) e listas de seleção múltipla (*list box*). *Radio buttons* e *check boxes* são variações de listas de seleção em que os projetistas mostram as opções para o usuário escolher. Dessa forma, tanto *check boxes* quanto *radio buttons* podem ser manipulados como *selection lists*. Na Figura 2.1, o campo `country` é um exemplo de *selection list*.

No contexto desta tese, é importante distinguir os conceitos de domínio do formulário e domínio do atributo. Domínio do formulário é o assunto, categoria ou tópico ao qual o formulário pertence. Um estudo completo sobre domínios de formulários na Web Oculta pode ser verificado em Chang et al. (2004). O domínio do atributo é o conjunto de valores que podem ser associados a um campo do formulário. Por exemplo, o domínio de um *select list* está inserido dentro do código HTML, juntamente com o rótulo *option* (Figura 2.2, linhas 10 a 16). Nesta tese, os termos consultas e/ou palavras podem ser referenciados com o mesmo significado de valores de um campo em um formulário.

Em resumo, existem dois tipos de campos: campos com um conjunto de valores pré-definidos (domínio de atributo finito) e campos sem valores pré-definidos (domínio de

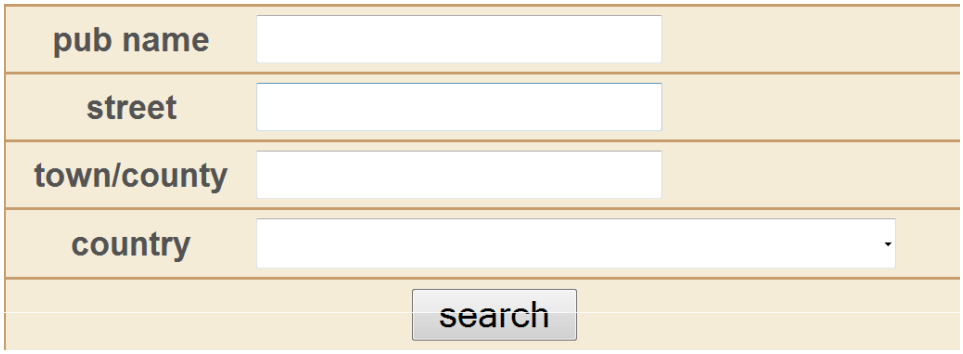


Figura 2.1 – Exemplo de Formulário HTML

atributo infinito). Alguns campos têm domínio pré-definido, isto é, o conjunto de valores está na página HTML, junto com o formulário. Outros campos possuem domínios de atributo indefinidos, por exemplo, o conjunto de todas as palavras com determinado comprimento. Além do domínio de atributo, cada campo possui um valor inicial que pode ser utilizado para submissão se o usuário não informar valor para o campo. Os valores iniciais são um elemento do domínio do atributo. O método descrito em Álvarez et al. (2007) utiliza o conceito de *bounded fields* para representar campos com domínio de atributo finito e *unbouded fields* para representar campos com domínio de atributo infinito.

Cada entrada ou campo de um formulário possui um nome associado que, normalmente, não é o mesmo que o usuário visualiza na página HTML. Campos podem ter rótulos (*labels*), que descrevem o significado do campo para o usuário. Ainda, um rótulo pode ser compartilhado com um grupo de campos. Por exemplo, um rótulo de `data` pode estar associado com três outros campos: `dia`, `mês` e `ano`. Em alguns casos, os rótulos podem ser ignorados para o entendimento da semântica dos campos. Por exemplo, na Figura 2.1, os campos cujos rótulos são *Pub Name* e *Street* possuem os nomes *n* e *st*, respectivamente, conforme as linhas 3 e 5 da Figura 2.2. Maiores detalhes sobre o entendimento de interfaces de busca podem ser encontrados em Dragut et al. (2009), Kaljuvee et al. (2001), Nguyen, Nguyen and Freire (2008), Shestakov, Bhowmick and Lim (2005), Wu et al. (2009), Khare and An (2009), Khare, An and Song (2010).

O trabalho de Wu et al. (2004) observou que as interfaces de busca podem ter uma estrutura hierárquica. Em mais detalhes, uma interface de busca é uma árvore ordenada de elementos na qual as folhas representam os campos da interface, os nós internos representam os grupos dos campos na interface e a ordem entre os nós irmãos dentro da árvore é similar à ordem dos campos na interface. O esquema em árvore captura a ordem semântica e o agrupamento dos campos na interface.

O trabalho apresentado em Dragut et al. (2009) descreve algumas regras que foram detectadas em quase todas as interfaces reais e que são parcialmente obedecidas por projetistas no projeto das interfaces. A extração automática de interfaces de busca é um desafio, pois elas são criadas de maneira autônoma e a linguagem HTML obedece a uma gramática flexível. O trabalho de Dragut et al. (2009) identificou um conjunto de regras que os projetistas de interfaces de busca seguem de maneira intuitiva. A seguir são listadas as regras identificadas por Dragut et al. (2009):

- Regra 0: interfaces de busca são organizadas de forma *top-down* e da esquerda para a direita.

```
1  <form  action="/pubs/results.shtml" method="get">
2    <span>pub name</span>
3    <input type="text" name="n">
4    <span>street</span>
5     <input type="text" name="st">
6    <span>town/county</span>
7     <input type="text" name="tc">
8    <span>country</span>
9    <select name="country">
10    <option value=""></option>
11   <option value="GB">United Kingdom</option>
12    <option value="RS"></option>
13    <option value="AG">Antigua and Barbuda</option>
14    <option value="AR">Argentina</option>
15    <option value="AU">Australia</option>
16    <option value="AT">Austria</option>
17    ............
18    </select>
19    <input type="submit" value="search" name="search">
20 </form>
```

Figura 2.2 – Código HTML da Figura 2.1

- Regra 1: campos dentro de uma interface são organizados em unidades semânticas de informação, ou seja, grupos.
- Regra 2: um rótulo é utilizado para denotar ou a semântica de um campo ou a de um grupo de campos, mas não ambos.
- Regra 3: se um campo `c` com um rótulo `rc` pertence a um grupo `g` com rótulo `rg` então o estilo do texto do rótulo `rc` é diferente do estilo do texto do rótulo `rg`.
- Regra 4: se um grupo `g` com rótulo `rg` é um subgrupo de um grupo `G` com rótulo `rG` então o estilo do texto do rótulo `rg` é diferente do estilo do texto do rótulo `rG`.
- Regra 5: os rótulos de todos os membros de um grupo têm o mesmo estilo de texto.
- Regra 6: a orientação de um rótulo de um campo está à esquerda, acima, à direita ou abaixo do campo. O rótulo de um grupo está ou acima ou à esquerda do grupo.
- Regra 7: os rótulos de todos os membros de um grupo têm a mesma orientação.
- Regra 8: Seja `G` um grupo e `g` um dos seus subgrupos. Suponha que um rótulo com estilo de texto `ET1` seja associado ao grupo `G` e um rótulo com estilo de texto `ET2` seja associado ao grupo `g`, então para qualquer grupo `H` e seu subgrupo `h`, o rótulo associado para `H` não pode ter o estilo de texto `ET2` quando o rótulo associado para `h` tem o estilo de texto `ET1`.

Outro ponto importante é o método de submissão de um formulário. A submissão de um formulário é realizada pela utilização de um dos dois métodos: *get* ou *post*. O método é definido pelo atributo *method*, utilizado em conjunto com o rótulo *form* da linguagem HTML. Na Figura 2.2, linha 1, é apresentado um exemplo do método *get*. Com o método *get* os parâmetros são adicionados na ação e incluídos como parte da URL na requisição HTTP. Com o método *post* os parâmetros são enviados dentro da requisição HTTP e a URL contém somente a ação a ser executada. Assim, as URLs obtidas de formulários que

utilizam o método *get* são únicas e dependentes dos valores submetidos enquanto que as que utilizam o método *post* não são únicas.

Para um conjunto de submissões de um formulário, são definidos dois conceitos: *Template* e *Template Instance*. Essas definições são extensões às definições propostas por Madhavan et al. (2008) de *query templates*. Formalmente, têm-se as seguintes definições:

Definição 1. (*Template*). Seja $F = f_1, f_2, ..., f_n$ um conjunto de campos localizados em um formulário Web F e seja $T = t_1, t_2, ..., t_m$, o conjunto de todas as combinações dos elementos de F. Cada elemento presente em T é definido como um *template*.

Definição 2. (*Template Instance*). Seja t um *template* presente em T e seja $V = v1, v2, ..., vm$ um conjunto de valores para t. Um *template instance* é um par atributo valor (t, v) em que v é um elemento de V.

Templates são combinações dos campos do formulário. Por exemplo, o formulário da Figura 2.2 tem quatro campos *pub name*(`n`), *street*(`st`), *town/county*(`tc`) e *country*(`country`). Os templates gerados partindo dos quatro campos são: `n`, `st`, `tc`, `country`, `n&st`, `n&tc`, `n&country`, `st&tc`, `st&country`, `tc&country`, `n&st&tc`, `n&st&country`, `n&tc&country`, `st&tc&country` e `n&st&tc&country`. Se existem `n` campos em um formulário, têm-se $(2^n - 1)$ templates possíveis. O número de campos que compõe um *template* é chamado de dimensão do *template*. Por exemplo, o template `n&st&country` tem dimensão 3, pois é composto pela combinação dos campos `n`, `st` e `country`. Os *template instances* empregam um valor de preenchimento para cada campo considerado na submissão do formulário. Por exemplo, no formulário da Figura 2.2, um *template instance* para o *template* `n&country` é `n=WhiteHorse&country=GB`. Na prática, *template instances* são representados por URLs. Essas URLs são submissões ao formulário, conforme o método seja *get* ou *post*.

2.2 Resumo do Capítulo

Neste capítulo, foram apresentados os conceitos e definições utilizados nesta tese assim como a uniformização de conceitos similares. Basicamente, os campos existentes em formulários podem ser divididos em dois tipos: aqueles que possuem valores predefinidos, denominados de campos de domínio finito e campos sem valores predefinidos, denominados de campos de domínio infinito. Outras definições importantes apresentadas neste capítulo são o *template* e o *template instance*. *Template* é a representação do formulário por meio dos seus campos e as combinações entre campos. *Template Instance* é a instanciação de um *template*, ou seja, quando o template possui valores associados as campos. Do ponto de vista prático, os *template instances* são as URLs submetidas aos formulários.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos que estão diretamente relacionados com o acesso a informações da Web Oculta. A seção 3.1 apresenta uma taxonomia para classificação dos trabalhos relacionados. As seções 3.2 a 3.5 descrevem os trabalhos relacionados com preenchimento de formulários na Web Oculta conforme a taxonomia definida na seção 3.1. Finalmente, a seção 3.6 apresenta uma análise comparativa entre os trabalhos.

3.1 Categorização do Estado da Arte

Esta seção tem como objetivo apresentar uma taxonomia para o agrupamento dos trabalhos relacionados. A taxonomia apresentada nesta tese leva em consideração dois eixos ortogonais de visualização dos trabalhos relacionados. Um dos eixos possui um viés da estratégia utilizada para a seleção de valores enquanto que o outro eixo possui um viés da estrutura da informação escondida por trás dos formulários.

Considerando o viés de preenchimento, existem dois tipos de estratégias utilizadas pelos trabalhos para selecionar os valores para submissão nos formulários. Alguns métodos tomam decisões baseadas em informações estatísticas sobre os valores e utilizam limiares como pontos de parada na busca de dados. Outros métodos utilizam uma abordagem diferente, por meio da utilização de algoritmos de aprendizagem de máquina e a utilização de conjuntos de treinamento para a construção de um modelo de aprendizagem para avaliação dos valores selecionados para os campos do formulário. Portanto, para fins de estudo e classificação, o viés de preenchimento é dividido em:

- **Heurísticas:** essa abordagem de preenchimento utiliza regras que mostram ser boas práticas para seleção de valores para preenchimento de formulários.
- **Aprendizagem de Máquina:** essa abordagem de preenchimento faz uso de informações avaliadas para a criação de um modelo, o qual é utilizado para predição dos novos valores para preenchimento de campos em formulários.

Dependendo do tipo de informação presente na Web Oculta, os BDOs por trás de formulários podem ser classificados como textuais ou estruturados. Um banco de dados textual é um sítio que contém especialmente documentos textuais. Uma vez que documentos textuais geralmente não possuem uma estrutura bem definida, a maioria dos bancos de dados textuais utiliza uma interface de busca simples em que os usuários informam uma lista de palavras-chave em uma única caixa de texto. Em outro sentido, um banco de dados estruturado, frequentemente, contém dados relacionais em vários atributos e as interfaces de busca possuem múltiplos atributos. Ntoulas, Zerfos and Cho (2005) apresentam uma classificação das fontes de dados na Web Oculta, conforme o tipo de informação, em bancos de dados textuais ou bancos de dados estruturados. Noor, Rashid

Figura 3.1 – Exemplos de formulário (a) complexo (b) textual livre (TJIN-KAM-JET; TRIESCHNIGG; HIEMSTRA, 2011)

and Rauf (2011) classificam a fonte de dados na Web Oculta como não estruturada e estruturada e associa essa classificação a dois tipos de interfaces: *simple* e *advanced* respectivamente. Tjin-Kam-Jet, Trieschnigg and Hiemstra (2011) classificam os formulários da Web Oculta em entradas textuais livres e formulários Web Complexos. Considerando o viés do tipo de informação escondida por trás dos formulários de busca, será utilizada a classificação proposta por (TJIN-KAM-JET; TRIESCHNIGG; HIEMSTRA, 2011). Assim, os bancos de dados estruturados podem ser mapeados para as interfaces de busca com múltiplos atributos ou formulários Web Complexos, enquanto que os bancos de dados textuais podem ser mapeados para interfaces de busca com entradas textuais livres ou formulários com palavras-chave. A Figura 3.1 mostra um exemplo de um formulário complexo (Figura 3.1a) e um formulário de entrada textual livre (Figura 3.1b) equivalente.

Classificar os trabalhos de preenchimento de formulários na Web Oculta sob esses dois vieses é uma tarefa inovadora. Os trabalhos sempre estarão associados à taxonomia sob, pelo menos, um viés de preenchimento, heurística ou aprendizagem, e um viés do tipo de interface, entrada textual livre ou formulário Web complexo. É importante salientar que as estratégias que compõem o mesmo viés não são disjuntas. Portanto, existe uma sobreposição entre as estratégias de um mesmo viés, assim como é possível uma sobreposição entre as estratégias de ambos os vieses. A Figura 3.2 ilustra a taxonomia proposta. Existem trabalhos que manipulam ambas as estratégias de preenchimento assim como as estruturas dos formulários.

As próximas seções apresentam os trabalhos relacionados existentes na literatura sobre consulta e preenchimento de formulários. Os trabalhos serão apresentados considerando a taxonomia apresentada na seção 3.1. As seções 3.2 e 3.3 descrevem os trabalhos que utilizam heurísticas. Já, a seção 3.4 descreve os trabalhos que usam a estratégia de aprendizagem de máquina e, por fim, a seção 3.5 apresenta os trabalhos que combinam mais de uma estratégia.

3.2 Heurísticas e Entradas Textuais Livres

Esta Seção descreve os trabalhos relacionados que utilizam heuríticas para o preenchimento de formulários com entradas textuais livres.

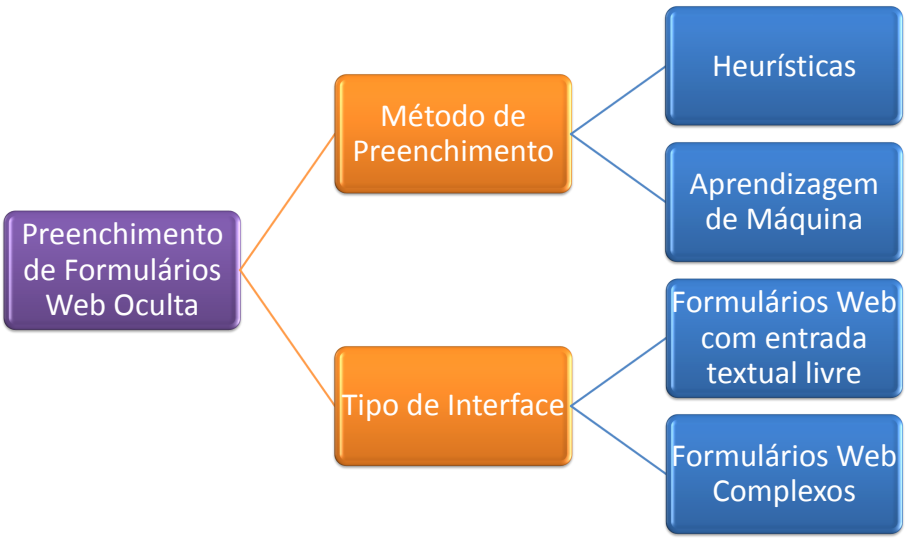


Figura 3.2 – Taxonomia

3.2.1 Barbosa and Freire (2004)

O trabalho de Barbosa and Freire (2004) estuda o problema de recuperação automática de dados da Web Oculta considerando formulários que recebem como entrada palavras-chave. Formulários baseados em palavras-chave simplificam a consulta porque não é preciso nenhum conhecimento sobre a estrutura ou o esquema dos dados.

Para recuperar os dados escondidos por meio de um formulário automaticamente, é preciso determinar quais as consultas devem ser usadas. Uma solução ingênua é enviar uma consulta para cada palavra presente em um dicionário. No entanto, essa solução pode ter um baixo desempenho em consequência do grande número de submissões que podem resultar em erro ou dados repetidos.

Dessa forma, o método descrito em Barbosa and Freire (2004) propõe uma abordagem baseada em amostragem para descobrir palavras que resultem em uma alta cobertura, isto é, uma taxa elevada de dados recuperados. A intuição é que palavras do próprio banco de dados ou da coleção de documentos são mais prováveis de resultar em alta cobertura do que palavras geradas aleatoriamente. A proposta consiste de duas fases:

- Uma amostra da coleção seleciona um conjunto de palavras-chave com alta frequência, conforme algoritmo descrito na Figura 3.3.
- Utilização dessas palavras com alta frequência para construir consultas com alta cobertura, segundo algoritmo apresentado na Figura 3.4.

O Algoritmo 1, visualizado na Figura 3.3, gera consultas de teste que são enviadas para aprender novas palavras-chave tomando por base o conteúdo do resultado de consultas e suas frequências relativas, considerando todos os resultados recuperados durante essa fase. O primeiro passo do algoritmo é recuperar a página na qual o formulário está localizado, selecionar valores e submeter as palavras-chave (linhas 1-16). Uma vez que a página resultante é recuperada com sucesso, o Algoritmo 1 constrói a lista de palavras candidatas iterativamente, submetendo uma consulta usando uma palavra selecionada (linha 22) e usando os resultados para inserir novas palavras com alta frequência no conjunto de palavras candidatas, assim como, atualizar a frequência das palavras existentes (linhas

23-28).

Algorithm 1 SampleKeywords(URL,form)

```
1:  page = get(URL);
2:  // retrieve the first set of results
3:  initialKeywordList = generateList(page);
4:  word = nextHigherOccurrency(initialKeywordList);
5:  resultPage = submitQuery(form,word);
6:  while resultPage == errorPage do
7:     word = nextHigherOccurrency(initialKeywordList);
8:     resultPage = submitQuery(form,word);
9:  end while
10: // initialize keyword list
11: acceptStopword = checkStopword(form);
12: if acceptStopword then
13:    keywordList = generateListWithStopWords(resultPage);
14: else
15:    keywordList = generateList(resultPage);
16: end if
17: // iterate and build list of candidate keyword/occurences
18: numSubs = 0;
19: while numSubs < maxSubs do
20:    // randomly selects a word
21:    word = selectWord(keywordList);
22:    resultPage = submitQuery(form,word);
23:    // adds new keywords, and updates the frequency of existing keywords
24:    if acceptStopword then
25:       keywordList += genListWithStopWords(resultPage);
26:    else
27:       keywordList += genList(resultPage);
28:    end if
29:    numSubs++;
30: end while
31: return keywordList;
```

Figura 3.3 – Algoritmo de seleção de palavras (BARBOSA; FREIRE, 2004)

As palavras-chave candidatas são a entrada para o Algoritmo 2 (Figura 3.4). O Algoritmo 2 utiliza uma estratégia gulosa para construir as consultas que possuem a mais alta cobertura. Iterativamente, o algoritmo seleciona, do conjunto de candidatas, a palavra-chave com a maior frequência e adiciona à consulta se ela conduz a um incremento na cobertura.

Uma questão que precisa ser abordada para o perfeito funcionamento da proposta de Barbosa and Freire (2004) é como a solução seleciona as palavras-chave. Barbosa and Freire (2004) selecionam uma palavra da página inicial do formulário. O Algoritmo 1, apresentado na Figura 3.3, encontra palavras-chave adicionais pela submissão iterativa de consultas usando palavras-chave obtidas de iterações anteriores. Os experimentos mostram que a utilização de *stopwords* na indexação dos resultados aumenta a cobertura do método. Isso ocorre porque *stopwords* aparecem em vários documentos e possuem uma alta frequência. O método (BARBOSA; FREIRE, 2004) utiliza oito sítios reais da Web

Algorithm 2 ConstructQuery(keywordList,form)

```
1:  numSubs = numWords = totalBefore = totalCurrent = 0;
2:  query = queryTemp = null;
3:  while (numSubs < maxSubs) && (numWords < maxTerms) do
4:     // selects word with highest frequency
5:     word = nextElement(listOfOccurrency);
6:     queryTemp = addWordToQuery(query,word);
7:     page = submit(form, queryTemp);
8:     totalCurrent = getNumberOfResults(page,form);
9:     if (totalBefore * minimumIncrease) <= totalCurrent then
10:       query = queryTemp;
11:       totalBefore = totalCurrent;
12:       numberWords++;
13:    end if
14:    numSubs++;
15: end while
```

Figura 3.4 – Algoritmo de construção de consultas (BARBOSA; FREIRE, 2004)

para realizar os experimentos e somente sítios nos quais é possível conhecer o tamanho da coleção de documentos. As métricas utilizadas para avaliação dos resultados são a cobertura e o número de requisições.

3.2.2 Ntoulas, Zerfos and Cho (2005)

Ntoulas, Zerfos and Cho (2005) apresentam uma solução para geração automática de consultas para preenchimento de formulários de busca na Web Oculta. O foco do trabalho proposto por (NTOULAS; ZERFOS; CHO, 2005) é em bancos de dados textuais com interfaces de busca com entrada textual livre. Dessa forma, Ntoulas, Zerfos and Cho (2005) propõem uma solução adaptativa que examina as páginas retornadas de submissões anteriores e adapta a política de seleção automática de consultas baseada nessas submissões.

O problema da seleção de consultas é formalizado da seguinte maneira (NTOULAS; ZERFOS; CHO, 2005): O *crawler* realiza o *download* de um sítio da Web que possui um conjunto de páginas *S*. Cada possível consulta q_i pode ser representada como um subconjunto de *S*, contendo todas as páginas que são retornadas quando é enviada a consulta q_i ao sítio. Cada subconjunto está associado a um peso que representa o custo da consulta. Assim, Ntoulas, Zerfos and Cho (2005) encontram quais subconjuntos (consultas) cobrem o número máximo de páginas Web com o mínimo peso (custo).

A solução adaptativa proposta por Ntoulas, Zerfos and Cho (2005) avalia os documentos que retornam de consultas anteriores, e são estimadas as palavras que possuem uma maior probabilidade de recuperar mais documentos. Baseado nessa análise, a palavra mais promissora é selecionada e submetida repetindo-se o processo.

Considere ainda uma consulta q_i, Ntoulas, Zerfos and Cho (2005) utilizam $P(q_i)$ para denotar a fração de páginas que q_i vai recuperar se for submetida para o sítio. Por exemplo, se um sítio contém 10 mil páginas no total, e se 3 mil páginas são recuperadas para a consulta $q_i = telephone$, então $P(q_i) = 0,3$. Outra métrica definida por Ntoulas, Zerfos

and Cho (2005) é o custo de submeter uma consulta q_i, denotado por $cost(q_i)$. O custo consiste de um número de fatores, incluindo o custo de submeter a consulta, recuperar os resultados e realizar o *download* das páginas. O custo de submissão de uma consulta c_q é fixo. O custo de recuperação dos resultados (c_r) é proporcional ao número de documentos que combinam com a consulta, enquanto que o custo de download (c_d) de um documento que combina com a consulta também é fixo. Assim, o custo total, $cost(q_i)$, de uma consulta é dado pela Equação 3.1:

$$cost(q_i) = c_q + c_r P(q_i) + c_d P(q_i) \tag{3.1}$$

A análise da palavra mais promissora é realizada por meio da estimativa da consulta q_i como a próxima consulta. Assim, assume-se que as consultas $q_1, ..., q_{i-1}$ foram submetidas, é preciso estimar a $P(q_1 \vee ... \vee q_{i-1} \vee q_i)$ para cada próxima consulta q_i a ser submetida. Ntoulas, Zerfos and Cho (2005) reescrevem $P(q_1 \vee ... \vee q_{i-1} \vee q_i)$ conforme a Equação 3.2:

$$P((q_1 \vee ... \vee q_{i-1}) \vee q_i = P(q_1 \vee ... \vee q_{i-1}) + P(q_i) - P(q_1 \vee ... \vee q_{i-1}) P(q_i / q_1 \vee ... \vee q_{i-1})) \tag{3.2}$$

O cálculo de $P(q_1 \vee ... \vee q_{i-1})$ é a união de todas as páginas que foram baixadas de $q_i, ..., q_{i-1}$. O valor de $P(q_i/q_1 \vee ... \vee q_{i-1})$ é a probabilidade condicional de q_i aparecer nas páginas das consultas $q_1, ..., q_{i-1}$, e pode ser calculada pela contagem do número de vezes que q_i aparece nas páginas retornadas das consultas $q_1, ..., q_{i-1}$. A probabilidade $P(q_i)$ é calculada pelo método proposto por Ipeirotis and Gravano (2002).

A seleção da consulta é baseada no número de novos documentos que podem ser recuperados da consulta q_i e do custo de submissão da consulta q_i. Por exemplo, se duas consultas q_i e q_j possuem o mesmo custo, mas q_i recupera mais dados novos que q_j, q_i é mais desejável que q_j. Similarmente, se q_i e q_j retornam o mesmo número de dados novos, mas q_i possui um custo menor que q_j, q_i é mais desejável. Dessa forma, Ntoulas, Zerfos and Cho (2005) utilizam a métrica denominada *Efficiency* para quantificar o quão desejável é uma consulta q_i. A Equação 3.3 apresenta a fórmula da Eficiência:

$$Efficiency(q_i) = \frac{P_{new}(q_i)}{cost(q_i)} \tag{3.3}$$

em que $P_{new}(q_i)$ representa a quantidade de novos documentos recuperados para q_i e $cost(q_i)$ representa o custo de submissão de q_i. $Efficiency(q_i)$ mede a quantidade de novos documentos que são recuperados por unidade de custo e pode ser usado como indicador de como os recursos estão sendo efetivamente utilizados quando q_i é submetida. Assim, (NTOULAS; ZERFOS; CHO, 2005) podem estimar a eficiência de cada consulta candidata q_i e selecionar aquela que possui o maior valor.

Uma limitação desse trabalho é a seleção da palavra inicial para a primeira consulta. A escolha dessa palavra não é realizada pelo algoritmo adaptativo proposto. A palavra para a primeira consulta é definida manualmente. Além disso, a palavra deve ser definida para cada sítio, pois uma palavra utilizada para um determinado sítio pode não ser útil para outro. Os experimentos realizados incluem quatro sítios da Web Oculta.

