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

TIAGO GUIMARÃES MORAES

Seleção de Valores para Preenchimento de Formulários *Web*

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Carlos Alberto Heuser
Orientador

Porto Alegre, junho de 2013

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Moraes, Tiago Guimarães

Seleção de Valores para Preenchimento de Formulários *Web* / Tiago Guimarães Moraes. – Porto Alegre: PPGC da UFRGS, 2013.

73 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2013. Orientador: Carlos Alberto Heuser.

1. Busca na *Web* oculta. 2. Busca na *Web* escondida. 3. Preenchimento automático de formulários. 4. Seleção automática consultas. I. Heuser, Carlos Alberto. II. Título.

“O impossível de hoje será possível amanhã, desde que façamos o possível de hoje, hoje.”

— PAULO FREIRE

AGRADECIMENTOS

Eu gostaria de agradecer muito a todos que me deram o suporte necessário e que de alguma forma contribuíram para a realização deste trabalho:

Primeiramente agradeço à UFRGS, ao Instituto de Informática e ao CNPq pela oportunidade de realização do curso de mestrado com apoio financeiro.

Em especial agradeço ao meu orientador. Desde os primeiros contatos até os últimos momentos do mestrado sempre se mostrando prestativo e disponível. Os direcionamentos e o apoio em todos os momentos foram fundamentais. Foi grande o aprendizado como seu orientado. Obrigado Carlos Alberto Heuser pela excelente orientação.

Gostaria de agradecer também a todos os professores que moldaram a minha formação acadêmica. Em especial aos professores José Palazzo, Renata Galante, Paulo Engel, Viviane Moreira e Carlos Heuser pelos ensinamentos nas disciplinas do mestrado. Principalmente a professora Viviane Moreira pelas reuniões e orientações para a escrita dos artigos.

Os meus colegas de laboratório Maurício Moraes, Marcelo Yamashita, Guilherme dal Bianco, Isabel Volpe e Gustavo Kantorski que foram de grande importância e ajuda para minha adaptação. Gostaria de agradecer ao Maurício e Guilherme pelo apoio técnico em diversas conversas. E ao Gustavo que foi um grande parceiro nas disciplinas, nos artigos e na construção deste trabalho. Auxiliando nas implementações e em várias definições.

Agradeço a todos meus amigos e amigas pelo suporte psicológico e apoio tático durante a realização deste trabalho. Especialmente a: Vanessa Mattis, Eduardo Chielle, Guilherme Vaz, Thiago Paes, Rafael Souza, André del Mestre, Roger Sá, Gabriel Oliveira, Daniel Detoni, Henrique Niencheski, Rafael Colares, Lucas Santos, Vanessa Baugarten, José Francisco, Marcelo Sobral, Osmar Huscza, Murilo Balinhas e Elcio Perez.

Por fim agradeço aos meus familiares, principalmente meu pai Antenor e minha mãe Ana. Obrigado pelas palavras duras e carinhosas, pelo apoio incondicional e pelos conselhos. Esses são os meus grandes mestres.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

AJAX	*Asynchronous Javascript and* XML
ADBIS	*Advances in Databases and Information Systems*
C	Cobertura
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
DF	*Document Frequency*
DEQUE	*Deep WEb QUery SystEm*
DOM	*Document Object Model*
EE	Eficiência de Execução
EI	Eficiência de Indexação
ExQ	*Extracting deep-Web Query interfaces*
HiWE	*Hidden Web Exposer*
HSP	*Hidden Syntax Parser*
HTML	*Hyper Text Markup Language*
HTTP	*Hypertext Transfer Protocol*
ISIT	*Incremental Search for Informative Query Templates*
ITP	*Instance Template Pruning*
ITTP	*Instance Template and Template Pruning*
kNN	*k Nearest Neighbors*
LabelEx	*Label Extractor*
LVS	*Label Value Set*
MDR	*Mining Data Record*
MEP	*Minimum Executable Pattern*
MLP	*Multi Layer Perceptron*
RIDF	*Residual Inverse Document Frequency*
SBBD	Simpósio Brasileiro de Banco de Dados
SQL	*Structured Query Language*

SVM	*Support Vector Machine*
TF	*Term Frequency*
TFIDF	*Term Frequency Inverse Document Frequency*
TP	*Template Pruning*
URL	*Uniform Resource Locator*
UFRGS	Universidade Federal do Rio Grande do Sul
W3C	*World Wide Web Consortium*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Os motores de busca tradicionais utilizam técnicas que rastreiam as páginas na *Web* através de *links* HTML. Porém a maior parte da *Web* não é acessada por essas técnicas. A parcela da *Web* não acessada é chamada de *Web* oculta. Uma enorme quantidade de informação estruturada e de melhor qualidade que a presente na *Web* tradicional está disponível atrás das interfaces de busca, os formulários que são pontos de entrada para a *Web* oculta. Essa porção da *Web* é de difícil acesso para os motores de busca, pois o preenchimento correto dos formulários representa um grande desafio, dado que foram construídos para a manipulação humana e possuem grande variabilidade e diversidade de línguas e domínios. O grande desafio é selecionar os valores corretos para os campos do formulário, realizando um número reduzido de submissões que obtenha a cobertura da maior parte da base de dados por trás do formulário. Vários trabalhos propõem métodos para busca na *Web* oculta, porém a maior parte deles apresenta grandes limitações para a aplicação automática na *Web*. Entre as principais limitações estão a dependência de informação prévia a respeito do domínio dos formulários, o não tratamento de todos os tipos de campos que um formulário pode apresentar e a correta seleção de um subgrupo do conjunto de todas as possibilidades de preenchimento de um formulário.

No presente trabalho é apresentada uma arquitetura genérica para o preenchimento automático de formulários. A principal contribuição dessa arquitetura consiste na seleção de valores para o preenchimento de formulários através do método ITP (*Instance template pruning*). Muitos formulários apresentam um número inviável de possibilidades de preenchimento quando combinam os valores dos campos. O método ITP consegue reduzir drasticamente o número de possibilidades. A poda de diversas consultas é possível à medida que as submissões são feitas e o conhecimento a respeito do formulário é obtido. Os experimentos realizados mostraram que o método proposto é superior ao método utilizado como *baseline*. A comparação foi feita com o método que representa o estado da arte. O método proposto pode ser utilizado em conjunto com outros métodos de forma a obter uma busca efetiva na *Web* oculta. Desta forma, os experimentos a partir da combinação do ITP com o *baseline* também implicaram em bons resultados.

Palavras-chave: Busca na *Web* oculta, busca na *Web* escondida, preenchimento automático de formulários, seleção automática consultas.

Selection of Values for Form Filling

ABSTRACT

The traditional search engines crawl the Web pages through HTML links. However, the biggest part of the Web is invisible for these crawlers. The portion of the Web which is not accessed is called hidden Web. An enormous quantity of structured data and with higher quality than in the traditional Web is available behind search interfaces, the forms that are the entry points to the hidden Web. Access this part of the Web by search engines is difficult because the correct filling of forms represent a big challenge. Since these forms are built for human manipulation and have big variability and diversity of domains and languages. The challenge is to select the correct values to fill the form fields, with a few number of submissions that reach good coverage of the database behind the form. Several works proposed methods to search the hidden Web. Most of these works present big limitations for an application that surfaces the entire Web in a horizontal and automatic way. The main limitations are the dependency of prior information about the form domains, the non-treatment of the all form field types and the correct selection of a subgroup of the set of all form filling possibilities.

In the present work is presented a generic architecture for the automatic form filling. The main contribution of this architecture is the selection of values for the form submission through the ITP (Instance Template Pruning) method. Several forms have an infeasible number of form filling possibilities when combining all fields and values. The ITP method can drastically reduce the number of possibilities. The prune of many possible queries is feasible as the submissions are made and the knowledge about the form is obtained. The results of the experiments performed indicate that the ITP method is superior to the baseline utilized. The comparison is made with the method that represents the state of the art. The proposed method can be used with other methods in order to an effective search in the hidden Web. Therefore, the results by the combination of ITP and baseline methods also have implicated in good results.

Keywords: Hidden Web crawling, Deep Web crawling, Automatic filling forms, Automatic query selection.

1 INTRODUÇÃO

A Internet consiste em um enorme repositório de informações que está disponível para um grande número de pessoas. Porém, o volume de informação é tão grande que muitas vezes é difícil para uma pessoa encontrar a informação desejada. Por isso a busca de informação na *Web* tem sido tão estudada. O presente trabalho busca contribuir para facilitar o acesso a informação desejada. Neste capítulo é apresentado o que é a *Web* Oculta (Seção 1.1), como é possível a busca na *Web* Oculta (Seção 1.2), o problema abordado (Seção 1.3) pelo presente estudo e a organização do trabalho (Seção 1.4).

1.1 A *Web* Oculta

Os motores de busca tradicionais funcionam de maneira simples (OLSTON; NAJORK, 2010). Primeiramente um conjunto de URLs iniciais é fornecido. O conteúdo dessas páginas HTML é "baixado" e todos seus *links* (para outras páginas) são acessados. A partir desses *links* é feita a navegação em novas páginas que passam pelo mesmo processo e assim sucessivamente. Cada página é indexada através de seu *link* e é montado um grafo com as conexões (*links*) entre as páginas. Desta forma um usuário pode fazer uma busca por palavra-chave. A porção da *Web* acessada por esses motores de busca é chamada *Web* superficial, *Web* tradicional ou *Web* pública e indexável (LAWRENCE; GILES, 1998).

Essa técnica consegue acessar e indexar apenas parte da *Web*. Outra parte se mantém oculta aos motores de busca tradicionais. Por exemplo, a informação de bancos de dados conectados a *Web*, que normalmente estão disponíveis através do preenchimento e submissão de formulários, ou interfaces de busca, não é acessada por *links*. O conteúdo acessível apenas pela submissão de formulários representa a maior parte da informação não atingida pela *Web* superficial. Essa porção da *Web* não indexada é chamada de *Web* profunda (BERGMAN, 2001) ou *Web* oculta (FLORESCU; LEVY; MENDELZON, 1998). Essa parte da *Web* é uma das principais lacunas dos motores de busca atuais.

O preenchimento e a submissão de forma automática dos formulários têm sido objeto de vários estudos, por exemplo, Raghavan e Garcia-Molina (2000), Ntoulas, Zerfos e Cho (2005), Madhavan et al. (2008) e Jiang et al. (2009). Porém, esta não é uma tarefa fácil por duas razões. Primeiro, apenas parte dos formulários presentes na *Web* são pontos de entrada para a *Web* oculta. Segundo, o processamento dos formulários por um algoritmo é difícil, pois são interfaces projetadas para manipulação pelo ser humano e apresentam grande diversidade.

Alguns estudos (BERGMAN, 2001; CHANG et al., 2004; HE et al., 2007; MADHAVAN et al., 2007, 2008) foram realizados para identificar e mensurar as características da *Web* oculta, como tamanho e tipo da informação disponível. Os resultados desses

motivam a comunidade de pesquisa com as seguintes estimativas:

- A *Web* oculta é de 400 a 500 vezes maior do que a *Web* superficial (BERGMAN, 2001);
- O conteúdo da *Web* oculta possui em torno de três vezes mais qualidade[1] que o da *Web* superficial (BERGMAN, 2001);
- A *Web* oculta cresce mais rápido que a *Web* normal (BERGMAN, 2001; HE et al., 2007);
- A maior parte (em torno de 77%) do conteúdo da *Web* oculta é estruturado (HE et al., 2007);
- Os motores de busca cobrem em torno de 37% da *Web* oculta (HE et al., 2007);
- A *Web* oculta apresenta grande diversidade de domínios (assuntos, tópicos), mais de 700 (MADHAVAN et al., 2008);
- A *Web* oculta possui 10 milhões de formulários de alta qualidade (MADHAVAN et al., 2007, 2008).

1.2 Busca na *Web* Oculta

O conteúdo da *Web* oculta pode ser acessado de duas maneiras diferentes, dependendo da finalidade da aplicação que executa a busca (MADHAVAN et al., 2009; Tjin-Kam-Jet; Trieschnigg; Hiemstra, 2011). A primeira forma é a integração virtual que segue o paradigma da integração de dados. Esse formato oferece ao usuário um formulário de busca, que é resultado da integração dos formulários de um determinado domínio de assuntos, por exemplo, carros ou passagens aéreas. Dessa forma, é possível uma busca vertical no conteúdo da *Web* oculta, porém a solução é vinculada ao domínio. A segunda forma é conhecida como *surfacing*, pois navega em toda a *Web* de maneira horizontal. Nesse paradigma, são feitas consultas (submissões) aos formulários de maneira automática e os resultados são indexados de forma que buscas por palavra-chave possam ser executadas. O foco do presente trabalho é na busca horizontal por toda *Web* (*surfacing*). Assim, um método ideal é o que pode ser aplicado em toda *Web*. O método que será apresentado busca ser independente ao tipo e formato do formulário ponto de entrada para *Web* oculta.

A busca na *Web* oculta de forma horizontal é feita através do correto preenchimento dos formulários que são pontos de entrada na *Web* oculta. Assim, um motor de busca irá procurar por todos os formulários existentes nas páginas que ele navegar. Para cada formulário achado na *Web* o motor de busca deve identificar se deve ou não preenchê-lo, e caso o faça, deve saber com que valores. Com isto, a busca na *Web* oculta é dividida em duas **tarefas** bem distintas (JIANG et al., 2009):

1. A busca dos pontos de entrada.
2. O preenchimento correto dos formulários.

[1] A qualidade do conteúdo é definida através de algumas métricas por Bergman (2001)

A primeira **tarefa** diz respeito à correta identificação dos formulários de interesse, os que estão relacionados ao conteúdo da *Web* oculta (BARBOSA; FREIRE, 2007). Formulários que executam a identificação e acesso de usuários, com os campos usuário e senha, por exemplo, são formulários que devem ser desconsiderados, pois não acessam o conteúdo da *Web* oculta. Já as interfaces de busca, formulários que operam buscas em bases de dados, são os principais pontos de entrada na *Web* oculta. O trabalho de Moraes et al. (2012) apresenta um *survey* com os principais métodos de identificação desses formulários.

A segunda **tarefa** diz respeito ao correto preenchimento e submissão de um formulário, ou seja, são abordados dois problemas: que campos serão considerados para o preenchimento? E, com quais valores? A ideia é a obtenção de uma grande cobertura do conteúdo por trás de um formulário com o mínimo possível de submissões.

A Figura 1.1 mostra o funcionamento de um motor de busca na *Web*. O processo cíclico 1-2-3 diz respeito à busca na *Web* superficial, onde cada URL é analisada e seus *links* são guardados para serem posteriormente analisados. Já as etapas 4 e 5 dizem respeito às **tarefas** 1 e 2 respectivamente e delimitam a busca na *Web* oculta. O motor de busca deve considerar a busca pelos *links*, mas também pelos formulários. O foco do presente trabalho é na segunda tarefa (etapa 5 da Figura 1.1). Assim, considera-se que os formulários processados pelo método apresentado já foram identificados e classificados como ponto de entrada na *Web* oculta.

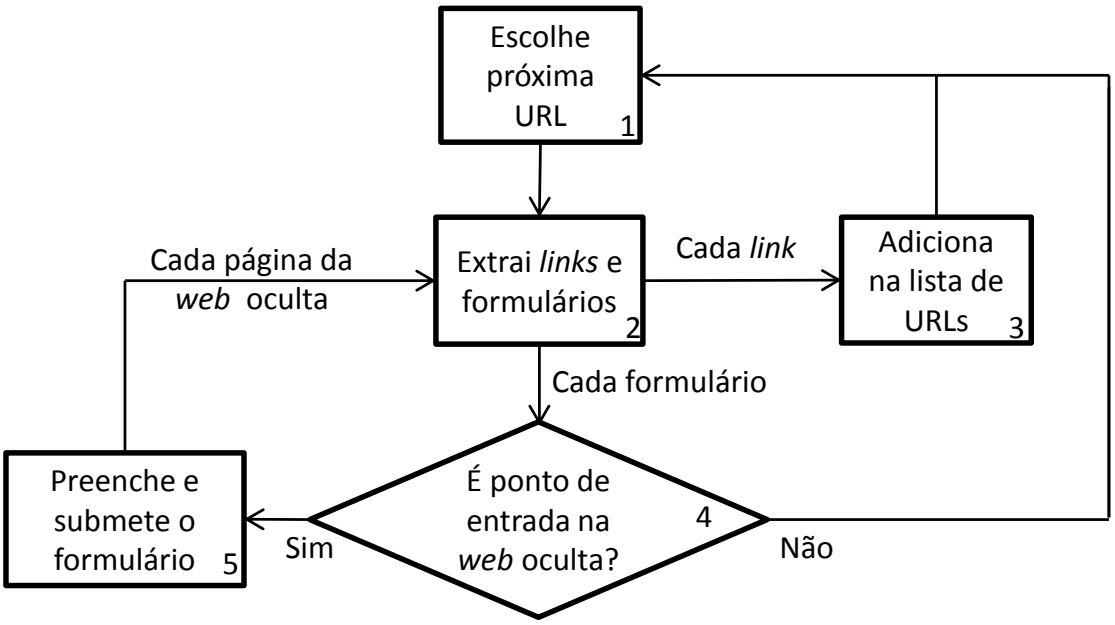

Figura 1.1: Execução básica de um motor de busca na *Web* superficial e na *Web* oculta

A maior parte dos trabalhos relacionados, que propõem métodos de busca na *Web* oculta, não executam o tratamento de todos os tipos de campos do formulário (LIDDLE et al., 2003; BARBOSA; FREIRE, 2004; NTOULAS; ZERFOS; CHO, 2005; WU et al., 2006; JIANG et al., 2009). Normalmente, quando os métodos tratam todos os campos necessitam de informação prévia (RAGHAVAN; GARCIA-MOLINA, 2000; LIU et al., 2009), de forma a se tornarem menos genéricos, pois ficam vinculados à qualidade e quantidade da informação passada previamente.

Alguns trabalhos (LIDDLE et al., 2003; BARBOSA; FREIRE, 2004; NTOULAS;

ZERFOS; CHO, 2005; WU et al., 2006) propõem que os valores mais genéricos (por exemplo: valores *default*, em branco ou o *token* 'a') e que retornem o maior número de dados possível como resposta são os melhores candidatos à submissão. Porém, os servidores normalmente limitam a resposta a um valor máximo de registros. É comum também que os resultados sejam apresentados de forma paginada, e a navegação nas páginas é uma tarefa difícil dado a variabilidade dos *sites*, com isso os métodos que executam essa navegação conseguem tratar uma parcela pequena de formulários. Dessa maneira, segundo Madhavan (2008) a execução de múltiplas consultas, ou submissões, aos formulários com a combinação de campos é a melhor saída para uma cobertura horizontal da *Web* oculta. Isso ocorre, pois além de ser uma forma mais independente a variabilidade dos *sites* e formulários também retorna partes distintas da base, já que realiza consultas mais específicas, com campos combinados.

O problema dessa estratégia de submissões de campos combinados é que existem muitas possibilidades de preenchimento. Principalmente quando o formulário possui muitos campos e quando os campos possuem muitos valores. Dessa forma, a correta seleção dos valores para o preenchimento se torna vital. O trabalho de Madhavan et al. (2008) é um dos trabalhos mais interessantes na abordagem desse problema, propondo um método que evita grande parte das possíveis submissões de um formulário que apresentariam resultados indesejados.

1.3 Definição do Problema

O preenchimento automático de formulários, que é a principal forma de busca de informações na *Web* oculta, apresenta alguns desafios. Abaixo, estão listados os principais:

1. Processamento do formulário: consiste da extração e manipulação de um formulário na *Web* oculta. A implementação dos formulários apresentam diversas heterogeneidades que dificultam a extração e correta manipulação do formulário, com seus campos, rótulos e outras informações importantes;

2. Processamento das páginas respostas a cada submissão: essas páginas, que formam a *Web* oculta, contém a informação de interesse. É importante um eficiente método de extração e manipulação dessa informação que normalmente é semi-estruturada;

3. Tratamento dos campos: escolher valores para os campos é uma tarefa simples para campos do tipo *select*, porém bastante complexa para campos do tipo *text*. É importante entender que não se conhece a base por trás do formulário, então a escolha das palavras ou conjuntos de palavras que combinam com os dados da base se torna uma tarefa difícil;

4. Número de submissões (a escalabilidade cartesiana): em muitos formulários, submeter o total de possibilidades de valores com todas suas possíveis combinações é inviável. Dessa forma, faz-se necessário a escolha de um subgrupo do total de possibilidades. Muitas vezes as bases de dados por trás dos formulários são pequenas e poucas submissões são suficientes. O ideal é que se faça o mínimo possível de submissões para mapear o máximo possível da base por trás de um formulário. Dessa forma, deve-se saber quais campos combinar e quais valores submeter.

O foco do presente trabalho é o último (quarto) problema listado. Para solucionar esse problema é proposto o método ITP (*Instance Template Pruning*). Esse faz a busca na *Web*

oculta sem necessitar de conhecimento prévio. A ideia é selecionar um grupo de valores para a submissão de cada formulário de forma a obter uma boa cobertura da base por trás desse formulário. Esse grupo é selecionado através da poda de valores considerados improdutivos (que não retornarão novos dados distintos). O método proposto aprende quais são os valores improdutivos baseado em submissões prévias ao formulário, com isso é possível reduzir drasticamente o número de consultas que são realizadas em um formulário. Os experimentos realizados mostraram que o método apresentado é, na maioria das vezes, superior ao estado da arte. A conclusão é de que as ideias propostas aumentam a eficiência e acurácia de um método de busca na *Web* oculta de forma horizontal.

1.4 Organização do Trabalho

O restante do trabalho está organizado da seguinte forma. As considerações sobre o processamento de formulários e as definições que são utilizadas durante o trabalho são apresentadas no Capítulo 2. O Capítulo 3 aborda os trabalhos relacionados à tarefa de preencher e submeter formulários, onde são apresentados e comparados os principais métodos correlatos ao proposto neste trabalho. Uma arquitetura genérica que foi implementada para a solução do problema e o método de seleção (ITP) de consultas executadas ao formulário (principal contribuição do trabalho) são descritas no Capítulo 4. Os experimentos realizados são mostrados no Capítulo 5. Por fim, o Capítulo 6 apresenta as conclusões.

2 CONCEITOS PRELIMINARES

Esse capítulo apresenta os conceitos preliminares acerca dos formulários HTML. Primeiramente são apresentadas as suas características, que são fatores determinantes para que se entenda de que forma pode-se fazer o processamento automático desses formulários. Por fim, são apresentadas algumas definições que serão utilizadas no restante do trabalho.

2.1 Processamento de Formulários

O acesso a *Web* oculta ocorre usualmente através do preenchimento de um formulário HTML. O conteúdo resultante como resposta a cada submissão de um formulário é comumente uma tabela com dados estruturados, proveniente de um banco de dados no servidor do *site*. Assim, entende-se que uma submissão de um formulário resulta em uma consulta SQL onde cada valor inserido nos campos do formulário são argumentos para a cláusula WHERE da consulta SELECT.

Um formulário é facilmente extraído do código HTML, pois é um bloco delimitado pelas *tags* <form> e </form>. Dentro desse bloco encontram-se os atributos e os campos do formulário. Os campos de interesse para o preenchimento podem ser de vários tipos: *radio*, *checkbox*, *select*, *select multiple*, *textarea*, *text*. A Figura 2.1 mostra um exemplo onde se pode ver todos os tipos de campos considerados como entrada de dados em um formulário.

Os formulários HTML são elementos feitos para interpretação e manipulação de seres humanos, na sua forma renderizada dentro de uma página *Web*. Além disso, a codificação dos formulários é apresentada de maneira bastante heterogênea. Dessa forma, um formulário possui algumas características que dificultam o seu processamento e preenchimento automático. Entre elas salientam-se: I - o método utilizado pelo formulário; II - a dependência entre seus campos; III - os campos obrigatórios; IV - os campos de apresentação; V - a utilização de tecnologias e metodologias como *JavaScript* e AJAX.

Os dados inseridos em um formulário podem ser submetidos ao servidor através de uma requisição HTTP de duas maneiras: pelo método POST e pelo método GET. Isso é delimitado pelo atributo *method* da *tag* <form>. Caso esse atributo não seja especificado o formulário é considerado do método GET. No método GET os pares de campo e valor são anexados ao atributo *action* do <form>, gerando URLs únicas para cada submissão do formulário. Já para o método POST os dados do preenchimento são anexados à mensagem que é enviada ao servidor, dessa forma, todas as páginas resposta possuem a mesma URL. Isso gera uma dificuldade tanto de implementação de um novo motor de busca, quanto de adaptação de motores de busca tradicionais, pois esses identificam cada página através de sua URL, que é um índice. As orientações para o uso do método POST e GET são

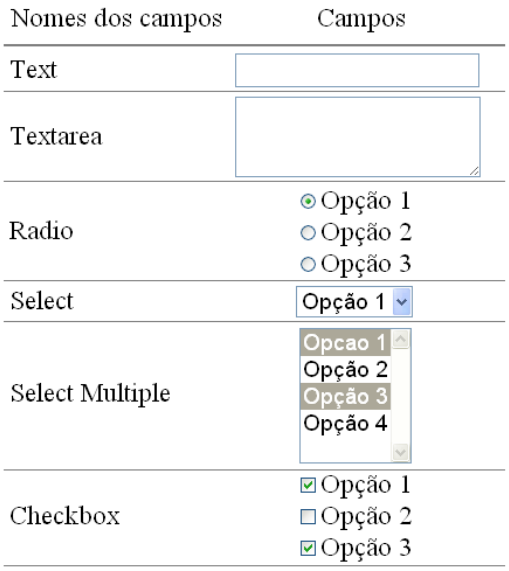

Figura 2.1: Tipos dos campos de entrada de um formulário

feitas pelo W3C (JACOBS, 2004) e, apesar de o padrão de utilização para formulários que são pontos de entrada na *Web* oculta ser o método GET (por exemplo, formulários de pesquisa), isso nem sempre ocorre na prática.

Outro aspecto importante é a **dependência entre campos** do formulário (campos correlacionados). Isso ocorre quando um campo é dependente do valor preenchido em outro campo (campos de seleção), normalmente, em subtipos delimitados pela prévia seleção de um tipo. Um exemplo de campos dependentes é um formulário que considera o preenchimento de um estado e de uma cidade. Nota-se que para cada estado selecionado um grupo de cidades esta correlacionado e cada cidade se relaciona com apenas um estado, assim, não se pode combinar uma cidade com todos os estados. Normalmente, esses formulários são dinâmicos (isso é feito, por exemplo, com a utilização de *JavaScript*), ou seja, tem seu conteúdo afetado pelo manuseio do formulário. Em alguns formulários a seleção de um campo pode, além de mudar as opções de outros campos, também tornar outros campos visíveis. Dessa forma, fica claro que o processamento de campos dependentes deve ser feito de forma diferente dos campos tradicionais e independentes.

Alguns formulários apresentam **campos obrigatórios**, ou seja, a consulta só retornará registros se o campo obrigatório tiver sido preenchido. Com isso, um campo não obrigatório submetido sozinho pode resultar em um erro mesmo que a consulta na base com esse argumento gere uma resposta com vários registros. Assim, os campos obrigatórios devem ser identificados. Isso é fundamental para a escolha e execução das consultas corretas que vão possibilitar assim as interpretações corretas das respostas do formulário.

Existem ainda formulários com **campos de apresentação**. Esses campos dizem respeito à forma que os resultados serão apresentados na página de resposta, de forma que não delimitam o conteúdo da resposta como os campos de interesse. Isto é, eles não representam um argumento a mais na cláusula WHERE da consulta feita ao banco de dados do servidor. Um exemplo desse tipo de campo é um campo de seleção que delimita o número de resultados por página. Esses campos devem simplesmente ser ignorados, ou seja, o valor *default* deve ser submetido sem qualquer tratamento. A dificuldade aqui é a identificação desses campos em um formulário.

Por fim, outro aspecto importante a ser salientado são os *scripts JavaScript*, muito

utilizados na construção de *sites* através da metodologia AJAX. Esses *scripts* executam processamento no lado do cliente, modificando o estado das páginas, bem como dos formulários. Isso é realizado pela manipulação dos atributos (do formulário) que controlam os eventos, por exemplo: *onselect*, *onclick*, *onsubmit*, etc. Nesses atributos os scripts são embutidos. Essa característica implica em um problema de processamento do formulário, dado que se deve simular a execução do *JavaScript* em todos os possíveis eventos para que se descubra os possíveis estados finais do formulário.

2.2 Definições

Existem várias formas de entrada de dados para um formulário, conforme visto na Figura 2.1. No entanto, considerando os valores que podem ser atribuídos a esses campos, esses diversos tipos podem ser divididos em dois grupos, como pode ser visto na Figura 2.2 : **campos de seleção** e **campos de texto**. A diferença é que as opções de preenchimento dos campos de seleção são limitadas e estão presentes no código HTML. Assim, o conjunto de possíveis valores associados a cada campo de seleção pode facilmente ser extraído do código HTML. Os campos do tipo *select*, *radio*, *checkbox* e *select multiple* são considerados como campos de seleção, sendo que os tipos *select* e *radio* aceitam apenas um valor por submissão. Já *checkbox* e *select multiple* aceitam a submissão simultânea de todos possíveis valores. Apesar dessa diferença, todos os campos de seleção podem ser tratados da mesma forma. Já os campos de texto compreendem os campos do tipo *textarea* e *text*. Esses campos não possuem um número delimitado de possibilidades de preenchimento podendo ser preenchidos por qualquer *token* ou conjunto de *tokens*, ou seja, existem infinitas opções de preenchimento. O fato do código HTML não conter um conjunto de possibilidades de preenchimento, como nos campos de seleção, torna o correto preenchimento dos campos de texto uma tarefa mais difícil comparada à dos campos de seleção.

A Figura 2.2 mostra também que os campos de texto podem ser divididos em outros dois tipos (MADHAVAN et al., 2008): campos de texto genéricos e campos de texto tipados. Os campos tipados são campos que pressupõem o preenchimento de valores de uma forma ou tipo específico. Por exemplo, um campo com os rótulos CEP ou CPF deve ser preenchido por um determinado tipo de *token* que obedeça as regras para número de CEP ou CPF. Já os campos de texto genéricos não pressupõem o preenchimento por um tipo específico de *token*, o caso mais clássico são os campos de palavras-chaves, que podem ser preenchidos com qualquer *token*.

No contexto deste trabalho é importante distinguir a definição de **domínio do formulário** e **domínio do campo**. O domínio do formulário diz respeito ao assunto ou tópico em que esse formulário está inserido, ou ao tópico do *site* onde se encontra o formulário. Por exemplo, *sites* de empregos, hotéis, venda de passagens aéreas, livros, músicas etc. Formulários de um mesmo domínio possuem características semelhantes que podem ser aproveitadas e usadas para a busca do conteúdo da *Web* oculta. Chang et al. (2004) apresentam um estudo sobre os domínios encontrados na *Web* oculta. Já o domínio do campo diz respeito ao conjunto de valores que está associado a determinado campo. Por exemplo, um campo *select* tem os seus valores dentro dos seus *options*. Esses valores são o seu domínio. Já um campo de texto, com o rótulo “Nome do autor”, por exemplo, possui um domínio associado a nomes de pessoas, pois possíveis valores que retornarão dados são nomes de pessoas. O domínio do campo é importante para se identificar os possíveis valores que serão relevantes para cada campo.

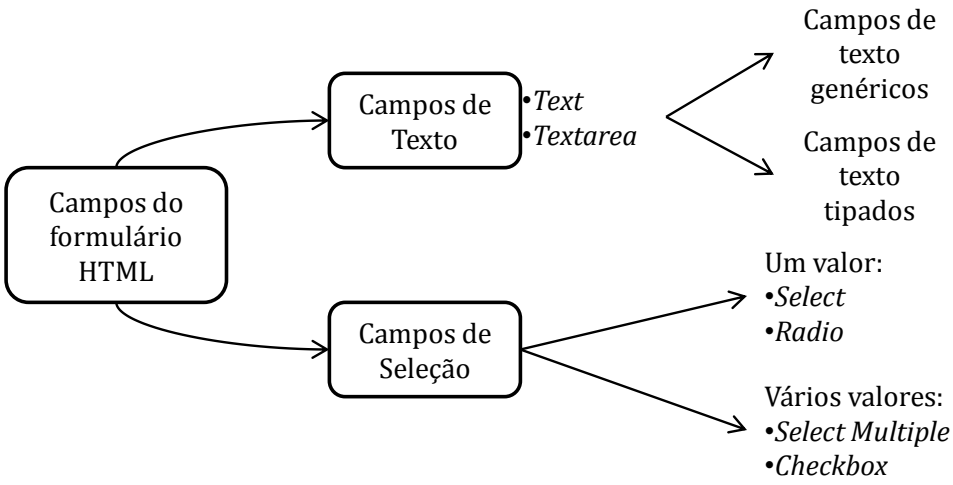

Figura 2.2: Classificação dos campos de entrada de dados nos formulários

Um formulário pode possuir campos que devem ser desconsiderados por um método de busca na *Web* oculta (campos de apresentação). Dessa forma, quando os campos de um formulário são referenciados será considerado os n campos que determinam a informação obtida pela página resposta a cada submissão e por isso necessitam ser preenchidos. Assim, n será um número menor ou igual ao total de campos do formulário. Para trabalhar com a submissão e preenchimento desses n campos dos formulários são definidos dois conceitos importantes para o restante do trabalho: ***template***[1] e **instância de *template***. O primeiro diz respeito aos campos que são considerados em uma submissão (o que preencher). Semelhante à ideia apresentada inicialmente por Rajaraman, Sagiv e Ullman (1995). Já a segunda definição diz respeito aos valores inseridos nos campos do formulário (como preencher). Assim, define-se de maneira formal:

Template: Considerando C=$\{C_1, C_2, C_3, ..., C_n\}$ como conjunto dos n campos considerados para submissão de um formulário *Web* e T=$\{t_1, t_2, t_3, ..., t_m\}$ o conjunto de todas as possíveis combinações dos elementos de C. Cada elemento de T é um *template*. Onde o tamanho do conjunto T ou número de *templates* (m) gerados é função do número de campos (n) dado por:

$$m = 2^n - 1 \tag{2.1}$$

Instância de *template*: Considerando V=$\{v_1,v_2,v_3,...,v_o\}$ como o conjunto de todas possibilidades de valores para o *template* $t \in$ T. Uma instância de *template* é um par *template*-valor (*t*,*v*) onde $v \in$ V.

A Figura 2.3 mostra um exemplo de um formulário renderizado (com os campos de seleção *A*, *B* e *C*) para ilustrar esses conceitos. Um formulário produz um total de *templates* que é função dos n campos que ele possui. Dessa forma, os templates gerados pelos 3 campos desse formulário são 7: *A*, *B*, *C*, *A&B*, *A&C*, *B&C* e *A&B&C*. Alguns exemplos de instâncias de *template* para os *templates* *A*(instância *A=ValorA*) e *A&B*(instância *A=ValorA&B=Valor1*). Por fim a Figura 2.3 também mostra a implementação de instância de *template*, a URL `http://www.exemplo.com?A=ValorA&B=Valor1` referente à instância *A=ValorA&B=Valor1*.

[1]Nomenclatura utilizada em inglês conforme os trabalhos relacionados para evitar uma tradução que atrapalhe o sentido.

Outro aspecto importante a respeito do conceito de template é a sua dimensão. Define-se **dimensão** ou **ordem** de um *template* como sendo o número de campos considerados pelo *template*. Assim, um formulário produz *templates* de dimensão ou ordem 1 a n. Por exemplo, os templates $A\&B$ e $A\&B\&C$, possuem ordem ou dimensão 2 e 3, respectivamente.

Por fim, a última definição apresentada por este capítulo é a **complexidade do formulário**. A complexidade diz respeito ao número de instâncias de *template* que um formulário pode gerar. Formulários que geram poucas instâncias de *template* são assumidos como menos complexos dos que geram mais instâncias de *template*. Esse número de instâncias é função de duas variáveis, o número de campos de um formulário e o número de valores associados a cada campo. O número de campos define o número de *templates*, assim afeta o número de submissões. Quanto mais campos um formulário possui, maior a sua complexidade. Já a quantidade de valores influencia no número de instâncias de *template* associadas a cada *template*. Assim como acontece com os campos, mais valores associados a cada campo significam uma complexidade maior. O desafio da seleção de valores para submissão em um formulário é proporcional à sua complexidade. Isso ocorre, pois formulários mais complexos tendem a ter a sua base de dados mapeada por um percentual pequeno do total de instâncias de *template* possíveis.