3.2.3 Wang et al. (2012)

Wang et al. (2012) estudam o problema de selecionar consultas para fazer a coleta em fontes de dados na Web Oculta usando um conjunto de documentos de amostra. A

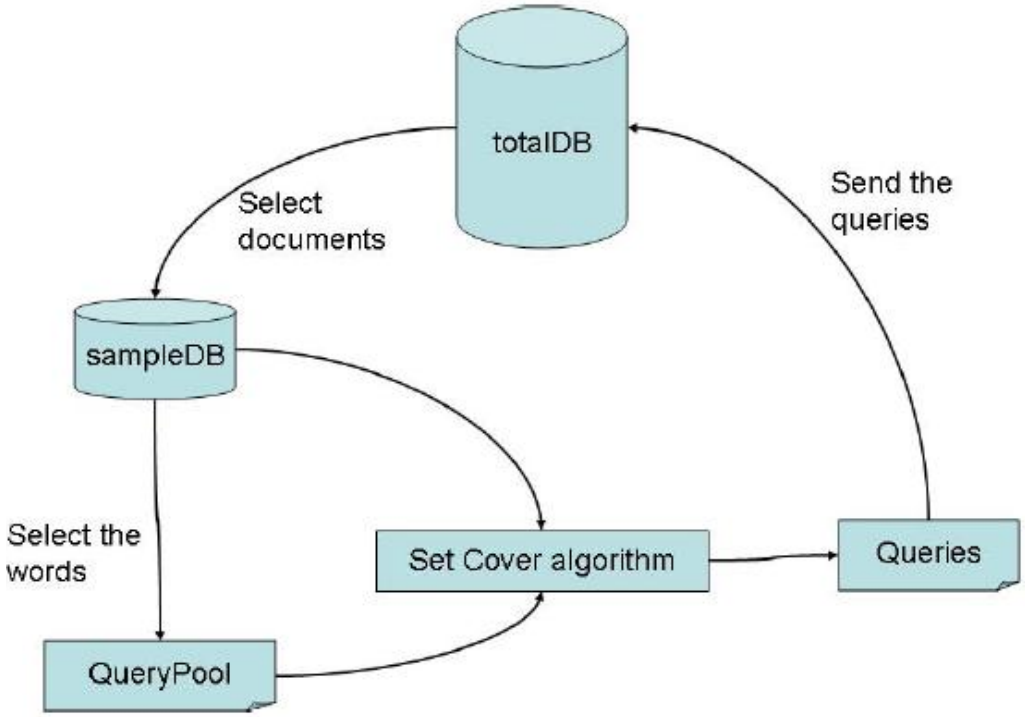


Figura 3.5 – Métodos de coleta baseado na Amostragem (WANG et al., 2012)

proposta de Wang et al. (2012) seleciona um conjunto de documentos da fonte de dados como uma amostra que representa a fonte de dados original. Com base na amostra, são selecionadas consultas que cobrem a maioria dos documentos da amostra com um baixo custo. Finalmente, o conjunto de consultas selecionadas partindo da amostra são usadas para extrair os dados da fonte original. A Figura 3.5 ilustra o método de Wang et al. (2012) .

O método de Wang et al. (2012) recebe como entrada a fonte de dados *db*, o tamanho da amostra *s*, e o tamanho do *pool* de consultas *p*. Dessa forma, é preciso estimar o número de documentos da amostra, ou seja, tamanho de *sampleDB*, e também, é preciso selecionar, partindo do conjunto de amostras, os termos que farão parte do *pool* de consultas, ou seja, o *QueryPool*. Finalmente, as *Queries* selecionadas do *QueryPool* dependem de vários critérios, tais como, o tamanho de *Queries*, o algoritmo selecionado para obter os termos em *SampleDB*, e os valores de *document frequency* (*df*) dos termos selecionados (Figura 3.5).

A afirmação de Wang et al. (2012) é que consultas que funcionam bem em uma amostra do banco de dados original, também produzem resultados satisfatórios atuando em toda a base de dados. Para isso, Wang et al. (2012) definem duas medidas: a taxa de acerto (*hit rate*) e a taxa de sobreposição (*overlapping rate*). A primeira tem como objetivo maximizar a extração de dados e a segunda, avaliar o custo da coleta.

Seja q uma consulta, DB um banco de dados e $S(q, DB)$ o conjunto de itens de dados que a consulta q recupera do banco de dados DB. Dado um conjunto de consultas $Q = q_1, q_2, ..., q_i$ e um banco de dados DB, a taxa de acerto do conjunto Q em DB, denotado por $HR(Q, DB)$, é definida como a relação entre o número de dados distintos coletados pelas consultas presentes em Q e o tamanho total do banco de dados DB, ou seja, pela Equação 3.4:

$$\mu = |\cup_{j=1}^{i} S(q_j, DB)| \quad e \quad HR(Q, DB) = \mu/|DB| \tag{3.4}$$

A taxa de sobreposição do conjunto Q em DB, denotado por $OR(Q, DB)$, é definida como a razão entre o número total de links coletados (η) e o número de links únicos recuperados pelas consultas (μ) presentes em Q segundo a Equação 3.5:

$$\eta = \sum_{j=0}^{i} |S(q_j, DB) \quad e \quad OR(Q, DB) = \eta/\mu \tag{3.5}$$

Assim, as hipóteses de Wang et al. (2012) são: (i.) os termos selecionados de *SampleDB* vão acertar muitos documentos em *TotalDB* e (ii.) a taxa de sobreposição em *TotalDB* será bem próxima da taxa de sobreposição em *SampleDB*.

A seleção de consultas para o *QueryPool* é realizada por termos de maneira aleatória e levando em consideração o valor do df dos termos no tamanho da amostra. Wang et al. (2012) selecionam os termos que estão localizados em certas faixas de *document frequencies*. Os experimentos realizados por eles selecionam termos que possuem *df* entre 2% e 20% do tamanho da amostra. Por exemplo, se o tamanho da amostra é 2.000, Wang et al. (2012) utilizam as palavras com valores de df entre 2 e 400. Palavras com $df = 1$ são consideradas mais raras e palavras que aparecem em mais de 400 documentos são muito populares para serem utilizadas.

Após a criação do *QueryPool* é preciso selecionar do pool quais os termos que serão enviados como consultas para o *TotalDB*. Wang et al. (2012) utilizam o seguinte critério para selecionar os valores de *QueryPool*(Equação 3.6):

$$HR(Queries, SampleDB) = 1 \ e \ OR(Queries, SampleDB) = minimo \tag{3.6}$$

Os experimentos realizados por eles têm como propósito avaliar como as consultas selecionadas de *SampleDB* se comportam quando executadas no *TotalDB*. Os experimentos foram executados em quatro conjuntos de dados de diferentes tamanhos. Um dos pontos negativos é que os experimentos não são executados em fontes de dados reais na Web Oculta. Como o método de Wang et al. (2012) depende do conhecimento sobre o total de dados da fonte e, geralmente, o número total de documentos é desconhecido, de fato é impossível calcular a taxa de acerto, HR, e avaliar os métodos de coleta em fontes reais. A geração da amostra é realizada de maneira aleatória partindo da fonte de dados. Uma segunda limitação do método de Wang et al. (2012) é a validação dos valores somente em interfaces com um único campo de entrada, baseado em palavras-chave.

3.2.4 Soulemane, Rafiuzzaman and Mahmud (2012)

Soulemane, Rafiuzzaman and Mahmud (2012) propõem um mecanismo automático de indexação para conteúdos dinâmicos encontrados na Web Oculta. O método de Soulemane, Rafiuzzaman and Mahmud (2012) apresenta uma abordagem automática para realizar a coleta no conteúdo dinâmico por meio da utilização de quatro fases: extração de conteúdo dinâmico, detecção de formulários, seleção de palavras-chave para preenchimento de formulários e detecção de URLs duplicadas.

A extração do conteúdo dinâmico consiste na extração dos dados por trás dos formulários de busca, por exemplo, os resultados da busca. A fase Detecção de formulários identifica os formulários de busca dentro de uma página. Soulemane, Rafiuzzaman and

Mahmud (2012) consideram somente formulários com um único campo texto para submissão. Formulários com mais de um campo são ignorados. A fase detecção de duplicatas tem por objetivo identificar quando dois valores diferentes usados em uma submissão podem produzir o mesmo resultado.

A fase de seleção de palavras-chave está diretamente relacionada com esta tese. Uma das etapas do *crawler* é a seleção de palavras para preenchimento de formulários. Soulemane, Rafiuzzaman and Mahmud (2012) selecionam os valores iniciais por meio da extração do conteúdo estático da página Web que contém o formulário de busca. Após a primeira geração de resultados, novas palavras são selecionadas partindo das páginas recuperadas. O limite de palavras é definido por um valor máximo, para evitar que o *crawler* entre em um *loop* infinito. A seleção de palavras-chave partindo da página Web, em vez da utilização de um dicionário ou repositório predefinido, permite o suporte a formulários multilíngues.

A prioridade na seleção de palavras-chave é calculada com base na frequência do termo F_{tf}, a qual determina a importância de um termo no conjunto de documentos. Suponha uma palavra W_i ocorra n_p vezes em uma página web P e existem N_p palavras no total (incluindo duplicatas) na página. A frequência do termo é dada por $F_{tf} = n_p/N_p$. Soulemane, Rafiuzzaman and Mahmud (2012) consideram *stopwords* para eliminar palavras que ocorram com muita frequência.

Os experimentos realizados avaliam várias iterações, mas estão limitados a somente um formulário de consulta com um único campo textual. Apesar dos formulários possuírem mais de um método de submissão, o trabalho de Soulemane, Rafiuzzaman and Mahmud (2012) trabalha somente com métodos do tipo *get*.

3.2.5 Wu et al. (2006)

Conforme Wu et al. (2006), existem duas maneiras de adquirir dados provenientes de fontes na Web. A mais eficiente e eficaz é que os provedores de dados disponibilizem seus dados para que os mecanismos de busca possam indexá-los. Infelizmente, em um ambiente totalmente autônomo, não cooperativo e competitivo como a Web pública, essa abordagem não é factível em razão do grande número de Web sítios ocultos e que potencialmente requerem um esforço humano e de tempo.

Wu et al. (2006) propõem a busca de conteúdo na Web Oculta por meio de técnicas de seleção de consultas nas quais são coletados os dados disponíveis no banco de dados. A busca inicia com consultas "sementes"preparadas nos formulários como pares (atributo, valor), por exemplo, *("Artist, 'Dire Straits'")*, *("Title, 'Brothers In Arms'")*, etc. Automaticamente são realizadas consultas na fonte de dados pela invocação dos valores de atributos nos campos apropriados do formulário. Os dados retornados são armazenados em um banco de dados local e decompostos em valores de atributos, que serão armazenados como candidatos para futuras consultas. Esse processo é repetido até que todas as possíveis consultas sejam enviadas ou algum limiar ou critério de parada seja atingido. Ele propõe um conjunto de técnicas para seleção de consulta que aperfeiçoam a taxa de recuperação, isto é, escolhe a consulta que melhor retorna o conteúdo do banco de dados.

A proposta de Wu et al. (2006) visualiza uma base de dados como um banco de dados estruturado com uma única tabela DB com n registros $t_1, t_2, ..., t_n$ sobre um conjunto de m atributos $AS = attr_1, attr_2, ..., attr_m$. O conjunto de valores de atributos distintos (DAV) consiste de todos os distintos valores de atributos que aparecem em DB. Um grafo atributo-valor (AVG), $G(V, E)$, para DB é um grafo não direcional que pode ser construído da seguinte maneira: para cada valor distinto $av_i \in DAV$ existe um único vértice

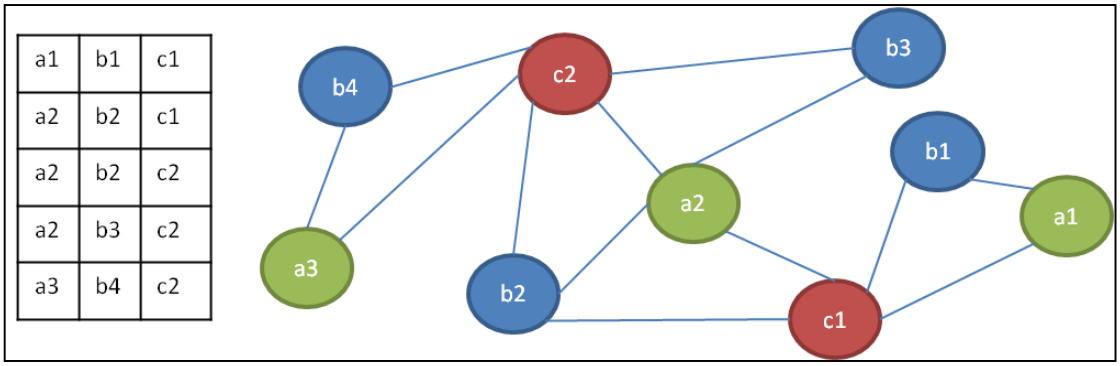


Figura 3.6 – Grafo Atributo-Valor (WU et al., 2006)

$v_i \in V$. Um arco não direcional $(v_i, v_j) \in E$, se, e somente se, av_i e av_j coexistem em uma instância relacional $t_k \in DB$. Cada arco em AVG representa uma ligação relacional entre av_i e av_j. De acordo com a definição descrita, os valores de atributo de cada instância relacional formam um grupo exclusivo. Se dois registros compartilham o mesmo valor de atributo, o vértice correspondente ao valor liga os dois grupos. Dependendo da estrutura do banco de dados, um grafo AVG não é necessariamente conectado.

A Figura 3.6 mostra um banco de dados relacional (representado como uma única tabela) e o correspondente grafo de valores de atributos distintos. Considerando que o motor de busca inicia com o valor a_2. Na primeira submissão, a consulta é realizada com o valor a_2. Os resultados retornados serão b_2, c_1, c_2, b_3. Na segunda submissão, se c_2 for selecionado e enviado, um novo registro é descoberto com dois novos valores de atributos: a_3, b_4. Na submissão final, se c_1 é selecionado, o último registro remanescente (primeira linha) é selecionado. Em resumo, o significado do AVG é que ele conceitualmente transforma a coleta de um banco de dados baseado em consulta em um problema de navegação em grafos, o qual é similar ao problema da coleta tradicional da Web, realizada através de *hiperlinks*.

O modelo de consulta proposto por Wu et al. (2006) considera um conjunto de atributos consultáveis $A_q = attr_{q1}, attr_{q2}, ..., attr_{qn}$ e um conjunto de atributos resultantes da consulta $A_r = attr_{r1}, attr_{r2}, ..., attr_{rm}$. Cada $attr_{qi} \in A_q$ representa um atributo que aparece em uma interface de busca (formulário) enquanto o atributo resultante $attr_{rj} \in A_j$ corresponde aos atributos que aparecem nas páginas resultantes da consulta. O trabalho de Wu et al. (2006) trata somente um predicado de igualdade para as consultas.

Para determinar qual a melhor consulta, Wu et al. (2006) definem a medida de custo de uma consulta e a medida de taxa de captura da consulta. O custo de uma consulta q_i em um banco de dados *DB* é definido como $cost(q_i, DB) = (num(q_i, DB)/k)$, em que $num(q_i, DB)$ representa o número total de registros em *DB* que combinam com q_i e k corresponde ao número máximo de registros visualizados em cada página de resultados no sítio. Por exemplo, se existem 98 registros de músicas em um banco de dados de músicas que combinam com o valor de atributo *"Artist,'Dire Straits'"* e cada página apresenta os próximos dez registros, o custo total para recuperar toda a resposta é 98/10=9,8.

O algoritmo proposto faz uso de três estruturas: uma tabela de estatísticas, uma lista que armazena os valores de atributos já consultados e uma lista dos atributos de valores já recuperados, mas que ainda não foram submetidos como consultas. O objetivo da tabela de estatísticas é auxiliar a tomada de decisão sobre qual atributo utilizar na seleção de

consulta. A cada consulta realizada, os dados são extraídos e adicionados em um banco de dados local.

As informações adicionadas no banco de dados local servem de requisitos para a definição de um valor denominado *Query Harvest Rate*. Dado um banco de dados na DB na Web, e um banco de dados local $DBlocal$ contendo os registros que já foram recuperados de DB, a taxa de captura (*harvest rate*) de uma consulta q_i é definida como $HR(q_i) = (num(q_i, DB) - num(q_i, DB_{local})/cost(q_i, DB))$ em que $num(q_i, DB)$ e $num(q_i, DB_{local})$ correspondem ao número de registros que combinam com q_i em DB e $DBlocal$ respectivamente; $cost(q_i, DB)$ representa o custo de obter todas as páginas resultantes da consulta. O objetivo de uma solução ótima é selecionar o valor de atributo que possui a maior taxa de captura como a próxima consulta.

Além da taxa de captura para determinar a próxima consulta a ser realizada, Wu et al. (2006) afirmam que algumas heurísticas podem ajudar no desempenho. Muitas fontes na Web relatam o número total dos resultados de uma consulta na primeira página. Assim, é possível determinar o número de novos registros que estarão nas páginas seguintes e é possível abortar a consulta se a taxa de captura não alcançou determinado limiar. Outra heurística possível, quando o número total de resultados não é informado, é cancelar as consultas que recuperam um número significativo de registros duplicados nas primeiras páginas.

Outra questão que o trabalho de Wu et al. (2006) aborda é a inclusão de conhecimento de domínio para a seleção da consulta. Como a decisão para seleção de consulta é feita somente baseada nas informações estatísticas disponíveis no $DBlocal$, a estimativa baseada na taxa de captura pode não ser eficiente. Além disso, somente os atributos que estão no $DBlocal$ são considerados para uma futura formulação de consulta, o que pode ocasionar a recuperação de somente uma parte do banco de dados visto que os grafos podem não ser totalmente conectados.

O conhecimento de domínio pode ajudar a minimizar os problemas abordados. Bancos de dados de um mesmo domínio são similares tanto em seus atributos quanto nos valores dos atributos. As informações recuperadas em bancos de dados semelhantes podem ser utilizadas como consultas em outros bancos de dados de um mesmo domínio. Por exemplo, se existe alguma informação sobre as músicas da `Amazon.com`, podemos utilizar essas informações para consultar outras fontes de músicas.

Wu et al. (2006) definem que uma tabela de estatísticas de domínio DT, associada a um determinado domínio DM, consiste de uma coleção de entradas na forma $< q_i, P(q_i, DM) >$, em que q_i representa a consulta candidata e $P(q_i, DM)$ representa a probabilidade de domínio da consulta q_i ocorrer em DM. Com a tabela de estatísticas de domínio, surgem dois grupos de consultas que podem ser enviadas ao banco de dados Q_{DB} e Q_{DT}. Q_{DB} consiste das consultas que correspondem a valores descobertos no banco de dados DB e Q_{DT} corresponde às consultas da tabela de domínio DT, mas que ainda não foram enviadas aos formulários.

O trabalho proposto não apresenta nenhuma abordagem sobre a escolha de quais campos de um formulário devem ser preenchidos, pois o foco é na seleção de consultas com somente um predicado de igualdade. Os valores são submetidos a todos os campos do banco de dados para determinar a cobertura. Com relação aos valores que serão gerados para preenchimento dos campos, a abordagem considera o relacionamento de valores iniciais e estima a taxa de captura das páginas resultantes para determinar o valor a ser preenchido nos campos do formulário. Os experimentos realizados consistem em um grupo de simulação controlada em um banco de dados local, que utiliza quatro bancos

de dados estruturados e uma coleta real em um sítio da Web. Para avaliar os resultados, Wu et al. (2006) utilizam a cobertura como métrica. Os valores iniciais são gerados randomicamente para a busca no sítio da Web.

3.3 Heurísticas e Formulários Web Complexos

3.3.1 Lage et al. (2004)

O trabalho de Lage et al. (2004) apresenta um método para preencher formulários usando um conjunto de heurísticas e um repositório de amostras para automaticamente encontrar formulários, preenchê-los e coletar as páginas resultantes. O método inicia com a coleta partindo da página principal e busca os formulários de maneira cega. Um conjunto de heurísticas é utilizado para descartar formulários indesejados. A seguir, o método extrai os rótulos dos campos e por meio de um repositório de amostras, identifica como preencher os campos. Finalmente, os formulários preenchidos são submetidos para identificar as páginas de dados. A Figura 3.7 ilustra a abordagem proposta por Lage et al. (2004). Cada módulo está representado por uma caixa com linhas pontilhadas.

Um componente chave do método é o repositório de amostras. O repositório é usado para identificar evidências que indiquem a qual domínio o formulário pertence. O repositório é um conjunto de pares atributo-valor da forma $< label(f_i), dom(f_i) >$ que descrevem os objetos do domínio de aplicação. Os repositórios são gerados pela extração de dados de fontes na Web de domínios específicos. Existem várias formas de geração do repositório. Valores podem ser extraídos de um banco de dados ou da saída de algum programa ou, ainda, pode ser construído manualmente.

A tarefa de preenchimento consiste em encontrar um mapeamento entre os campos do formulário e os atributos do repositório. Para isso, heurísticas são utilizadas para selecionar os valores. As heurísticas extraem os rótulos que estão acima dos campos. Depois de identificar os rótulos, o repositório de amostras é usado para verificar se o formulário é relevante. Portanto, o método tenta encontrar mapeamentos entre os rótulos do formulário e os atributos do repositório. Antes de realizar o mapeamento, os atributos do repositório e os rótulos do formulário são normalizados, por meio de um algoritmo de *stemming* e remoção de *stopwords*. Os rótulos da linguagem HTML também são removidos. Se nenhum mapeamento é encontrado, o formulário é descartado.

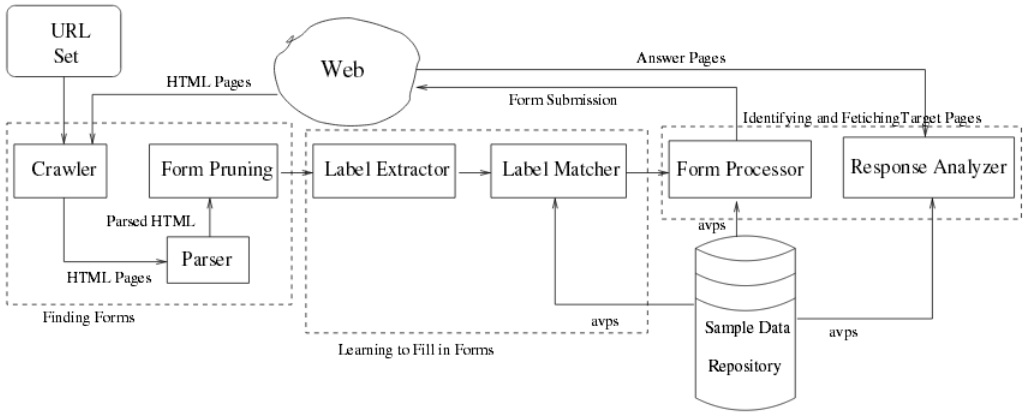


Figura 3.7 – Abordagem proposta por (LAGE et al., 2004)

É importante observar que o mapeamento é simples para restringir os formulários de busca, uma vez que é possível facilmente obter o mapeamento dos atributos, os quais estão explicitamente disponíveis nos rótulos HTML $< SELECT >$. Se o mapeamento não é encontrado, o agente sabe como preencher o formulário e todas as demais informações para a submissão. Experimentos foram realizados em trinta sítios da Web em três domínios: *books*, *CDs* e *software*.

3.3.2 Raghavan and Garcia-Molina (2001)

Raghavan and Garcia-Molina (2001) apresentam uma solução de coleta na Web Oculta com o objetivo de tratar o problema de extração de conteúdo de páginas dinâmicas. O *crawler* desenvolvido por Raghavan and Garcia-Molina (2001) é chamado HiWE (*Hidden Web Exposer*). A submissão dos formulários e as páginas geradas pelo *crawler* são armazenadas em um repositório. As páginas, estáticas e dinâmicas, presentes no repositório podem ser indexadas para busca. Além disso, as páginas escondidas podem ser acessadas sem a necessidade de submissão de consultas para a fonte original.

A Figura 3.8 mostra a arquitetura do HiWE. Ela é composta por seis módulos funcionais e por duas estruturas de dados internas. Uma das estruturas é a *URL List* que contém todas as URLs que foram descobertas pelo *crawler*. O módulo *Crawl Manager* controla o processo de captura, decidindo qual *link* visitar e realiza as conexões de rede. Para processar o formulário e extrair o conteúdo por trás do formulário existem os demais módulos e a estrutura da tabela LVS (*Label Values Set*). Os módulos *Form Analyzer, Form Processor* e *Response Analyzer* implementam a sequência de passos iterativos para recuperar as informações. O módulo *LVS Manager* é o responsável pelo gerenciamento da tabela LVS. A estrutura LVS é explicada em detalhes mais adiante.

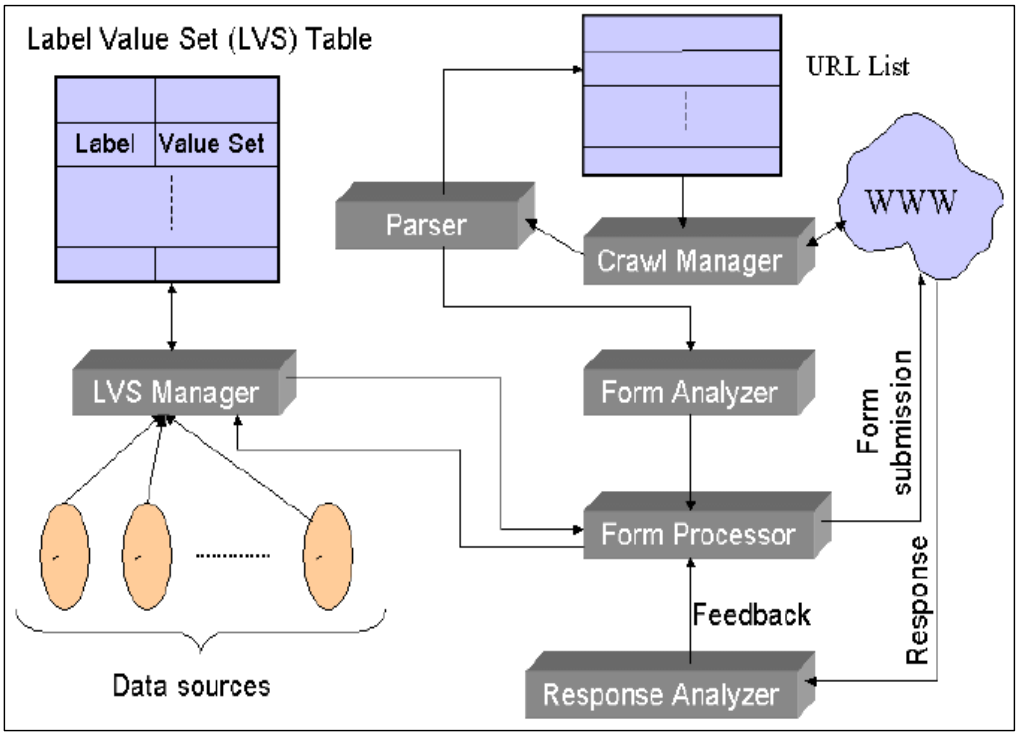


Figura 3.8 – Arquitetura HiWE (RAGHAVAN; GARCIA-MOLINA, 2001)

Para entender como Raghavan and Garcia-Molina (2001) preenchem formulários é preciso entender como os formulários são modelados. Um formulário F é modelado como um conjunto de pares (elemento, domínio), $F = (E_1, D_1), (E_2, D_2), ..., (E_n, D_n)$, em que E_is são os elementos (campos) e D_is são os domínios do elemento. Um elemento do formulário pode ser qualquer um dos objetos de entrada padrão: lista de seleção, caixas de texto, áreas de texto, *checkboxes* ou *radio buttons*. O domínio de um elemento é o conjunto de valores que podem ser associados com os correspondentes elementos do formulário. Alguns elementos têm domínio finito, em que o conjunto de valores válido está embutido na própria página do formulário. Por exemplo, se E_j é uma lista de seleção, indicado pelo rótulo<*SELECT*> no formulário HTML, então D_j é o correspondente conjunto de valores que está contido na lista. Outros elementos, como caixas de texto, têm valores infinitos. Ainda, muitos elementos de um formulário estão associados com algum texto descritivo que serve de ajuda para o entendimento semântico do elemento. Esses textos descritivos são os rótulos do elemento, e $label(E_i)$ representa o rótulo associado com o i^{th} elemento do formulário.

Um usuário preenche um formulário pela associação de um valor ou parte de texto com cada elemento do formulário. Para o preenchimento automático é preciso uma associação similar de valores aceitáveis para o domínio para cada elemento do formulário. A escolha de valores aceitáveis depende da semântica do elemento. Para elementos com domínios finitos pequenos, uma possibilidade é o preenchimento exaustivo dos valores. Por exemplo, para o campo *Country*, na Figura 2.2, o *crawler* irá recuperar todos os dados que estão associados ao campo, por exemplo, *GB*, *RS*, *AD*, etc (ver Figura 2.2, linhas 10 a 16). No entanto, para elementos infinitos, é preciso definir valores significativos para o domínio e relevantes para a aplicação. Por exemplo, para o campo *Town/County*, na Figura 2.2, o *crawler* deve selecionar somente valores referentes a nomes de cidades.

Raghavan and Garcia-Molina (2001) propõem para o preenchimento de elementos em um formulário uma tabela denominada *Label Value Set* (Figura 3.8). Cada linha na tabela *LVS* é da forma *(L,V)*, em que L é um rótulo e $V = v_1, v_2, ..., v_n$ é um conjunto de valores difusos (ZIMMERMANN, 1992) pertencentes ao rótulo. O conjunto V tem uma função M_V que associa pesos, no intervalo entre [0,1], para cada membro do conjunto. Cada v_i representa um valor possível que pode ser associado a um elemento E se $label(E)$ combinar com L. $M_V(v_i)$ representa uma estimativa de quão correto o valor v_i é em relação a E. A tabela *LVS* também suporta o conceito de apelidos e sinônimos para representar um mesmo conceito, como por exemplo, *cidade* e *município*. (RAGHAVAN; GARCIA-MOLINA, 2001) afirmam que a distância visual é uma estratégia bastante robusta e efetiva para extrair rótulos e domínios de atributos.

Dado um formulário $F = (E_1, D_1), (E_2, D_2), ..., (E_n, D_n)$, a geração de valores realizada pela combinação entre o rótulo de um elemento E_i e um rótulo v_i na tabela LVS é um conjunto V_i definido como:

- Se E_i é um elemento de domínio infinito e *(L,V)* representa uma entrada na tabela LVS em que L é similar com $label(E_i)$, então $V_i = V$ e $M_{Vi} = M_V$.
- Se E_i é um elemento de domínio finito, então $V_i = D_i$ e $M_{Vi}(x) = 1$, para todo e qualquer $x \in V_i$.

O conjunto de valores associados a cada elemento que irão compor as submissões a um formulário F e que estão contidos na LVS é dado por $ValAssign(F, LVS) = V_1XV_2X...XV_n$. Seja S_{max} o valor máximo de submissões para um formulário, o *crawler* seleciona o número de submissões como o $minS_max, |V_1| \times ... \times |V_n|$ para enviar

ao formulário. O valor de S_{max} é um parâmetro definido na inicialização do *crawler*.

Para identificar os rótulos de um dado elemento do formulário e os domínios de atributos, Raghavan and Garcia-Molina (2001) utilizam as seguintes heurísticas:

- Identificar os trechos de um texto que estão visualmente adjacentes ao elemento do formulário, nas direções horizontal e vertical. Para isso, baseado no leiaute, são computadas as distâncias entre os centros dos elementos do formulário e o centro dos trechos de texto. Essa etapa retorna uma lista de, no máximo, quatro candidatos possíveis.
- Se existem candidatos à esquerda e/ou acima do elemento, os candidatos à direita e abaixo são descartados.
- Se ainda existem dois candidatos, será escolhido aquele que está em negrito ou em um tamanho de fonte maior.
- Se o empate não for resolvido, então um dos dois candidatos é escolhido ao acaso.

A proposta de Raghavan and Garcia-Molina (2001), HiWE, possui quatro mecanismos para preencher a tabela LVS. O primeiro é a inicialização explícita da tabela LVS em que são fornecidos os rótulos e o conjunto de valores associados. Esses dados são carregados para a tabela LVS durante o processo de inicialização do *crawler*. Conforme o domínio de conhecimento, os rótulos e o conjunto de valores podem ser adaptados. O segundo é utilizar certas categorias comuns a todos os domínios, tais como, dias da semana, mês, etc., que podem ser úteis em vários tipos de aplicações. O terceiro mecanismo permite a comunicação da LVS com fontes de dados que podem ser consultadas para popular a tabela, por meio de uma interface bem definida. As interfaces possuem dois tipos de consultas: i. dado um rótulo, retornar um conjunto de valores que podem ser associados ao rótulo e ii. dado um valor, retornar outros valores que pertencem ao mesmo conjunto de valores. Finalmente, no momento de coletar os formulários, é possível verificar os valores de elementos e adicioná-los à tabela LVS. Esse mecanismo é útil com elementos de domínio finito.

Uma limitação do trabalho de Raghavan and Garcia-Molina (2001) é que a tabela LVS deve ser previamente preenchida por um usuário especializado, ou uma fonte de dados pode ser indicada para uma carga inicial. Existe também a possibilidade de se utilizar um conjunto de valores existente em um formulário para outro formulário. Por exemplo, um elemento de um formulário ser do tipo lista de seleção e em outro formulário ser do tipo caixa de texto. Se os elementos forem semelhantes é possível utilizar os valores da lista de seleção de um formulário como valores para a caixa de texto do outro formulário. Raghavan and Garcia-Molina (2001) executaram experimentos em 50 sítos. Os experimentos utilizam como métricas a cobertura, que representa a quantidade de informação recuperada e a eficiência de submissão que consiste na relação entre o total de submissões e o total de submissões que resultam em páginas resultado que contém um ou mais resultados válidos.

3.3.3 Liddle et al. (2003)

O trabalho proposto por Liddle et al. (2003) recupera dados que estão por trás de um formulário em particular. Essa tarefa é realizada por meio de três passos: (i) automatizar o preenchimento dos formulários; (ii) simplificar o processo de coleta de maneira eficiente e (iii) analisar os resultados encontrados para manipular erros de forma inteligente e descartar dados duplicados.

O preenchimento automático de formulários é realizado pela análise do formulário

HTML por meio da identificação dos rótulos de início e fim, <*form*> e <*/form*> respectivamente. Se o formulário está presente ele é extraído. Somente formulários com *action* do tipo *get* são suportados pelo método proposto por Liddle et al. (2003). As informações armazenadas sobre os formulários incluem a fonte URL da página, o número de campos e os detalhes de cada campo, incluindo nome, informações de domínio e as opções disponíveis para listas de seleção e valores *default*. Um formulário é representado como uma tupla $F =< U, (N_1, V_1), (N_2, V_2), ..., (N_n, V_n) >$, em que U é a URL e (N_i, V_i) são pares (nome,valor) que são enviados na submissão do formulário.

O preenchimento do formulário é iniciado com a associação de valores *default* para cada campo. Com as associações, é realizada a requisição do formulário. Existem vários tipos de campos e vários domínios para cada campo. Alguns campos oferecem valores *default*, por exemplo, valores do tipo `Indiferente`, `Todos`, `Qualquer`. Assim, é preciso ter um dicionário de valores *default* para campos, para determinar qual o valor *default* correspondente. A construção de consultas é feita para campos do tipo *selection lists*, *radio-buttons* e *check boxes*. Campos do tipo *text boxes* são ignorados e somente são utilizados, se o usuário associar valores às caixas de texto.

Após a submissão do formulário, são analisadas as informações recuperadas. A resposta mais comum é que o sítio retorne pequenos resultados a cada vez, mostrando n resultados por página. Normalmente existe um *link* ou um botão para recuperar a próxima página e assim por diante. A questão importante é se a consulta inicial, considerando valores *default*, recupera todos os dados ou apenas um fragmento do banco de dados. A estratégia utilizada envolve a análise da consulta *default* e a verificação da amostra resultante para verificar se a resposta da consulta *default* é abrangente o suficiente. Caso não seja, é preciso consultar exaustivamente até que um determinado limiar seja alcançado. Por exemplo, o tamanho do banco de dados escondido atrás do formulário pode ser estimado, e então são enviadas várias consultas até que um determinado limiar de completude seja atingido.

Liddle et al. (2003) apresentam cinco limiares que podem ser informados pelo usuário para interromper a submissão de consultas. Os limiares são: (i) a porcentagem de dados recuperados, (ii) o número de consultas enviadas, (iii) o número de bytes recuperados, (iv) a quantidade de tempo gasto e (v) o número de consultas que não retornaram dados. O limiar (i) verifica a qualidade do motor de busca e o limiar (v) mostra que a probabilidade de encontrar novos dados reduz significativamente quando o número de consultas aumenta. Cada um dos limiares constitui uma regra de parada sequencial que pode finalizar a busca antes de experimentar todas as consultas. Entende-se por todas as consultas o produto cartesiano de valores dos campos com domínios finitos. Campos com domínio infinito não são considerados no cálculo do produto cartesiano.

A última questão aborda o tratamento de exceções ou erros. Basicamente, duas situações são possíveis: (i). quando uma consulta não retorna dados ou (ii). quando um campo de texto é de preenchimento obrigatório. Para o primeiro caso o sistema continua e para o segundo caso é necessária a intervenção do usuário. O usuário aborta a operação ou fornece valores para o campo obrigatório e o sistema continua a execução da consulta. Para casos nos quais um erro ocorre o sistema mantém uma rotina de *timeout* para terminar a operação se alguma rotina anormal acontecer.

Para determinar o tamanho da amostra de consultas que serão submetidas, a solução de Liddle et al. (2003) caracteriza cada campo finito (não considera *text boxes*) como sendo um fator no espaço de busca. Assim, seja $f_1, f_2, ..., f_n$ os n fatores correspondentes aos campos do formulário e seja $|fi|$ o número de possibilidades de valores para o i^{th}

fator. O número total de combinações N para o formulário é dado por $N = \prod_{(i=1)}^{n} |f_i|$ e a cardinalidade do maior fator é dada por $c = max(|f_1|, |f_2|, ..., |f_n|)$. O tamanho da amostra de consultas C é dado por $C = max(c, log_2 N)$.

Outra questão sobre as consultas da amostra é a escolha das C consultas. Por exemplo, suponha que um formulário contenha dois campos com quatro e sete possibilidades de valores. Neste caso, o valor de C será $max(7, log_2 28) = 7$. Se forem escolhidas sete consultas de maneira aleatória pode acontecer de valores repetidos serem escolhidos e a amostra não ser representativa. Uma solução possível é manter uma trilha dos valores escolhidos para cada consulta. Se as C consultas produzirem dados novos, uma nova amostra de C consultas é gerada até que um determinado limiar seja alcançado ou todas as possibilidades tenham sido testadas.

O preenchimento de formulários proposto por Liddle et al. (2003) é independente de domínio e parte do princípio que uma consulta que associa valores *defaults* a todos os campos não textuais pode recuperar a base de dados escondida por trás do formulário. A técnica de preenchimento é ingênua, pois somente os campos que possuem domínios finitos são preenchidos. Campos que possuem valores infinitos não são preenchidos. Se, por acaso, forem obrigatórios é necessária a intervenção do usuário. É escolhida uma amostra de consultas e é verificada se a amostra escolhida recupera toda a base de dados. Os experimentos realizados incluem 15 diferentes sítios da Web de vários domínios de formulários.

3.3.4 Álvarez et al. (2007)

O trabalho proposto por Álvarez et al. (2007) apresenta uma arquitetura denominada *DeepBot* para *crawler* na Web Oculta. O protótipo desenvolvido recebe como entrada um conjunto de páginas "semente" e recursivamente obtém novas páginas. Quando uma nova página é encontrada, o formulário é examinado e é verificada qual a relevância dos campos com as definições de domínio. Se o formulário é considerado relevante, ele é usado para executar um conjunto de consultas definidas pelo domínio, com o objetivo de alcançar novas páginas.

Álvarez et al. (2007) utilizam o conceito de definições de domínio para determinar se um formulário é relevante. Uma definição de domínio (*domain definition*) é composta por três elementos:

- Um conjunto de atributos $A = a_1, a_2, ..., a_n$, no qual cada atributo a_i possui associado um nome, um conjunto de apelidos $a_i\_alias_1, ..., a_i\_alias_k$ e um índice de especificidade s_i.
- Um conjunto de consultas $Q = q_1, q_2, ..., q_m$ que são executadas para descobrir os formulários relevantes, em que cada consulta q_j é uma lista de pares (atributo, valor), onde o atributo é um atributo do domínio e valor é uma string.
- Um limiar de relevância denotado por μ.

Um atributo representa um campo que aparece em um formulário. Os apelidos representam rótulos alternativos que podem identificar um atributo em um formulário. Por exemplo, o atributo *pub name* (Figura 2.2), de um domínio sobre *pubs*, pode ter apelidos como *name*. O índice de especificidade, s_i, de um atributo a_i é um número entre 0 e 1 indicando a probabilidade do atributo ser relevante para o domínio. Por exemplo, em um domínio de coleção de livros, o atributo *ISBN* possui um alto valor (0,95) uma vez que esse atributo é específico para sítios de livros, enquanto um atributo como *preço* pode ter um valor baixo (0,50), uma vez que qualquer tipo de sítio possa ter esse atributo.

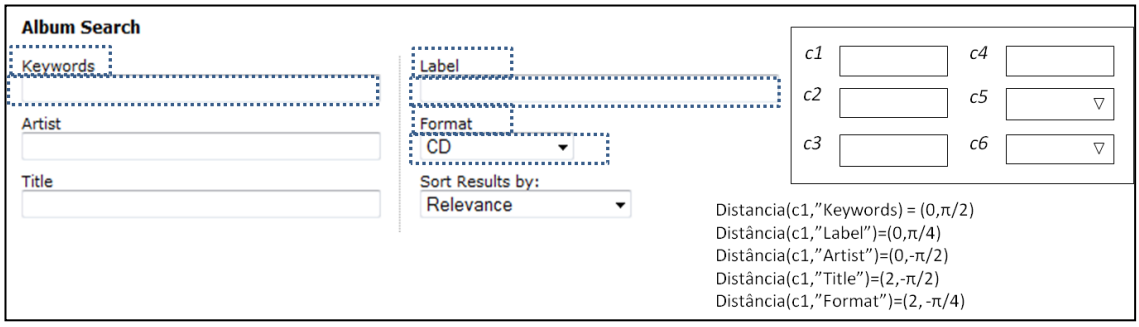


Figura 3.9 – Exemplo de Formulários — distâncias visuais e ângulos (ÁLVAREZ et al., 2007)

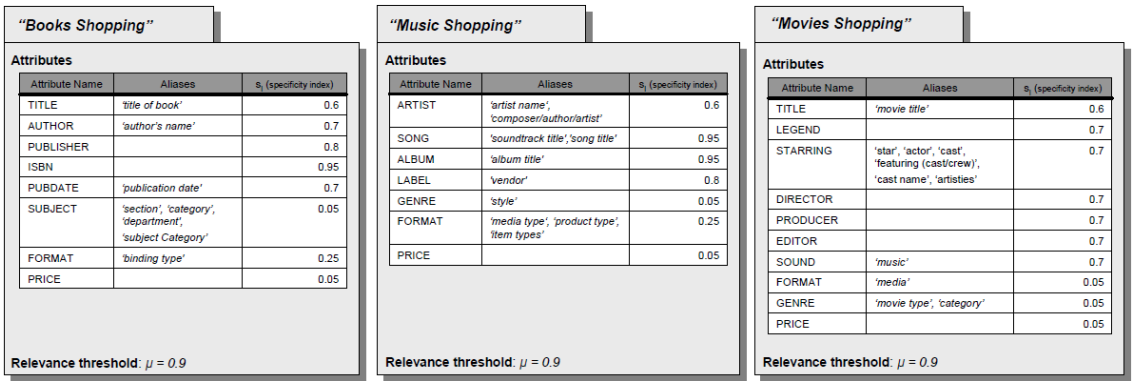
A proposta de Álvarez et al. (2007) analisa cada formulário por intermédio da associação de campos do formulário aos atributos do domínio, a relevância do formulário para o domínio específico e a execução das consultas no formulário. Para associar os campos do formulário com os atributos de um domínio considere um formulário f localizado em uma página HTML e um domínio d que descreve uma coleção de dados. O objetivo é determinar se o formulário f permite a execução de consultas para os atributos presentes em d. Para verificar essa questão, Álvarez et al. (2007) primeiro determinam quais os rótulos estão associados com quais campos do formulário por meio de heurísticas usando distância visual entre os campos e os rótulos. Na sequência, relacionam os campos do formulário f com os atributos de d por meio da medida de similaridade entre os rótulos associados aos campos de f e os rótulos associados com cada atributo na definição de domínio d.

A distância visual entre um campo c e um rótulo r não é computada em *pixels*, mas em uma unidade maior. Uma API foi utilizada para obter as coordenadas do retângulo em torno do campo c e o retângulo em torno de r. A distância visual é obtida pela distância mínima entre os retângulos. Além da distância é obtido o ângulo da menor linha entre os retângulos. O ângulo é aproximado e múltiplo de $\pi/4$. A Figura 3.9 mostra um exemplo para o campo $c1$ e suas distâncias e ângulos.

Partindo das distâncias e ângulos é possível determinar qual rótulo r está associado a um determinado campo c. Para selecionar os "melhores rótulos"para um determinado campo c, são aplicados três passos:

1. Todos os rótulos que possuem a menor distância d com o respectivo campo c são adicionados em uma lista.
2. Aqueles rótulos que possuem uma distância inferior a $k.d$ em relação ao campo c são adicionados em ordem de distância. (k é um fator configurável e usualmente é utilizado o valor 5). Esse passo descarta aqueles rótulos mais distantes de um campo.
3. Rótulos com a mesma distância são ordenados conforme o seu ângulo. A ordem privilegia rótulos que estão acima e à esquerda em relação aos rótulos que estão abaixo e à direita.

Dessa forma, tem-se uma lista ordenada de rótulos associada a cada campo do formulário. Álvarez et al. (2007) asseguram que um determinado rótulo está presente somente na lista de um campo e que cada campo tem pelo menos um rótulo associado. Em páginas reais, um campo de um formulário sempre tem algum rótulo associado para que o usuário



"Books Shopping"

Attributes

Attribute Name	Aliases	s_i (specificity index)
TITLE	*'title of book'*	0.6
AUTHOR	*'author's name'*	0.7
PUBLISHER		0.8
ISBN		0.95
PUBDATE	*'publication date'*	0.7
SUBJECT	*'section', 'category', 'department', 'subject Category'*	0.05
FORMAT	*'binding type'*	0.25
PRICE		0.05

Relevance threshold: $\mu = 0.9$

"Music Shopping"

Attributes

Attribute Name	Aliases	s_i (specificity index)
ARTIST	*'artist name', 'composer/author/artist'*	0.6
SONG	*'soundtrack title','song title'*	0.95
ALBUM	*'album title'*	0.95
LABEL	*'vendor'*	0.8
GENRE	*'style'*	0.05
FORMAT	*'media type', 'product type', 'item types'*	0.25
PRICE		0.05

Relevance threshold: $\mu = 0.9$

"Movies Shopping"

Attributes

Attribute Name	Aliases	s_i (specificity index)
TITLE	*'movie title'*	0.6
LEGEND		0.7
STARRING	'star', 'actor', 'cast', 'featuring (cast/crew)', 'cast name', 'artisties'	0.7
DIRECTOR		0.7
PRODUCER		0.7
EDITOR		0.7
SOUND	*'music'*	0.7
FORMAT	*'media'*	0.05
GENRE	*'movie type', 'category'*	0.05
PRICE		0.05

Relevance threshold: $\mu = 0.9$

Figura 3.10 – Definições de Domínio (ÁLVAREZ et al., 2007)

possa identificar qual a sua função. Por exemplo, se a lista de um campo c_1 contém os rótulos r_1 e r_2 (nessa ordem) e a lista de um campo c_2 somente contém r_1, então r_1 é removida da lista de c_1.

Até o momento, foram determinados os rótulos e seus campos correspondentes. Agora é preciso associar os campos com os atributos de domínio e, dessa forma, preencher os campos com os valores. Álvarez et al. (2007) distinguem dois tipos de campos: campos delimitados (*bounded fields*) e campos irrestritos (*unbounded fields*). Campos delimitados são aqueles que possuem uma lista finita de valores possíveis para consulta, e campos irrestritos são aqueles que não possuem valores finitos, tais como, caixas de texto.

A ideia básica para classificar a similaridade entre um campo c e um atributo a é medir a similaridade textual entre os rótulos associados com c na página e os rótulos associados com a no domínio (o nome do atributo e seus apelidos). Quando o campo é delimitado, o sistema também leva em conta as semelhanças entre os valores possíveis do campo c na página e os valores de entrada da consulta para o atributo a no domínio. As medidas de similaridade de texto são obtidas pela combinação dos algoritmos TF-IDF e Jaro-Winkler, conforme proposto por Cohen, Ravikumar and Fienberg (2003).

O resultado obtido é uma tabela com as similaridades estimadas entre cada campo do formulário e cada atributo. Então, são descartados os pares que não alcançam um valor de limiar mínimo. A saída dessa etapa é um conjunto de associações entre campos do formulário e atributos de domínio.

O próximo passo antes de preencher o formulário é verificar se o formulário é relevante para um determinado domínio. O conjunto de associações entre os campos e os atributos possui uma determinada confiança, expressa por um número entre zero e um. Para cada associação A_i tem-se uma confiança c_i. Para determinar se o formulário é relevante para um determinado domínio, é realizado o somatório do produto entre o índice de especificidade s_i do atributo envolvido e a confiança c_i da associação A_i e verificado se o valor é maior que o limiar de relevância μ do domínio sendo considerado. Se o valor for superior, o formulário é relevante. A Figura 3.10 mostra um exemplo de definições de domínio apresentadas por Álvarez et al. (2007).

O trabalho proposto por Álvarez et al. (2007) não utiliza nenhuma técnica para verificar quais os campos de um formulário devem ser preenchidos. Apenas associa os rótulos presentes em um formulário com atributos de um determinado domínio e, partindo da associação, atribui valores para os campos com o objetivo de executar consultas nos dados escondidos por trás do formulário. Os valores para preenchimento de campos em formu-

lários é realizado de maneira similar ao trabalho de Raghavan and Garcia-Molina (2001), em que uma tabela de definições de domínio contém os valores a serem associados aos campos do formulário. Álvarez et al. (2007) não apresentam experimentos para avaliação do método de preenchimento de formulários.

3.4 Aprendizagem de Máquina e Formulários Web Complexos

3.4.1 Toda et al. (2010)

Toda et al. (2010) apresentam uma solução para preenchimento de formulários partindo de um documento textual. A abordagem de Toda et al. (2010), chamada $iForm$, consiste na seleção automática de segmentos (conjuntos de *tokens*) de um documento textual e na associação desses segmentos com os campos do formulário. Assim, o usuário informa um documento textual ou partes de um documento e o *iForm* automaticamente extrai valores para o preenchimento do formulário baseando-se no conhecimento de informações de valores anteriores associados a cada campo. Usuários podem verificar o formulário preenchido, corrigir os campos, se for o caso, e proceder com a submissão dos valores no formulário. Após a submissão, os novos valores associados são armazenados e usados como novas evidências quando um novo documento textual for fornecido pelo usuário.

O preenchimento automático de formulários definido por Toda et al. (2010) é dividido em dois subproblemas: i. extração de valores de um documento textual e ii. preenchimento dos campos de um formulários usando os valores extraídos. Um documento textual é definido como uma sequência de tokens $t_1, ..., t_N$, que representam palavras. A tarefa de extração consiste na identificação de segmentos do documento textual, ou seja, uma sequência de *tokens* contínuos. Um segmento s_{ij} é composto por tokens consecutivos, tal que $i \leq j$, $i \geq 1$ e $j \leq N$. Um conjunto válido de valores extraídos do documento textual segue duas condições: i. um segmento está associado a somente um campo do formulário e ii. todo segmento extraído não possui sobreposição, ou seja, não existe um segmento s_{ab} e s_{cd} para $a < c$ tal que $b \geq c$.

A abordagem *iForm* consiste em avaliar os segmentos candidatos do documento textual e estimar a probabilidade condicional de um campo dado o segmento. Os autores consideram dois tipos de características dos valores previamente submetidos nos formulários: i. características associadas ao conteúdo, tais como, os valores propriamente ditos e os *tokens* que compõem esses valores e ii. características associadas ao estilo dos valores, tais como, pontuação, capitalização, etc.

Assim, Toda et al. (2010) associam segmentos de texto para campos de um formulário usando três características distintas. Toda et al. (2010) modelam a computação da probabilidade de um campo f_j, dado um segmento S_{ab}, por meio de um modelo de rede bayesiana de maneira similar à proposta de Ribeiro and Muntz (1996) para ranking de documentos em tarefas de busca. Uma ilustração da rede proposta por Toda et al. (2010) é mostrada na Figura 3.11.

Segmentos e campos de um formulário são representados por *tokens*, valores completos e estilos. O nó S_{ab}, no topo da rede, representa um segmento de um documento textual. Na parte de baixo (Figura 3.11), cada nó F_j modela um campo do formulário que está sendo preenchido. Nós de V_1 a V_q representam os valores distintos submetidos anteriormente para os campos, nós de T_1 a T_w modelam os *tokens* que aparecem nesses valores e os nós E_1 a E_r representam cada um dos distintos símbolos das máscaras encontrados quando o processamento dos valores submetidos anteriormente foi realizado. Os vetores

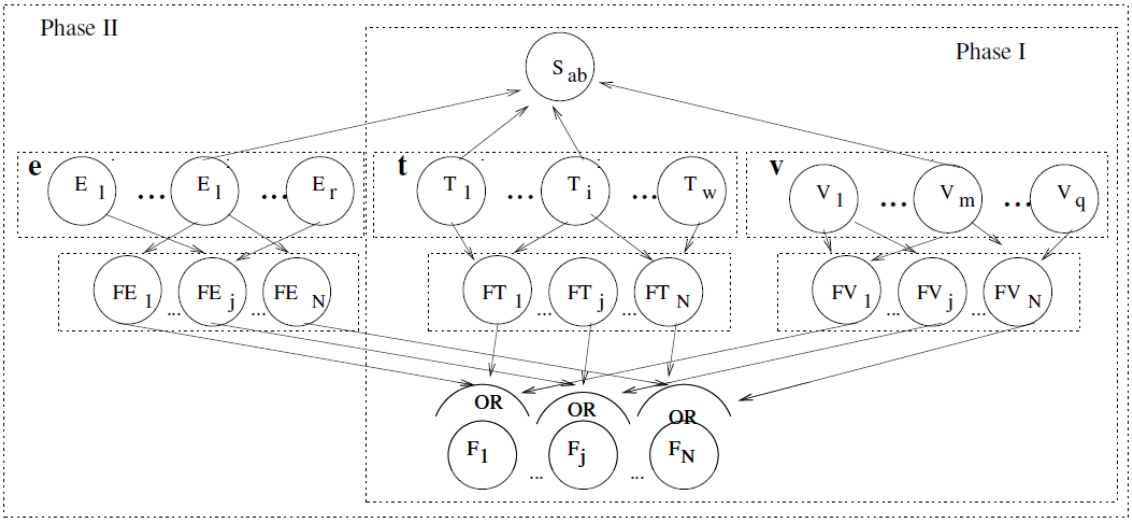


Figura 3.11 – Modelo de Rede Bayesiana proposto por (TODA et al., 2010)

v, t e e são usados para referenciar qualquer possível estado dos nós raiz de V_1 a V_q, T_1 a T_w, e E_1 a E_r respectivamente.

Os nós de FV_1 a FV_N modelam a probabilidade de cada campo F_1 a F_N dada a ocorrência de um valor no segmento S_{ab}, isto é, a ocorrência da *string* inteira que representa o segmento. De maneira semelhante, FT_1 a FT_N modelam a probabilidade dos campos dada a ocorrência de um conjunto distinto de *tokens* encontrado no segmento e, os nós de FE_1 a FE_N modelam a probabilidade dos campos, dada a ocorrência de um estilo que combine com a máscara do segmento. As informações disponíveis são combinadas por meio de um operador OR para computar a probabilidade final de cada campo dado um segmento S_{ab}.