Formulário renderizado	Campo A ValorA → A Campo B Valor1 → B Campo C ValorX → C
Templates	*A, B, C, A&B, A&C, B&C, A&B&C*
Instâncias de *Templates*	*A=ValorA, A=ValorA&B=Valor1*
URL: Implementação da instância ***A=ValorA&B=Valor1***	http://www.exemplo.com?**A=ValorA&B=Valor1**

Figura 2.3: Exemplo de *templates* e instâncias de *template* para um formulário

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos relacionados ao preenchimento de formulários *Web*. Os maiores esforços de pesquisa na área ocorreram na última década. Principalmente após os estudos realizados no trabalho de Bergman (2001), que revelou a existência de várias bases de dados semiestruturadas na *Web* oculta, o que motivou o interesse da comunidade de pesquisadores de banco de dados.

Muitos estudos abordam técnicas que não realizam o preenchimento de formulários *Web*, porém são relevantes para a tarefa de escolher valores para o preenchimento de formulários. Alguns trabalhos, por exemplo, propõem a extração de informação. A informação a ser extraída pode ser de conteúdo não estruturado, como um texto, ou de conteúdo semiestruturado, como em uma página resposta à submissão de um formulário. Assim, os métodos que propõem extração automática de informação semiestruturada são os mais relevantes no contexto da *Web* oculta. Chang et al. (2006) apresenta um *survey* com diferentes métodos de extração de informação. O método MDR (LIU; GROSSMAN; ZHAI, 2003), por exemplo, busca na estrutura em árvore do HTML os registros de dados individuais, ou regiões do código HTML onde se concentram os dados. É necessário apenas uma página para que o método descubra a região de dados, por isso o método pode ser executado apenas uma vez por formulário, já que todas as páginas resposta de um mesmo formulário possuem as mesmas *tags* que delimitam a região de dados no HTML. Outros métodos de extração mais novos (ZHAI; LIU, 2005; LI; LIU; OBREGON, 2007; HONG; SIEW; EGERTON, 2010), normalmente, apenas apresentam extensões à ideia do MDR, como a consideração também de informações visuais.

Outros trabalhos focam na extração de todas as informações do formulário, realizando a tarefa de "entendimento de formulários", ou "entendimento de interfaces de buscas". Essa tarefa foca, principalmente, na vinculação de campos a seus rótulos, podendo também identificar campos correlacionados e obrigatórios. Um recente *survey* (KHARE; AN; SONG, 2010) faz um balanço das técnicas mais recentes. DEQUE (SHESTAKOV; BHOWMICK; LIM, 2005), HSP (ZHANG; HE; CHANG, 2004), LabelEx (NGUYEN; NGUYEN; FREIRE, 2008) e ExQ (WU et al., 2009) são alguns exemplos. Alguns métodos, como o HSP, utilizam uma gramática e um *parser* que permite que os usuários especifiquem regras de extração. Outros métodos utilizam técnicas de aprendizado de máquina supervisionados. Por exemplo, o LabelEx utiliza classificadores bayesianos e árvores de decisão. Há ainda alguns estudos que consideram elementos visuais como distância e posição relativa de elementos do formulário, como o ExQ. O correto entendimento de um formulário é útil para a identificação do domínio e tipo dos campos, o que pode melhorar a maioria dos métodos de preenchimento de formulários.

Os trabalhos de entendimento de interfaces de busca normalmente não apresentam soluções para o preenchimento do formulário. Porém, em alguns estudos (ÁLVAREZ

et al., 2007; RAGHAVAN; GARCIA-MOLINA, 2000) são propostos motores de busca que primeiro passam por um processo que correlaciona os rótulos com os campos e após usam a informação para efetuar o preenchimento. Em Alvarez et al. (2007), por exemplo, são utilizadas informações visuais para o entendimento do formulário. Cada rótulo *r* identificado e relacionado a um campo é vinculado a um atributo (por uma função de similaridade) da base de dados com as *definições de domínio*, que também possui os valores candidatos ao preenchimento do campo correspondente ao rótulo *r*. Porém, o método não apresenta detalhes sobre a obtenção desses valores das *definições de domínio*.

Existem ainda alguns trabalhos (TODA et al., 2010; KRISTJANSSON et al., 2004; ALI; MEEK, 2009) que propõem métodos para o preenchimento automático de formulários em um contexto diferente. Enquanto o presente trabalho pensa a seleção de valores dos formulários para a busca na *Web* oculta, esses métodos pretendem, por exemplo, poupar o esforço do usuário para preencher um formulário de registro de ofertas em sites de leilão. Essas soluções normalmente utilizam aprendizado de máquina para sugerir ao usuário valores a serem preenchidos nos campos de um formulário.

Os principais trabalhos que propõem soluções especificamente para preenchimento automático dos campos de um formulário, de forma a obter o conteúdo da *Web* oculta, são explorados na Seção 3.1. Já a Seção 3.2 apresenta uma comparação dos métodos.

3.1 Métodos de Preenchimento de Formulários

Existem vários estudos que trazem soluções para o problema do preenchimento de formulários *Web*. Nessa seção são mostrados os principais trabalhos (RAGHAVAN; GARCIA-MOLINA, 2000; LIDDLE et al., 2003; BARBOSA; FREIRE, 2004; NTOULAS; ZERFOS; CHO, 2005; WU et al., 2006; MADHAVAN et al., 2008; LIU et al., 2009; JIANG et al., 2009). Para uma melhor visualização e posterior comparação, os métodos de cada trabalho são separados em subseções com o nome do primeiro autor de cada trabalho (Subseções 3.1.1 a 3.1.8).

3.1.1 Método de Raghavan

Em seu estudo (RAGHAVAN; GARCIA-MOLINA, 2000, 2001), Raghavan e Garcia-Molina propuseram o método HiWE. Esse é um trabalho chave, pois representou o início de um novo momento na pesquisa sobre a *Web* oculta. O HiWE consiste de um motor de busca na *Web* oculta que vai além de escolher valores para o preenchimento de um formulário, ele realiza todo o processo de busca, ou seja, identifica os formulários que são pontos de entrada na *Web* oculta e os preenche. O HiWE é inicializado com alguns *sites* fazendo a varredura dos formulários desses *sites*. Após, os formulários identificados são preenchidos com os valores escolhidos. Por fim, o método extrai informações das páginas resposta de cada formulário.

No que tange a seleção dos valores para o preenchimento de cada formulário, o método se baseia em uma tabela chamada LVS, onde cada linha possui um rótulo e os valores associados a esse rótulo, ou seja, candidatos ao preenchimento de campos com esse rótulo. Para os campos de seleção, os seus próprios valores é que serão escolhidos na tabela. Já para os campos de texto, os valores são escolhidos pela análise mais profunda da tabela LVS. A tabela é alimentada diretamente pelo usuário ou pelo processamento de bases públicas indicadas pelo usuário antes do processo. Durante o processo, cada campo de seleção tem o seu rótulo e os seus valores inseridos na tabela, ou na linha do rótulo mais similar achado na tabela, ou em uma nova linha, caso nenhum rótulo tenha similaridade

suficiente com o rótulo do campo. A esses valores da tabela, são associados pesos de 0 a 1 que representam a ordem em que serão utilizados para o preenchimento de um campo referente àquele rótulo. Os valores iniciais tem peso alto, principalmente os informados pelo usuário e obtidos a partir dos campos de seleção. O peso de cada valor candidato vai diminuindo na medida em que não geram páginas resposta com dados para alguma submissão anterior, por isso é importante a extração de informação de cada página resposta. São esses valores da tabela que são considerados para a seleção de valores para a submissão dos campos de texto dos formulários.

Os autores consideram a cobertura da base como uma ótima métrica, mas justificam o seu não uso pela dificuldade de obtenção. A métrica utilizada é a eficiência de submissão, ou seja, a razão entre o total de consultas submetidas que retornaram dados e o total de consultas submetidas. As comparações são feitas com variações nos parâmetros do método. Primeiramente, são apresentados os resultados para a variação do parâmetro α que delimita a utilização de apenas formulários com no mínimo α campos (α é variado de 2 a 5). Em um segundo momento, é apresentado a comparação dos resultados para a utilização ou não da atualização automática da tabela LVS durante o processo. Por fim, é apresentada a comparação dos resultados para as três funções de ordenação dos valores da tabela LVS apresentadas no trabalho.

O método depende de uma forte intervenção inicial do usuário, para a seleção de alguns parâmetros e para a inicialização da tabela LVS. Assim, o desempenho do método se mostra sensível ao usuário e ao domínio do formulário, pois diferentes domínios representam diferentes rótulos e valores associados aos campos. Os valores presentes na tabela LVS podem apresentar um bom desempenho para um formulário de um domínio e um fraco desempenho para outro formulário semelhante de outro domínio.

3.1.2 Método de Liddle

Liddle et al. (2003) propõem um método de preenchimento e submissão de formulário pontos de entrada na *Web* oculta que não depende de uma base de informação, o que é um avanço em relação ao trabalho anterior (RAGHAVAN; GARCIA-MOLINA, 2000). O método é dividido em três etapas:

1. Primeiramente o método faz uma submissão do formulário, com os valores *default* dos campos, extraídos do HTML, ou seja, os valores padrão e genéricos para os campos de seleção e a *string* vazia para os campos de texto. Os dados da página resposta são extraídos por um método ingênuo (na época do trabalho os métodos mais recentes de extração de informação semiestruturada não existiam). É feita a navegação pelas outras páginas do resultado pela busca das palavras "*next*" ou "*more*" dentro do código HTML;

2. Após, é feito um conjunto de x submissões que determinarão se o percentual de cobertura desejado já foi atingido. Para essa determinação são feitas algumas estimativas estatísticas para estimar o tamanho da base por trás do formulário. Caso a cobertura tenha atingido o percentual desejado o método para;

3. Caso o percentual não seja obtido o algoritmo começa um processo exaustivo de submissões dividas em conjuntos com o mesmo tamanho x. Ao fim de cada conjunto de submissões o algoritmo novamente decide se para ou não. Nessa fase o algoritmo pode parar ao atingir algum dos limiares do método, determinados pelo usuário.

A intervenção do usuário é necessária caso o preenchimento de um campo texto seja obrigatório (pois o método não executa esse tipo de preenchimento). Cabe ao usuário decidir também alguns limiares, que funcionam como critérios de parada. São eles: mínimo percentual coberto da base, número máximo de submissões realizadas, número máximo de *bytes* retornados, tempo de execução máximo e número máximo de submissões consecutivas que não retornam novos dados.

O método é testado em 13 formulários, porém os resultados de cobertura em *bytes* são mostrados para apenas 2 formulários. É citado que em 5 dos 13 formulários a execução é finalizada na fase 2, ou seja, o método busca uma cobertura suficiente apenas com a consulta *default*.

Uma importante restrição do trabalho é a necessidade de intervenção do usuário, o que afeta diretamente a sua performance. O método também não oferece um tratamento para os campos de texto. Outra limitação é que na fase automática, em caso de formulários com respostas paginadas, o método tem seu desempenho em função da obtenção ou não dos *links* para as próximas páginas, pela procura de duas palavras da língua inglesa. Caso a página resposta apresente um *link* para as próximas páginas sem a utilização dessas duas palavras, o que é bastante comum, o desempenho do método cai bastante.

3.1.3 Método de Barbosa

Barbosa e Freire (2004) apresentam um método de busca na *Web* oculta focado em solucionar o problema de submissão de campos de texto, mais especificamente campos de texto de palavras-chave ou campos de texto genéricos. Dessa forma é apresentada uma solução apenas para este tipo de campo, sendo desconsiderados os campos de seleção, por exemplo.

Primeiramente o método extrai o conteúdo da página onde o formulário se localiza. Todas as palavras (*tokens*) são extraídas sendo filtradas as palavras indesejadas como as de propagandas, por exemplo. Após, é montada a primeira listagem de palavras candidatas ao preenchimento. Essa listagem é ordenada pelo número de ocorrências de cada termo, dessa forma os termos mais frequentes são utilizados primeiro. Para cada submissão feita, a informação da página resposta é extraída e a listagem dos *tokens* é atualizada. A ideia é de que as consultas com termos mais frequentes tendem a retornar mais elementos da base.

O método identifica quando um formulário aceita a submissão de *stopwords* e adapta a geração das palavras para o preenchimento, com a utilização ou não de *stopwords*. Como o método considera que cada consulta retorna o total de dados encontrados na mesma página, ele apresenta problemas para formulários que retornam um número fixo e reduzido de dados por página resposta, por exemplo, os formulários com resultados paginados. Sendo que em alguns *sites* os resultados além de paginados são também limitados a um número máximo para visualização por consulta.

A obtenção de uma lista de termos que são ordenados pela frequência é uma variação da proposta de Callan e Connell (2001) e consiste da primeira proposta, de seleção de valores para preenchimento de campos de texto, independente do usuário e da informação de domínio do formulário. Essa proposta influenciou a maioria dos trabalhos sucessores no que tange o tratamento de campos de texto genéricos.

Nos experimentos são apresentados resultados para cobertura em oito formulários diferentes. Não é utilizado nenhum outro trabalho como base de comparação, são utilizadas apenas algumas variantes do próprio método, como: seleção das palavras considerando ou não as *stopwords*, e o local da página onde as palavras são obtidas.

A principal limitação do método é considerar apenas o preenchimento de campos texto genéricos, o que representa apenas uma porção dos formulários disponíveis como ponto de entrada na *Web* oculta.

3.1.4 Método de Ntoulas

Ntoulas, Zerfos e Cho (2004; 2005) propõem um *framework* para o estudo do problema de busca na *Web* oculta. Nesse *framework* é apresentado uma modelagem para as bases de dados da *Web* oculta, a formalização do problema e um algoritmo genérico para busca na *Web* oculta. Posteriormente, o *framework* é utilizado para a apresentação de um método de busca.

Esse método se assemelha ao de Barbosa, pois foca no problema de submissão de campos texto e utiliza uma técnica estatística baseada na frequência dos termos das páginas resposta a submissões anteriores. Porém, o método apresenta duas diferenças claras.

Primeiramente, apesar de ser utilizada uma tabela com a frequência dos termos nas páginas retornadas, os *tokens* não são escolhidos diretamente pela sua frequência. A frequência é utilizada para se estimar o percentual de cobertura da base por trás do formulário para a submissão de um termo q, dado por $P(q_i)$. Com isso, o número de novos dados buscados na base após a submissão de um termo q pode ser estimado ($P_{new}(q_i)$). O custo ($Cost(q_i)$) para a submissão de cada termo também é estimado. E assim, é feito o cálculo da eficiência estimada de cada termo (dado pela razão entre $P_{new}(q_i)$ e $Cost(q_i)$). E é essa eficiência que é considerada para a escolha de um *token*, sendo sempre escolhido para a próxima submissão o termo com a melhor estimativa de eficiência que ainda não foi submetido.

Outra diferença importante consiste na inicialização do processo, pois na primeira submissão ainda não existem dados para o cálculo da eficiência estimada. Assim, o processo é iniciado pelo usuário que manualmente informa o primeiro valor para preenchimento do campo texto.

São apresentadas também outras duas técnicas mais ingênuas para escolha de valores para campos de texto. A primeira escolhe valores para submissão randomicamente dado um conjunto de termos e a segunda gera uma lista de palavras ordenadas baseada na frequência dos termos de uma base de dados estática.

Os experimentos são realizados em quatro formulários de três *sites* de domínios diferentes. Entre outros experimentos menos importantes, é apresentada a cobertura, para comparar as três técnicas apresentadas e para medir o impacto da escolha do termo inicial (os experimentos apresentados sugerem que o método converge mesmo para diferentes termos escolhidos). Nenhuma trabalho é utilizado como base de comparação nos experimentos.

O método apresenta o mesmo problema do método de Barbosa ao considerar apenas campos de texto genéricos para o preenchimento. Além disso, o método é dependente do usuário que escolhe o termo inicial. Apesar de o autor argumentar que o método converge independente do termo inicial escolhido, ainda assim ele não é completamente automático.

3.1.5 Método de Wu

Wu et al. (2006) apresentam um método baseado na modelagem da base por trás de cada formulário como um grafo. Assim, a questão da busca pelo menor conjunto de valores para submissão de um formulário que retornem uma cobertura significativa dos dados é tratado como o problema de encontrar uma cobertura de vértices mínima do

grafo. Diferentemente dos trabalhos anteriores (BARBOSA; FREIRE, 2004; NTOULAS; ZERFOS; CHO, 2005), que fazem busca em bases textuais (formulários baseados em palavras-chave), o método faz a busca em bases estruturadas (com diferentes atributos), que possuem a interface de busca baseada em formulários estruturados (com vários campos). Porém, é considerada a submissão de apenas um campo por vez.

A base de dados por trás de um formulário é modelada como uma única tabela com m atributos. O grafo é montado a partir dessa tabela da seguinte forma: cada valor v de uma coluna da tabela é considerado como um nodo do grafo. A conexão desse nodo com outros nodos (valores) é feita pela análise dos valores presentes junto com v nas linhas da tabela em que v ocorre. Assim, como a base por trás do formulário, o grafo é descoberto em tempo de execução. A análise da submissão de v permite a identificação de todos os seus vizinhos, dessa forma novos nodos e novas conexões do grafo são descobertos.

O método é inicializado por valores randômicos e a medida que novas submissões são feitas, uma lista de valores a serem submetidos (inicialmente vazia) vai sendo preenchida com os valores descobertos. Essa lista é ordenada por um coeficiente para que se decida a ordem com que os valores serão escolhidos. Este coeficiente é determinado de duas formas:

1. Inicialmente, os valores candidatos à submissão são ranqueados proporcionalmente ao grau do seu nodo no grafo, ou seja, um valor é correlacionado com a frequência dos termos nas páginas resposta às submissões, pois valores que ocorreram mais com outros já submetidos são os que representam nodos de grau maior;

2. Após a obtenção de certa cobertura da base, em torno de 85% (segundo o autor), um **ponto de saturação** é obtido e a tática para escolha de valores muda. Assim, a ordenação dos valores passa a ser feita de maneira inversamente proporcional ao grau do seu nodo no grafo (correlação dos valores candidatos com os valores obtidos), ou seja, serão priorizados os valores mais raros ou que retornaram menos vezes nas páginas resposta. O problema é como identificar o **ponto de saturação**. O trabalho apenas relata que utiliza "algumas heurísticas" e que uma forma automática de detecção deve ser estudada em um trabalho futuro.

Paralelamente a essa lista o autor considera outra tática que pode ser utilizada para melhorar o método: a utilização das informações de domínio do formulário. É apresentada uma tabela com a probabilidade de valores ocorrerem em um determinado domínio. Essa tabela é utilizada para obtenção de outra lista de valores. As duas listas são utilizadas paralelamente para a seleção de valores. Segundo o autor, essa tática melhora o desempenho do método, porém essa lista dependente da informação de domínio do formulário pode não ser utilizada pelo método. Assim, o método não é considerado dependente da informação de domínio.

Os experimentos são conduzidos com variantes das técnicas apresentadas sem a comparação com outro método de um trabalho relacionado. É apresentada a cobertura pelo número de comunicações com o servidor para cinco formulários. A principal lacuna deste método é o fato de considerar o preenchimento de apenas um campo por vez. O método também não apresenta uma solução para a identificação do **ponto de saturação**. Outro problema, é quando a base possui a chamada "ilha de dados", que ocorre quando o grafo não é totalmente conectado, ou seja, possui valores (nodos) inacessíveis, ilhados.

3.1.6 Método de Madhavan

O trabalho proposto por Madhavan et al. (2008) descreve o motor de busca na *Web* oculta do *Google*. O trabalho propõe uma solução para a submissão de um ou mais campos de um formulário combinados que evite o problema do produto cartesiano. Também é proposta uma solução para o preenchimento de campos texto genérico e de campos texto tipados.

A estratégia de seleção de campos para submissão é baseada no conceito de *template*. Um *template* diz respeito aos campos que são considerados para o preenchimento. A estratégia consiste na submissão de todos os valores de cada *template* de ordem 1. Após, eles são avaliados no teste de informação de um *template* que delimitará a criação dos *templates* de ordem superior. A ideia é de que se um campo *c* não retorna informação relevante quando submetido sozinho, também não retornará quando combinado com outros campos (devido ao fato de que um campo a mais apenas aumenta a especificidade da consulta SQL feita na base por trás do formulário). Dessa forma, a geração dos *templates* de uma ordem é influenciada pela avaliação em **informativo** ou **não informativo** dos *templates* de ordens inferiores. Por exemplo, se o *template* de ordem 1 com o campo *c* é dito não informativo, todos outros *templates* de ordem superior que consideram o valor *c* não são gerados e são descartados. Dessa forma, primeiro todos os *templates* de ordem um são gerados, após os de ordem dois, e assim sucessivamente.

O teste de informação (algoritmo ISIT) consiste na avaliação da distinção do conteúdo das páginas resposta às submissões de um mesmo *template*, ou seja, submissões com conteúdo repetido, que não retornam conteúdo algum, ou que retornam erro são consideradas indistintas e influenciam negativamente na informação de um *template*. Um *template* é avaliado pela razão entre o número de páginas com conteúdo distinto e o total de submissões que realizou. Essa razão, que é chamada de **fração de distinção**, é comparada a um limiar para determinar se um *template* é **informativo** ou **não informativo**. Para que a validação dos *templates* seja feita, é necessário primeiro a obtenção de todos os valores candidatos de cada campo. Para os campos de seleção, os valores estão presentes no HTML e para os campos de texto, são obtidos da seguinte forma:

- A escolha de valores candidatos para os campos de texto genérico é feita com base na página onde o formulário se encontra e nas páginas resposta às submissões obtidas, caso já se tenha feito alguma submissão. São consideradas as estatísticas de frequência, de forma similar a métodos anteriores (NTOULAS; ZERFOS; CHO, 2004; BARBOSA; FREIRE, 2004). Assim, é feito o ordenamento dos valores candidatos, inspirado na métrica TF-IDF (SALTON; MCGILL, 1983);

- A escolha dos valores candidatos para campos de texto tipados é feita através da informação de domínio de alguns campos. Isso ocorre, pois determinados campos aparecem muitas vezes em diversos formulários de diversos domínios. O método apresenta informação prévia para os seguintes padrões de campos: cidade (nomes em inglês), data (no formato mm/dd/aa), código postal americano (*zip code*) e preço (preço baixo, de 0 a 5000, e preço alto, de 50000 a 500000).

Um passo importante é a identificação de um campo de texto, se genérico ou tipado e o tipo (caso seja um campo de texto tipado). Dessa forma, quando o método necessita preencher um campo texto, ele procede da seguinte forma: é executado algumas submissões com os diversos valores dos tipos pré-definidos (valores dos quatro domínios de campos apresentados), após se utiliza o algoritmo ISIT para identificar o tipo (ou domínio) do

campo. Caso nenhum tipo tenha sido identificado ele é tratado como um campo de texto genérico.

O trabalho apresenta diversos experimentos para avaliar as afirmações e heurísticas apresentadas, além de avaliar o método em si. Dessa forma, ao avaliar o algoritmo ISIT é apresentada a cobertura e o número de URLs submetidas para 12 formulários com apenas campos de seleção. Já para a avaliação da técnica de submissão dos campos texto genéricos, é apresentada a cobertura e o número de termos gerados para 10 formulários. Nenhuma base de comparação é apresentada nos experimentos.

O método de Madhavan, que resultou em duas patentes (MADHAVAN et al., 2013; MADHAVAN; HALEVY; KO, 2013), representa a melhor solução estudada neste capítulo, e assim, o atual estado da arte. Mesmo assim, o trabalho apresenta algumas questões em aberto. É sugerido um limiar *k*, número máximo de submissões por *template*. Em muitos *templates* esse subconjunto representa menos de 10% do total de possibilidades para um *template*, porém o trabalho não explica como é feita a seleção destes valores. O método não apresenta uma solução para a extração dos dados de cada página resposta, fator crucial para execução de várias partes da técnica, assim como em outros métodos que fazem busca na *Web* oculta. Apesar de os autores sugerirem sempre uma solução automática e escalável, de forma a se diferenciar dos outros trabalhos, isso não ocorre no preenchimento de campos de texto tipado, onde são utilizadas informações prévias de domínio.

3.1.7 Método de Liu

O trabalho de Liu et al. (2009) aborda um método baseado em dois trabalhos anteriores (NTOULAS; ZERFOS; CHO, 2004; RAGHAVAN; GARCIA-MOLINA, 2000) para obter o conteúdo da *Web* oculta. O trabalho também apresenta o MEP, um conceito semelhante ao conceito de *template*, ou seja, diz respeito aos campos que serão considerados em uma submissão, assim como uma consulta de um MEP é o mesmo que uma instância de *template*. Com a utilização do MEP, o trabalho consegue propor uma solução que considera o preenchimento de campos de texto e de seleção, assim como a combinação de campos. O método funciona em duas etapas:

- No início, *fase de acumulação de dados*, o método se baseia em uma tabela, semelhante a tabela LVS (RAGHAVAN; GARCIA-MOLINA, 2000) para fazer suas primeiras submissões e assim obter um determinado conhecimento inicial, mapeando parte da base. A ideia aqui é a obtenção de parte significativa da base por trás do formulário, sem a utilização de apenas valores da própria base, de forma a ser evitado o fenômeno das "ilhas de dados". Essa fase se mantém em execução até a consulta (submissão) s, um limiar fixo do método;

- Em um segundo momento, tem início a *fase de predição*, que é uma extensão do método de Ntoulas. Com parte da base extraída, é possível ranquear os *templates* (conjunto com os MEP's gerados), assim como os valores a partir das frequências estatísticas das páginas resposta. Com isso, é estimado o *template* que retornará mais dados e é submetida a instância de *template* que tem a maior probabilidade de retornar novos dados distintos. O método para de buscar em um determinado formulário quando a razão entre os novos dados distintos obtidos e as últimas x submissões atinge um valor menor ou igual a determinado limiar. Para que seja detectado o "amortecimento na busca", é feito o cálculo com os dados das últimas

x submissões, onde x é o tamanho de uma janela deslizante com parte do total de submissões.

Os experimentos realizados apresentam a cobertura obtida pelo número de consultas realizadas. O método é testado em seis formulários diferentes, sendo apresentadas comparações com um método base em três casos. Em dois formulários o resultado é comparado ao método de Ntoulas e no terceiro com o método de Raghavan. O método proposto, baseado no MEP, foi um pouco superior nos três casos apresentados.

O MEP apresenta uma solução que considera todos os campos do formulário de forma a evitar as "ilhas de dados", fenômeno observado quando parte da base não é atingida mesmo após um número suficientemente grande de consultas (estagnação do método), ficando "ilhada". Porém o método apresenta uma solução que necessita de informação anterior (tabela LVS), o que deixa a solução vinculada ao domínio do formulário e a sua eficiência vinculada a qualidade dessa informação.

3.1.8 Método de Jiang

Jiang et al. (2009) apresentam um método de busca na *Web* oculta que faz o preenchimento de campos texto baseados em palavras-chave, campos de texto genéricos. O método se baseia nos dados buscados pelas submissões anteriores. Enquanto os trabalhos que propõem o preenchimento de campos de texto genéricos (MADHAVAN et al., 2008; BARBOSA; FREIRE, 2004; NTOULAS; ZERFOS; CHO, 2004; LIU et al., 2009) utilizam os dados estatísticos de frequência de termos nas páginas resposta (TF, DF e TFIDF) para ordenamento dos termos (valores candidatos ao preenchimento), o método de Jiang utiliza outros atributos para caracterizar os termos das páginas resposta, como as características da língua e da análise do código HTML, além das tradicionais estatísticas de frequência de termos. O conjunto de tuplas, termo e suas características, forma a base de treinamento para obtenção de um modelo de regressão utilizando aprendizado de máquina.

Inicialmente, se submete alguns valores da página HTML onde o formulário se encontra, as páginas respostas servem como base de treinamento de um modelo. Os valores candidatos à submissão são ranqueados por um coeficiente que é calculado pelo modelo obtido. Esse coeficiente diz respeito ao número de registros distintos produzidos por uma consulta. Cada página resposta é analisada e o coeficiente real do valor que gerou a resposta pode ser determinado. De tempos em tempos, os novos valores submetidos são utilizados para mais uma rodada de treinamento e obtenção de um novo modelo. As características apresentadas para o treinamento são as seguintes:

- Propriedades linguísticas: classe gramatical (pronome, substantivo e verbo entre outros), tamanho da palavra e o idioma que a palavra pertence;

- Propriedades do HTML: a *tag* vinculada à palavra, a profundidade dentro da árvore DOM e o destaque da palavra no documento (uma fórmula calcula um coeficiente em função do tamanho da fonte e formatação, como negrito e itálico);

- Propriedades estatísticas: TF, DF, TFIDF e RIDF (YAMAMOTO; CHURCH, 2001).

Para escolha do método de aprendizagem de máquina é apresentado um experimento com um formulário onde são comparados quatro políticas de aprendizado de máquina: Bayes ingênuo, kNN, SVM e MLP. O kNN apresentou melhor desempenho e por isso é utilizado

nos demais experimentos. O método encerra de maneira similar ao método de Liu, pela utilização da janela deslizante.

Os experimentos analisando a cobertura obtida à medida que as consultas são realizadas foram conduzidos sobre três formulários. O método proposto apresentou bons resultados ao ser comparado com uma implementação integrada das soluções (MADHAVAN et al., 2008; BARBOSA; FREIRE, 2004; NTOULAS; ZERFOS; CHO, 2004; LIU et al., 2009) que se assemelham ao considerar estatísticas da frequência de termos.

Em Jiang et al. (2010) o trabalho é complementado, sendo proposto um método semelhante, onde é utilizando aprendizado por reforço. Porém, o principal problema do método persiste, assim como em trabalhos anteriores (BARBOSA; FREIRE, 2004; NTOULAS; ZERFOS; CHO, 2004) não é considerada uma solução para todos tipos de campos de um formulário, sendo proposta uma solução apenas para os campos de texto genéricos.

3.2 Comparação dos Métodos

O preenchimento de formulários de forma automática obtendo cobertura satisfatória e de maneira escalável (um método que possa buscar em *sites* de diversos domínios e línguas) é um grande desafio. A maior parte dos trabalhos apresenta algumas soluções para parte do problema, ou então recaem em utilização de informação anterior, o que resulta em uma solução vertical, vinculada a poucos formulários de determinados domínios.

Para uma melhor comparação dos trabalhos apresentados na Seção 3.1 são definidas oito características importantes para um trabalho que proponha um método de busca na *Web* oculta. São elas:

- **VC (Vários Campos)** - Preenchimento e submissão de vários campos de um formulário ao mesmo tempo;
- **Se (Seleção)** - Preenchimento e submissão de campos de seleção;
- **TG (Texto Genérico)** - Preenchimento e submissão de campos de texto genérico;
- **TT (Texto Tipado)** - Preenchimento e submissão de campos de texto tipado;
- **DF (Domínio dos Formulários)** - Independência do conhecimento do domínio dos formulários;
- **DC (Domínio dos Campos)** - Independência do conhecimento do domínio dos campos;
- **Us (Usuário)** - Independência das informações fornecidas pelo usuário;
- **BC (Base de Comparação)** - Se o trabalho apresentou algum outro método como base de comparação.

A Tabela 3.1 mostra a comparação dos métodos através das características apresentadas. Os campos entre parênteses significam que o trabalho oferece apenas indícios de que possuem ou não determinada característica, sem relatar de forma explícita.

Se por um lado, é desejável que um motor de busca trate todos os tipos de formulário (características VC, Se, TT e TG), por outro lado é desejável que o método seja automático e não utilize conhecimento anterior para escolha de valores para submissão (características DF, DC e Us). Na prática, os métodos não conseguem reunir esses dois

Tabela 3.1: Comparação entre os métodos de preenchimento de formulário

Método	**Itens avaliados**							
	VC	**Se**	**TG**	**TT**	**DF**	**DC**	**Us**	**BC**
Raghavan	**sim**	**sim**	**sim**	(**sim**)	não	não	não	não
Liddle	**sim**	**sim**	não	não	**sim**	**sim**	não	não
Barbosa	não	não	**sim**	não	**sim**	**sim**	**sim**	não
Ntoulas	não	não	**sim**	não	**sim**	**sim**	não	não
Wu	não	(não)	**sim**	não	**sim**	**sim**	**sim**	não
Madhavan	**sim**	**sim**	**sim**	**sim**	**sim**	não	**sim**	não
Liu	**sim**	**sim**	**sim**	(**sim**)	não	não	não	**sim**
Jiang	não	não	**sim**	não	**sim**	**sim**	**sim**	**sim**

grupos de características. A Tabela 3.1 mostra que os métodos mais automatizados (Liddle, Barbosa, Ntoulas, Wu, Jiang) não consideram todos formulários. E os que consideram, como Raghavan e Liu utilizam informação de domínio dos formulários, o que os torna ruins para utilização em toda *Web*. O método que mais se aproxima do desejado é o de Madhavan.

Essas características delimitam também que existem dois tipos de métodos. Os que fazem uma busca vertical em determinados formulários de um mesmo domínio, de forma a obter a maior cobertura possível destes *sites*. E os que fazem uma busca horizontal na *Web* inteira em todos os formulários (de diferentes domínios), a fim de buscar uma maior cobertura da *Web* oculta como um todo e não de um grupo de *sites* específicos.

Muitas das propostas apresentam semelhanças. Uma solução recorrente, por exemplo, é a utilização de conhecimento obtido através de submissões anteriores para escolha de valores para o preenchimento de campos de texto genérico. Boa parte dos trabalhos utilizam informações estatísticas, como frequência de termos, para escolher os seus valores (MADHAVAN et al., 2008; BARBOSA; FREIRE, 2004; NTOULAS; ZERFOS; CHO, 2004; LIU et al., 2009). Essa estratégia é comprovada pelo estudo de Wang, Lu e Chen (2009), que utiliza bases locais para demostrar que um pequeno subgrupo de valores da base pode retornar todos valores da base. Essa parece ser uma tática consolidada e bastante eficiente para campos de consulta a palavras chave (campos de texto genérico), porém o mesmo não ocorre para campos de texto tipados.

Campos fracamente tipados (por exemplo, nome da cidade, nome de autor) que possuem um domínio restrito e principalmente campos fortemente tipados (por exemplo, data) que além de domínio restrito apresentam também formatação específica representam um dos maiores desafios para o preenchimento automático de formulários. As poucas soluções existentes recaem em utilização de informação prévia, por exemplo, acerca do domínio dos campos, como no método de Madhavan.

A maior parte das soluções não considera a submissão de todas as possibilidades de preenchimento de um determinado formulário (LIDDLE et al., 2003; BARBOSA; FREIRE, 2004; NTOULAS; ZERFOS; CHO, 2004; WU et al., 2006; JIANG et al., 2009), ignorando o problema da combinação cartesiana. A correta seleção de um subgrupo de instâncias de *template* a partir do produto cartesiano é uma importante tarefa ainda sem uma resposta definitiva.

Outra lacuna importante presente na maioria dos trabalhos é a falta da utilização de um método como base de comparação com o método proposto (RAGHAVAN; GARCIA-MOLINA, 2000; LIDDLE et al., 2003; BARBOSA; FREIRE, 2004; NTOULAS; ZERFOS; CHO, 2005; WU et al., 2006; MADHAVAN et al., 2008). Na maioria dos casos

as poucas comparações são feitas em termos conceituais e não experimentais. Os dois trabalhos que apresentam comparações experimentais (JIANG et al., 2010; LIU et al., 2009), o fazem de maneira simplificada e reduzida.

O método apresentado neste trabalho, no Capítulo 4, apresenta uma solução que busca a maioria das características presentes na Tabela 3.1. O método proposto possui as seguintes contribuições:

- um estudo sobre como selecionar um subgrupo de consultas ou instâncias de *template* para um mesmo *template*. A relevância desse estudo aumenta para *templates* de ordem dois ou superior, pois possuem um grande número instâncias candidatas;

- um método que pode reduzir drasticamente o número de submissões (quando se considera a submissão de formulários com mais de um campo), o que aumenta a eficiência de um motor de busca na *Web* oculta.