A estrutura de dados proposta por Toda et al. (2010) considera valores utilizados em submissões anteriores do formulário e as características dos valores extraídos partindo de uma entrada textual para determinar os novos valores a serem associados aos campos do formulário. A principal limitação é que a solução depende de valores anteriores e de uma entrada textual. Além disso, o preenchimento de formulários proposto por Toda et al. (2010) não se aplica a interfaces de busca. Nos experimentos, foram utilizados cinco formulários com múltiplos atributos para cadastro de informações.

3.5 Combinações Sobrepostas

3.5.1 Jiang et al. (2009) e Zheng et al. (2013)

O trabalho proposto por Jiang et al. (2009) apresenta um método para busca na Web Oculta em que cada palavra obtida nas páginas de resultado é codificada como uma tupla representando sua linguística, estatística e características HTML. Essa tupla é utilizada para selecionar valores para uma entrada textual livre. As palavras selecionadas são usadas para treinar um modelo usando algoritmos de aprendizagem de máquina. Esse modelo é usado para avaliar a taxa de seleção (*hr - harvest rate*) de novas palavras-chave.

Jiang et al. (2009) utilizam um algoritmo adaptativo para realizar a coleta na Web Oculta. A questão chave é como estimar a taxa de seleção baseado nos dados recuperados

1: Select the keyword with maximum harvest rate from candidate set;
2: If termination condition is satisfied Exit;
Else generate and issue a query to the database using the selected keyword;
3: Download and parse response pages then compute the harvest rate of the current query;
4: Renew the training set and the candidate set;
5: If error > s, retrain the learning model;
6: Estimate the harvest rate for each keyword in candidate set; Goto Step 1;

Figura 3.12 – Algoritmo Adaptativo (JIANG et al., 2009)

até o momento. A ideia básica do algoritmo é estimar as taxas de seleção para palavras não utilizadas baseada nas submissões anteriores. O algoritmo proposto por Jiang et al. (2009) é mostrado na Figura 3.12.

Para inicializar o algoritmo, o *crawler* tem que adicionar valores iniciais para o conjunto de candidatos. Geralmente, essas palavras são obtidas de páginas Web dos formulários de consulta. O passo 2 determina a condição de parada da busca. O processo de análise, no passo 3, extrai e segmenta todas as palavras-chave que aparecem em novos registros e computa a taxa de seleção da consulta submetida. Essa informação é utilizada para atualizar o conjunto de treinamento e de candidatos. O conjunto de treinamento T_r compreende o conjunto de tuplas (q, hr) de todas as consultas submetidas, em que q é uma consulta e hr é sua taxa de seleção. Dada a consulta corrente, o conjunto candidato C é o conjunto das palavras disponíveis para serem submetidas. Os elementos em T_r e C são codificados em um *Vector Space Model* e a principal diferença entre eles é que hr em T_r é conhecido enquanto que em C não.

O modelo será atualizado quando o erro quadrático acumulado (Equação 3.7) alcançar um limiar s no passo 5 do algoritmo.

$$error = \sum_{j=t}^{i} (hr_j - ehr_j)^2 \tag{3.7}$$

em que, hr_j é a taxa de seleção da j^{th} consulta enquanto que ehr_j é a taxa de seleção estimada pelo modelo de aprendizagem até a consulta $j - 1^{th}$. O valor t é o número da consulta quando o mais recente modelo de aprendizagem foi treinado, e i é o número da consulta atual. Finalmente, o passo 6 tenta estimar a taxa de seleção das consultas que estão no conjunto candidato generalizando partindo do conjunto de treinamento.

Uma questão importante para o método apresentado por Jiang et al. (2009) são os tipos de características extraídas das palavras para compor o conjunto de treinamento e codificar a taxa de seleção. As características foram dividas em três grupos: linguística, estatística e HTML. As características linguísticas incluem o tipo de palavra (verbo, substantivo, adjetivo, etc), o comprimento da palavra, ou seja, o número de caracteres da palavra, e o idioma da palavra. As características estatísticas incluem o TF (*term frequency*), DF (*Document Frequency*), TF-IDF (*term frequency-inverse document frequency*) e RIDF

(*Residual IDF*) da palavra. As características HTML incluem os rótulos HTMLs na qual a palavra está inserida, a localização dentro da árvore DOM, e *markedness* que significa o destaque da palavra dentro do conteúdo HTML. *Markedness* leva em consideração se a palavra está em negrito, itálico, sublinhada ou outro tipo de destaque.

Os experimentos apresentam uma melhoria em comparação com a combinação dos métodos Madhavan et al. (2008) e Barbosa and Freire (2004). Os experimentos realizados por Jiang et al. (2009) utilizam como algoritmo de aprendizagem de máquina somente o kNN (SHAKHNAROVICH; INDYK; DARRELL, 2006) e foram realizados em apenas dois formulários e, somente em um campo textual do tipo de domínio infinito (*text box*). Os experimentos nos dois formulários alcançaram uma média de 70% de cobertura para o método proposto por Jiang et al. (2009) enquanto que a combinação de Madhavan et al. (2008) e Barbosa and Freire (2004) alcançou uma cobertura de 50% para o mesmo número de submissões. Em Jiang et al. (2010) é apresentada uma solução que utiliza aprendizagem por reforço para a coleta na Web Oculta. Os experimentos de Jiang et al. (2010) foram realizados em entradas textuais livres. Zheng (ZHENG et al., 2013) estende o trabalho de Jiang et al. (2009), Jiang et al. (2010) apresentando um algoritmo de aprendizado por reforço *Q-value* que permite ao *crawler* selecionar um valor por meio da aprendizagem de valores submetidos anteriormente. A recompensa dos valores é estimada pela métrica de *Q-value*.

3.5.2 Dong and Li (2012)

Dong and Li (2012) apresentam uma abordagem de busca na Web Oculta baseada em um modelo de aprendizagem construído partindo das taxas de busca (*harvest rate*), de maneira similar ao trabalho proposto por Wu et al. (2006). O método de Dong and Li (2012) utiliza um banco de dados local, DB_{sample}, para extração de características e construção de um modelo de aprendizagem que é utilizado para busca no banco de dados na Web. A abordagem proposta por Dong and Li (2012) funciona para formulários com entrada textual livre.

Dong and Li (2012) extraem uma amostra do banco de dados destino e seleciona vários tipos de características dessa amostra. Por meio dessas características, a abordagem proposta aprende um modelo de consulta. Finalmente, o modelo é usado para selecionar consultas para submeter ao banco de dados da Web até uma determinada condição de parada. O banco de dados de amostra DB_{sample} é obtido partindo da escolha de um atributo textual do banco de dados DB. Dessa forma, Dong and Li (2012) utilizam a abordagem proposta em Cao et al. (2006) para gerar o DB_{sample}. Depois de obter o DB_{sample}, o conjunto de treinamento é construído. A abordagem para geração do conjunto de treinamento é simular a busca no DB_{sample}.

Para formalizar o problema de busca na Web Oculta, Dong and Li (2012) utilizam um modelo de custo e uma taxa de cobertura. O custo é o número total de comunicações entre o *crawler* e o servidor Web. Assim, o custo é dado por $cost(q_i, DB) = (|R(q_i, DB|)/k$, em que $|R(q_i, DB|$ é o número total de registros em ***DB*** que combinam com q_i e k corresponde ao número máximo de registros mostrados em cada página resultado. A taxa de cobertura de uma consulta q_i é definida como $CR(q_i, DB) = (|R(q_i, DB)|)/(|DB|)$ em que $|DB|$ corresponde ao número total de registros no banco de dados real na Web. Portanto, o problema é encontrar uma lista de consultas $Q = q_1, q_2, ..., q_n$ que maximiza $CR(q_1 \cup ... \cup q_n, DB)$, sobre a restrição $\sum_{i=1}^{n} cost(q_i, DB) \leq t$, onde $CR(q_1 \cup ... \cup q_n, DB)$ denota a taxa de cobertura de Q, $cost(q_i)$ indica o custo da consulta q_i e t é o número máximo de consultas.

O objetivo é selecionar as consultas que recuperam mais linhas como menor custo. Dong and Li (2012) definem uma métrica chamada *query harvest rate* (HR) para capturar a produtividade de cada consulta candidata. Essa fórmula é semelhante à fórmula definida em (WU et al., 2006). A métrica $HR(q_i, DB)$ é dada pela Equação 3.8:

$$HR(q_i, DB) = \frac{|R(q_i, DB)| - |R(q_i, DB_{sample})|}{cost(q_i, DB)} \tag{3.8}$$

Ainda, Dong and Li (2012) substituem na equação 3.8 o valor de $cost(q_i, DB)$ por $cost(q_i, DB) = (|R(q_i, DB|)/k$ e a equação pode ser reescrita pela Equação 3.9:

$$HR(q_i, DB) = k \times (1 - \frac{|R(q_i, DB_{sample})|}{|R(q_i, DB)|}) \tag{3.9}$$

O problema na solução é que o valor de $|R(q_i, DB)|$ não pode ser calculado no processo de busca, pois somente o DB_{sample} é conhecido. Portanto, a solução extrai características do DB_{sample} para gerar uma base de treinamento. A diferença entre Dong and Li (2012) e Wu et al. (2006) é que o último utiliza heurísticas para determinar o valor de $|R(q_i, DB)|$. Outra observação na técnica utilizada por Dong and Li (2012) é que para calcular a taxa de cobertura é preciso saber o valor de $|DB|$ e, na maioria dos bancos de dados disponíveis na Web essa informação não está disponível. Dong and Li (2012) realizaram experimentos em três bancos de dados e escolheram um campo em cada *database* para ser utilizado como atributo para a coleta, isto é, o campo do formulário para preenchimento foi previamente selecionado.

3.5.3 Madhavan et al. (2008)

Madhavan et al. (2008) tratam sobre a possibilidade de gerar as páginas dinamicamente de maneira *off-line* e em seguida indexá-las como qualquer página HTML, utilizando as vantagens dos motores de busca tradicionais. Assim, é possível, segundo Madhavan et al. (2008), aumentar a acessibilidade do conteúdo da Web Oculta para os usuários de máquinas de busca tradicionais.

Um problema é que, geralmente, formulários HTML possuem mais de uma entrada (campo) e o preenchimento ingênuo dessas entradas pode provocar uma explosão de possibilidades, mais especificamente, o produto cartesiano das entradas e seus valores. Ainda, muitas das possibilidades retornarão resultados vazios, o que não é interessante. Dessa maneira, Madhavan et al. (2008) propõem um teste que avalia as combinações de entradas do formulário, denominado teste de informatividade (*informativeness test*). As combinações de entradas de um formulário são chamadas de *query templates*. Os *templates* são examinados com diferentes conjuntos de valores para cada entrada e é verificada se as páginas HTML obtidas são suficientemente distintas uma da outra. *Templates* que geram páginas distintas são considerados bons candidatos para a busca.

A Figura 3.13 apresenta um esquema para o modelo proposto por Madhavan et al. (2008). Partindo da extração do formulário HTML são gerados todos os *templates* candidatos conforme o número de campos existentes no formulário. Para cada *template* candidato, são selecionados os valores de preenchimento conforme a característica de cada campo presente no *template*. A seguir, as instâncias do *template* são montadas, juntando-se os campos do *template* candidato com os valores escolhidos para preenchimento. As instâncias do *template* são submetidas e o algoritmo ISIT (*Incremental Search for Informative query Templates*) determina se o *template* é informativo.

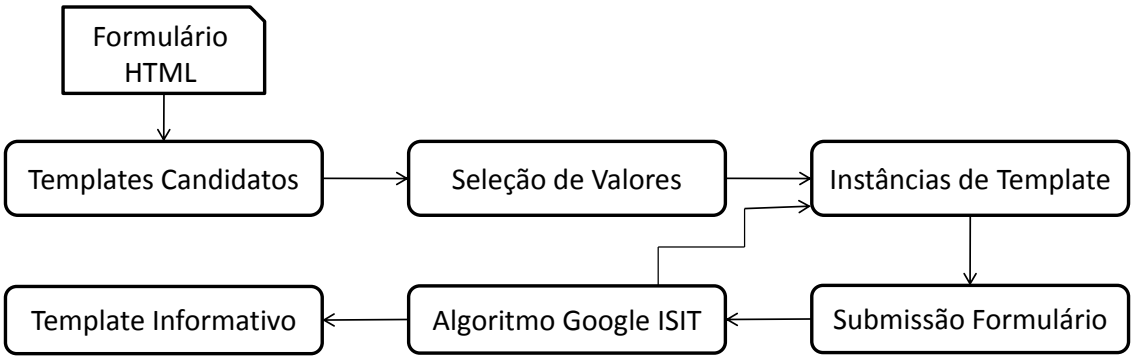


Figura 3.13 – Arquitetura Google Deep Web (MADHAVAN et al., 2008)

Madhavan et al. (2008) modelam o conteúdo escondido atrás dos formulários como um banco de dados D com uma única tabela de m atributos. O formulário f_D, usado para consultar D, possui n entradas, $X_1, ..., X_n$. Uma submissão do formulário consiste em encontrar os valores para cada entrada e retornar o subconjunto de registros em D.

O conceito de *query template* pode ser expandido como sendo a composição de um subconjunto de entradas do formulário f_D, denominado de *binding inputs*. As demais entradas são chamadas de *free inputs*. Múltiplas submissões de formulários podem ser geradas pela associação de diferentes valores para os *binding inputs*. Madhavan et al. (2008) respondem às questões de preenchimento por meio da seleção de um conjunto apropriado de *query templates* e pela seleção de valores de entrada apropriados para os *binding inputs*, isto é, a instanciação dos *query templates* com os valores dos campos. Para campos com menus de seleção todos os valores no menu são utilizados, mas para campos texto, os valores não são conhecidos e não existe nenhum conhecimento a priori sobre o domínio no qual o formulário está inserido. Madhavan et al. (2008) assumem que o conjunto de valores com a qual uma entrada é instanciada é o mesmo para todos os *query templates* nos quais a entrada aparece.

Para a seleção dos *templates*, existem duas preocupações. A primeira é selecionar aqueles *templates* que não possuem *bindings inputs* de apresentação, pois esses *templates* recuperam as mesmas informações que outros *templates* sem os *bindings inputs* de apresentação. A segunda é escolher a dimensão correta para os *templates*. Uma estratégia poderia ser a escolha de *templates* com a maior dimensão possível, ou seja, com muitos *bindings inputs* quanto possível. Tais *templates* garantiriam uma cobertura máxima gerando todas as consultas possíveis. No entanto, esse método tem como desvantagem o aumento do tráfego e, provavelmente, resultaria em muitos resultados vazios.

A escolha de *templates* com dimensão muito pequena tem a vantagem de gerar poucas submissões do formulário. Ainda, se caracteres "curingas"puderem ser escolhidos para todas as entradas, uma grande cobertura pode ser alcançada. No entanto, essas submissões de formulário devem gerar um grande número de registros. Isso pode ser um problema, pois na prática, vários sítios da Web limitam o número de registros recuperados por consulta. Essa limitação poder ser realizada por *links* do tipo *next*, *more* ou até mesmo pelo truncamento do resultado.

Para determinar os *templates* a serem utilizados, Madhavan et al. (2008) apresentam o conceito de clareza ou nitidez (*distinctness*) das páginas Web resultantes das diversas submissões do formulário. O número de páginas Web distintas que o *template* gera é

estimado pelo agrupamento delas baseado na similaridade do seu conteúdo. Se o número de páginas Web distintas é pequena em comparação com o número de submissões do formulário é provável que alguma das situações a seguir ocorra:

- O *template* inclui entradas de apresentação e, de fato, múltiplas páginas possuem o mesmo conteúdo.
- A dimensão do *template* é muito alta para os dados do banco de dados e existem muitas páginas sem registros.
- Existe um problema no *template*.
- Existe um problema no formulário que conduz a páginas de erros.

A medida de nitidez é realizada pela comparação do conteúdo das páginas Web recuperadas partindo das submissões dos formulários. Se as páginas geradas são suficientemente distintas, é dito que o template é informativo. De forma mais específica, são computadas as assinaturas para o conteúdo de cada página resultante da submissão e os *templates* são considerados não informativos se eles computam muito menos assinaturas que o número de possíveis submissões. Dessa forma, é criado o conceito de informatividade (*informativeness*) da seguinte forma: seja T um *query template* e *Sig* uma função que computa assinaturas para páginas HTML. Seja G o conjunto de todas as possíveis submissões para um formulário geradas pelo template T e seja S o conjunto $Sig(p)/p \in G$.

O *template* T é informativo se $|S|/|G| \geq \tau$. Caso contrário, o *template* T é não informativo. A taxa $|S|/|G|$ é chamada de fração de nitidez (*distinctness*). O valor do limiar τ é definido experimentalmente por Madhavan et al. (2008). A função da assinatura das páginas HTML não deve levar em consideração questões como formatação, ordenação de resultados e informações que não são importantes para o propósito do trabalho, como propagandas, sobretudo em sítios comerciais. A função também não deve considerar os valores aplicados nas entradas do formulário. Madhavan et al. (2008) não esclarecem como a função de assinatura é calculada para determinar as assinaturas das páginas HTML.

Para evitar o teste de todos os possíveis *templates* $(2^n - 1)$, o método realiza a busca de *templates* considerados informativos de maneira *bottom-up*, ou seja, partindo de *templates* com dimensão 1 e aumentando a dimensão conforme a medida de informatividade. A intuição principal que conduz essa estratégia é que a informatividade de um *template* é provavelmente dependente dos *templates* que ele estende, isto é, com um *binding input* adicional. O algoritmo utilizado é mostrado na Figura 3.14.

O algoritmo mostrado na Figura 3.14 inicia com os *templates* candidatos de dimensão 1. O método *GetCandidateInputs* contém as entradas do formulário W que serão analisadas, ou seja, contém as listas de seleção e caixas de texto. As entradas de texto são preenchidas se os valores são conhecidos e as demais entradas são preenchidas com valores *default*. O método *checkInformative* verifica se os *templates* candidatos são considerados informativos. Se o *template* for considerado informativo, o método *Augment* constrói os *templates* candidatos de dimensão 2. Cada novo candidato é testado para verificar se é informativo. Caso afirmativo, são gerados os candidatos de dimensão 3 e, assim sucessivamente até que não existam mais candidatos considerados informativos.

Para executar os experimentos, Madhavan et al. (2008) utilizam alguns refinamentos práticos para diminuir a escala do algoritmo. Madhavan et al. (2008) filtraram a dimensão dos *templates*, pois constataram que *templates* com dimensão maior que três eram quase sempre não informativos. Ainda, quando o número de URLs geradas partindo de um template é muito grande, somente uma amostra é utilizada. Com base em experimentos,

```
GetInformativeQueryTemplates (W: WebForm)
Ī : Set of Input = GetCandidateInputs(W)
   candidates: Set of Template  { T: Template | T.binding = {I}, I ∈ Ī}
informative: Set of Template = Φ
while (candidates  <> Φ)
  newcands: Set of Template = Φ
  foreach (T: Template in candidates)
    if ( CheckInformative(T, W) )
       informative = informative  U  { T }
       newcands = newcands   U  Augment(T, I)
   candidates = newcands
return informative

Augment (T: Template, I: Set of Input)
 return { T' | T'.binding = T.binding [ {I}, I ∈ P, I ∈ T.binding }
```

Figura 3.14 – Algoritmo ISIT (MADHAVAN et al., 2008)

o número da amostra foi fixado em 200.

A próxima questão que deve ser analisada é a geração de valores para as entradas do formulário. Para entradas do tipo listas de seleção, os valores para preenchimentos são extraídos do próprio formulário HTML. Um grande número de formulários HTML possui caixas de texto. Os autores tratam as caixas de texto de duas formas diferentes. Na primeira, as palavras preenchidas são usadas para recuperar todos os documentos contidos no banco de dados que contenham aquela palavra. Exemplos comuns desse tipo são campos para busca de nomes de *pubs* (*pub name*) e nomes de ruas (*street*) (ver Figura 2.2). A segunda, as caixas de texto estão associadas a valores com um conjunto bem definido, tais como, unidades da federação ou CEPs, ou podem ser instâncias de valores contínuos, por exemplo, preços ou datas. Assim, o método divide as caixas de texto em dois tipos: genéricas e tipadas. Entradas inválidas para caixas de texto tipadas geralmente levam a uma página de erro e, por isso, é importante identificar os dados corretos para esses tipos de entrada.

Para o preenchimento de caixas de texto genéricas é preciso identificar boas palavras-chave. Uma solução seria possuir uma lista de palavras em vários domínios para preencher as caixas de texto. Essa estratégia não é muito prática visto que existem muitos domínios e muitos conceitos. Além disso, para campos de texto, mesmo se forem identificadas entradas em dois formulários diferentes, com o mesmo conceito dentro do mesmo domínio, não necessariamente as mesmas palavras-chave sejam de boa utilização em ambos os formulários. Como um dos objetivos do trabalho proposto por Madhavan et al. (2008) é alcançar um grande número de sítios, essa estratégia não é utilizada.

Madhavan et al. (2008) utilizam um método iterativo para encontrar as palavras-chave para as caixas de texto. Por meio de um conjunto inicial de palavras para as caixas de texto e a utilização de um *query template* com dimensão 1, contendo a caixa de texto, são geradas as submissões para extrair palavras-chave adicionais. As palavras extraídas

são utilizadas para atualizar os valores candidatos para as caixas de texto. Essa técnica é repetida até que seja alcançada uma condição de parada. No final, um subconjunto das palavras extraídas é escolhido como o conjunto de valores para a caixa de texto.

Para determinar as palavras-chave que serão utilizadas pela caixa de texto, o algoritmo aplica o teste para saber se o template que possui a caixa de texto é informativo ou não partindo dos valores iniciais. Madhavan et al. (2008) salientam que se o template não passa no teste existe uma grande probabilidade de a caixa de texto não ser do tipo genérica. Os valores iniciais são gerados partindo da própria página do formulário e as demais são geradas das diversas submissões do formulário.

Como a quantidade de palavras é muito grande, é utilizada a medida TF-IDF (SALTON; MCGILL, 1986) e seleciona as $N_{initial}$ palavras iniciais na página do formulário classificadas conforme o valor de TF-IDF. Para selecionar as palavras candidatas na iteração i+1, considere W_i o conjunto de todas as páginas geradas e analisadas até a iteração i e C_i, o conjunto de todas as palavras que aparecem entre as N_{probe} em qualquer página do conjunto W_i. Do conjunto C_i, são eliminadas palavras que ocorrem na maioria das páginas, tais como propagandas e menus e palavras que ocorrem em somente uma página e que não representam o conteúdo do sítio. As palavras restantes em C_i são as novas palavras-chave candidatas para a iteração i + 1. A escolha de $N_{initial}$ e N_{probe} determinam a agressividade do algoritmo. A escolha de valores pequenos pode não extrair palavras-chave suficientes, e valores altos podem gerar muitas palavras-chave não representativas. Os experimentos indicam $N_{initial} = 50$ e $N_{probe} = 25$ como bons valores.

Para caixas de texto tipadas foi observado que existem poucos tipos e que são usados em vários formulários. Por exemplo, o `CEP` (Código de Endereçamento Postal) é utilizado em muitos formulários como automóveis, imóveis, registros públicos. O tipo que representa uma data é frequentemente utilizado em muitos domínios também. Assim, uma caixa de texto tipada pode produzir bons resultados se os valores apropriados forem utilizados. Dessa forma, pode-se utilizar esses campos para realizar o teste de informatividade para valores conhecidos de tipos populares. As caixas de texto tipadas podem ser classificadas em tipos finitos e contínuos. Tipos finitos, por exemplo, são `CEPs` e abreviações de unidades da federação e podem ser testados por meio do uso de valores conhecidos. Tipos contínuos podem ser testados com valores distribuídos uniformemente em diferentes ordens de magnitude, por exemplo, preços pequenos e preços grandes. Os experimentos apresentados em Madhavan et al. (2008) consideram a análise da cobertura em dez formulários HTML.

3.6 Análise Comparativa

Esta seção apresenta uma comparação entre os trabalhos relacionados e a taxonomia descrita na seção 3.1 por meio da classificação dos trabalhos nos dois eixos ortogonais. Além disso, uma comparação mais específica entre os trabalhos é realizada, verificando-se como cada trabalho trata a geração de valores iniciais, métodos de submissão do formulário, quantidade de experimentos, métricas utilizadas e necessidade de intervenção humana ou não. A Figura 3.15 apresenta onde cada trabalho se encaixa e sua respectiva abrangência dentro da taxonomia.

A maioria dos trabalhos (BARBOSA; FREIRE, 2004; NTOULAS; ZERFOS; CHO, 2005; WANG et al., 2012; SOULEMANE; RAFIUZZAMAN; MAHMUD, 2012; RAGHAVAN; GARCIA-MOLINA, 2001; LIDDLE et al., 2003; WU et al., 2006; ÁLVAREZ et al., 2007; LAGE et al., 2004) utiliza um viés heurístico para a solução dos

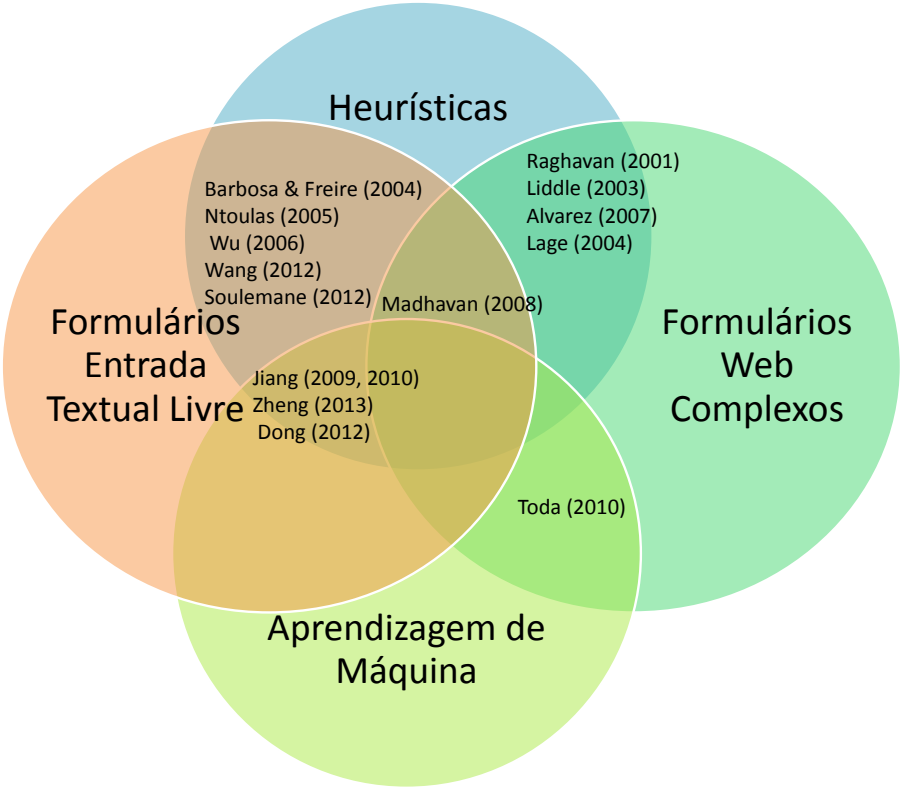


Figura 3.15 – Trabalhos Relacionados

problemas de preenchimento, tanto em formulários de entrada textual simples quanto em formulários complexos. Para formulários com entrada textual livre Jiang et al. (2009), Dong and Li (2012) utilizam uma abordagem de aprendizagem de máquina. Madhavan et al. (2008) utilizam várias heurísticas, como os demais trabalhos, mas trabalha tanto para formulários de entrada textual livre quanto formulários Web complexos.

Para bancos de dados estruturados que possuem formulários complexos, a maioria dos trabalhos também utiliza um viés heurístico para seleção de valores. Toda et al. (2010) utilizam aprendizagem de máquina para seleção de valores. A desvantagem dos trabalhos que utilizam um viés de aprendizagem de máquina é a dependência de um conjunto de treinamento para criar o modelo de aprendizagem. O trabalho de Toda et al. (2010) depende do conhecimento obtido de valores submetidos anteriormente para cada campo e validados pelos usuários. O trabalho proposto por Dong and Li (2012) utiliza somente formulários com entradas textuais livres.

A Tabela 3.1 apresenta um resumo comparativo dos trabalhos relacionados. As colunas representam os critérios analisados, em complementação à taxonomia definida na seção 3.1. Os critérios definidos para comparação são:

- Geração de sementes: apresenta como cada trabalho seleciona os valores iniciais dos campos textuais.
- Valores dos campos textuais: apresenta como os demais valores são selecionados. As estratégias utilizam heurísticas e técnicas de aprendizagem de máquina e, em alguns casos, simplesmente solicitam a intervenção do usuário em campos textuais.
- Conhecimento a *priori*: apresenta se é necessário algum conhecimento preliminar

para selecionar os valores.

- Método: especifica qual o método de envio de dados para a URL que a solução proposta utiliza. As opções possíveis são *get* e *post*.
- Intervenção Humana: indica se é necessária intervenção do usuário em algum momento durante a execução do método.
- #Formulários: mostra o número de formulários utilizados nos experimentos. Os experimentos podem utilizar sítios reais ou simulados.
- Geração de Sementes (*seeds*): descreve de que forma o trabalho trata os valores iniciais que serão submetidos para seleção de novas palavras.
- Métricas de avaliação: apresenta as métricas de avaliação utilizadas em cada método.

A Tabela 3.1 mostra alguns pontos interessantes. Vários trabalhos (RAGHAVAN; GARCIA-MOLINA, 2001; WU et al., 2006; ÁLVAREZ et al., 2007; WANG et al., 2012; NTOULAS; ZERFOS; CHO, 2005) dependem de um conjunto inicial de dados para o formulário. Esse conjunto, geralmente, é definido manualmente ou preparado com antecedência para cada domínio ou formulário. Outros trabalhos (WANG et al., 2012) utilizam heurísticas de seleção de informações para determinar os valores que serão selecionados, mas possuem o inconveniente de ter acesso a toda a base de dados por trás do formulário. A maioria dos sítios na Web Oculta não informa o total de dados da base de dados. Os experimentos realizados por Wang et al. (2012) são realizados em ambiente controlado, no qual é possível conhecer o total da base de dados e não são realizados experimentos em sítios reais na Web. O trabalho de Wu et al. (2006) utiliza formulários com entrada textual livre para acessar bancos de dados estruturados.

Além disso, diferentes critérios são usados pelas abordagens para selecionar os valores para preenchimento. O método de Liddle et al. (2003) não seleciona valores automaticamente para campos textuais, sendo necessária a participação do usuário. Madhavan et al. (2008) usam medidas tradicionais da área de recuperação de informação tais como TF-xIDF enquanto que Barbosa and Freire (2004), Soulemane, Rafiuzzaman and Mahmud (2012) adotam somente a frequência do termo (valor selecionado). O trabalho de Wang et al. (2012) utiliza a taxa de acerto (*hit rate* - HR) e a taxa de sobreposição (*overlapping rate* - OR). O valor de HR é uma limitação do trabalho de Wang et al. (2012), porque o número total de informações escondidas por trás do formulário é necessária e, para a maioria das fontes reais da Web Oculta, esse número é desconhecido. Finalmente, existem métodos (JIANG et al., 2009; JIANG et al., 2010; DONG; LI, 2012; ZHENG et al., 2013) que utilizam características sobre as submissões para selecionar valores.

A maioria das abordagens manipulam o método de submissão *get* (LIDDLE et al., 2003; MADHAVAN et al., 2008; SOULEMANE; RAFIUZZAMAN; MAHMUD, 2012) enquanto outros podem trabalhar com ambos os métodos, *get* e *post* (DONG; LI, 2012; NTOULAS; ZERFOS; CHO, 2005; RAGHAVAN; GARCIA-MOLINA, 2001; TODA et al., 2010; WANG et al., 2012; WU et al., 2006). Essa informação, no entanto, não está clara em alguns dos trabalhos estudados (ÁLVAREZ et al., 2007; BARBOSA; FREIRE, 2004; JIANG et al., 2009; LAGE et al., 2004).

O número de formulários utilizados nos experimentos varia significativamente entre os métodos (de 1 até 50). Experimentos com formulários reais na Web são complicados, uma vez que eles mudam com frequência, podem ter altos tempos de resposta ou, até mesmo, não estarem disponíveis em determinados períodos. Essas dificuldades impactam a escala dos testes. Alguns trabalhos (MADHAVAN et al., 2008) citam que realizam experimentos

Tabela 3.1 – Resumo dos Trabalhos Relacionados

Trabalho	Geração de Sementes	Valores de Campos Textuais	Conhecimento a *priori*	Método de Submissão	Intervenção Humana	# Formulários	Métricas de Avaliação
BARBOSA; FREIRE (2004)	página que contém o formulário	submissões de consultas usando valores obtidos em iterações anteriores baseadas na frequência dos termos	Não	não informado	Não	8	Cobertura (> 90% — 5 forms) (> 30 < 87 — 3 forms)
SOULEMANE et al. (2012)	página que contém o formulário	submissões de consultas usando valores obtidos em iterações anteriores baseadas na frequência dos termos	Não	get	Não	1	Número de valores
NTOULAS (2005)	manualmente definido	submissões de consultas usando valores obtidos em iterações anteriores baseadas na probabilidade dos termos	Não	get/post	Sim	4	Cobertura (84%)
WANG et al. (2012)	amostra aleatório do domínio	submissões de consultas usando valores obtidos em iterações anteriores baseados no conjunto de dados recuperados e na quantidade de dados duplicados recuperados	Não	get/post	Não	4	Tamanho da amostra (2.000)
LAGE et al. (2004)	repositório de dados de amostra	repositório de dados de amostra	Sim	não informado	Não	30	Precisão e revocação (100%)
RAGHAVAN; GARCIA-MOLINA (2001)	tabela LVS (*Label Value Set)*	Tabela LVS (*Label Value Set)*	Sim	get/post	Sim	50	Eficiência de Submissão
LIDDLE et al. (2003)	valores *default*	não aplicável	Não	get	Sim	13	Cobertura (80%)
WU et al. (2006)	valores selecionados aleatoriamente do banco de dados	Attribute Value Graph *Domain Statistics Table* (DT)	Sim	get/post	Não	5	Cobertura (sem DT 90%) (com DT 95%)
ÁLVAREZ et al. (2007)	atributos de domínio prédefinidos	tabela de atributos de domínio	Sim	não informado	Sim	30	Precisão e revocação (> 90%) exceto em um caso
TODA et al. (2010)	documento textual informado pelo usuário	probabilidade de um campo dada uma palavra n-gram	Sim	get/post	Sim	5	Precisão, revocação e F-Measure (0,95)
JIANG et al. (2009) ZHENG, et al. (2013)	página que contém o formulário	taxa de seleção (*harvest rate*) por meio do uso de características dos valores	Não	não informado	Não	3	Cobertura (> 80%)
DONG;LI (2012)	repositório de dados de amostra	taxa de seleção (*harvest rate*) por meio do uso de características dos valores	Não	get/post	Não	3	Cobertura (95%)
MADHAVAN et al. (2008)	página que contém o formulário	submissões de consultas usando valores obtidos em iterações anteriores baseadas na medida de TFxIDF	Não	get	Não	10	Cobertura (> 55%)

com muitos formulários, mas os resultados apresentados para preenchimento de campos são de apenas dez formulários.

A maior parte dos métodos apresenta a avaliação dos resultados em termos de cobertura (LIDDLE et al., 2003; BARBOSA; FREIRE, 2004; WU et al., 2006; NTOULAS; ZERFOS; CHO, 2005; MADHAVAN et al., 2008; JIANG et al., 2009; DONG; LI, 2012). As medidas de precisão, revocação e *F-measure* também são utilizadas (LAGE et al., 2004; ÁLVAREZ et al., 2007; TODA et al., 2010). Outros métodos descrevem novas medidas de avaliação dos experimentos (RAGHAVAN; GARCIA-MOLINA, 2001; SOULEMANE; RAFIUZZAMAN; MAHMUD, 2012; WANG et al., 2012).

A intervenção do usuário é requerida em cinco dos 14 métodos estudados (RAGHAVAN; GARCIA-MOLINA, 2001; LIDDLE et al., 2003; NTOULAS; ZERFOS; CHO, 2005; ÁLVAREZ et al., 2007; TODA et al., 2010). O grau de intervenção varia desde a rotulagem de dados até a entrada de valores manualmente.

A Figura 3.15 permite visualizar que existe uma lacuna na manipulação de ambas as interfaces de busca. Geralmente, as soluções são específicas para um tipo de interface. Com exceção de Madhavan et al. (2008), nenhum dos trabalhos relacionados apresenta experimentos com os dois tipos de interface. Além disso, poucos trabalhos (JIANG et al., 2009; JIANG et al., 2010; DONG; LI, 2012) combinam heurísticas e aprendizagem de máquina para a seleção de valores.

3.7 Resumo do Capítulo

Neste capítulo foi definida uma taxonomia para categorização dos trabalhos relacionados e foram apresentados 15 trabalhos relevantes existentes na literatura classificados segundo quatro categorias da taxonomia proposta. A taxonomia considera dois vieses: o método de preenchimento e o tipo de interface. O método de preenchimento pode ser dividido em heurísticas e técnicas de aprendizagem de máquina. O tipo de interface é classificado como entrada textual livre, o qual representa aqueles formulários com somente um campo texto e formulários Web complexos, o qual representa aqueles formulários com mais de um campo para preenchimento. A análise comparativa entre os trabalhos verificou que a maioria dos trabalhos manipula somente interfaces com entrada textual livre e existem poucos trabalhos que trabalham com os dois tipos de interface. Ainda, não existem trabalhos na literatura que manipulem os dois tipos de interface combinados com os dois métodos de preenchimento.

4 UMA SOLUÇÃO PARA PREENCHIMENTO AUTOMÁTICO DE FORMULÁRIOS NA WEB OCULTA

Este capítulo apresenta em detalhes a tese desenvolvida para preenchimento de formulários na Web Oculta. A Seção 4.1 descreve a arquitetura implementada nesta tese. A Seção 4.2 apresenta uma visão geral do método proposto para a solução do problema, enquanto que as Seções 4.3 e 4.4 detalham as fases de seleção de valores para campos de formulários.

Esta tese apresenta uma solução automática de preenchimento de formulários na Web Oculta, denominada *smartFTF*, que engloba formulários complexos e formulários de entrada textual livre. A solução proposta combina as duas estratégias de preenchimento, heurísticas e aprendizagem de máquina, em uma combinação de modelos. A ideia é utilizar as regras práticas alcançadas pelas heurísticas para apoiar os algoritmos de aprendizagem de máquina na predição de bons valores para os campos dos formulários. O foco principal está no preenchimento de campos com domínio infinito, sejam eles, campos de palavras-chave ou não.

Considerando a classificação apresentada no Capítulo 3, a solução descrita nesta tese se enquadra na interseção de todas as estratégias. Ainda, considerando os métodos analisados, verifica-se que não existem trabalhos que estejam localizados dentro desta interseção. O trabalho que mais se aproxima desse contexto preenche formulários com entradas textuais simples e formulários complexos, mas utiliza heurísticas como técnica de preenchimento dos campos (MADHAVAN et al., 2008). O método proposto nesta tese também preenche campos em ambos os tipos de formulários, mas em vez de utilizar somente heurísticas, faz uso de algoritmos de aprendizagem de máquina para melhorar a predição de valores para campos de formulários. A Figura 4.1 ilustra o contexto em que a tese está classificada, conforme taxonomia descrita na Seção 3.1.

Esta tese apresenta um método para seleção de valores para preenchimento automático de formulários Web. O problema é como escolher valores para preenchimento de campos do formulário de maneira a maximizar a recuperação de instâncias e minimizar o número de submissões ao formulário. A formulação do problema estudado nesta tese pode ser como segue: 'Dado um formulário HTML, identificar os campos e encontrar valores para preenchê-los'. A seção 4.1 descreve a arquitetura implementada para a coleta na Web Oculta. A seção 4.2 apresenta uma visão geral do método proposto nesta tese e as seções 4.3 e 4.4 descrevem em detalhe as fases do método, especialmente para campos textuais.

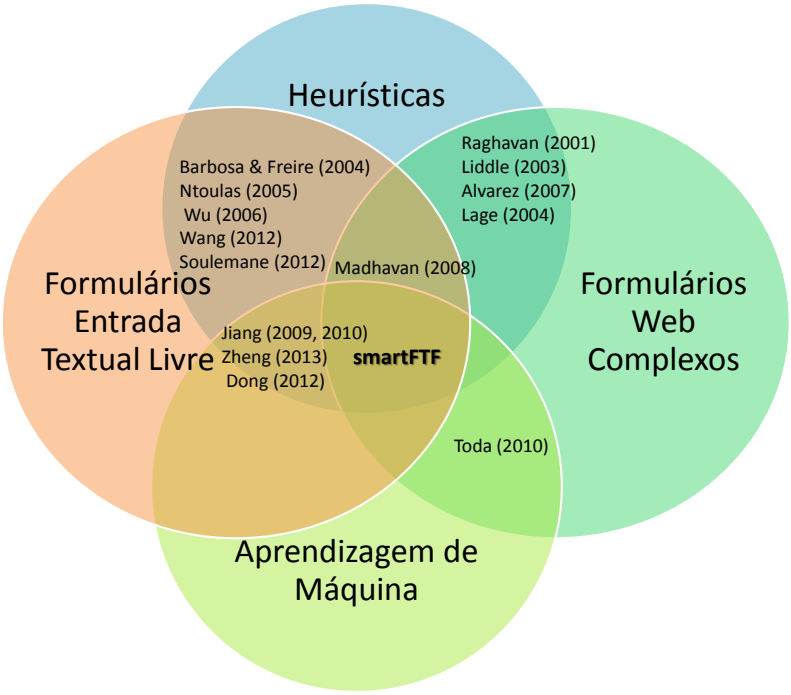


Figura 4.1 – Taxonomia e tese

4.1 Arquitetura

O método de solução descrito nesta tese é dividido em uma série de passos, organizados em uma arquitetura. De forma similar ao trabalho proposto por Madhavan et al. (2008), o método de solução utiliza o conceito de *template*.

A Figura 4.2 mostra a arquitetura proposta para coleta de dados na Web Oculta. O módulo *"Geração Templates Candidatos"* representa o processamento e análise do formulário HTML com a finalidade de gerar todos os possíveis *templates*. A análise do formulário implica na identificação e extração dos campos do formulário e na montagem dos *templates*. O módulo *"Seleção de Valores"* tem como objetivo encontrar valores para preenchimento dos campos do formulário. O foco principal neste módulo é encontrar bons valores para campos com domínio infinito. A seção 4.2, apresenta em detalhes como a seleção de valores é realizada. O próximo módulo, *"Geração e Submissão de Template Instances"*, adiciona os valores de entrada para cada campo correspondente de cada *template*, criando o conjunto de *template instances*. Os *templates* são processados em ordem crescente de dimensão. Assim, primeiro todos os *template instances* de dimensão 1 são gerados e submetidos, seguidos dos *templates* de dimensão 2, e assim por diante. O módulo *"Extração de Dados"* extrai as informações das páginas resultantes das submissões do conjunto de *template instances*. Essa extração é realizada para encontrar na página resultado a região de dados. Informações sobre publicidade e apresentação encontradas na página de resultados são descartadas. Os dados extraídos são utilizados para avaliação de cada *template*. Esses dados são armazenados em um banco de dados, que será utilizado para avaliação da cobertura e custo das submissões. O módulo *"Avaliação da Informatividade de Templates"* (MADHAVAN et al., 2008) verifica se um *template* é considerado informativo após a submissão de todo o conjunto de *template instances* correspondente. Um *template* será considerado informativo se o conjunto de *template instances* recupera

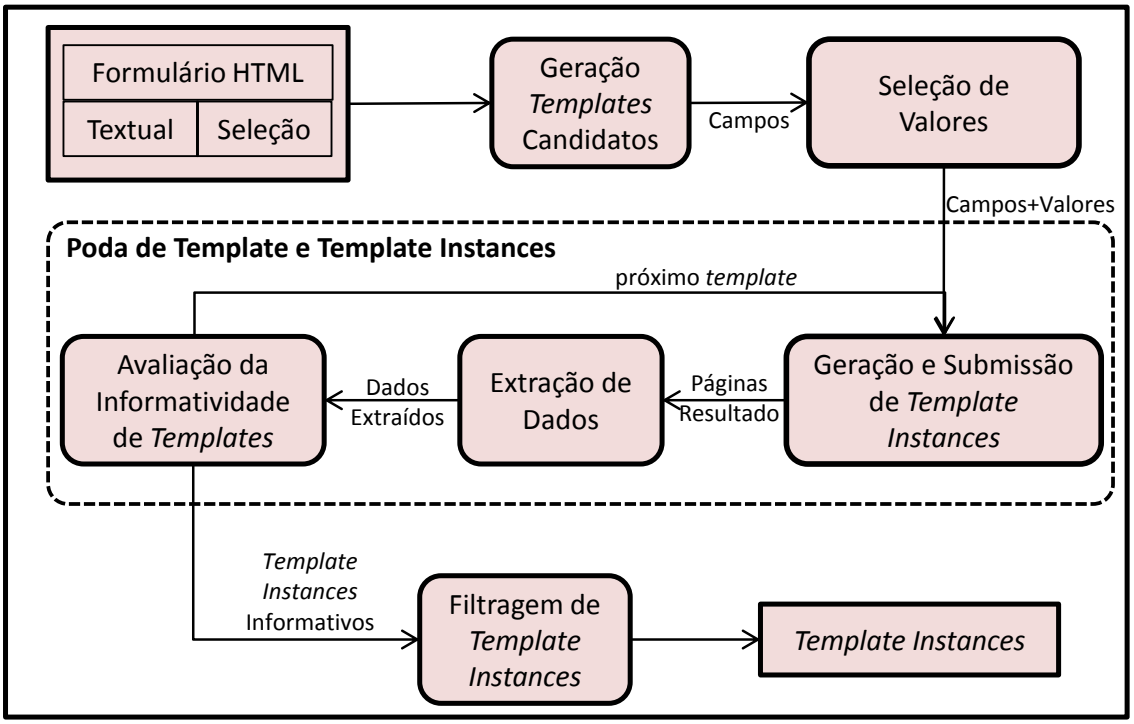


Figura 4.2 – Visão Geral da Arquitetura

dados suficientemente distintos. Se um template t é considerado não informativo, todos os *templates* de dimensão superior a t e que incluem t também são considerados não informativos e serão descartados. No final do processo, após processar todos os *templates*, o módulo *"Filtragem de Template Instances"* determina o número mínimo de *template instances* necessários para recuperar a maior quantidade de linhas distintas extraídas das páginas resultantes das submissões ao formulário.

Todos os módulos apresentados na arquitetura visualizada na Figura 4.2 foram implementados. O foco principal da presente tese é o módulo *"Seleção de Valores"*. O módulo *"Geração e Submissão de Template Instances"* é parte integrante da dissertação de mestrado do aluno Tiago Guimarães Moraes (MORAES, 2013). A Seção 4.2 descreve a abordagem utilizada para a seleção de valores, em especial para campos de domínio infinito.

4.2 Visão Geral

Esta seção apresenta uma visão geral dos métodos desenvolvidos na presente tese para preenchimento de formulários. O objetivo de selecionar valores para *campos textuais* é descobrir os valores que devem ser usados nas submissões do formulário que recuperam a maior quantidade de informação escondida por trás do formulário com um custo aceitável. O primeiro método, denominado FTF, utiliza heurísticas para seleção de valores para campos de um formulário. O segundo método, denominado $smartFTF$, combina as heurísticas do método FTF com algoritmos de aprendizagem de máquina para selecionar valores para os campos. O método automaticamente escolhe os valores para os campos do formulário, evitando a rotulagem manual das classes.

A Figura 4.3 mostra dois tipos de campos: *campo textual* e *campo de seleção*. *Campo de seleção* refere-se a campos de domínio finito. A geração de valores para campos com domínio finito é mais simples, uma vez que os valores que devem ser selecionados estão presentes no formulário. Esses campos são preenchidos pelos valores extraídos do código do formulário, por exemplo, valores presentes no rótulo *option* no código HTML do formulário (ver Figura 2.2).

A seleção de valores para campos de domínio infinito (*campo textual*) não é tão simples como a seleção para campos de domínio finito. Isso acontece pois não é possível saber de antemão a quantidade e a qualidade de valores dos campos. A quantidade mede o número total de valores desejados para alcançar a maior cobertura dos dados escondidos por trás do formulário e por qualidade entende-se a seleção de valores que recuperam mais dados distintos. Essas características tornam o processo de encontrar valores para *campos textuais* uma tarefa desafiadora. Uma forma de encontrar os valores iniciais pode ser uma lista de valores associados com o *campo textual*. No entanto, isso não é factível uma vez que existem muitos formulários na Web localizados em vários domínios que podem ter campos similares nos quais os valores podem ter uma grande variação.

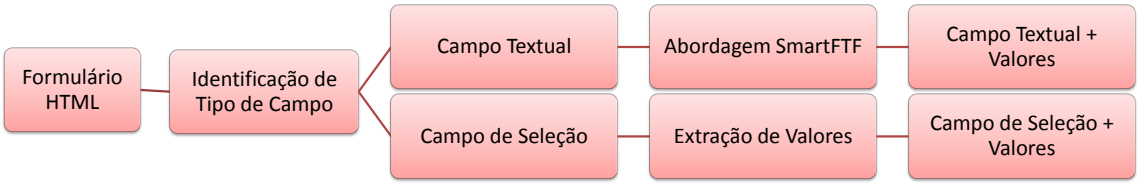


Figura 4.3 – Tipos de Campo e Seleção de Valores

Na abordagem *smartFTF* a seleção de valores para os *campos textuais* é dividida em duas fases. Durante a primeira fase, regras heurísticas são aplicadas com o objetivo de gerar valores para o conjunto de treinamento. As páginas resultantes das submissões são processadas para encontrar onde está a região de dados da página. Informações sobre propagandas, por exemplo, são descartadas. Os dados extraídos são utilizados para avaliar cada *template*. Esses dados também são usados como base para a geração de valores para os *campos textuais*. O método MDR (LIU; GROSSMAN; ZHAI, 2003) foi utilizado no processo de extração dos dados da página. A intuição é que uma maior cobertura pode ser alcançada com os dados de submissões anteriores em vez de usar dados gerados aleatoriamente. Na segunda fase, o conjunto de treinamento é utilizado para a construção de um modelo que é usado para predição e seleção dos valores para preenchimento dos *campos textuais*. A primeira fase é executada somente uma vez. Depois que o conjunto de treinamento está pronto, os valores são sempre escolhidos pelo modelo de aprendizagem de máquina.

A Figura 4.4 apresenta uma visão geral da abordagem *smartFTF*. A numeração indica a sequência de execução da cada etapa. Na fase heurística, submissões são realizadas ao formulário (1) e as páginas resultantes da submissão (2) são analisadas. A análise das páginas extrai os dados assim como várias características sobre as submissões. As heurísticas são aplicadas sobre o conjunto de dados resultante para selecionar os valores dos campos. As características das submissões geram o conjunto de treinamento (3). Partindo do conjunto de treinamento é construído um modelo de aprendizagem (4) que será utilizado na fase de aprendizagem para prever os novos valores. Quando novas submissões

são geradas com base em novos valores (5) as características das submissões (6) são avaliadas conforme o modelo construído na etapa (4). Com base na avaliação do modelo, valores serão selecionados (7) para preenchimento dos campos do formulário.

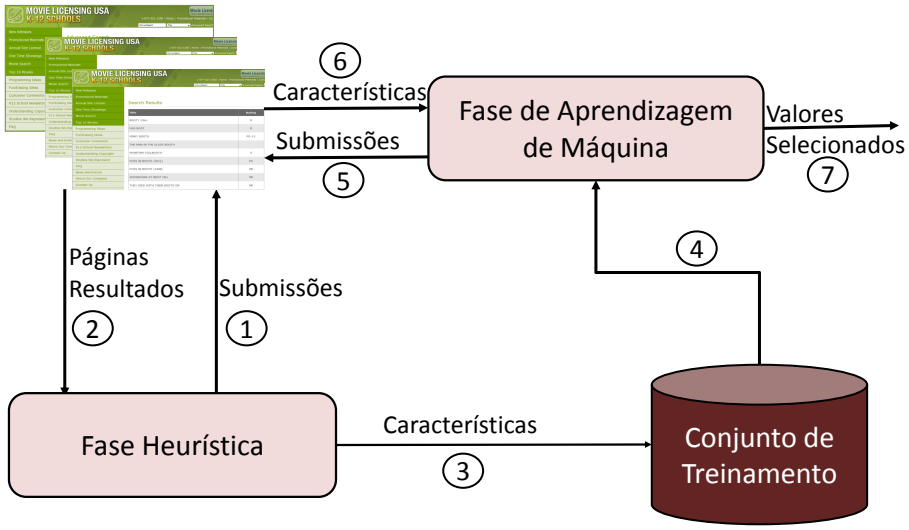


Figura 4.4 – Visão Geral da Abordagem *smartFTF*

As próximas seções apresentam cada uma das fases em detalhes. A seção 4.3 descreve a abordagem *FTF* e apresenta as heurísticas usadas para geração do conjunto de treinamento. A seção 4.4 mostra a geração do conjunto de treinamento e como é realizada a predição dos valores para preenchimento dos campos.

4.3 Fase Heurística

Esta fase tem como objetivo a geração do conjunto de valores que serão usados para a construção do conjunto de treinamento que será utilizado na próxima fase. O conjunto de treinamento é gerado automaticamente por intermédio de heurísticas que selecionam valores para os campos dos formulários. O método de seleção de valores por meio das heurísticas, denominado *FTF*, foi publicado em Kantorski et al. (2013).

A fase *Heurística* gera valores para campos textuais baseado na frequência dos termos, na ordem de submissões do conjunto total de submissões realizadas e na importância do valor. Cabe salientar que somente *templates* de dimensão 1 são utilizados para geração dos valores dos campos textuais. Os valores gerados são submetidos, a página resultante da submissão é processada, os dados são extraídos e as heurísticas são aplicadas para a seleção de novos valores a serem submetidos. Esse processo é repetido até que determinado limiar seja alcançado. A fase *heurística* funciona em dois passos. O primeiro passo consiste na geração de valores para submissão enquanto o segundo seleciona os melhores valores gerados no primeiro passo.

Os valores iniciais são extraídos da página do formulário e de submissões anteriores. Como o número de valores extraído pode ser muito grande, são selecionados k valores para submissão. Os k valores são selecionados conforme um escore r_1, mostrado nas equações 4.1 e 4.2. A variável cf_t representa o número de vezes que o valor t aparece na coleção, idf_t representa a frequência inversa, N é o número total de submissões e df_t é o

número de páginas resultado que contém o valor t. A coleção representa o resultado de todas as submissões realizadas. Cada página resultante de uma submissão é considerada um documento pertencente à coleção. A ideia é que quanto maior for variável cf_t mais dados podem ser recuperados. O valor de idf_t é usado para eliminar valores que aparecem no resultado de todas as submissões, como por exemplo, cabeçalhos e notas de rodapé. O idf representa a importância do valor entre todos os resultados, beneficiando mais valores distintos.

$$r_{1t} = cf_t \times idf_t \tag{4.1}$$

$$idf_t = \log_2 \frac{N}{df_t} \tag{4.2}$$

O valor *k* é variável e determinado conforme o número de linhas distintas recuperadas a cada submissão. Conforme o número de linhas distintas diminui, é determinado um limiar que limita o valor de *k*. O limiar definido para seleção dos *top-k* valores é 1/4 do total de linhas distintas recuperadas na submissão corrente. Para evitar um valor *k* muito pequeno, além do limiar, é definido um valor mínimo para *k*. Observações empíricas mostraram que o valor 50 se comportou melhor para o valor mínimo de *k*.