As técnicas propostas no presente trabalho podem ser utilizadas em conjunto com a maioria dos métodos estudados. Os resultados experimentais apresentados são comparados com o estado da arte (método de Madhavan) e mostram a validade da técnica de poda de instâncias de *template* (ITP).

4 PREENCHIMENTO AUTOMÁTICO DE FORMULÁRIOS

Neste capítulo é explicado o problema da escalabilidade cartesiana , ou seja, quando se considera todas as combinações de campos e valores para gerar todas as possíveis consultas a um formulário. Após é mostrada a arquitetura implementada para viabilizar a busca na *Web* oculta. Serão mostrados e explicados os módulos dessa arquitetura, sendo que será dada maior ênfase para o módulo que traz a contribuição do presente trabalho.

4.1 O Problema da Escalabilidade Cartesiana

A necessidade de uma forma de seleção das submissões a um formulário varia de acordo com a complexidade desse, ou seja, para formulários com poucos campos e poucas opções de preenchimento, a submissão de todas as possibilidades é viável. Porém, isso normalmente só ocorre para formulários com um único campo. Para formulários com dois campos ou mais, a possibilidade de seleção torna-se uma necessidade, que cresce à medida que se tem mais campos e que cada campo tenha mais possibilidades de preenchimento. Isso ocorre porque os *templates* de ordem maior são os que têm um produto cartesiano das possibilidades maior também.

A Figura 4.1 mostra um gráfico onde se pode observar como o número total de submissões se comporta em função do número de campos do formulário. É mostrado duas curvas, uma (mais escura e contínua) que considera todos possíveis *templates* gerados e outra (mais clara e tracejada) que considera todos possíveis templates de no máximo ordem 3, de acordo com o que é proposto por Madhavan (2008). No gráfico cada campo tem 30 possíveis valores associados e são somadas todas possibilidades de preenchimento, ou seja, todas as instâncias de *template* geradas por todos *templates* gerados. Por exemplo, um formulário com três campos tem três *templates* na ordem um, cada um com 30 instâncias de *template*, para ordem dois são 3 *templates* com 900 (30*30) instâncias de *template* cada e por fim um *template* de ordem três com 27000 (30*30*30) instâncias de *template*, totalizando 29790 instâncias de *template*, ou possibilidades de preenchimento.

O gráfico mostra que o número de submissões cresce exponencialmente. Submeter dois ou mais campos juntos é uma tarefa custosa, pois, muitas vezes, o produto de combinações é muito grande. Esse problema ocorre mesmo para a curva tracejada que considera no máximo *templates* de ordem 3. Dessa forma, se torna importante para qualquer método que considere o tratamento de mais de um campo, para o preenchimento automático de formulários, uma estratégia para seleção de um subgrupo do total de possibilidades para efetivamente ser submetido. A resposta para quais campos devem ser combinados se refere a quais *templates* devem ser considerados. E a resposta para quais valores submeter

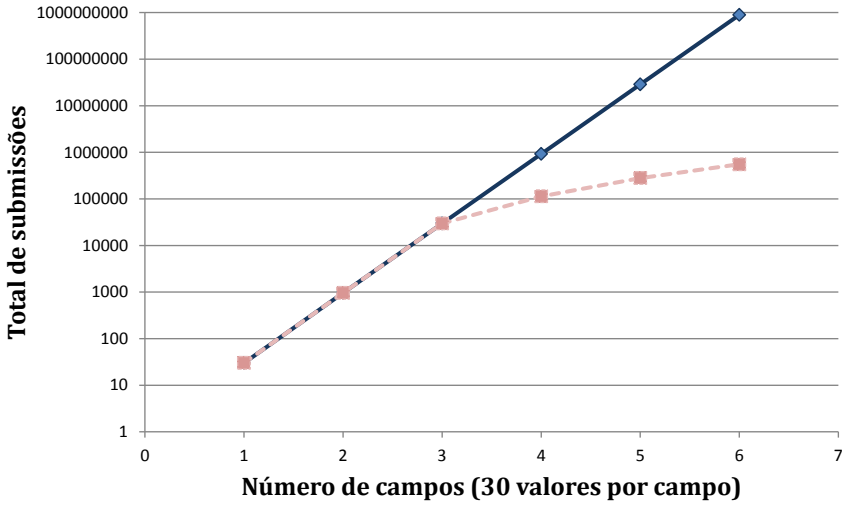

Figura 4.1: Gráfico com o exemplo para ilustrar o problema da submissão de todas as possibilidades de preenchimento

diz respeito a quais instâncias de *template*[1] considerar para cada *template*.

O restante deste capítulo discutirá a arquitetura do motor de busca implementado (Seção 4.2), bem como as estratégias usadas para a seleção desse subgrupo (Seções 4.3 e 4.4), que é o objetivo deste trabalho.

4.2 Arquitetura Implementada

Para que seja possível avaliar as estratégias de seleção de instâncias de *template* para submissão no formulário, foi implementada uma arquitetura para o preenchimento e submissão automática de formulários *Web*. Essa arquitetura é semelhante à proposta por Madhavan et al. (2008), principalmente pela forma com que os valores para os campos de texto genérico são selecionados. Outra semelhança é a consideração de submissões com a combinação de campos, ou seja, são considerados todos os *templates*. Por outro lado, a arquitetura proposta difere do método de Madhavan em alguns aspectos, principalmente na forma como seleciona quais instâncias de *template* serão submetidas. A Figura 4.2 apresenta a arquitetura proposta.

A arquitetura implementada parte do princípio que os formulários são conhecidos e são pontos de entrada para a *Web* oculta. Portanto, a entrada do processo é um formulário *Web* que deverá ser processado automaticamente. É considerado também que os campos que não são interessantes para submissão, como os campos de apresentação, já foram identificados e excluídos do processo.

No começo, o formulário é extraído no módulo *Extração dos campos (de texto e de seleção)*. Seus campos e atributos (características) como *method* (POST ou GET) e *action* são extraídos. Isso é feito através da representação da árvore DOM do documento HTML da página onde o formulário se encontra. Essa representação em árvore auxilia e facilita a navegação, pois o nodo *form* é processado facilmente, obtendo-se seus atributos e seus filhos. Assim, pode-se extrair os campos de seleção e de texto. Nesse momento, os campos de apresentação que não devem ser considerados são filtrados, isso é feito através de uma lista passada pelo usuário em um arquivo de configuração XML (essa intervenção

[1] Os conceitos de *template* e instância de *template* foram apresentados na Seção 2.2

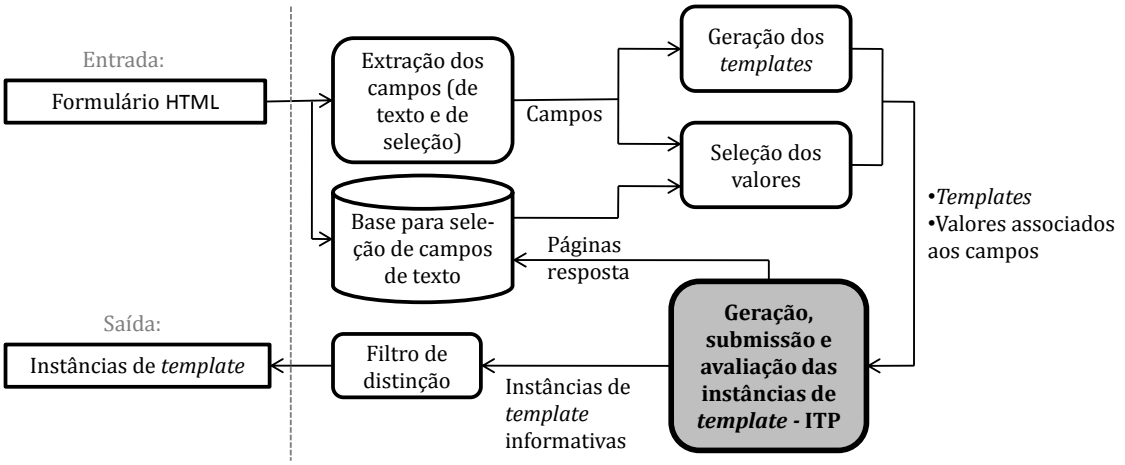

Figura 4.2: Arquitetura implementada para busca na *Web* oculta

do usuário se justifica, tendo em vista que a identificação destes campos não é o foco deste trabalho). Esse arquivo é lido e processado no início da aplicação. Nele estão presentes, além dessa lista, o endereço dos sites que contém os formulários (pontos de entrada a *Web* oculta).

Após ter sido executada a extração dos campos, o módulo de *Geração dos templates* é executado. Nesse momento, cabe salientar que segundo Madhavan (2008), *templates* de ordem superior a três não representam muitas possibilidades de submissão que retornarão dados, dado o grau de especificidade da consulta, e assim podem ser desconsiderados. Dessa forma, a presente arquitetura considera a geração de todos os *templates* de ordem um, dois e três.

No módulo de *Seleção dos valores* são relacionados os valores para cada campo de interesse do formulário. Para os campos de seleção esse processo é simples, pois os valores são de domínio finito e constam no HTML, podendo ser facilmente extraídos pela navegação na árvore DOM do nodo *form*. Já para os campos de texto, é utilizada uma base de dados. A partir dessa base são ranqueados e extraídos os *top-p tokens*. Esses *p tokens* serão os valores selecionados para os campos de texto. Essa base é composta pela extração dos dados da página inicial que contém o formulário e dos dados retornados das submissões feitas ao formulário até o momento. Com a finalidade de garantir que a base seja rica o suficiente em dados (*tokens*), primeiramente todos *templates* de primeira ordem relacionados aos campos de seleção são submetidos. Caso o formulário apresente apenas campos de texto, será utilizada apenas a página inicial (a página que contém o formulário). A lista criada com os valores para o primeiro campo de texto que necessita ser preenchido é armazenada e utilizada para todos os outros *templates* que contenham campos de texto.

Os documentos da base de dados são obtidos pela extração apenas do conteúdo textual do documento HTML, assim todas as *tags*, *scripts* e comentários presentes no código são filtradas. Os documentos gerados a partir de páginas resultantes de submissões ao formulário passam primeiramente por um outro filtro. Esse filtro faz a extração da região de dados (local do HTML que estão os registros retornados da base). Isso é feito com a utilização de um método de extração de informação. A intuição aqui é que provavelmente *tokens* presentes na base por trás do formulário são bons candidatos para submissões que retornem novos dados (BARBOSA; FREIRE, 2004; NTOULAS; ZERFOS; CHO, 2005;

WU et al., 2006; WANG; LU; CHEN, 2009).

A forma com que os *p tokens* são ranqueados é inspirada na medida de recuperação de informações TF-IDF (SALTON; MCGILL, 1983). Dessa forma, o conteúdo extraído de cada página (a inicial que contém o formulário e as respostas às submissões) será um documento dentro da coleção (base de dados). Cada um dos *tokens* dessa coleção recebe uma pontuação para que possam ser ranqueados. Essa pontuação é o escore r_t mostrado na equação 4.1. O componente cf_t (*collection frequency*) é o número de vezes que cada *token* aparece na base de dados inteira. *N* é o número de documentos e df_t (*document frequency*) é o número de documentos que contém *t*. O componente idf_t (*inverse document frequency*) é usado para remover *tokens* presentes em todos os resultados, por exemplo, o nome dos campos da tabela resposta. Quando não existem páginas resposta na base e necessita-se selecionar valores para campos de texto, são ranqueados os *tokens* apenas da página inicial, de forma que é feita somente a contagem dos termos.

$$r_t = cf_t \times idf_t \quad onde \quad idf_t = log\frac{N}{df_t} \tag{4.1}$$

O conjunto de *templates*, com seus possíveis valores associados são passados para o módulo *Geração, submissão e avaliação das instâncias de template - ITP*. Nesse módulo, está a contribuição do presente trabalho. Nele são geradas e submetidas as instâncias de *template*. Nele também é feita extração das páginas resposta para alimentar a *Base para seleção de campos de texto* e para avaliar a qualidade de cada instância de *template*, que irá influenciar na criação de novas instâncias de *template*. As primeiras instâncias de *template* geradas e submetidas são as de ordem um, resultantes dos campos de seleção. Isto é feito para que se possa alimentar a *Base para seleção de campos de texto*. Após são submetidas as instâncias de ordem um, referentes aos campos de texto. Posteriormente, são geradas e submetidas as instâncias de *template* de ordem dois e três respectivamente. É obtido como saída desse módulo todas as instâncias de *template* consideradas informativas. A Seção 4.3 apresenta com maiores detalhes como este módulo atua, ou seja, como a seleção das instâncias de *template* informativas é feita.

O fluxograma contido na Figura 4.3 demonstra melhor o processo de geração dos valores para campos de texto e geração das instâncias de *template*. Assim, a geração de valores para campos de texto é tratada de forma diferente para os formulários com campos de texto e de seleção e para os formulários com apenas campos de texto. Para os formulários com apenas campos de texto, a *Base para seleção de campos de texto* contém apenas a página onde está contido o formulário. Já quando o formulário possui campos de seleção a base contém também as páginas respostas às submissões de todas instâncias de *template* de ordem um, relativas aos campos de seleção. A obtenção dos *top-p tokens* é feita uma única vez e essa lista é vinculada a todos os campos de texto presentes no formulário.

O foco do presente trabalho não é a seleção dos valores para preenchimento de campos de texto. Por isso, é mais simples fazer a seleção da lista de valores para um campo textual uma única vez. Essa característica também facilita a comparação dos resultados experimentais presentes no Capítulo 5.

Após o fim da geração e submissão de todas instâncias de *template* de todos *templates* gerados é feito um último processamento no módulo *Filtro de distinção*. Esse módulo recebe todas as instâncias de *template* consideradas informativas e todas as páginas respostas retornadas por cada uma dessas instâncias. Cabe salientar neste momento que muitas instâncias de *template* informativas retornam os mesmos registros já encontrados por

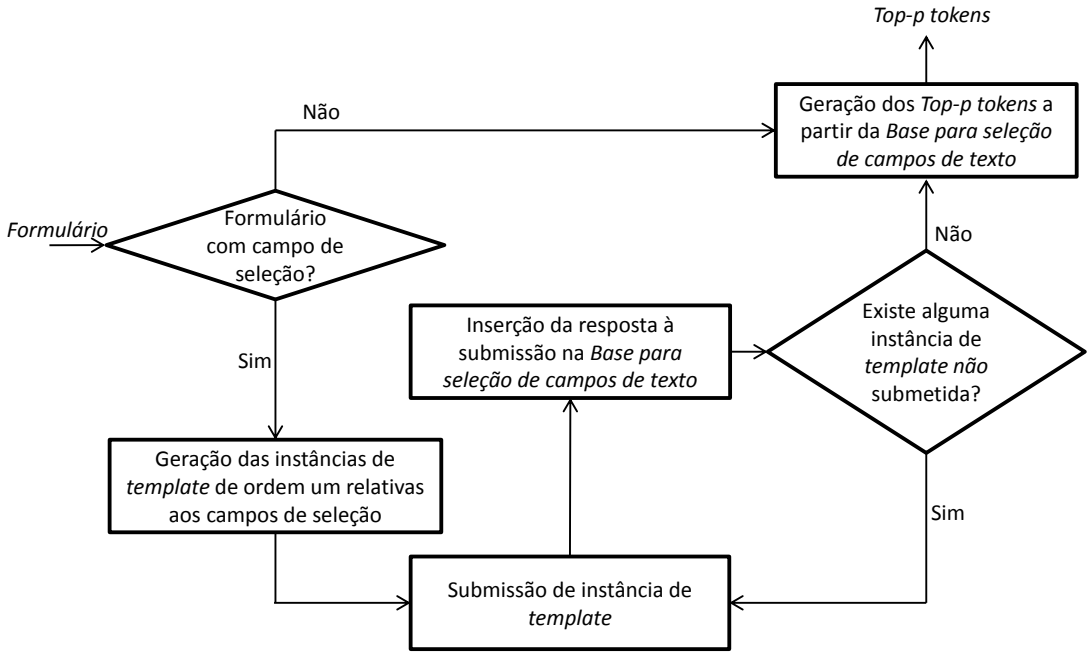

Figura 4.3: Fluxograma que demonstra a obtenção dos *top-p tokens* para geração de valores para os campos texto

outras instâncias de outros *templates* já submetidas e consideradas também informativas. Principalmente em formulários com bases de dados pequenas. Assim, esse módulo seleciona o menor número de instâncias de *template* que retornam todos os dados distintos da base por trás do formulário retornados através de pelo menos uma submissão de instância informativa feita durante todo processo. A saída final da arquitetura é um número reduzido de instâncias de *template*, que produzem todos os dados distintos retornados durante o processo, e suas páginas respostas que podem ser indexadas para eventuais buscas.

4.3 Seleção Dinâmica

Nesta Seção é apresentada a estratégia para a seleção de valores (instâncias de *template*) para a submissão, denominada *Instance Template Pruning* (ITP). Como visto anteriormente é feita a exploração das instâncias de *template* dos *templates* de no máximo ordem três. Mesmo com essa limitação a maior parte dos formulários com mais de três campos pode gerar um número muito grande de instâncias de *template*, inviabilizando a realização de todas submissões. A ideia geral é aprender à medida que as submissões são realizadas e, assim, descobrir e descartar futuras submissões indesejadas, ou seja, que não retornam dados. Como as instâncias de *template* que serão submetidas são selecionadas durante o processo, a seleção é dita dinâmica.