Para *templates* de dimensão 1 compostos por campos de domínio finito, as submissões de URLs são geradas de acordo com as opções de cada campo. Para campos de domínio infinito, os *top-k* valores da página que contém o formulário são ordenados pelo escore r_1 e são usados como valores iniciais para submissão. O valor de *k* depende de três variáveis: o número total de instâncias recuperadas, o número total de instâncias distintas recuperadas e a ordem da submissão. A Equação 4.3 mostra a dependência entre as novas instâncias distintas e o total de instâncias. O limiar *h* representa o critério de parada, em que *h* é menor que 25% do total de instâncias que retornam de uma consulta.

$$h = \frac{\sum_{t=1}^{k} nd_t \times \log_2 ord_t}{\sum_{t=1}^{k} nr_t \times \log_2 ord_t} \tag{4.3}$$

em que nd_t é o número de linhas distintas recuperadas da submissão t; nr_t é o número total de linhas recuperadas na submissão t; e ord_t é a ordem da submissão t.

Para *templates* com *n* dimensões, em que $n > 1$, existem várias estratégias de seleção. (KANTORSKI; MORAES; HEUSER, 2011; KANTORSKI; MORAES; HEUSER, 2012) discutem quatro estratégias para a geração de URLs para *templates*. Nesta tese, é empregada a estratégia *k-allValues*, a qual utiliza todos os valores pelo menos uma vez. A intuição desse método é que a seleção de todos os valores pode alcançar uma maior cobertura. Todas as URLs são geradas, mas somente um subconjunto é submetido para o formulário de acordo com a estratégia *k-allValues*.

Para cada submissão, são analisadas características como total de linhas recuperadas, total de linhas distintas recuperadas até o presente momento, ordem da submissão no conjunto total de submissões. Após a submissão dos *k* valores, um segundo escore r_{2t} (Equação 4.4) é calculado para cada valor. O escore r_2 é baseado no número de linhas distintas recuperadas (nd_t) pelo valor t e pela ordem de submissão (ord) no conjunto total de submissões. A intuição é que valores usados em ordens maiores e que tenham recuperado muitos dados distintos são melhores. Por exemplo, se a terceira e a décima submissão recuperam cem linhas distintas, então o valor utilizado na décima submissão recebe um escore maior, uma vez que é mais fácil recuperar mais linhas distintas nas primeiras submissões.

$$r_{2t} = nd_t \times \log_2 ord \tag{4.4}$$

Os resultados de cada submissão são avaliados pelo escore r_{2t} e as seguintes heurísticas são aplicadas para descarte de valores:

1. Valores que foram submetidos e que não retornaram dados.
2. Valores que retornam apenas uma linha.
3. Valores que retornam o menor número de linhas.
4. Valores que aparecem em todos os resultados de submissões.

Valores que retornam apenas uma linha são descartados porque é provável que registros com somente uma linha contenham apenas cabeçalhos. Os valores que retornam o menor número de linhas também são descartados. A intuição é que descartando valores com poucas linhas, estamos liberando mais espaço para a seleção de novos termos, que podem recuperar mais linhas na próxima iteração. Esse processo é repetido até que o número mínimo de *k* valores seja alcançado.

Para fins de processamento, os formulários foram divididos em dois tipos: aqueles que contêm somente campos texto e aqueles que contém campos texto e campos de domínio finito (*select fields*). Essa divisão é necessária, pois influencia na geração dos valores iniciais para os campos. Em campos de domínio finito (*select fields*), são obtidos diretamente da página HTML. As submissões sempre iniciam pelos campos de domínio finito seguidos pelos campos de domínio infinito. Os resultados das submissões são armazenados em um repositório que será usado mais adiante para gerar valores que serão utilizados para os demais campos. A informação localizada na página HTML na qual o formulário está inserido é extraída somente quando o formulário possui apenas campos de domínio infinito.

4.4 Fase de Aprendizagem de Máquina

Esta fase tem como finalidade, com base nos valores gerados na primeira fase, extrair metadados sobre as submissões realizadas na fase *heurística*, construir um modelo baseado no conjunto de treinamento e classificar novos valores usando o modelo gerado. Os metadados são usados como instâncias de treinamento para construção do modelo de aprendizagem. O propósito desse modelo é decidir se um valor não rotulado deve ser selecionado para preenchimento de campos do formulário ou deve ser descartado. Esta seção apresenta como gerar o conjunto de treinamento para *templates* de dimensão 1 formados por campos textuais, como construir o modelo de aprendizagem e como classificar valores não rotulados.

O Algoritmo 1 mostra como o conjunto de treinamento é gerado. Novos dados de treinamento são gerados para cada *template* de dimensão 1. As variáveis *nSubmissions* e *nIteration*, nas linhas 3 e 4, representam o número de valores selecionados e o número de iterações respectivamente. A variável *nIteration* é aplicada se, e somente se, na linha 5, o valor de *nSubmissions* não seja alcançado.

O método *generateTemplateInstance*, na linha 6, gera o conjunto de *template instances* conforme o escore r_1 (Equação 4.1). Por exemplo, a página na qual o formulário está localizado e resultados de submissões anteriores são extraídos, *tokenizados*, e ranqueados de acordo com o escore r_1. Os *k* maiores valores são usados para a construção de *template instances* (URLs) que serão submetidas para o formulário. A seguir, os *templates instances* são submetidos e os resultados são avaliados conforme visualizado na linha 7.

O método *selectValues* (linha 8) decide quais os valores serão selecionados e quais serão descartados. Esse método extrai os dados resultantes da submissão e aplica as regras heurísticas para rotular os valores. Os valores selecionados pelas regras heurísticas são rotulados como exemplos positivos e os valores descartados são rotulados como exemplos negativos, de acordo com o escore r_2. Dessa maneira, a rotulagem manual de instâncias de treinamento é evitada. Os experimentos demonstram que a rotulagem automática melhora a qualidade das métricas, comparado com o método que utiliza somente regras heurísticas. O conjunto de todos os valores, positivos e negativos, serão usados para a geração do conjunto de treinamento. A variável *nSubmissions* é atualizada para o número de valores positivos (linha 8). A seguir, o método *generateTrainingData* (linha 9), extrai as características para cada submissão para criar o conjunto de treinamento. No fim de cada iteração, o conjunto de treinamento é atualizado para o *TrainingDataset* (Figura 4.5).

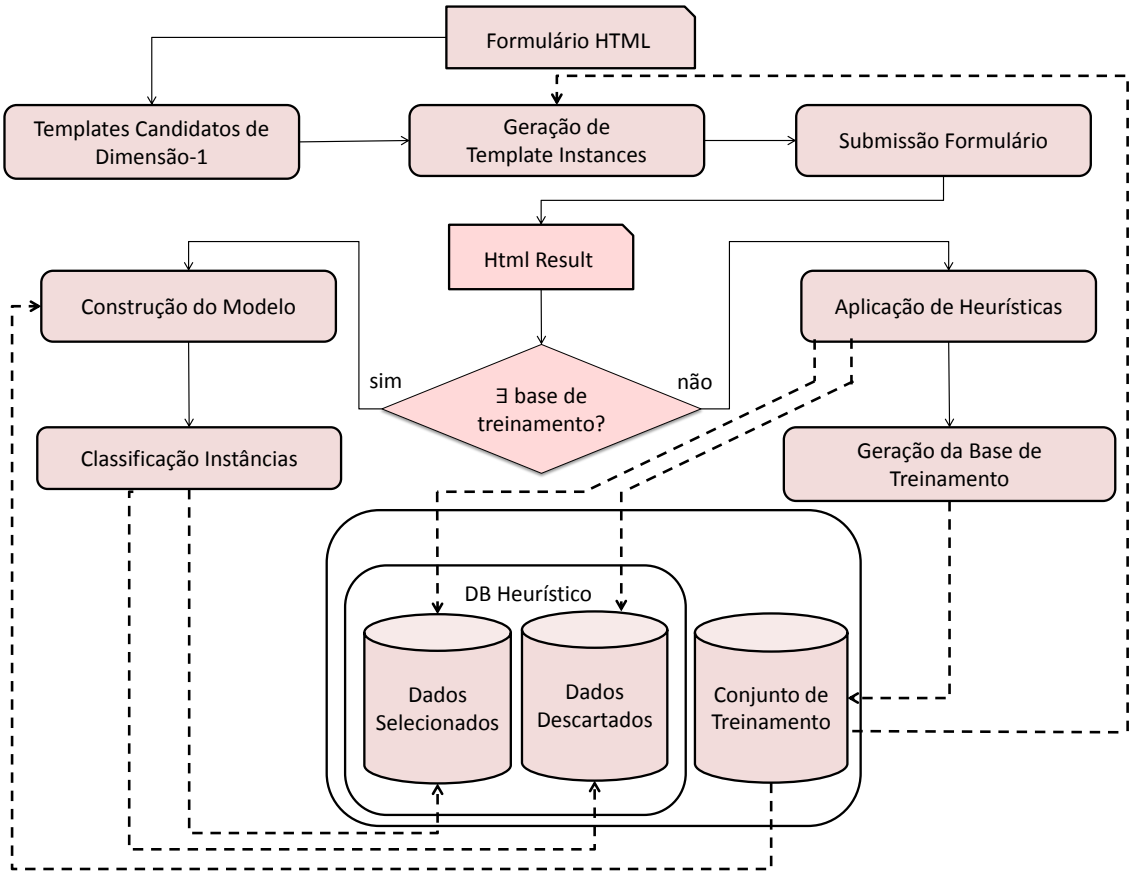


Figura 4.5 – Abordagem *smartFTF*

O número total de conjuntos de treinamento deve ser igual ao número de *templates* de dimensão 1. Por exemplo, para o formulário mostrado na Figura 2.1 vão existir três conjuntos de treinamento. Um conjunto para o *template pub name*, um para o *template street* e outro para o *template town/county*. O número de exemplos de treinamento deve ser igual ao número total de valores gerados para submissão de cada *template*. Um exemplo de conjunto de treinamento para o *template pub name* é apresentado na Figura 4.6. O rótulo $@relation$ descreve o nome do campo do formulário, enquanto que os rótulos $@attribute$ apresentam as características extraídas das submissões. O primeiro rótulo $@attribute$ é mostrado apenas para fins de explicação do método. O valor submetido

Algoritmo 1 Geração do Conjunto de Treinamento

```
1: begin
2:  for each 1-dimension template do
3:    nSubmissions ← ∅
4:    nIteration ← ∅
5:    while nSubmissions < m and nIteration < i do
6:      generateTemplateInstance()
7:      submitTemplateInstance()
8:      nSubmissions ← selectValues()
9:      generateTrainingData()
10:     nIteration + +
11:  return TrainigDataset
12: end
```

(*token*) não é parte integrante do conjunto de treinamento. Finalmente, o rótulo $@data$ indica os dados que compõe o conjunto de treinamento. O valor *yes* ou *no* no fim de cada exemplo de treinamento indica se aquele valor submetido (*token*) foi selecionado ou descartado.

```
@relation pubName

@attribute token string
@attribute cfidf real
@attribute linhas real
@attribute distintas real
@attribute cf real
@attribute idf real
@attribute ordem real
@attribute select {yes,no}

@data
wood,120.10,22,22,3,1.0,44,yes
kingdom,11.81,5,2,2,1.0,60,yes
green,56.86,22,22,10,1.0,6,yes
gardens,116.27,22,22,3,1.0,39,yes
.
.
.
wimbledon,2.0,1,0,2,1.0,68,no
upminster,0.0,1,0,2,1.0,54,no
```

Figura 4.6 – Exemplo de Conjunto de Treinamento

Cada conjunto de treinamento associado a cada *template* tem um número de características que são usadas para classificar os novos valores para o preenchimento do formulário. As características extraídas de cada submissão são:

- r_{2t}: conforme calculado pela Equação 4.4.

- cf_t: *collection frequency*, isto é, o número de vezes que um valor *t* aparece nos resultados.
- df_t: *document frequency*.
- idf: *inverse document frequency*, ou seja, a importância do valor entre todas as submissões.
- r_{1t}: conforme calculado pela Equação 4.1.
- $linha$: número de linhas recuperadas pela submissão.
- $linhasdistintas$: número total de linhas distintas recuperada pela submissão.
- $ordem$: a ordem de submissão do valor no total de submissões realizadas.
- $select$: a classe do valor submetido (positivo = $selecionado$ ou negativo = $descartado$).

Finalmente, em submissões futuras, o conjunto de treinamento é usado para a seleção de valores para os campos do formulário. Assim, o componente *Construção do Modelo*, na Figura 4.5, cria um modelo de aprendizagem baseado no conjunto de treinamento. O Algoritmo 2, linha 3, mostra a geração do modelo. O algoritmo utilizado para construção do modelo foi o *Multilayer Perceptron* (MLP). MLP é um classificador que utiliza *backpropagation* para classificar instâncias. Existem parâmetros que devem ser definidos para o perfeito funcionamento do algoritmo MLP. O parâmetro que define o número de camadas ocultas *hidden layers* foi definido para o mesmo número de atributos existentes no conjunto de treinamento. Embora tenha sido utilizado um algoritmo específico, o método permite que vários tipos de algoritmos de aprendizagem podem ser utilizados para a geração do modelo. O componente *Classificação Instâncias*, na figura 4.5, mostra a classificação dos novos valores como positivos ou negativos. Nessa etapa, somente as características das submissões são obtidas. Os dados das tuplas resultantes da submissão não são processadas. Esse componente é representado pelo método *classifyTestInstances*, mostrada na linha 9 do algoritmo 2. Por exemplo, considere o template *pub name* no formulário mostrado na figura 2.1. O conjunto de treinamento associado ao template *pub name* é selecionado e o modelo é construído conforme mostrado na linha 3 do algoritmo 2. A seguir, novos valores são gerados e submetidos para o formulário. As características para cada submissão são calculadas e usadas pelo modelo de classificação. Esse processo será repetido até que um dos limiares (*m* ou *i*) seja alcançado.

Algoritmo 2 `Classificação`

```
    begin
 2: for each 1-dimension template do
      buildClassifier(TrainingDataset)
 4:   nSubmissions ← ∅
      nIteration ← ∅
 6:   while nSubmissions < m and nIteration < i do
        generateTemplateInstance()
 8:     submitTemplateInstance()
        nSubmissions ← classifyTestInstances()
10:     nIteration + +
     return selectedValues
12: end
```

Para ilustrar de maneira global como a abordagem *smartFTF* funciona, considerando o campo *pub name* do formulário visualizado na Figura 2.1. Esse campo também é um

template de dimensão 1, chamado *pub name*. Se ainda não existe conjunto de treinamento para o campo, o Algoritmo 1 é executado para o *template*. O Algoritmo 1 equivale à execução dos componentes *Aplicação de Heurísticas* e *Geração da Base de Treinamento* mostrados na Figura 4.5. Os valores iniciais são gerados pelo método *generateTemplateInstance()* (linha 6). Os valores iniciais são selecionados da página que contém o formulário e dos resultados de submissões de campos de domínio finito (*template country*) (Figura 2.1). A página que contém o formulário e os resultados da submissão do *template country* são extraídas e *tokenizadas*. Esses termos (*tokens*) são ordenados pelo escore apresentado na Equação 4.1. Depois, *k* termos são selecionados para criar o conjunto de *template instances*, que serão submetidos pelo método *submitTemplateInstance()* (linha 7). Após as submissões de todos os *template instances*, o método *selectValues()* gera um ranking baseado na Equação 4.4 e aplica as seguintes heurísticas para descartar valores para o campo *pub name*:

1. Valores que recuperam menos que duas linhas;
2. Valores que recuperam o menor número de linhas;
3. Valores que aparecem em todos os resultados.

Os valores, que não se enquadram nas heurísticas acima, são selecionados. A variável *nSubmissions* é atualizada para o número de valores selecionados. Então, o método *generateTrainingData()* (linha 9) computa, para cada submissão, as características que serão usadas pelo algoritmo de aprendizagem de máquina. Os valores selecionados são rotulados automaticamente como positivos e os valores descartados são rotulados como negativos. Finalmente, na linha 10, a variável *nIteration* é incrementada e *nSubmissions* e *nIteration* são verificadas. Se os limiares *m* ou *i* não são alcançados, o processo é repetido, ou seja, páginas resultantes de submissões anteriores são extraídas e tokenizadas e assim por diante.

Caso o conjunto de treinamento exista para o *template pub name*, a abordagem *smartFTF* utiliza o Algoritmo 2. A Figura 4.5 ilustra esse caminho por meio dos componentes *Construção do Modelo* e *Classificação Instâncias*. Primeiro, o método *buildClassifier()*, na linha 3, lê o conjunto de treinamento para o *template pub name* e constrói o modelo de aprendizagem. Na avaliação experimental, foram utilizados vários algoritmos de predição. Os resultados foram muito próximos, o que mostra que a abordagem *smartFTF* funciona bem com diferentes algoritmos de aprendizagem. A seguir, os métodos *generateTemplateInstance()* e *submitTemplateInstance()* funcionam de maneira similar ao Algoritmo 1. Após as submissões, o método *classifyInstances()* (linha 9) analisa as características das submissões. As características são computadas e formam o conjunto de teste que será submetido ao modelo construído. Cada instância de teste representa um valor submetido e é classificado como positivo ou negativo. Subsequentemente, a variável *nSubmissions* é atualizada para o total de valores que o método *classifyInstances()* classificou como positivo. O resultado final é um conjunto de valores que serão usados para o preenchimento do campo *pub name*. O processo é repetido para os demais *templates* de dimensão 1, *street* e *town/county*.

4.5 Resumo do Capítulo

Neste capítulo, foi discutida a abordagem *smartFTF* para seleção de valores para preenchimento de campos em formulários na Web Oculta. O foco principal do método apresentado é na seleção de valores para campos textuais. O método é dividido em duas fases:

heurística e aprendizagem de máquina. A fase heurística é responsável pela geração do conjunto de treinamento, o qual é usado para geração do modelo de aprendizagem. A fase de aprendizagem constrói o modelo e aplica o modelo construído para selecionar os valores dos campos do formulário.

5 AVALIAÇÃO EXPERIMENTAL

Neste capítulo, experimentos são executados para avaliar o método proposto nesta tese para seleção de valores para campos de formulários. A seção 5.1 apresenta a configuração utilizada nos experimentos, descrevendo as métricas utilizadas, os métodos utilizados como *baseline* para comparação e os formulários reais utilizados na avaliação experimental. A seção 5.2 mostra os resultados alcançados com os experimentos sob vários pontos de vista.

5.1 Configuração dos Experimentos

Todos os experimentos foram conduzidos em cinquenta formulários reais existentes na Web. Os métodos propostos e suas implementações são independentes de domínio. Formulários Web de vários domínios (tais como, `books`, `events`, `food&drink`, `jobs`, `movies` e `people`), tamanhos e números de campos foram utilizados. O objetivo é avaliar os métodos sem considerar o viés de domínio.

O método proposto nesta tese foi comparado com dois outros métodos: o método proposto por (MADHAVAN et al., 2008), o qual será denominado de *baseline*, e o método *FTF* (KANTORSKI et al., 2013), que utiliza somente heurísticas. O método proposto por (MADHAVAN et al., 2008) foi selecionado como *baseline*, porque é o único trabalho relacionado que manipula os dois tipos de interface (textual livre e complexa).

Uma vez que os experimentos foram executados em formulários reais na Web, os quais são dinâmicos e podem ficar indisponíveis, os experimentos foram executados em um curto espaço de tempo. Para complementar, o formulário com identificação 16 não estava disponível quando o *baseline* foi executado. Portanto, os valores médios das métricas para o *baseline* desconsideram esse formulário. O número de formulários usados nos experimentos é similar ou maior que o número usado nos trabalhos relacionados (BARBOSA; FREIRE, 2004; LIDDLE et al., 2003; RAGHAVAN; GARCIA-MOLINA, 2001; TODA et al., 2010; KANTORSKI et al., 2013). A lista completa de formulários é visualizada na Tabela 5.1.

As métricas de avaliação são *cobertura* (BARBOSA; FREIRE, 2004) e duas extensões do conceito de eficiência (*efficiency*) introduzido por (NTOULAS; ZERFOS; CHO, 2005):

- Cobertura (Cov_f): é o número de instâncias distintas extraídas durante todo o processo.
- Eficiência de Execução (EE_f): avalia a cobertura Cov_f em relação ao custo de execução, isto é, a taxa entre a cobertura alcançada e o total de submissões realizadas ($nSubmissions$). Considera-se custo de execução todas as URLs submetidas

para o formulário, independente se o resultado retorna dados ou não. Na prática, essa métrica avalia a porcentagem de informações recuperadas em razão do total de URLs submetidas (Equação 5.1).

$$EE_f = \frac{Cov_f}{nSubmissions} \tag{5.1}$$

- Produtividade ($Prod_f$): avalia o relacionamento entre a cobertura Cov_f e o custo das submissões úteis, isto é, o valor médio de instâncias obtidas para cada *template instance* que retornou dados ($nProductive$). $nProductive$ contém o total de URLs que retornaram dados distintos de outras URLs. Essa variável representa um valor menor ou igual ao custo de execução. Na prática, essa métrica avalia a porcentagem de informações recuperadas em virtude do total de URLs submetidas que retornarem informações úteis (Equação 5.2).

$$Prod_f = \frac{Cov_f}{nProductive} \tag{5.2}$$

A Cobertura, Eficiência de Execução e Produtividade foram normalizadas pelo melhor método para cada métrica e formulário. Por exemplo, a cobertura de 100% não significa que todas as informações foram recuperadas por trás do formulário. Isso mostra que o método que alcançou o valor de 100% apresentou um desempenho melhor que os demais métodos.

Os experimentos foram divididos em duas fases. A primeira é a geração do conjunto de treinamento (aplicação da fase *heurística* da abordagem *smartFTF*) e a segunda fase é a seleção de valores para preenchimento de formulários por intermédio de técnicas de aprendizagem de máquina que utilizam o conjunto de treinamento gerado na primeira fase.

A ferramenta *Weka* (HALL et al., 2009) foi integrada na implementação e o algoritmo *Multilayer Perceptron* (MLP) foi utilizado como algoritmo de classificação. A ferramenta *Weka* contém uma coleção extensa de algoritmos de aprendizagem de máquina.

5.2 Resultados

A Tabela 5.2 mostra os detalhes dos formulários usados nos experimentos. A coluna *nSubmitted URLs* representa o número total de URLs submetidas ao formulário para cada um dos métodos (*baseline*, *FTF* e *smartFTF*). A coluna *nProductive URLs* descreve o subconjunto de URLs submetidas que resultaram em dados úteis. A coluna *nExploration URLs* representa o número total de URLs submetidas para o formulário para descobrir os valores para os campos textuais. Essas URLs representam a primeira fase da abordagem *smartFTF*, em que são gerados os conjuntos de treinamento para cada *template* de dimensão 1, ou seja, para cada campo textual existente no formulário. Por exemplo, considere o formulário com identificador (*id*) igual a 1. A coluna *nExploration FTF* possui o valor 242 enquanto que a coluna *nExploration smartFTF* possui o valor 642. Isso significa que foram necessárias 242 submissões para o método *FTF* e 642 submissões para o método *smartFTF* descobrir os valores para os campos textuais do formulário. Os valores entre parêntesis mostram o valor médio de submissões por campo textual do formulário. Finalizando, também são mostrados os números de campos existentes no formulário por meio da coluna *#Campos* e o domínio no qual o formulário está inserido pela coluna *Domínio do Formulário*.

Id	Formulário Web
1	http://onlineraceresults.com/search/my_results.php
2	http://jobs.careerbuilder.com/Jobseeker/Jobs/JobResults.aspx?
3	http://www.beerintheevening.com/pubs/search.shtml
4	http://www.posteritati.com/advanced_search.php
5	http://www.mldb.org/search-bf
6	http://www.e4s.co.uk/
7	http://dojapp.doj.ca.gov/missing/default.asp
8	http://www.putporkonyourfork.com/recipes/search.html
9	http://www.drinkgenius.com/advanced-search.php
10	http://www.pamperedchef.com/recipe_search/search.jsp
11	http://www.islandnet.com/~bolen/search.php
12	http://www.bookstellyouwhy.com/advSearch.php?action=search
13	http://www.movlic.com/k12/advancedResults.asp
14	http://college.swankmp.com/business/advancedResults.asp
15	http://www.jewocity.com/event/list/by-advanced-search.htm?
16	http://www2.electrolux.com.br/servicos/gastronomia/pagina_receita.asp
17	http://empregocerto.uol.com.br/search/jobs.html
18	http://www.guiadeempregos.com.br/db/area_oferta.php3
19	http://www.drinkjockey.com/search.html
20	http://www.cocktailmaking.co.uk/searchexecute.php
21	http://www.islamicity.com/food/AdvanceSearch.asp
22	http://www.conagrafoods.com/consumer/recipes/index.jsp
23	http://www.organicvalley.coop/recipes/search-recipes
24	http://www.seattlejobs.org/listings.php
25	http://minnesotajobs.com/
26	http://minnesotajobs.com/browse-by-company/
27	http://www.biographi.ca/009004-100.01-e.php
28	http://www.filmbug.com/site/advanced.php
29	http://www.filmarchive.org.nz/the-catalogue/advanced-search/
30	http://www.movieflix.com/search.php?advanced=1
31	http://www.barnesandnoble.com/search.asp
32	http://www.powells.com/s
33	http://www.hachettebookgroup.com/search/advanced/
34	http://www.dauntbooks.co.uk/search.asp?TAG=&CID=
35	http://events.sas.ac.uk/support-research/events/advanced-search
36	http://runeberg.org/search.pl
37	http://soundcloud.com/people/search?q[country_code]=&advanced=1
38	http://womenwriters.library.emory.edu/ewwrp/advancedsearch.php
39	http://www.eventim.co.uk/Tickets.html?affiliate=EUK&doc=search/extendedSearch
40	http://www.eventim.de/Tickets.html?affiliate=TUG&doc=search/extendedSearch
41	http://www.cliccalavoro.it/
42	http://www.talentmanager.com/elenco_annunci.php
43	http://www.ivid.it/cerca-film
44	http://www.maremagnum.com/ricerca-avanzata/
45	http://www.ticketone.it/biglietti.html?affiliate=ITT&doc=search/extendedSearch&fun=search&action=power&show=for
46	http://www.unive.it/nqcontent.cfm?a_id=415
47	http://bur.rcslibri.corriere.it/libri/ricerca/catalogo_nuova.action
48	http://www.bath.ac.uk/contact/?search=advanced
49	http://www.eventfinder.co.nz/search
50	http://www.submarino.com.br/portal/BuscaAvancadaLivros/

Tabela 5.1 – Lista de Formulários Web

Custo das Abordagens. O número de URLs determina o custo da abordagem. O número de URLs submetidas (*nSubmitted URLs* na Tabela 5.2) mostra o total de consultas enviadas para o formulário. O número de URLs produtivas (*nProductive URLs*) é o subconjunto de URLs submetidas que retornam dados úteis. Para a maioria dos formulários, o número de URLs submetidas foi maior para a abordagem *FTF* que para a abordagem *smartFTF*. O ganho da abordagem *smartFTF* em relação a *FTF* foi pouco menos de 10%. Para a maioria dos formulários, o número de URLs produtivas na abordagem *FTF* foi superior à abordagem *smartFTF*. Isso significa que o custo da abordagem *smartFTF* é menor que o custo da *FTF*, e são necessárias menos submissões para alcançar uma cobertura igual ou maior. Em ambas as abordagens, a probabilidade de *templates* serem informativos é alta, comparada com o *baseline*. Embora *templates* com dimensão maior que 1 tenham alta probabilidade de serem informativos, foi observado que para formulários que contém somente campos textuais, os *templates* com dimensão maior que 1 possuem uma pequena probabilidade de serem informativos. O *baseline* tem o menor número de URLs (*nProductive URLs*). Isso ocorre porque o algoritmo utilizado pelo *baseline* seleciona os valores somente com a medida de TFxIDF.

Custo e Eficiência. Outra avaliação interessante é a relação entre o custo e a eficiência das abordagens. Existem dois tipos de custos envolvidos no processo: o custo de descobrir os valores para os campos textuais e o custo de recuperar os dados escondidos por trás dos formulários. O *baseline* não possui custo de descoberta de valores, porque os valores são escolhidos pela medida de seu TFxIDF. *FTF* e *smartFTF* possuem o custo de descoberta de valores, representados pela coluna *nExploration* na Tabela 5.2.