Primeiramente, torna-se necessário identificar os critérios que determinaram se os valores submetidos em um formulário continuarão a ser explorados ou não. Cada submissão ao formulário gera uma página resposta, e é a informação dessa página que deve ser analisada. Dessa forma, são definidos os seguintes critérios que determinam quando uma instância de *template* é não informativa:

- **Página de erro**: quando uma submissão não retorna uma página resposta com dados do banco e sim um erro. É provável que esse erro aconteça novamente quando

se submete estes valores outra vez, mesmo que combinados com outros valores;

- **Página em branco**: quando uma submissão não retorna nenhum dado, não há nenhuma ocorrência dos valores submetidos na base de dados por trás do formulário dos valores submetidos;

- **Página com pouco conteúdo**: quando uma submissão retorna muitos dados a página resposta usualmente retorna parte desses dados, pois o resultado é expresso com a utilização de várias páginas. Como a submissão de um valor em um campo aumenta a especifidade da consulta, se uma submissão retorna menos dados que o máximo apresentado por página, significa que a utilização dos valores submetidos combinados com outro valor irá retornar no máximo o mesmo número de dados retornados anteriormente.

Assim, define-se como **não informativas** todas instâncias de *template* submetidas que recaírem em uma dessas três situações citadas. Já as instâncias de *template* que retornarem mais dados que o máximo apresentado por página são consideradas **informativas**. Toda instância de *template* submetida é avaliada e classificada em informativa ou não. As classificadas como não informativas são inseridas em uma lista de poda. Essa lista representa o conhecimento obtido até o momento através das submissões, e é considerada sempre que as instâncias de um determinado *template* são geradas. Dessa forma, é reduzido o número de instâncias geradas e por sua vez o número de submissões realizadas.

A Figura 4.4 mostra com mais detalhes o processo de geração, submissão e avaliação das instâncias pelo método ITP. No início, a lista de poda é vazia e vai sendo preenchida à medida que as instâncias de *template* vão sendo submetidas. No módulo *Geração das instâncias*, são criadas as instâncias de cada *template*, levando em conta a lista de poda e os valores associados a cada campo. Nas instâncias de primeira ordem a lista de poda não é considerada, pois as instâncias são independentes. Isso não representa um problema, pois são geradas poucas instâncias nos *templates* de ordem um. Para as ordens dois e três, valores que não podem mais explorar novos dados (presentes na lista de poda) são ignorados. As instâncias de *template* geradas são submetidas uma a uma. Cada página resposta à submissão de uma instância passa pelo módulo *Extração de Informação*, que é responsável por extrair apenas os dados retornados da base de dados por trás do formulário. Esses dados contêm a informação necessária para a avaliação das instâncias de *template* e classificação em informativa ou não informativa. Páginas de erro ou em branco são facilmente identificadas pela extração da informação. Descobrir o número máximo de registros retornados por página pode ser feito por uma submissão ao formulário que retorne muitos dados, por exemplo, o formulário submetido em branco. As instâncias não informativas são inseridas na lista de poda, já as informativas são a saída do processo. Esse processo, que é repetido para todos *templates*, evita a submissão desnecessária de instâncias de *template*. Uma instância de *template* considerada não informativa implica na poda de outras instâncias de *template* de ordem superior. Para uma visualização alternativa do processo descrito, é apresentado o algoritmo com o método ITP na Figura 4.5.

Para um melhor entendimento do método ITP é apresentado um exemplo na Figura 4.6. Supõe-se um formulário com três campos (A, B e C), cada um com três possibilidades de preenchimento, valores A1, A2 e A3 (associados ao campo A), B1, B2 e B3 (associados ao campo B) e C1, C2 e C3 (associados ao campo C). Os valores sublinhados são instâncias de *template* submetidas e consideradas não informativas. Já os valores tachados são as instâncias de *template* que nem foram geradas para submissão

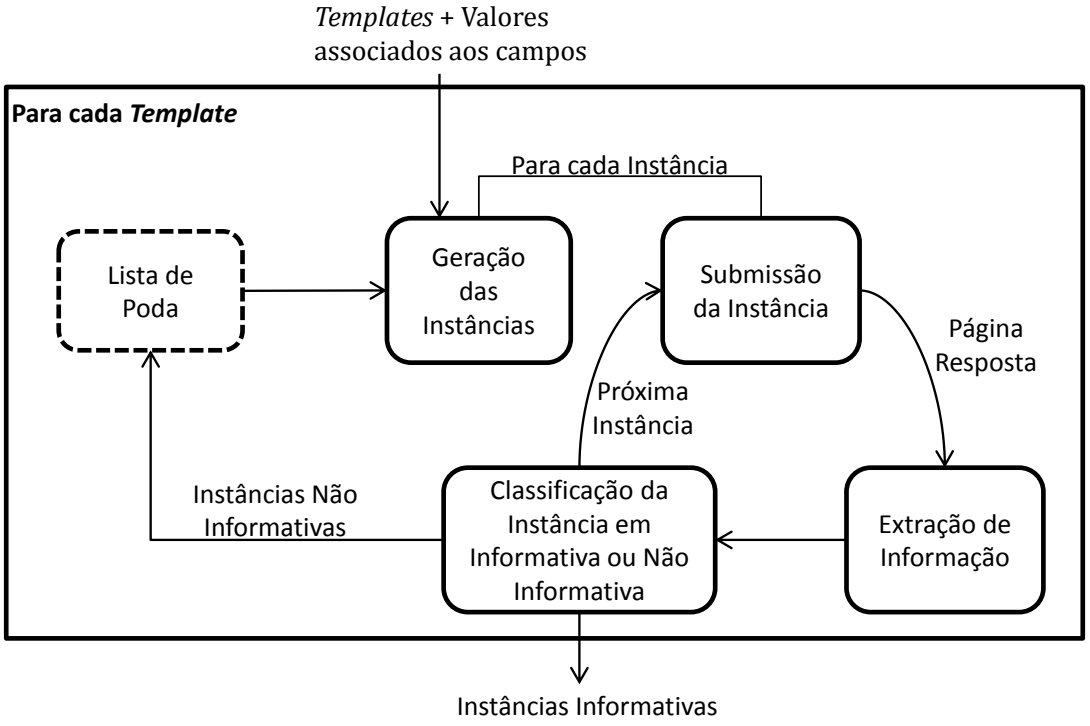

Figura 4.4: Método ITP

após a avaliação da lista de poda. E os demais valores representam as instâncias de *template* submetidas e consideradas informativas. As setas representam quais valores da lista de poda são considerados para geração das instâncias de cada *template*. Por exemplo, a instância do *template* A, A = A1 foi considerada não informativa, assim os *templates* que consideram o preenchimento do campo A (*templates* AB, AC e ABC) tem todas as instâncias de *template* com o valor A1 para o campo A removidas do grupo de instâncias geradas e consideradas para submissão. Com isto, o número de instâncias geradas é reduzido. Um método que considera o produto cartesiano das possibilidades, sem nenhuma forma de poda, submeteria 63 instâncias. Com o método ITP o número de submissões cai para 33, onde 4 são classificadas como não informativas e 29 como informativas. O método descobriu 30 instâncias de *template* que não retornam dados antes que fossem submetidas e assim as ignorou, isto com a classificação de apenas 4 instâncias de *template* como não informativas.

O ITP pode ser visto como um método que aprende quais são as submissões ruins (que não retornam informação relevante) e utiliza esse conhecimento para evitar outras submissões ruins. A lista de poda representa o conhecimento obtido durante o processo. E os critérios utilizados, para identificar se uma instância de *template* vai para a lista de poda ou não, são as heurísticas utilizadas.

O método ITP pode ainda ser utilizado em conjunto com o teste de informatividade [2] de *template* proposta por Madhavan et al. (2008). Dessa forma, primeiramente é decidido se um *template* é informativo e, só caso seja, é que serão geradas as instâncias de *template* pelo método ITP. Com isso, as duas podas, de *templates* e de instâncias de *template*, são complementares na diminuição do grupo de instâncias de *template* submetidas para cada formulário. O método que considera as duas podas é definido como *Instance Template*

[2] O algoritmo ISIT para avaliação da informatividade de *templates* é descrito na Subseção 3.1.6

```
ITP(lista_templates, lista_valores)
  lista_de_inst_informativas = novaLista();
  lista_de_poda = novaLista();
  tamI = tamanho(lista_templates);
  para i de 1 até tamI faça
    temp = renornaNaPos(lista_templates, i);
    vals = valoresDoTemplate(temp, lista_valores);
    lista_de_instâncias = geraInst(lista_de_poda,temp,vals);
    tamJ = tamanho(lista_de_instancias);
    para j de 1 até tamJ faça
      instancia = retornaNaPos(lista_de_instancias,j);
      pagina = submete(instancia);
      info = extraiInformacao(pagina);
      se (testeInformatividade(info)) então
        //é uma instância informativa
        insere(lista_de_inst_informativas, instancia);
      senão
        //é uma instância não informativa
        insere(lista_de_poda, instancia);
      fim se
    fim para
  fim para
  retorne lista_de_inst_informativas;
```

Figura 4.5: Algoritmo: Método ITP

and Template Pruning (ITTP).

Existem situações, principalmente para *templates* de ordem três, que mesmo esses métodos apresentados (ITP e ITTP) geram um conjunto instâncias de *template* para submissão ainda muito grande. De modo a evitar muitas submissões do mesmo *template* torna-se necessário limitar o número de instâncias de *template* geradas por *template*, ou seja, gerar e submeter apenas um subgrupo do total de possibilidades após as podas das instâncias. A Seção 4.4 faz a discussão de táticas de seleção prévia de um subgrupo de instâncias de *template*.

4.4 Seleção Estática

Quando o número de instâncias de *template* geradas pelo método ITP é ainda muito grande, um critério de parada na geração das instâncias é necessário. Dessa forma, garante-se um número máximo de instâncias de *template* geradas para cada *template*. Como a seleção das instâncias de *template* que serão submetidas é feita antes do processo de submissão e sem considerar submissões anteriores, a seleção é dita estática. Considerando k como sendo esse número máximo de instâncias de *template* geradas, esta Seção aborda formas para selecionar k instâncias. Assim, dado um grupo de instâncias de *template* deseja-se selecionar um subgrupo de instâncias. Uma instância de *template* pode ser escolhida em detrimento de outra, dependendo do critério utilizado para a escolha do

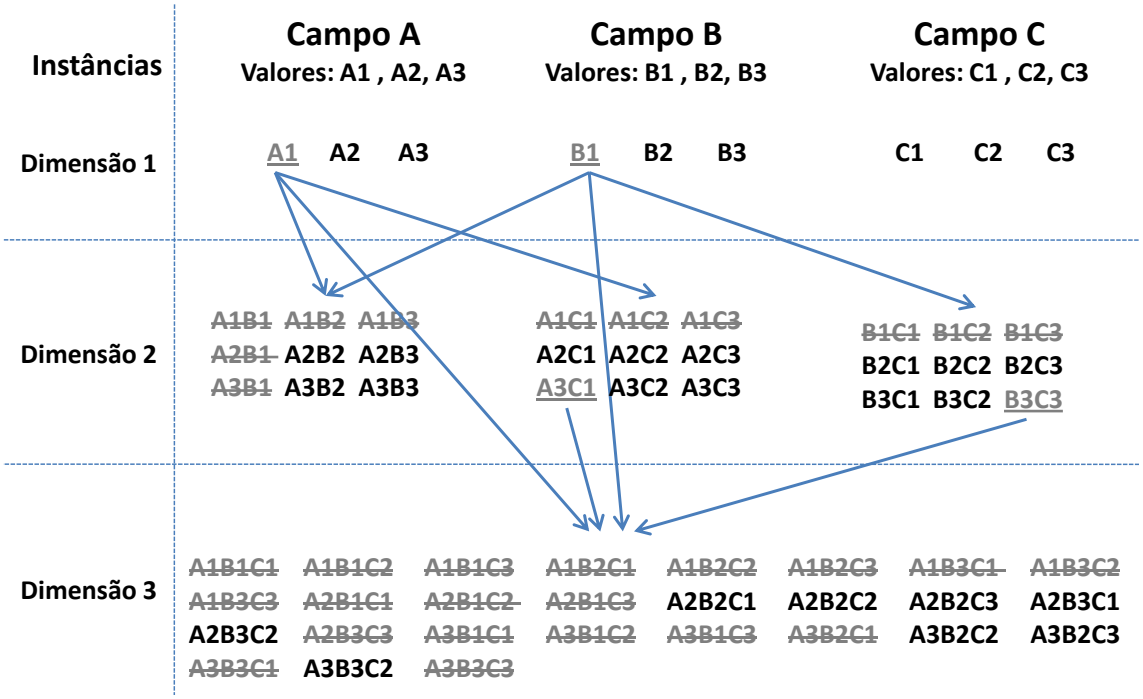

Figura 4.6: Exemplo da utilização do método ITP

subconjunto. Essa Seção aborda quatro estratégias para seleção de um subgrupo de k instâncias de *template*

Para melhor discussão das estratégias de seleção é considerado um exemplo de um formulário com dois campos, A e B. Onde o campo A tem quatro possibilidades de preenchimento, a_1, a_2, a_3 e a_4, e o campo B tem seis possíveis valores, b_1, b_2, b_3, b_4, b_5 e b_6. Os *templates* para esse formulário são A, B e AB. O número de possíveis submissões, instâncias de *template*, que podem ser geradas por esse formulário é dado pelas submissões de cada campo individualmente mais o produto da combinação deles. Assim existem 34 possibilidades de preenchimento, 4 para o campo A, 6 para o campo B e 24 para a combinação dos dois campos. Com isso as instâncias geradas são: $\{A = a_1, A = a_2, A = a_3, A = a_4, B = b_1, B = b_2, B = b_3, B = b_4, B = b_5, B = b_6, A = a_1 \& B = b_1, A = a_1 \& B = b_2, ..., A = a_4 \& B = b_5, A = a_4 \& B = b_6\}$.

Para esse exemplo simples se poderia submeter todas instâncias de *template*. Porém, para formulários com mais campos e mais valores por campo isso se torna proibitivo. Uma parte apenas das submissões podem ser necessárias para se obter como resposta a maior parte da base por trás do formulário. Para ilustrar isto, considera-se que o valor k é 6. Assim, para os dois primeiros *templates* do exemplo, A e B, não é necessário efetuar poda de instâncias de *template*. Mas para o *template* AB deve-se podar 18 instâncias de *template*, selecionando 6 instâncias para a submissão. As Figuras 4.7, 4.8, 4.9 e 4.10 mostram quatro possibilidades de escolha destas seis instâncias de *template* baseado em quatro estratégias:

- Estratégia 1: *k-first*. O algoritmo para seleção das instâncias de *template* começa combinando o primeiro valor do primeiro campo com cada um dos valores do segundo campo, e assim por diante. O segundo valor do primeiro campo só é utilizado após a seleção de todas as instâncias de *template* que possuem o primeiro valor para o primeiro campo. Com isso a estratégia seleciona as primeiras k instâncias de *template* do espaço de possibilidades para a submissão. Para o exemplo da Figura 4.7

com k sendo 6, o *template* AB com 24 instâncias de *template*, tem as seguintes instâncias selecionadas: $A = a_1 \& B = b_1, A = a_1 \& B = b_2, A = a_1 \& B = b_3, A = a_1 \& B = b_4, A = a_1 \& B = b_5, A = a_1 \& B = b_6$.

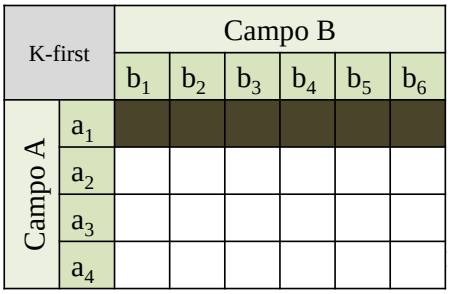

Figura 4.7: Estratégia *k-first*

- Estratégia 2: *k-random*. Nessa estratégia a seleção do subconjunto com k instâncias de *template* do total de possibilidades é feito de forma aleatória. Nenhuma análise prévia é feita para seleção, sendo um processo estocástico. A Figura 4.8 mostra um exemplo de uma possível seleção.

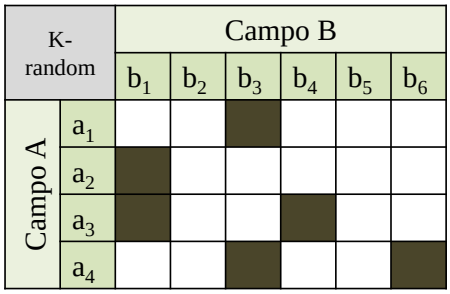

Figura 4.8: Estratégia *k-random*

- Estratégia 3: *k-linear*. Neste processo a seleção do subconjunto com as k instâncias de *template* é feita de forma linear. É escolhido um passo linear de acordo com o total de instâncias de *template* geradas no *template* (N_{tot}), e k, o tamanho do subconjunto. Assim o passo é definido pela proporção N_{tot}/k. Para o exemplo da Figura 4.9, $k = 6$ e $N_{tot} = 24$, assim o passo linear calculado é 24/6=4. E as instâncias de *template* selecionadas são: $A = a_1 \& B = b_1, A = a_1 \& B = b_5, A = a_2 \& B = b_3, A = a_3 \& B = b_1, A = a_3 \& B = b_5, A = a_4 \& B = b_3$.

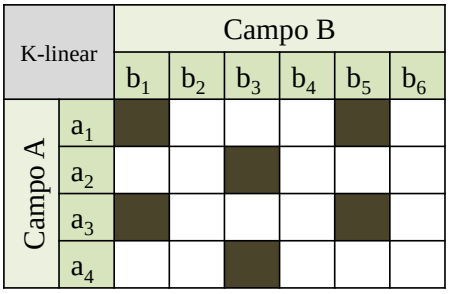

Figura 4.9: Estratégia *k-linear*

- Estratégia 4: *k-all values*. A idéia desta técnica é selecionar uma instância de *template* com valores diferentes da instância selecionada anteriormente. Inicialmente, seleciona-se a primeira instância de *template* e a seleção de uma próxima é obtida pela variação dos valores utilizados em cada campo. Dessa forma, se o valor k for maior ou igual que o número de valores associados ao campo com mais opções de preenchimento, a estratégia garante a utilização de todos valores de todos campos, pelo menos uma vez. A Figura 4.10 mostra um exemplo para k=6 onde as instâncias selecionadas são: $A = a_1 \& B = b_1, A = a_2 \& B = b_2, A = a_3 \& B = b_3, A = a_4 \& B = b_4, A = a_1 \& B = b_5, A = a_2 \& B = b_6$.