A relação entre o número de URLs utilizadas para descobrir os valores (*nExploration URLs*) e o número de campos textuais (*#campos*), mostrada em parêntesis na Tabela 5.2, apresenta resultados interessantes. O número médio de submissões para descobrir valores que aumentam a cobertura é de 278 URLs por campo texto para a abordagem *FTF* e de 574 para a abordagem *smartFTF*. Esse valor pode crescer mais quando os campos textuais não são do tipo *keyword* em ambas as abordagens. A abordagem *smartFTF* necessita mais submissões (*template instances*) para encontrar valores comparada com *FTF*. Isso ocorre porque *smartFTF* precisa computar as características para o modelo de aprendizagem. Em alguns formulários Web, mais especificamente os formulários 2, 10, 11, 14, 19 e 20, *smartFTF* seleciona valores para campos textuais com menos submissões. A Tabela 5.2 mostra essa informação por meio das colunas *nExploration URLs FTF* e *nExplorationURLs smartFTF*.

Cobertura, Eficiência de Execução e Produtividade. A Tabela 5.3 ilustra os resultados para as três métricas de avaliação. A *Eficiência de Execução* e a *Produtividade* são representadas pelo número médio de registros recuperados pelas submissões. Os melhores valores são mostrados em negrito.

Analisando os resultados para a cobertura, o ganho médio obtido pela abordagem *smartFTF* em relação ao *baseline* foi de 84% e, em relação à abordagem *FTF* foi de 18%. Isso mostra que a abordagem *smartFTF* foi eficiente para descobrir mais dados no banco de dados escondido por trás do formulário. Por exemplo, no formulário 17, o *baseline* e *FTF* possuem menos *templates* informativos que *smartFTF*, e *smartFTF* alcançou uma cobertura bem superior. *smartFTF* também possui melhor cobertura para a maioria dos formulários que contém somente campos textuais. *FTF* foi melhor que *smartFTF* nos formulários 3, 8, 12, 23 e 38, porque o modelo gerado selecionou menos valores que o

	nSubmmited URLs			nProductive URLs			nExploration URLs		#Campos		Domínio do
Id	Baseline	FTF	smartFTF	Baseline	FTF	SmartFTF	FTF	SmartFTF	Texto	Seleção	Formulário
1	94	919	913	29	180	179	242 (121.00)	642 (321.00)	2	0	Events
2	76	76	143	73	76	143	1100 (550.00)	1100 (550.00)	2	0	Jobs
3	214	289	286	173	288	284	300 (100.00)	900 (300.00)	3	1	Food & Drink
4	87	92	93	56	87	87	100 (100.00)	300 (300.00)	1	1	Movies
5	1400	955	916	512	604	496	400 (100.00)	1669 (417.25)	4	0	Movies
6	55	100	62	55	100	62	100 (100.00)	200 (200.00)	1	2	Jobs
7	50	160	154	20	160	154	822 (274.00)	2821 (940.33)	3	0	People
8	595	1057	276	506	962	242	100 (100.00)	200 (200.00)	1	4	Food & Drink
9	109	126	128	83	125	127	681 (227.00)	731 (243.67)	3	1	Food & Drink
10	700	4448	3103	671	4448	3103	5685 (1421.25)	5661 (1415.25)	4	0	Food & Drink
11	100	159	166	45	159	166	2615 (653.75)	2615 (653.75)	4	0	Books
12	3200	3275	3328	3200	1287	529	336 (84.00)	1332 (333.00)	4	0	Books
13	100	171	123	57	171	123	500 (166.67)	3000 (1000.00)	3	0	Movies
14	150	187	157	78	187	157	3000 (1000.00)	3000 (1000.00)	3	0	Movies
15	52	269	96	48	269	96	150 (150.00)	900 (900.00)	1	2	Events
16	n/a	50	82	n/a	50	81	320 (160.00)	687 (343.50)	2	0	Food & Drink
17	106	120	950	88	120	782	768 (384.00)	1000 (500.00)	2	1	Jobs
18	42	165	96	32	144	84	213 (106.50)	750 (375.00)	2	1	Jobs
19	50	84	84	42	84	84	100 (100.00)	100 (100.00)	1	0	Food & Drink
20	50	122	97	33	122	96	439 (219.50)	370 (185.00)	2	0	Food & Drink
21	68	139	70	57	136	64	530 (176.67)	903 (301.00)	3	2	Food & Drink
22	104	524	439	100	397	322	100 (100.00)	200 (200.00)	1	3	Food & Drink
23	511	554	473	437	469	425	100 (100.00)	200 (200.00)	1	4	Food & Drink
24	40	86	87	30	86	87	100 (100.00)	200 (200.00)	1	1	Jobs
25	92	154	137	84	149	131	1550 (775.00)	1617 (808.50)	2	2	Jobs
26	50	27	49	27	25	49	900 (450.00)	1527 (763.50)	2	1	Jobs
27	2491	3419	2453	1639	2420	1910	696 (232.00)	1873 (624.33)	3	2	People
28	1271	1295	1287	986	1014	1006	150 (150.00)	300 (300.00)	1	6	Movies
29	541	1069	2138	513	917	1580	412 (68.67)	884 (147.33)	6	2	Movies
30	100	162	265	77	162	265	1550 (310.00)	4007 (801.40)	5	0	Movies
31	100	143	175	72	143	175	200 (100.00)	400 (200.00)	2	0	Books
32	799	920	911	601	727	756	398 (99.50)	1200 (300.00)	4	0	Books
33	100	76	151	60	76	151	1000 (333.33)	2700 (900.00)	3	0	Books
34	100	147	196	69	144	195	700 (233.33)	1925 (641.67)	3	0	Books
35	74	95	74	65	95	74	100 (100.00)	1350 (1350.00)	1	2	Events
36	70	685	199	45	681	199	867 (173.40)	3363 (672.60)	5	0	People
37	1004	1111	1100	846	866	788	300 (100.00)	900 (300.00)	3	1	People
38	150	588	225	88	553	225	1350 (270.00)	4893 (978.60)	5	0	People
39	77	110	152	60	104	145	1050 (350.00)	2400 (800.00)	3	1	Events
40	151	194	208	110	185	199	750 (250.00)	2400 (800.00)	3	1	Events
41	50	95	117	49	94	117	200 (100.00)	1100 (550.00)	2	0	Jobs
42	50	81	128	43	81	128	150 (75.00)	905 (452.50)	2	0	Jobs
43	133	269	251	107	267	248	846 (169.20)	2100 (420.00)	5	1	Movies
44	1750	1860	1855	1199	1379	1467	300 (100.00)	900 (300.00)	3	0	Books
45	97	107	160	66	92	145	1000 (333.33)	2700 (900.00)	3	1	Events
46	100	99	142	32	72	142	1350 (450.00)	4050 (1350.00)	3	0	People
47	50	60	66	35	60	66	600 (300.00)	800 (400.00)	2	0	Books
48	50	80	58	44	80	58	550 (275.00)	1000 (500.00)	2	0	People
49	100	192	184	93	185	183	200 (200.00)	200 (200.00)	1	0	Events
50	100	350	107	19	237	102	454 (151.33)	1207 (402.33)	3	0	Books
Média	**359,24**	**550,30**	**502,20**	**274,57**	**430,38**	**369,54**	**728 (278)**	**1523 (581)**			

Tabela 5.2 – Detalhes dos formulários utilizados nos experimentos

método heurístico.

A abordagem *smartFTF* é melhor que *FTF* em termos de *Eficiência de Execução* em 41 formulários. Nos nove formulários restantes, a *Eficiência de Execução* de ambas as abordagens foram similares. *SmartFTF* executou melhor que o *baseline* em 37 formulários. A superioridade em *Eficiência de Execução* em relação ao *baseline* foi muito superior. Por exemplo, para o formulário 36 a superioridade foi de 1281% em relação ao *baseline* e para o formulário 8 a superioridade foi 267% em relação ao *FTF*. Isso significa que, no formulário 36, enquanto o *baseline* recupera quatro registros por URL, *smartFTF* recupera 55 registros por URL. Considerando a média geral em todos os formulários, o ganho em *Eficiência de Execução* para a abordagem *smartFTF* foi 71% em relação ao *baseline* e 37% em relação à *FTF*. O *baseline* foi mais eficiente nos formulários 10 e 42 porque o *baseline* alcançou apenas *templates* informativos de dimensão 1. A tabela 5.2 apresenta que o *baseline* realizou apenas 700 e 50 submissões enquanto que a abordagem *smartFTF* submeteu 3.103 e 128, para os formulários 10 e 42 respectivamente.

Considerando a *Produtividade*, *smartFTF* foi melhor que *FTF* em 42 formulários e melhor que o *baseline* em 33 formulários. Isso significa que, para alcançar a mesma cobertura, o número de URLs submetidas por *smartFTF* é menor que *FTF* e *baseline*. Por exemplo, para o formulário 12 (ver Tabela 5.3), *smartFTF* recupera, em média, 21 linhas por submissão enquanto *FTF* recupera oito linhas e o *baseline* recupera apenas duas linhas. A média geral da produtividade entre todos os formulários para *smartFTF* foi de 48% em relação ao *baseline* e de 42% em relação a *FTF*.

Para a maioria dos formulários, *smartFTF* foi melhor que as outras abordagens em todas as métricas. A maior melhoria foi em termos de cobertura comparado com o *baseline*. Em média, *smartFTF* necessita menos submissões para alcançar alta cobertura. Esses resultados indicam que *smartFTF* pode simultaneamente maximizar a cobertura e minimizar o custo.

Resultados para Tipos de Campos. A Figura 5.1 apresenta os resultados experimentais para as três métricas agrupadas por tipos de campos (somente campos textuais e a combinação de campos textuais com outros tipos de campos). Existem 27 formulários com campos textuais somente e 23 formulários com campos textuais e outros tipos de campos. A Figura 5.1(a) mostra os resultados para a cobertura para as três abordagens separadas por tipos de campos. A cobertura da *smartFTF* foi superior para ambos os tipos de campos. Nota-se que o maior ganho foi em formulários que possuem somente campos textuais, os quais são os mais difíceis de preencher.

Os resultados mostrados na Figura 5.1(b) para eficiência de execução também mostram vantagem para a abordagem *smartFTF*. O principal ganho aqui é que *smartFTF* melhora os resultados comparado à abordagem *FTF* para a combinação de campos textuais com os demais campos. A comparação entre o *baseline* e *FTF* apresenta vantagens para o *baseline*. A introdução de técnicas de aprendizagem de máquina agregou ganho em relação a *FTF* e o *baseline*.

A respeito da produtividade, *smartFTF* foi melhor que *FTF* para ambos os tipos de campos. O *baseline* foi melhor que *smartFTF* em formulários que combinam campos textuais e campos de seleção. Essa foi a única métrica em que *smartFTF* obteve desempenho inferior comparado ao *baseline*, no entanto, a diferença foi muito pequena.

Atributos e Domínios de Formulários. Outro resultado interessante é que existem relações entre os valores selecionados e o domínio dos atributos. Embora

	Baseline			Abordagem FTF			Abordagem SmartFTF		
Id	Cobertura (#registros)	Eficiência Execução	Produtividade	Cobertura (#registros)	Eficiência Execução	Produtividade	Cobertura (#registros)	Eficiência Execução	Produtividade
1	4.320	**45,96**	**148,97**	14.042	15,28	78,01	**17.072**	18,70	95,37
2	1.651	21,72	22,62	1.262	16,60	16,60	**3.330**	**23,28**	**23,28**
3	1.659	7,75	9,59	**3.515**	**12,16**	**12,20**	3.117	10,89	10,97
4	2.605	29,94	**46,52**	2.863	31,11	32,90	**2.908**	**31,26**	33,04
5	6.347	4,53	12,40	7.676	8,03	12,70	**8.539**	**9,32**	**17,21**
6	320	5,82	5,82	482	4,82	4,82	**493**	**7,95**	**7,95**
7	73	1,46	3,65	670	4,18	4,18	**867**	**5,63**	**5,63**
8	285	0,48	0,56	**287**	0,27	0,29	274	**0,99**	**1,13**
9	1.100	10,09	13,25	1.949	15,47	15,59	**2.507**	**19,58**	**19,74**
10	621	**0,89**	**0,93**	737	0,16	0,16	**757**	0,24	0,24
11	1.105	11,05	20,09	2.891	18,18	18,18	**4.575**	**27,56**	**27,56**
12	6.483	2,03	2,03	**11.373**	**3,47**	8,84	11.217	3,37	**21,20**
13	2.438	24,38	42,77	7.089	41,46	41,46	**9.924**	**80,68**	**80,68**
14	4.151	27,67	53,22	9.129	48,82	48,82	**16.960**	**108,02**	**108,02**
15	**249**	**4,79**	**5,19**	206	0,76	0,76	211	2,20	2,20
16	-	-	-	162	**3,24**	**3,24**	**212**	2,58	2,61
17	615	**5,80**	**6,99**	670	5,58	5,58	**4.096**	4,25	5,17
18	205	**4,88**	**6,41**	337	2,04	2,34	**342**	3,56	4,07
19	1.922	38,44	45,76	4.264	50,76	50,76	**4.284**	**51,00**	**51,00**
20	451	9,02	13,67	1.722	14,11	14,11	**1.801**	**18,57**	**18,76**
21	76	**1,12**	**1,33**	77	0,55	0,57	**77**	1,10	1,20
22	373	**3,59**	**3,73**	821	1,57	2,07	**825**	1,88	2,56
23	**1.938**	**3,79**	**4,43**	1.913	3,45	4,08	1.471	3,11	3,46
24	648	16,20	**21,60**	1.095	12,73	12,73	**1.823**	**20,95**	20,95
25	809	**8,79**	**9,63**	924	6,00	6,20	**931**	6,79	7,11
26	37	0,74	1,37	47	**1,74**	**1,88**	**68**	1,39	1,39
27	4.994	2,00	3,04	7.335	2,14	3,03	**7.957**	**3,24**	**4,16**
28	5.314	4,18	5,39	5.428	4,19	5,35	**5.595**	**4,35**	**5,56**
29	7.212	**13,33**	**14,06**	8.704	8,14	9,49	**14.477**	6,17	9,16
30	267	2,67	2,67	634	**3,91**	**3,91**	**1.005**	3,79	3,79
31	708	7,08	9,83	1.369	9,57	9,57	**1.732**	**9,90**	**9,90**
32	10.229	12,80	17,02	13.471	14,64	14,53	**14.509**	**15,93**	**19,19**
33	614	6,14	10,23	949	12,49	12,49	**1.987**	**13,16**	**13,16**
34	944	9,44	13,68	1.886	12,83	13,10	**2.714**	**13,85**	**13,92**
35	535	7,23	5,23	537	5,65	5,65	**538**	**7,27**	**7,27**
36	281	4,01	6,24	10.395	15,17	15,26	**11.023**	**55,39**	**55,39**
37	25.363	**25,26**	29,98	27.111	24,40	31,31	**27.328**	24,84	**34,68**
38	1.713	11,42	19,46	**10.936**	18,59	19,77	5.137	**22,83**	**22,83**
39	509	6,61	**8,48**	668	6,07	6,42	**1.111**	**7,31**	7,66
40	1.735	11,49	15,77	2.812	14,49	15,20	**3.348**	**16,10**	**16,82**
41	570	**11,40**	**11,63**	806	8,48	8,57	**1.276**	10,91	10,91
42	948	**18,96**	**22,05**	1.440	17,78	17,78	**1.707**	13,34	13,34
43	1.657	12,46	**15,49**	2.928	10,88	10,97	**3.486**	**13,89**	14,06
44	15.666	8,95	13,07	17.426	9,37	12,64	**21.386**	**11,53**	**14,58**
45	554	5,71	8,39	953	8,91	10,36	**1.618**	**10,11**	**11,16**
46	313	3,13	9,78	570	5,76	7,92	**1.898**	**13,37**	**13,37**
47	222	4,44	6,34	398	6,63	6,63	**510**	**7,73**	**7,73**
48	792	15,84	18,00	**3.499**	43,74	43,74	2.729	**47,05**	**47,05**
49	802	8,02	8,62	1.663	8,66	8,99	**1.686**	**9,16**	**9,21**
50	90	0,90	4,74	1.576	4,49	6,65	**1.948**	**18,21**	**19,10**
Média	2.500	10,29	15,95	3.994	11,99	13,97	**4.708**	**16,49**	**19,13**

Tabela 5.3 – Resultados das Métricas de Avaliação

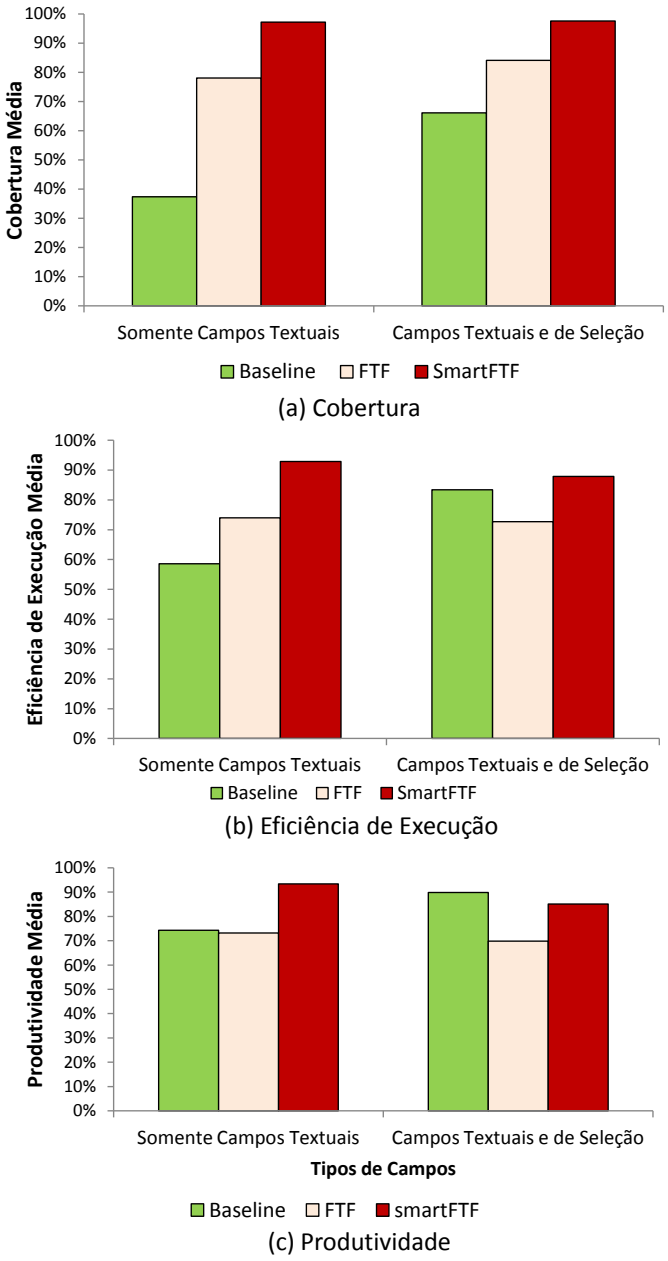


Figura 5.1 – Avaliação das métricas para tipos de campos nos formulários

smartFTF seja independente de domínio, os valores selecionados para cada campo textual têm um relacionamento com o domínio do atributo. Por exemplo, se o campo é o nome de uma pessoa, os valores selecionados são, usualmente, nomes próprios. Se o campo solicita um ingrediente em uma receita, os valores associados são relacionados a alimentação. Na Figura 5.2, são mostrados os primeiros dez valores selecionados pela abordagem *smartFTF* para o formulário disponível em `http://www.foodandwine.com/search/recipe.cfm` para os campos *recipe*, *chef* e *ingredient*.

Outros resultados interessantes estão relacionados ao comportamento do domínio do formulário. A Figura 5.3 mostra os resultados alcançados para todas as métricas nas três abordagens separadas por domínio de formulário (`books`, `events`, `food&drink`,

Recipe	Chef	Ingredient
Ribs	Clark	Ham
Cake	Jaconson	Hearts
Beef	Rubel	Peanut
Goat	Steven	Ricotta
Pasta	Melissa	Parsley
Sandwiches	Chris	Apple
Salmon	Michael	Butter
Cocktail	Vongerichten	Lemon
Stew	Jacques	Mushroom
Potatoes	Nancy	Nuts

Figura 5.2 – Exemplo de valores selecionados para o formulário Web em `http://www.foodandwine.com/search/recipe.cfm`

`jobs`, `movies` e `people`).

A Figura 5.3(a) mostra os resultados para a cobertura. A cobertura da abordagem *smartFTF* foi superior em todos os domínios comparado com o *baseline*. No domínio `people`, a abordagem *smartFTF* se comportou melhor que a *FTF*, mas a diferença foi muito pequena. Isso ocorre porque na abordagem *FTF*, por exemplo para o formulário com `id` igual a 38, *templates* com dimensão maior que 1 foram considerados informativos enquanto que para a abordagem *smartFTF*, esses *templates* foram considerados não informativos.

Os resultados para a eficiência de execução são mostrados na Figura 5.3(b). A eficiência de execução da abordagem *smartFTF* foi superior em todos os domínios comparado à *FTF*. O *baseline* foi melhor que *smartFTF* em dois domínios: `events` e `jobs`, mas a diferença foi muito pequena. Por exemplo, no formulário 1, o número de submissões realizada pelo *baseline* foi muito menor que a abordagem *smartFTF*.

Finalmente, a Figura 5.3(c) apresenta os resultados para a produtividade. A produtividade da *smartFTF* foi superior em todos os formulários comparado com *FTF*. Em quatro domínios, `books`, `food&drink`, `movies` e `people`, a abordagem *smartFTF* foi melhor que o *baseline*. Nos domínios de formulários que *FTF* foi pior que o *baseline*, *smartFTF* melhorou os resultados (`movies` e `food&drink`).

Outra questão interessante é como o número total de linhas recuperadas se comporta com o número de submissões realizadas. A Figura 5.4 apresenta essa visualização para o formulário com `id` igual a 44 (ver Tabela 5.1). O comportamento inicial é similar para todas as abordagens, especialmente para *FTF* e *smartFTF*. O *baseline* alcança um platô no gráfico quando o número de submissões está em torno de 1.420. Partindo deste valor, nenhuma nova informação é recuperada. Até esse ponto, a abordagem *FTF* recuperou menos linhas que o *baseline*, mas ela nivelou depois de 1.650 submissões, atingindo uma cobertura maior que o *baseline*. Ainda, no ponto em que o *baseline* estabilizou, a abordagem *smartFTF* recuperou quase a mesma cobertura, mas *smartFTF* continuou mantendo a recuperação de novas linhas, estabilizando em torno de 1.850 submissões. Enquanto o *baseline* parou de submeter consultas em torno de duas mil consultas, as abordagens *FTF* e *smartFTF* pararam de submeter em torno de 2.100, mas o ganho em

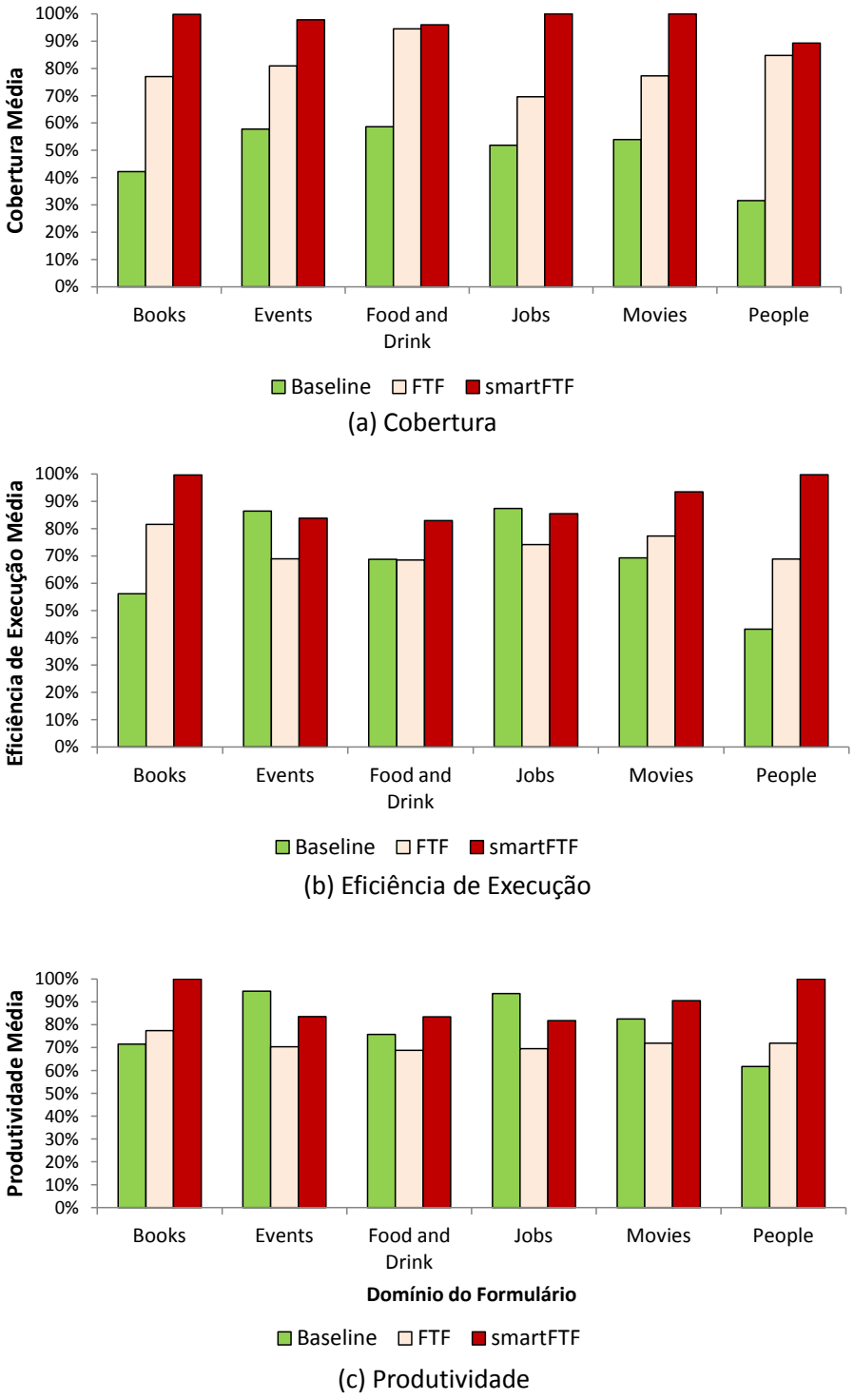


Figura 5.3 – Métricas de Avaliação Agrupadas por Domínio de Formulário

termos de novas informações é maior que 5 mil. A maioria dos formulários usados nos experimentos tem um comportamento similar.

Significância estatística. A Figura5.5 mostra a média das três métricas de avaliação para todos os formulários. Percebe-se que o *smartFTF* é superior em todas as métricas. Para

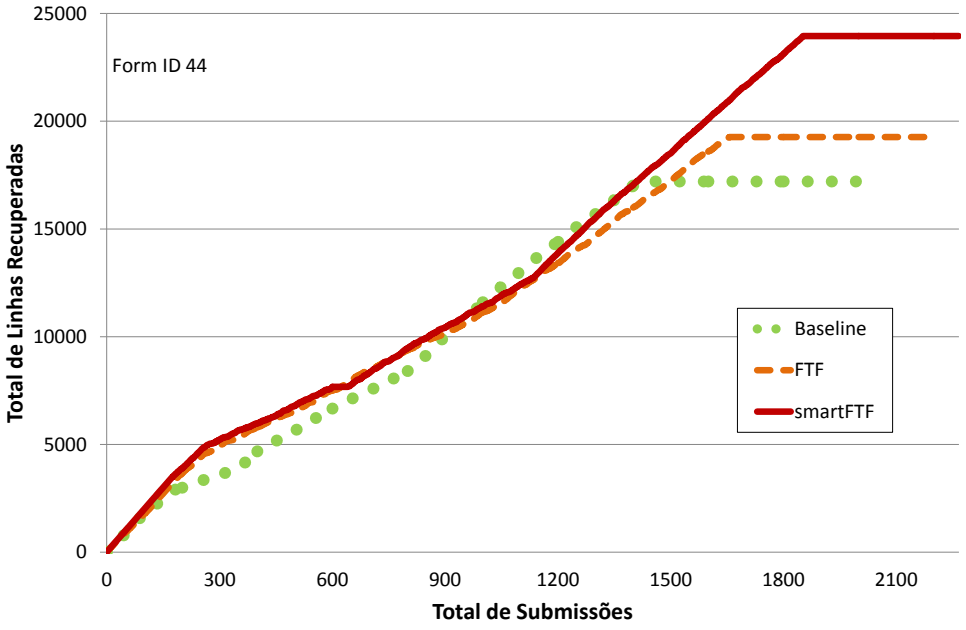


Figura 5.4 – Número de linhas recuperadas como função do número de submissões

determinar se os experimentos produziram melhoria estatisticamente significantes, um *test-t* pareado foi executado. O limiar padrão para significância estatística (α) de 0,05 foi utilizado. Os *p-values* calculados foram muito menores que α em todos os casos, o que demonstra que os ganhos são, de fato, significativos.