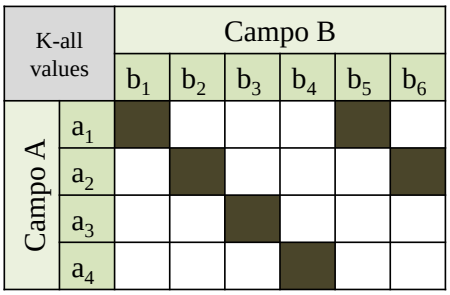

Figura 4.10: Estratégia *k-all values*

As estratégias *k-first*, *k-random* e *k-linear* não buscam a garantia de submeter todos os possíveis valores em cada campo pelo menos uma vez, diferentemente da técnica *k-all values*. Com isso, o mesmo valor é submetido várias vezes em detrimento de outros valores que podem não ser submetidos. Por exemplo, na estratégia *k-first* os valores do final do conjunto de possibilidades dificilmente são selecionados. Na Figura 4.7 todas instâncias de *template* com o valor a_4 para o campo A encontram-se no final do conjunto e assim não são selecionadas para valores baixos de k. A ideia é que com a submissão de um número maior de valores, como é feito no *k-all values* pode-se atingir uma cobertura maior da base por trás do formulário.

5 EXPERIMENTOS

Este capítulo apresenta os experimentos que foram realizados para avaliar os métodos de seleção de valores para o preenchimento de formulários propostos e apresentados no Capitulo 4. São mostradas ainda as métricas utilizadas (baseadas na cobertura e no número de submissões) e a discussão dos resultados obtidos.

Os experimentos foram divididos em duas partes. Primeiramente, se realizou um experimento para avaliar as quatro técnicas de poda estática apresentadas na Seção 4.4 com uma base de dados. Em outro momento, com outra base de dados, se realizou os experimentos que avaliaram a arquitetura inteira implementada, com variações no método de poda dinâmica (Seção 4.3).

Para a realização dos experimentos, foram utilizados formulários de *sites* reais. Como os métodos propostos e a arquitetura implementada são independentes do domínio do formulário dado como entrada, foram utilizados formulários de domínios diversos, como por exemplo: livros, empregos, filmes, músicas, etc. Alguns formulários foram utilizados nas duas partes dos experimentos. As bases de dados reais, como as existentes por trás dos formulários utilizados, apresentam uma variabilidade grande com o passar do tempo. Assim, cada conjunto de experimentos foi realizado em sequência, com um tempo mínimo entre diferentes execuções do mesmo formulário. Com isso, essa variabilidade foi minimizada, o que facilita as comparações pretendidas pela realização dos experimentos. As duas partes dos experimentos foram realizadas em períodos diferentes. Dessa forma, mesmo que as duas bases de teste apresentem alguns formulários em comum, são consideradas independentes e diferentes, pois as bases por trás dos formulários são dinâmicas.

Inicialmente, realizou-se alguns experimentos para o ajuste dos parâmetros. Foi utilizado p sendo 50, ou seja foram escolhidos 50 *tokens* como candidatos a submissão de campos texto (MADHAVAN et al., 2008). O valor máximo de instâncias de *template* geradas por *template*, k, foi escolhido de acordo com o número de campos do formulário. Para formulários com três ou menos campos foi considerado k sendo 800, já formulários com quatro campos, k foi considerado igual a 600 e para formulários com cinco ou mais campos foi considerando k igual a 400.

Para a extração da informação das páginas resposta do formulário foi utilizado o método MDR (LIU; GROSSMAN; ZHAI, 2003). Esse é um método automático de extração de dados semiestruturados baseado nos padrões do HTML DOM.

Para a avaliação dos experimentos foram levados em consideração os seguintes aspectos: a quantidade de informação da *Web* oculta que foi buscada, o custo para se indexar essa informação e o custo do processo. Dessa forma, são utilizadas as seguintes métricas:

- **Cobertura (*C*)**: a cobertura diz respeito aos dados buscados na base por trás do formulário, ou seja, o número de registros distintos obtidos por todas as submissões

feitas ao formulário;

- **Eficiência de Execução (*EE*)**: a eficiência de execução refere-se ao custo de execução em contra partida com os dados buscados. O custo para executar um método que preenche formulários *Web* é dado pelo número de submissões feitas. Assim, considerando *TS* como o total de submissões feitas ao formulário durante todo o processo, a eficiência de execução, *EE*, é dada pela razão entre *C* e *TS* (equação 5.1);

$$EE = \frac{C}{TS} \tag{5.1}$$

- **Eficiência de Indexação (*EI*)**: a eficiência de indexação refere-se ao custo para se indexar o total de dados retornados no processo. O custo de indexar as informações retornadas por um método diz respeito ao número de URLs que serão indexadas, onde cada URL é a implementação de uma instância de *template*. Considerando o total de instâncias de *template* retornadas por um método (candidatas a indexação) sendo *TI*, a eficiência de indexação, *EI*, é dada pela razão entre *C* e *TI* (equação 5.2).

$$EI = \frac{C}{TI} \tag{5.2}$$

A intuição básica é que um bom método deve ter uma boa cobertura, obtendo isso através de um número mínimo de submissões ao formulário, o que representa valores altos para as três métricas. Cabe ainda salientar que dependendo da aplicação que faz busca na *Web* oculta, pode ser mais interessante que um método tenha, ou melhor eficiência de execução ou melhor eficiência de indexação. Por exemplo, para uma aplicação que apenas faz busca por dados em um *site* e não indexa a informação, será mais interessante um método de execução eficiente. Já para uma aplicação como um motor de busca que indexa a informação obtida para outras consultas, será mais interessante uma maior eficiência de indexação.

5.1 Avaliação da Seleção Estática

Esta seção apresenta os experimentos feitos para avaliar as diferentes estratégias de seleção estática de valores para o preenchimento do formulário. Como visto anteriormente, é necessário limitar o número máximo de instâncias de *template* geradas para um *template*, ou seja, caso a seleção dinâmica não reduza suficientemente o número de instâncias que devem ser submetidas, torna-se relevante a utilização de um critério de parada estático, ou um número (k) máximo de instâncias de *template* gerado por *template*.

A comparação entre as estratégias de seleção de k instâncias de *template* é independente à poda dinâmica e, por isso, os experimentos que buscam determinar a melhor estratégia foram realizados também de forma independente. Foi procurado que a variabilidade entre diferentes execuções seja mínima (para execuções do mesmo formulário e das diferentes estratégias). Com isso, estes experimentos desconsideram o método de seleção de valores dinâmicos. Já que, ao se submeter diferentes instâncias de *template* utilizando diferentes estratégias, obteria-se diferentes listas de poda e isso poderia atrapalhar uma melhor visualização dos resultados.

A poda de *templates* [1] (MADHAVAN et al., 2008) foi utilizada para que se evitasse os *templates* de ordem superior que possuem várias instâncias de *template* e não retornam

[1] O algoritmo ISIT que poda os *templates* não informativos é descrito com mais detalhes na Subseção 3.1.6

praticamente nenhum dado. Isso é ainda mais relevante pelo fato de se ter utilizado como base de comparação a submissão de todos valores de um *template*, o que, para alguns casos, é muito custoso. O único cuidado necessário, para a melhor visualização dos dados experimentais, foi garantir que, para um mesmo formulário, todas estratégias selecionem os mesmos *templates* como informativos.

A base de dados experimental 1 é apresentada na Tabela 5.1. Nela são mostradas as páginas *Web* com os formulários utilizados nos experimentos dessa seção. A Tabela 5.1 ainda mostra o número de campos de texto e de campos de seleção de cada formulário. Como se pode notar, foram utilizados formulários de tamanhos diversos, apenas foram ignorados formulários com um campo. Já que, para esses, não se faz necessário a poda de valores, como eles não possuem *templates* que combinem campos (ordem 2 ou superior), também não apresentam um número grande de instâncias de *template* geradas. O tamanho da base por trás de cada formulário não é apresentado, pois ele não é necessário para se comparar uma estratégia com outra e a maioria dos *sites* omite essa informação, mesmo porque, para alguns sites, esse dado varia muito rápido.

Tabela 5.1: Propriedades dos formulários da base de dados experimental 1

Id	**Formulário *Web***	**Campos de texto**	**Campos de seleção**
1	http://www.global-standard.org	1	3
2	http://www.hcareers.com/seeker/search	1	5
3	http://www.rtbookreviews.com/rt-search/books	1	3
4	http://www.boston.com/jobs	1	2
5	http://www.shadetrees.org/search.php	0	4
6	http://pipa.gov.ps/results.asp	0	4
7	http://www.usajobs.gov/	2	0
8	http://formovies.com/search/combined.html	3	0
9	http://www.mymusic.com/advancedsearch.asp?curr=1	6	2
10	http://www.onlineraceresults.com/search/index.php	1	3

A cobertura da base já é suficiente para a avaliação das estratégias de seleção estática. Porém a eficiência de execução também é utilizada para demonstrar, primeiramente, a necessidade de uma poda estática de k valores. E em um segundo plano, demonstrar com dados reais o problema da submissão de todas as instâncias possíveis pelo produto cartesiano (Figura 4.1), que é visível mesmo com a poda dos *templates* não informativos. Dessa forma os experimentos realizados na base de dados experimental 1 apresentam as quatro estratégias de seleção de k valores vistos na Seção 4.4 (*k-first*, *k-linear*, *k-random*, *k-all values*) e a submissão de todas as possibilidades dos *templates* informativos (que é o base de comparação adotada) chamada de "produto cartesiano".

A Tabela 5.2 apresenta o total de instâncias de *template* submetidas pelo produto cartesiano e pelas quatro estratégias. Já a Tabela 5.3 apresenta os resultados obtidos da cobertura normalizada e eficiência de execução. Como os resultados variam muito de um formulário para outro e o importante é a comparação das estratégias em um mesmo formulário, foi utilizado a apresentação gráfica dos resultados de forma normalizada. As Figuras 5.1 e 5.2 apresentam os gráficos da cobertura e eficiência de execução respectivamente. A normalização é feita pelo melhor resultado obtido em cada formulário e para cada métrica, de forma que uma melhor visualização dos resultados é obtida. O eixo x está relacionado com o Id de cada formulário apresentado na Tabela 5.1.

Tabela 5.2: Instâncias de *template* submetidas para base de dados experimental 1

Id	***TS* estratégias**	***TS* produto cartesiano**
1	1609	2209
2	9197	1913840
3	3591	41299
4	876	1375
5	2041	77549
6	5474	3343199
7	972	2600
8	150	150
9	3357	8057
10	2224	17437

Tabela 5.3: Cobertura e eficiência de execução obtidos para a base de dados experimental 1

Id	**Produto cartesiano**		***k-all values***		***k-linear***		***k-random***		***k-first***	
	C	***EE***	***C***	***EE***	***C***	***EE***	***C***	***EE***	***C***	***EE***
1	1139	0,52	1077	0,67	1076	0,67	**1130**	**0,70**	1061	0,66
2	6176	0,00	4021	0,44	3665	0,40	**4136**	**0,45**	3709	0,40
3	32765	0,79	11130	3,10	10309	2,87	**12144**	**3,38**	10751	2,99
4	56481	41,08	28714	32,78	**35603**	**40,64**	32592	37,21	29379	33,54
5	1756	0,02	**1756**	**0,86**	**1756**	**0,86**	**1756**	**0,86**	**1756**	**0,86**
6	409	$1{,}2\text{x}10^{-4}$	**408**	**0,07**	389	0,07	404	0,07	**408**	**0,07**
7	2713	1,04	1158	1,19	1592	1,64	**2639**	**2,72**	2295	2,36
8	886	5,91	993	6,62	1134	7,56	1163	7,75	**1198**	**7,99**
9	11294	1,40	**5567**	**1,66**	5230	1,56	5135	1,53	3847	1,15
10	6842	0,39	5212	2,34	5260	2,37	5240	2,36	**5500**	**2,47**

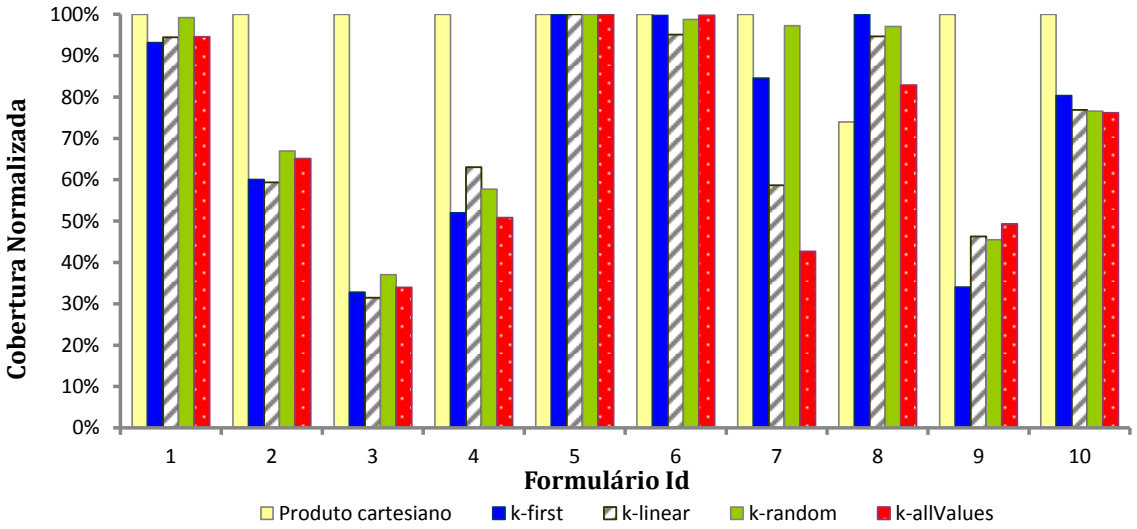

Figura 5.1: Cobertura normalizada pelos melhores resultados obtidos em cada formulário da base de dados experimental 1

A análise dos dados permite perceber que o produto cartesiano obtém, em quase todos os formulários, uma cobertura maior ou igual, assim como submete mais instâncias

de *template*. A exceção é o formulário 8, onde não há *templates* de ordem dois ou superior informativos, dessa forma a poda de *k* valores não é feita e o número de submissões é igual para as estratégias e o produto cartesiano. A cobertura varia, pois os campos desse formulário são campos textuais e os valores submetidos podem sofrer leve variação, provavelmente devido à variabilidade dos *tokens* obtidos no *site*, outra possibilidade é a base de dados ser dinâmica o suficiente para ter modificado os resultados. Em cinco formulários as estratégias de seleção obtiveram cobertura próxima ou igual ao produto cartesiano, o que demonstra claramente que não se necessita submeter todas as possibilidades, mesmo para os *templates* informativos.

Ao analisar a eficiência de execução (gráfico da Figura 5.2) é fácil perceber que o custo da execução do produto cartesiano é, muitas vezes, proibitivo, como nos formulários 2, 3, 5, 6 e 10. Conforme esperado, a eficiência de execução das estratégias de seleção é melhor que o produto cartesiano para a maioria dos casos. Esse comportamento se acentua para formulários de alta complexidade e/ou com poucos dados presentes na base por trás do formulário.

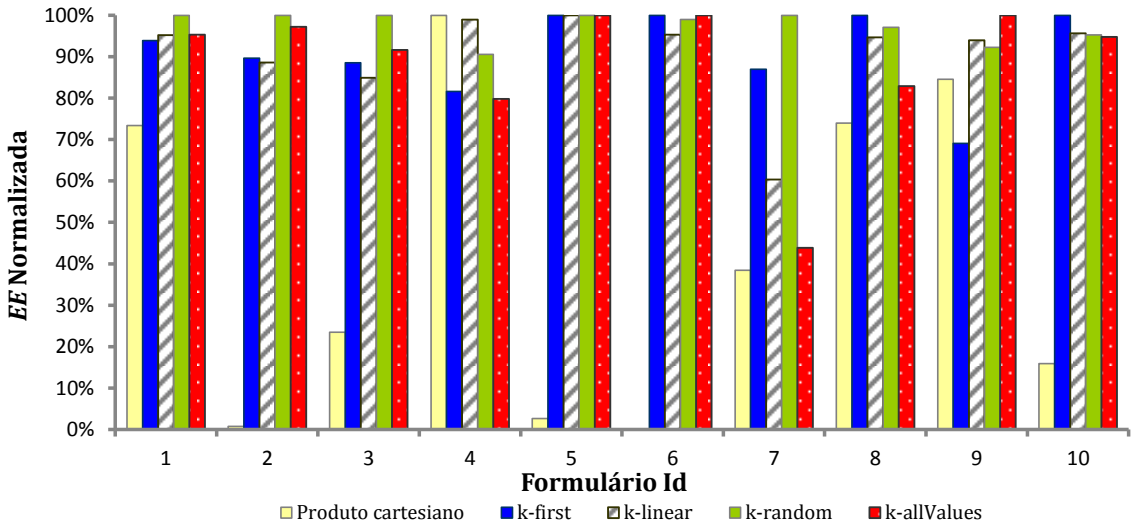

Figura 5.2: Eficiência de execução (EE) normalizada pelos melhores resultados obtidos em cada formulário da base de dados experimental 1

A análise comparativa das quatro estratégias tanto em cobertura quanto em eficiência de execução não permite definir que uma estratégia seja vantajosa em relação às outras. A técnica com melhores resultados varia de um formulário para outro e, muitas vezes, os resultados são muito semelhantes. A conclusão é de que se pode selecionar *k* valores com qualquer uma das 4 estratégias.

5.2 Resultados da Seleção de Valores Proposta

Nesta seção são mostrados os principais experimentos realizados. Eles buscam validar a arquitetura inteira proposta principalmente no que tange às ideias da seleção dinâmica apresentadas na Seção 4.3.

Como visto anteriormente, qualquer uma das quatro estratégias de seleção estática pode ser utilizada, pois obtiveram resultados semelhantes. Dessa forma, para os experimentos dessa seção foi considerado o método *k-first* como forma de seleção estática. A Tabela 5.4 apresenta a base de dados experimental 2, com os *sites* em que cada formulário

está presente. Também é mostrado o número de campos de texto e o número de campos de seleção de cada formulário. Novamente a base de dados é bastante heterogênea (assim como a base de dados experimental 1), possuindo formulários de domínio variado, bem como diferentes combinações de campos de texto e campos de seleção. Mais uma vez os valores do total da base por trás de cada formulário não são apresentados, pelo mesmo motivo explanado para a base de dados experimental 1.