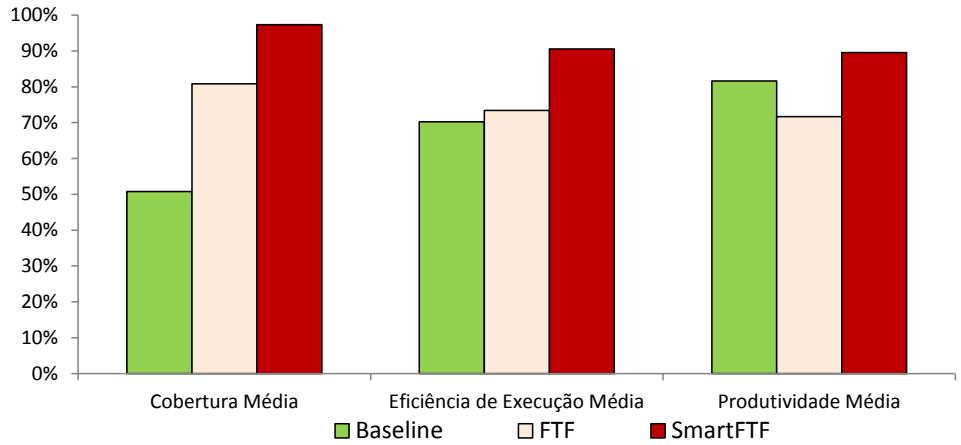


Figura 5.5 – Média das Métricas de Avaliação

Teste com outros algoritmos de aprendizagem de máquina. Todos os experimentos realizados até o presente momento empregaram o *Multilayer Perceptron* (MLP) como algoritmo de aprendizagem. A fim de avaliar outros algoritmos de aprendizagem, foram realizados experimentos com mais quatro algoritmos que empregam diferentes técnicas de aprendizagem, mas especificamente, árvores de decisão (C4.5-J48) (QUINLAN,

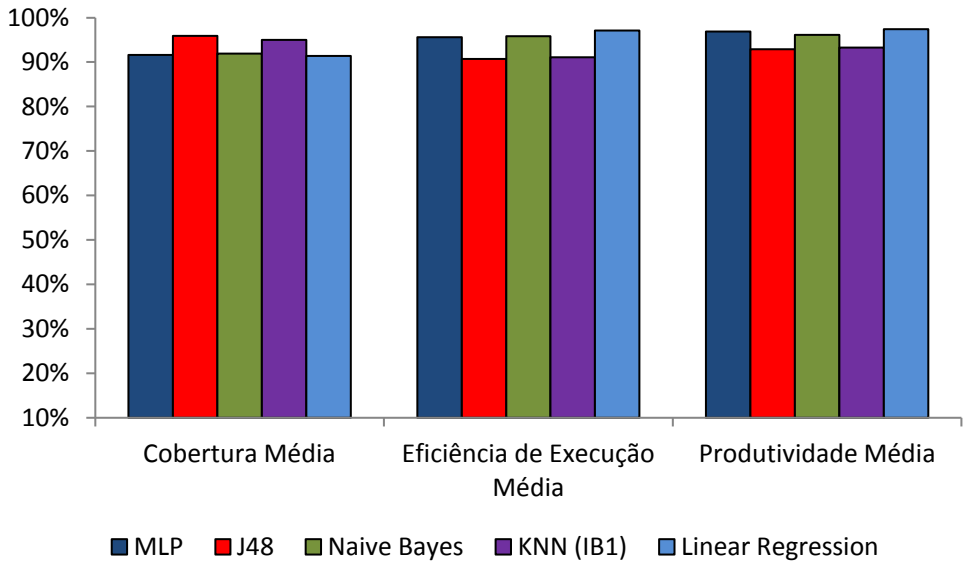


Figura 5.6 – Média das Métricas de Avaliação para vários algoritmos de aprendizagem de máquina

1993), naïve bayes (JOHN; LANGLEY, 1995), kNN (*k-nearest-neighbor*) (AHA; KIBLER; ALBERT, 1991) e classificação por regressão. A classificação por regressão é uma combinação de modelos. A ideia é adicionar um novo atributo que possui a saída da regressão. Então, um algoritmo de classificação é usado para otimizar o ponto de divisão entre as classes. Para a avaliação experimental, foi utilizado o algoritmo *OneR* (HOLTE, 1993). Os testes foram realizados em um subconjunto de 12 formulários aleatoriamente selecionados. Esse subconjunto inclui os formulários 1, 11, 15, 31, 33, 34, 35, 39, 40, 44, 45 e 47 apresentados na Tabela 5.1. A Figura 5.6 mostra os resultados dessa comparação. Pode-se observar que o desempenho médio para os cinco algoritmos é muito semelhante. Todos os algoritmos alcançaram valores acima de 90% em todas as métricas. A cobertura média para o algoritmo MLP, KNN e J48 foi em torno 95%. O algoritmo de regressão linear alcançou a menor cobertura, em torno de 92%, mas alcançou as maiores médias para eficiência de execução e produtividade, em torno de 97%. Um *test-t* pareado produziu *p-values* maiores que $\alpha = 0,05$ em todos os casos, o que demonstra que qualquer um dos cinco algoritmos podem ser utilizados para a seleção de valores.

Seleção de Características. Um outro ponto interessante é avaliar as características utilizadas no classificador para decidir quais são as melhores. Uma avaliação foi realizada para um formulário (id = 44) usando *correlation-based feature selection* (CFS). CFS avalia a qualidade de um subconjunto de características considerando a capacidade preditiva individual de cada característica, juntamente com o grau de redundância entre elas (HALL, 1999). Subconjuntos de características que possuem alta correlação com a classe são preferidas em relação àquelas que têm baixa correlação. Os experimentos demonstraram que características diferentes são consideradas mais úteis para campos diferentes. Por exemplo, para cada um dos três campos no formulário 44: *parola chiave*, *autore* e *titolo* existe um conjunto de treinamento. Os resultados para *parola chiave* mostram que somente *cfidf* e *cf* são necessários para a previsão. Para o campo *autore*, as

características selecionadas são *cfidf* e *linhasdistintas*. Finalmente, para o campo *titolo* as características selecionadas para geração do modelo são *cfidf*, *linha* e *linhasdistintas*. Isto evidencia que a combinação de características é uma necessidade para prover previsões para uma grande faixa de campos.

5.3 Resumo do Capítulo

Este capítulo apresentou a avaliação experimental do método proposto na tese. Os experimentos foram realizados em vários formulários reais existentes na Web. Uma vasta avaliação experimental foi realizada considerando várias dimensões, tais como, tipos de campos, domínios de formulários, diferentes algoritmos de aprendizagem. Os resultados demonstraram que o método é eficiente, selecionando valores para campos de formulários aumentando a cobertura e minimizando o custo de submissões ao formulário. A utilização de vários algoritmos de aprendizagem de máquina combinados com o método da tese alcançou resultados muito similares. A seleção de características também foi avaliada e a necessidade de várias características é mandatória pois dependendo do campo que está sendo processado a combinação de características para seleção pode variar para campos distintos.

6 CONCLUSÃO

Este capítulo apresenta as conclusões alcançadas com esta tese e discute algumas direções futuras no preenchimento de formulários na Web Oculta. As contribuições da tese são resumidas e associadas à produção científica resultante.

A Web Oculta ainda representa um grande desafio de pesquisa. Encontrar informações escondidas por trás de formulários e que não estão indexadas por máquinas de busca tradicionais é um desafio. Além disso, a pesquisa no preenchimento automático de formulários na Web justifica-se no momento em que esta não é uma tarefa usual. O projeto de formulários é diversificado e as interfaces são projetadas para serem utilizadas por pessoas.

Esta tese apresentou uma visão sobre uma das três fases para busca de conteúdo na Web Oculta: identificação, preenchimento e submissão de valores para campos em formulários Web. No decorrer do texto, foram abordados os trabalhos relacionados (RAGHAVAN; GARCIA-MOLINA, 2001; LIDDLE et al., 2003; BARBOSA; FREIRE, 2004; WU et al., 2006; NTOULAS; ZERFOS; CHO, 2005; ÁLVAREZ et al., 2007; TODA et al., 2010; DRAGUT et al., 2009; LAGE et al., 2004; SOULEMANE; RAFIUZZAMAN; MAHMUD, 2012; MADHAVAN et al., 2008; WANG et al., 2012) ao tema desta tese, e foi apresentada uma classificação dos trabalhos segundo dois vieses: um viés de estratégia de seleção de valores, heurísticas e/ou aprendizagem de máquina e um viés de estrutura dos bancos de dados por trás de formulários baseado na interface, complexa ou entrada textual livre. Verifica-se que a maioria dos trabalhos (RAGHAVAN; GARCIA-MOLINA, 2001; LIDDLE et al., 2003; BARBOSA; FREIRE, 2004; WU et al., 2006; NTOULAS; ZERFOS; CHO, 2005; ÁLVAREZ et al., 2007; LAGE et al., 2004; SOULEMANE; RAFIUZZAMAN; MAHMUD, 2012; MADHAVAN et al., 2008; WANG et al., 2012) utiliza heurísticas para seleção de valores e manipula somente um tipo de interface associada à estrutura do banco de dados por trás do formulário.

A contribuição principal desta tese é uma solução automática de preenchimento de formulários complexos e textuais na Web Oculta. A solução descrita no decorrer do texto combina heurística e aprendizagem de máquina para melhorar a seleção de valores de campos. A intenção é utilizar as regras práticas das heurísticas para a geração de conjuntos de treinamento associados aos campos do formulário. Esses conjuntos de treinamento são utilizados para a construção de modelos de classificação para seleção de valores de campos de formulários. A solução proposta demonstrou na experimentação funcionar bem para formulários com ambos os tipos de interface: textuais livres e interfaces complexas.

Uma arquitetura foi desenvolvida para o preenchimento de formulários. A arquitetura contempla todas as etapas para a coleta na Web Oculta. Um componente é destacado nessa arquitetura: seleção de valores. A seleção de valores para campos textuais não é

trivial e ainda é um desafio de pesquisa. Esta tese apresentou uma abordagem chamada *smartFTF* que possui duas fases. A primeira fase, denominada *fase heurística* utiliza regras práticas que geram conjuntos de treinamento para campos textuais, baseado em características das submissões. A segunda, denominada *fase de aprendizagem*, um modelo de classificação é criado partindo dos conjuntos de treinamento e os novos valores são previstos baseados nesse modelo. A abordagem é totalmente automática e não necessita de intervenção de especialistas. Uma extensa avaliação experimental foi realizada e os resultados comprovam o bom comportamento da abordagem comparada a outras técnicas de seleção de valores.

Esta tese responde a algumas das questões apresentadas na Introdução. Por meio do conceito de informatividade de *templates* e *template instances* é possível responder quais os campos de um formulário devem ser preenchidos. Os campos que pertencem a *templates* considerados informativos serão selecionados para o preenchimento. A questão sobre quais os valores que devem ser colocados nesses campos está da mesma forma atendida, uma vez que esta tese apresenta uma abordagem de seleção automática de valores para campos de um formulário.

A abordagem *smartFTF* foi avaliada em cinquenta formulários reais existentes na Web e em vários domínios de formulários, tais como, `Jobs`, `Events`, `Movies`, `Food&Drink`, `Books`, `People`. Os resultados alcançados com os experimentos apresentam melhorias em relação ao *baseline* utilizado nos experimentos. Como produção científica resultante dos experimentos é possível citar:

- XXVI Simpósio Brasileiro de Banco de Dados (SBBD), 2011. Seleção de Valores para Preenchimento Automático de Formulários Web. pp. 139–146.
- 6º Alberto Mendelzon International Workshop on Foundations of Data Management, 2012. Automatic Filling of Web Forms. 215–219.
- 16ª East-European Conference on Advances in Databases and Information Systems (ADBIS). 2012. Choosing Values for Text Fields in Web Forms. Advances in Databases and Information Systems. Advances in Intelligent Systems and Computing. Volume 186, 2013, pp 125–136.
- Submissão para o periódico *Sigmod Record* do artigo intitulado *Automatically Filling Complex Hidden Web Forms: A Survey*, 2014. Em processo de revisão.

Trabalhos futuros podem ser direcionados para formulários Web Complexos visto que existem várias soluções propostas para formulários textuais livres. Formulários Complexos são, na maioria das vezes, associados a fontes de dados estruturadas na Web Oculta. Fontes de dados estruturadas são mais significantes quantitativa e qualitativamente, comparada a fontes de dados não estruturadas, e elas compreendem em torno de 80% de toda a Web Oculta (NOOR; RASHID; RAUF, 2011). Em formulários Web complexos, as futuras pesquisas devem se concentrar no relacionamento de múltiplos atributos do formulário. Esse desafio está diretamente associado ao entendimento de formulários Web. Entendimento de formulários é o processo de extração de informação semântica de uma interface (DRAGUT; MENG; YU, 2012; FURCHE et al., 2012; KHARE; AN; SONG, 2010). A combinação de extração de informação semântica e métodos de preenchimento é o próximo passo na melhoria do preenchimento automático de formulários Web.

Baseada nas abordagens de aprendizagem de máquina, investigações futuras podem ser realizadas na avaliação de vários modelos para determinar qual melhor se adapta à tarefa de preenchimento de formulários. Embora uma análise tenha sido realizada nesta tese, considerando cinco algoritmos (k-NN, *multilayer perceptron*, J48, classificação por

regressão e naïve bayes), abordagens de *meta learning* e combinação de modelos podem ajudar na descoberta de qual subconjunto de algoritmos de aprendizagem são mais recomendados para a tarefa de seleção de valores para preenchimento de campos em formulários. Ainda, o reuso de aprendizagem anteriores pode ser realizada para melhorar a conjunto de treinamento.

Outra questão é a possibilidade de busca incremental na Web Oculta. Coletores na Web Oculta capturam os dados, armazenam em bancos de dados locais e focam na coleta de todos os dados de uma única vez. Os sítios na Web Oculta são atualizados periodicamente e realizar a busca de todos os dados sempre que necessário pode ter um custo muito alto. Assim, autalizar os bancos de dados locais de maneira seletiva e incremental é um estudo interessante que direciona possíveis trabalhos futuros.

Finalizando, muitas das abordagens de preenchimento de formulários são projetadas para uma coleta geral na Web Oculta. Outra questão que pode ser tratada está relacionada para aqueles casos nos quais existe o interesse em uma coleta particular de um formulário na Web Oculta. Por exemplo, uma máquina de busca focada em livros especializados que procura somente o subconjunto relacionada a uma categoria específica.

REFERÊNCIAS

AHA, D. W.; KIBLER, D.; ALBERT, M. K. Instance-based learning algorithms. **Machine learning**, Springer, v. 6, n. 1, p. 37–66, 1991.

ÁLVAREZ, M. et al. Crawling the content hidden behind web forms. **ICCSA 2007**, Springer, p. 322–333, 2007.

BARBOSA, L.; FREIRE, J. Siphoning hidden-web data through keyword-based interfaces. In: **Proceedings of the XIX Brazilian Symposium on Databases**. [S.l.: s.n.], 2004. p. 309–321.

BERGMAN, M. The deep web: Surfacing hidden value. **Journal of Electronic Publishing**, v. 7, n. 1, p. 07–01, 2001.

CAO, Y. et al. Adapting ranking svm to document retrieval. In: ACM. **Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval**. [S.l.], 2006. p. 186–193.

CAVERLEE, J.; LIU, L.; BUTTLER, D. Probe, cluster, and discover: focused extraction of qa-pagelets from the deep web. In: **Data Engineering, 2004. Proceedings. 20th International Conference on**. [S.l.: s.n.], 2004. p. 103 – 114. ISSN 1063-6382.

CHANG, C.-H. et al. A survey of web information extraction systems. **Knowledge and Data Engineering, IEEE Transactions on**, IEEE, v. 18, n. 10, p. 1411–1428, 2006.

CHANG, K. et al. Structured databases on the web: Observations and implications. **ACM SIGMOD Rec**, ACM, v. 33, n. 3, p. 61–70, 2004.

COHEN, W.; RAVIKUMAR, P.; FIENBERG, S. A comparison of string metrics for matching names and records. In: **KDD Workshop on Data Cleaning and Object Consolidation**. [S.l.: s.n.], 2003. v. 3, p. 73–78.

CRESCENZI, V.; MECCA, G.; MERIALDO, P. Roadrunner: Automatic data extraction from data-intensive web sites. In: **Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data**. New York, NY, USA: ACM, 2002. (SIGMOD '02), p. 624–624. ISBN 1-58113-497-5. Available from Internet: <http://doi.acm.org/10.1145/564691.564778>.

DONG, Y.; LI, Q. A deep web crawling approach based on query harvest model. **Journal of Computational Information Systems**, v. 8, n. 3, p. 973–981, 2012.

DRAGUT, E. C. et al. A hierarchical approach to model web query interfaces for web source integration. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 2, n. 1, p. 325–336, 2009.

DRAGUT, E. C.; MENG, W.; YU, C. T. **Deep Web Query Interface Understanding and Integration**. [S.l.]: Morgan & Claypool Publishers, 2012. ISBN 1608458946, 9781608458943.

FLORESCU, D.; LEVY, A.; MENDELZON, A. Database techniques for the world-wide web: a survey. **SIGMOD Rec**, ACM, New York, NY, USA, v. 27, p. 59–74, September 1998. ISSN 0163-5808. Available from Internet: <http://doi.acm.org/10.1145/290593.290605>.

FURCHE, T. et al. Opal: automated form understanding for the deep web. In: **Proceedings of the 21st international conference on World Wide Web**. New York, NY, USA: ACM, 2012. (WWW '12), p. 829–838. ISBN 978-1-4503-1229-5. Available from Internet: <http://doi.acm.org/10.1145/2187836.2187948>.

HALL, M. et al. The weka data mining software: an update. **SIGKDD Explor. Newsl.**, ACM, New York, NY, USA, v. 11, n. 1, p. 10–18, nov. 2009. ISSN 1931-0145. Available from Internet: <http://doi.acm.org/10.1145/1656274.1656278>.

HALL, M. A. **Correlation-based feature selection for machine learning**. Thesis (PhD) — The University of Waikato, 1999.

HE, B. et al. Accessing the deep web. **Communications of the ACM**, ACM, v. 50, n. 5, p. 94–101, 2007.

HOLTE, R. C. Very simple classification rules perform well on most commonly used datasets. **Machine learning**, Springer, v. 11, n. 1, p. 63–90, 1993.

IPEIROTIS, P. G.; GRAVANO, L. Distributed search over the hidden web: Hierarchical database sampling and selection. In: VLDB ENDOWMENT. **Proceedings of the 28th international conference on Very Large Data Bases**. [S.l.], 2002. p. 394–405.

JIANG, L. et al. Efficient deep web crawling using reinforcement learning. In: **Proceedings of the 14th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining - Volume Part I**. Berlin, Heidelberg: Springer-Verlag, 2010. (PAKDD'10), p. 428–439. ISBN 3-642-13656-7, 978-3-642-13656-6. Available from Internet: <http://dx.doi.org/10.1007/978-3-642-13657-3_46>.

JIANG, L. et al. Learning deep web crawling with diverse features. In: **Proc. Int. Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01**. [S.l.: s.n.], 2009. p. 572–575.

JOHN, G. H.; LANGLEY, P. Estimating continuous distributions in bayesian classifiers. In: MORGAN KAUFMANN PUBLISHERS INC. **Proceedings of the Eleventh conference on Uncertainty in artificial intelligence**. [S.l.], 1995. p. 338–345.

KALJUVEE, O. et al. Efficient web form entry on pdas. In: ACM. **Proceedings of the 10th international conference on World Wide Web**. [S.l.], 2001. p. 663–672.

KANTORSKI, G.; MORAES, T.; HEUSER, C. Seleção de valores para preenchimento automático de formulários web. In: SBBD. **Proceedings of the XXVI Brazilian Symposium on Databases**. [S.l.], 2011. p. 139–146.

KANTORSKI, G.; MORAES, T.; HEUSER, C. Strategies for automatically filling web forms. **Technical Report RP367-PPGC, Instituto de Informatica, UFRGS, Brazil**, 2012.

KANTORSKI, G. Z. et al. Choosing values for text fields in web forms. In: **ADBIS**. [s.n.], 2013. (Advances in Intelligent Systems and Computing), p. 125–136. ISBN 978-3-642-32740-7. Available from Internet: <http://dx.doi.org/10.1007/978-3-642-32741-4_12>.

KHARE, R.; AN, Y. An empirical study on using hidden markov model for search interface segmentation. In: ACM. **Proceedings of the 18th ACM conference on Information and knowledge management**. [S.l.], 2009. p. 17–26.

KHARE, R.; AN, Y.; SONG, I. Understanding deep web search interfaces: A survey. **ACM SIGMOD Rec**, ACM, v. 39, n. 1, p. 33–40, 2010.

LAENDER, A. H. et al. A brief survey of web data extraction tools. **ACM SIGMOD Rec**, ACM, v. 31, n. 2, p. 84–93, 2002.

LAGE, J. P. et al. Automatic generation of agents for collecting hidden web pages for data extraction. **Data Knowl. Eng.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 49, n. 2, p. 177–196, may 2004. ISSN 0169-023X. Available from Internet: <http://dx.doi.org/10.1016/j.datak.2003.10.003>.

LAWRENCE, S.; GILES, C. L. Searching the world wide web. **Science**, American Association for the Advancement of Science, v. 280, n. 5360, p. 98–100, 1998.

LERMAN, K. et al. Using the structure of web sites for automatic segmentation of tables. In: **Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data**. New York, NY, USA: ACM, 2004. (SIGMOD '04), p. 119–130. ISBN 1-58113-859-8. Available from Internet: <http://doi.acm.org/10.1145/1007568.1007584>.

LIDDLE, S. et al. Extracting data behind web forms. **Advanced Conceptual Modeling Techniques**, Springer, p. 402–413, 2003.

LIU, B.; GROSSMAN, R.; ZHAI, Y. Mining data records in web pages. In: **SIGKDD**. [S.l.: s.n.], 2003. p. 601–606.

LU, Y. et al. Annotating structured data of the deep web. In: IEEE. **Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on**. [S.l.], 2007. p. 376–385.

MADHAVAN, J. et al. Google's deep web crawl. **Proc. of the VLDB Endowment**, VLDB Endowment, v. 1, n. 2, p. 1241–1252, 2008.

MORAES, M. C. et al. Pre-query discovery of domain-specific query forms: A survey. **IEEE Transactions on Knowledge and Data Engineering**, IEEE Computer Society, Los Alamitos, CA, USA, v. 99, n. PrePrints, 2012. ISSN 1041-4347.

MORAES, T. G. **Seleção de Valores para Preenchimento de Formulários Web**. Dissertation (Master) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2013.

NGUYEN, H.; NGUYEN, T.; FREIRE, J. Learning to extract form labels. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 1, n. 1, p. 684–694, 2008.

NOOR, U.; RASHID, Z.; RAUF, A. A survey of automatic deep web classification techniques. **International Journal of Computer Applications**, v. 19, n. 6, p. 43–50, abr. 2011.

NTOULAS, A.; ZERFOS, P.; CHO, J. Downloading textual hidden web content through keyword queries. In: **JCDL**. [S.l.: s.n.], 2005. p. 100–109.

QUINLAN, J. R. **C4. 5: programs for machine learning**. [S.l.]: Morgan kaufmann, 1993.

RAGHAVAN, S.; GARCIA-MOLINA, H. Crawling the hidden web. In: **Proceedings of the International Conference on Very Large Data Bases**. [S.l.: s.n.], 2001. p. 129–138.

RIBEIRO, B. A.; MUNTZ, R. A belief network model for ir. In: ACM. **Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval**. [S.l.], 1996. p. 253–260.

SALTON, G.; MCGILL, M. Introduction to modern information retrieval. McGraw-Hill, Inc., 1986.

SHAKHNAROVICH, G.; INDYK, P.; DARRELL, T. **Nearest-neighbor methods in learning and vision: theory and practice**. [S.l.: s.n.], 2006.

SHERMAN, C.; PRICE, G. **The invisible Web: Uncovering information sources search engines can't see**. [S.l.]: Information Today, Inc., 2001.

SHESTAKOV, D.; BHOWMICK, S. S.; LIM, E.-P. Deque: querying the deep web. **Data & Knowledge Engineering**, Elsevier, v. 52, n. 3, p. 273–311, 2005.

SHESTAKOV, D. et al. Search interfaces on the web: Querying and characterizing. TUCS Dissertations 104, 2008.

SOULEMANE, M.; RAFIUZZAMAN, M.; MAHMUD, H. Article: Crawling the hidden web: An approach to dynamic web indexing. **International Journal of Computer Applications**, v. 55, n. 1, p. 7–15, Oct. 2012.

SOUZA, A.; MELLO, R. Definição e avaliação de uma abordagem para extração e catalogação de conteúdo obtido da deep web. In: SBBD. **Proceedings of the XXIX Brazilian Symposium on Databases**. [S.l.], 2014. p. 77–86.

TJIN-KAM-JET, K.; TRIESCHNIGG, D.; HIEMSTRA, D. Free-text search over complex web forms. In: **Multidisciplinary Information Retrieval**. [S.l.]: Springer, 2011. p. 94–107.

TODA, G. et al. A probabilistic approach for automatically filling form-based web interfaces. **Proc. of the VLDB Endowment**, VLDB Endowment, v. 4, n. 3, p. 151–160, 2010.

WANG, J.; LOCHOVSKY, F. H. Data extraction and label assignment for web databases. In: **Proceedings of the 12th International Conference on World Wide Web**. New York, NY, USA: ACM, 2003. (WWW '03), p. 187–196. ISBN 1-58113-680-3. Available from Internet: <http://doi.acm.org/10.1145/775152.775179>.

WANG, Y. et al. Selecting queries from sample to crawl deep web data sources. **Web Intelligence and Agent Systems**, v. 10, p. 75–88, January 2012.

WU, P. et al. Query selection techniques for efficient crawling of structured web sources. In: **ICDE**. [S.l.: s.n.], 2006. p. 47–47.

WU, W. et al. Modeling and extracting deep-web query interfaces. In: **Advances in Information and Intelligent Systems**. [S.l.]: Springer, 2009. p. 65–90.

WU, W. et al. An interactive clustering-based approach to integrating source query interfaces on the deep web. In: ACM. **Proceedings of the 2004 ACM SIGMOD international conference on Management of data**. [S.l.], 2004. p. 95–106.

YI, L.; LIU, B.; LI, X. Eliminating noisy information in web pages for data mining. In: **Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2003. (KDD '03), p. 296–305. ISBN 1-58113-737-0. Available from Internet: <http://doi.acm.org/10.1145/956750.956785>.

ZHAI, Y.; LIU, B. Structured data extraction from the web based on partial tree alignment. **Knowledge and Data Engineering, IEEE Transactions on**, IEEE, v. 18, n. 12, p. 1614–1628, 2006.

ZHAO, H. et al. Fully automatic wrapper generation for search engines. In: ACM. **Proceedings of the 14th international conference on World Wide Web**. [S.l.], 2005. p. 66–75.

ZHENG, Q. et al. Learning to crawl deep web. **Information Systems**, Elsevier, v. 38, n. 6, p. 801–819, 2013.

ZIMMERMANN, H. **Fuzzy Set Theory and Its Applications Second, Revised Edition**. [S.l.]: Springer, 1992.