Tabela 5.4: Propriedades dos formulários da base de dados experimental 2

Id	**Formulário *Web***	**Campos de texto**	**Campos de seleção**
1	http://www.foodandwine.com/search	3	1
2	http://www.global-standard.org	1	3
3	http://www.onlineraceresults.com/search/index.php	1	3
4	http://www.phillyfunguide.com	1	2
5	http://www.whoprofits.org	1	5
6	http://www.hcareers.com/seeker/search	1	5
7	http://www.rtbookreviews.com/rt-search/books	1	3
8	http://formovies.com/search/combined.html	3	0
9	http://www.careerbuilder.com	2	1
10	http://www.policechiefmagazine.org/magazine	1	2

O *baseline* adotado é o método de seleção de valores pela avaliação dos *templates* em informativos ou não (MADHAVAN et al., 2008), que aqui será também chamado de TP (*Template Pruning*). A comparação com o *baseline* foi feita com a execução de duas versões da arquitetura implementada. A primeira é com a avaliação das instâncias de *template* em informativas ou não, método ITP. E, a segunda versão consiste da avaliação da informação tanto dos *templates* como das instâncias de *template*, método ITTP. O *baseline* também necessita de uma forma de poda de k valores, bem como escolha dos parâmetros k e p. Para uma melhor comparação, as três execuções (TP, ITP e ITTP) foram realizadas com os mesmos parâmetros. A forma como os valores texto são escolhidos é a mesma para as três execuções. Isto é feito para que os métodos se diferenciem apenas na forma que selecionam as instâncias de *template* para submissão e assim se obtenha uma melhor comparação dos métodos de seleção de valores.

A Tabela 5.5 mostra os valores de instâncias de *template* submetidas durante todo o processo (*TS*) e retornadas por cada método (*TI*). Para o método ITP e ITTP o valor *TI* se refere a saída do módulo *Filtro de distinção* (Figura 4.2). Para o *baseline* as instâncias obtidas como saída do método (*TI*) são todas as instâncias de *template* dos *templates* informativos e distintas entre si (são subtraídas, por exemplo, as instâncias que retornam erro e páginas em branco).

Os resultados para as três métricas apresentadas, cobertura (*C*), eficiência de execução (*EE*) e eficiência de indexação (*EI*), com as três execuções realizadas (TP, ITP, ITTP) na base de dados experimental 2, podem ser vistos na Tabela 5.6. As Figuras 5.3, 5.4 e 5.5 apresentam graficamente esses resultados. Para uma melhor visualização na forma gráfica, os resultados foram normalizados. A normalização seguiu a mesma lógica dos resultados da Seção 5.1, ou seja, cada valor foi normalizado pelo método que obteve o melhor resultado para a métrica e para o formulário. O eixo x está relacionado com o Id de cada formulário apresentado na Tabela 5.4.

A análise dos resultados obtidos (Tabelas 5.5 e 5.6 e Figuras 5.3, 5.4 e 5.5) permite algumas conclusões. Pode-se perceber que o método ITP, possui uma cobertura superior

Tabela 5.5: *TS* e *TI* para a base de dados experimental 2

Id	***Baseline***		**ITP**		**ITTP**	
	TS	***TI***	***TS***	***TI***	***TS***	***TI***
1	3889	1601	6768	1288	4768	971
2	1865	70	4353	268	1153	58
3	4024	1361	3410	417	2210	417
4	2870	321	832	335	601	316
5	4106	119	8066	140	910	99
6	6397	488	12981	944	4981	572
7	4991	3998	4817	2197	4817	2197
8	150	39	1404	116	150	39
9	759	598	1971	798	759	536
10	147	64	118	12	79	12

Tabela 5.6: Cobertura, eficiência de execução e eficiência de indexação obtidos para a base de dados experimental 2

Id	***Baseline*(TP)**			**ITP**			**ITTP**		
	C	***EE***	***EI***	***C***	***EE***	***EI***	***C***	***EE***	***EI***
1	8690	**2,23**	5,43	**10824**	1,60	8,40	9992	2,10	**10,29**
2	511	0,27	7,30	**1271**	0,29	4,74	511	**0,44**	**8,81**
3	5284	1,31	3,88	**5372**	1,58	**12,88**	**5372**	**2,43**	**12,88**
4	2543	0,89	7,92	**2662**	3,20	7,95	2569	**4,27**	**8,13**
5	364	0,09	3,06	**449**	0,06	3,21	406	**0,45**	**4,10**
6	4549	0,71	9,32	**6832**	0,53	7,24	5404	**1,08**	**9,45**
7	10054	2,01	2,51	**10806**	**2,24**	**4,92**	**10806**	**2,24**	**4,92**
8	535	**3,57**	**13,72**	**1089**	0,78	9,39	535	**3,57**	**13,72**
9	10866	14,32	18,17	**14627**	7,42	18,33	11363	**14,97**	**21,20**
10	**2215**	15,07	34,61	**2215**	18,77	**184,58**	**2215**	**28,04**	**184,58**

aos outros métodos em todos os formulários. Isso se deve ao fato de possuir um espaço de busca maior mesmo com a poda estática de *k* valores por *template*. Esse espaço é maior pois a avaliação da informação dos *templates* não é realizada. Apesar de uma cobertura superior, o método apresenta um custo de execução elevado na maioria dos casos. Esses, custo e cobertura, refletem em uma eficiência de execução moderada. A *EE* é sempre menor comparada ao ITTP, e em metade dos casos menor que o *baseline*. Para a eficiência de indexação o método ITP se mostra um pouco melhor, obtendo resultados melhores que o *baseline* em sete dos dez formulários e, apesar de não obter uma *EI* melhor em nenhum formulário, obtém resultados mais próximos do método ITTP.

A intuição da combinação da poda de *templates* e instâncias de *template* realizada pelo método ITTP é de que a eficiência do método será alta. E, como se pode ver nos resultados, é isto o que ocorre. Ao submeter menos instâncias o custo é quase sempre menor ou igual ao TP e ITP. Isso só muda caso a lista de poda tenha modificado as instâncias de *template* submetidas de um *template* suficientemente para que esse *template* seja considerado informativo para o ITTP e não informativo para o TP. O que por sua vez implicará novas submissões em *templates* de ordem superior derivados desse, e assim mais instâncias são submetidas, gerando um custo maior em relação ao TP. Para os experimentos realizados isto ocorre de forma bastante visível no formulário 1. Apesar desse custo, a cobertura do ITTP é igual ou superior ao *baseline* em todos os casos. Assim, a

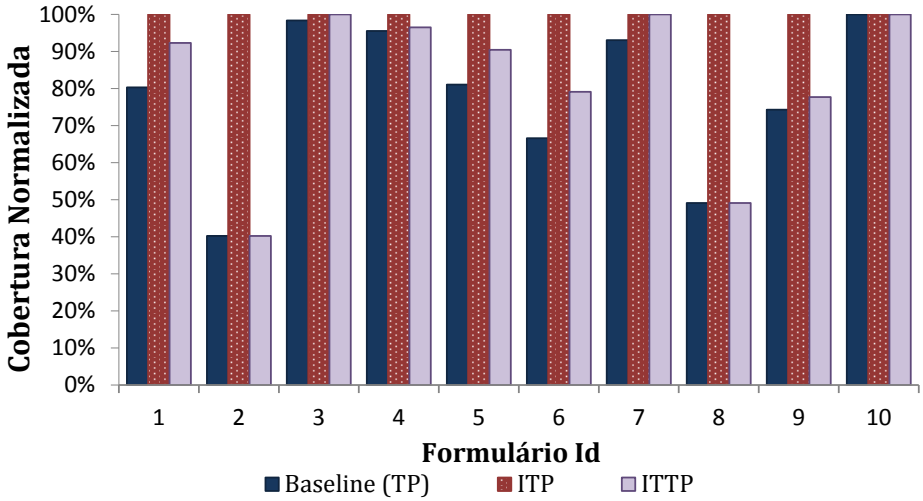

Figura 5.3: Cobertura normalizada pelos melhores resultados obtidos em cada formulário da base de dados experimental 2

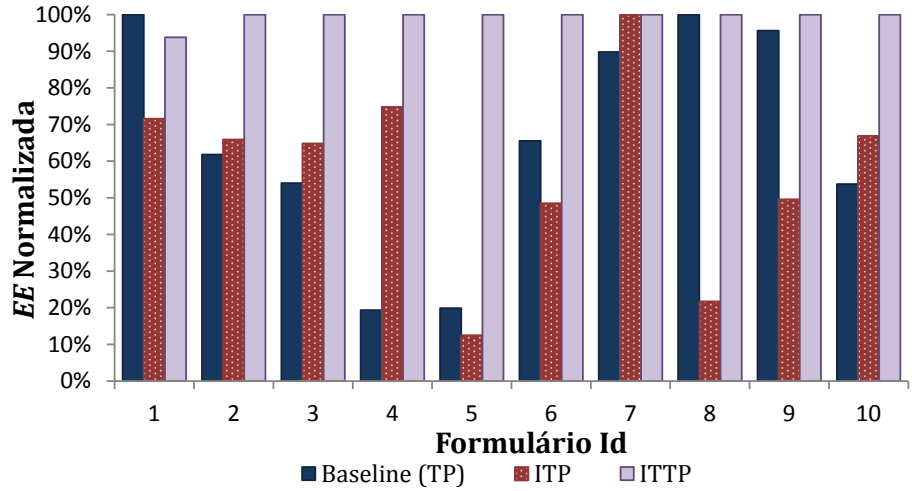

Figura 5.4: Eficiência de execução (*EE*) normalizada pelos melhores resultados obtidos em cada formulário da base de dados experimental 2

eficiência de execução do ITTP é a maior para a maioria dos formulários, com exceção do formulário 1, onde obtém *EE* maior que 90% do *baseline*. E, para a avaliação pela eficiência de indexação o método ITTP é sempre superior aos demais.

Os resultados sugerem que o método ITTP é melhor que o *baseline* e, que dependendo da aplicação desejada, pode-se utilizar o ITP ou ITTP. Uma aplicação com foco em uma maior cobertura e que não se importe com um custo maior poderá utilizar o método ITP. E outra aplicação que foca em uma maior eficiência poderá utilizar o método ITTP. Porém, uma melhor análise dos dados experimentais, como a feita pela Subseção 5.2.2 permite a conclusão que o método ITP é o melhor em ambos os sentidos, necessitando apenas de um ajuste no critério de parada.

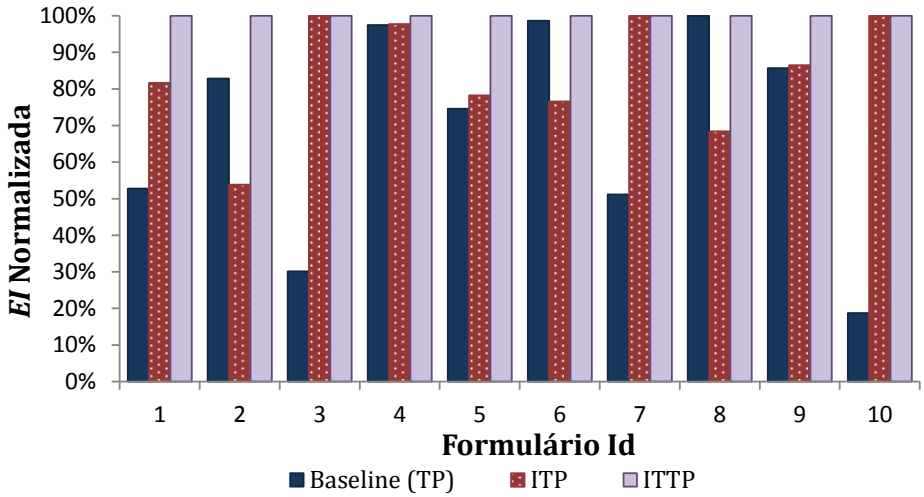

Figura 5.5: Eficiência de indexação (*EI*) normalizada pelos melhores resultados obtidos em cada formulário da base de dados experimental 2

5.2.1 Análise dos Critérios de Poda de Instâncias de *Template* do Método ITP

Para que o método ITP tenha sucesso é fundamental que a lista de poda seja suficientemente grande. Essa lista diz respeito ao quanto o método aprendeu sobre o formulário. Quanto mais ele tiver aprendido quais são as submissões indesejadas feitas a um formulário, melhores serão os resultados do método, pois outras instâncias de *template* indesejadas serão previamente identificadas e evitadas. Esse comportamento é mais evidente quando se identifica instâncias de *template* de ordem um como sendo não informativas. Por isso, uma avaliação das heurísticas que determinam a informatividade de uma instância de *template* é importante. Uma forma de efetuar essa avaliação é a identificação de quantas instâncias de *template* foram consideradas não informativas e entraram na lista de poda em cada *template*.

A Figura 5.6 mostra um gráfico com o percentual de instâncias de *template* consideradas não informativas nos *templates* de ordem 1 e 2 para cada formulário. Os dados de ordem 3 não são necessários pois não significam que futuras instâncias de *template* serão desconsideradas. Para cada métrica e formulário é mostrado o percentual de instâncias de *template* submetidas nos *templates* de cada ordem (1 e 2) que foram consideradas não informativas e entraram na lista de poda. Foram considerados os métodos ITP e ITTP, pois o TP não realiza poda de instâncias de *template*. Dessa forma, os dados são considerados sem *baseline*, pois a ideia é mostrar a validade das heurísticas em descartar instâncias de *template* e diminuir o espaço de busca.

Como esperado, o método ITP identifica um percentual maior de instâncias de *template* não informativas. Isso ocorre pois é esperado que *templates* não informativos tenham mais instâncias de *template* não informativas e o ITTP faz a poda dos *templates* não informativos, sendo desconsiderado esses *templates* para o cálculo do percentual de instâncias consideradas não informativas. Por isso também o gráfico não mostra valores para o percentual podado nos *templates* de ordem 2 para os formulários 2 e 8, pois não foram submetidos *templates* de ordem 2. Como pode-se ver, os resultados obtidos são bastante interessantes. Em alguns casos, como no formulário 10, as heurísticas implementadas foram suficientes para descartar a grande maioria das instâncias de *template*.

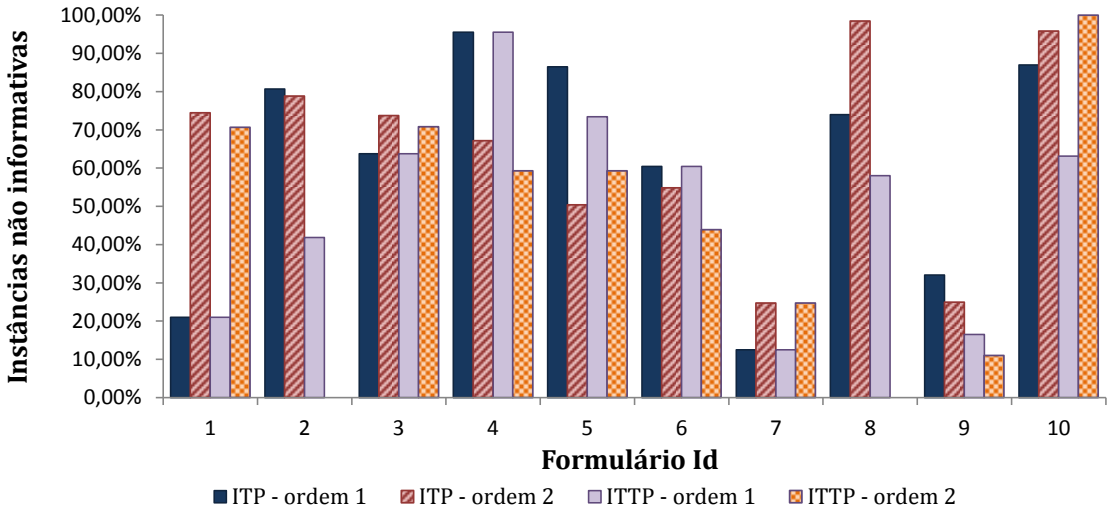

Figura 5.6: Percentual de instâncias de *template* classificadas como não informativas

Os formulários 2, 3, 4, 5 e 8 também tiveram destaque na poda das instâncias de ordem 1. Em média, se classificou como não informativa 61,3% das instâncias de ordem 1 e 64,3% das instâncias de ordem 2 para o método ITP. E para o método ITTP, em média se classificou como não informativa 50,6% das instâncias de ordem 1 e 35,8% das instâncias de *template* de ordem 2. Esses resultados são bastante significativos e explicam o sucesso dos métodos na comparação com o *baseline*. O elevado percentual obtido mesmo para o método ITTP, mostra que as heurísticas podem melhorar os resultados após a poda de *templates*.

5.2.2 Análise da Obtenção dos Dados da Base por Trás do Formulário

A análise dos experimentos realizados com as métricas utilizadas permitem diversas conclusões. Mas ainda deixam uma dúvida. De que maneira cada método busca os dados da base por trás do formulário à medida que as submissões vão sendo feitas? Esses dados podem ser buscados gradualmente conforme as submissões ao formulário vão sendo realizadas, ou de maneira aleatória devido a poucas submissões, concentradas ou não em um momento durante o processo de busca.

Para responder essa questão e realizar uma melhor avaliação dos experimentos, foi feito um gráfico que mostra o número de valores distintos obtidos até o momento de cada submissão, ou seja, cada ponto das curvas fornece a eficiência de execução após cada submissão. Foram gerados dez gráficos para os formulários da base de dados experimental 2, apresentados da Figura 5.7 até a Figura 5.16.

Como no início do processo nenhum dado da base foi obtido, a intuição é que as primeiras submissões colaboram com muitos dados distintos e que à medida que a base começa a ser mapeada os dados retornados são na sua maioria duplicatas, com poucos dados distintos sendo buscados. Até que em determinado momento o processo fique estagnado, sem obter novos distintos até o seu encerramento. Esse comportamento com algumas variantes é o que acontece nos gráficos obtidos. Existem os casos em que os métodos param de submeter antes de estagnar (como nos formulários da Figuras 5.13 e 5.15), o que sugere a ideia de que outras instâncias de *template* ainda poderiam retornar novos dados distintos. Isso ocorre normalmente para bases muito grandes mapeadas por formulários

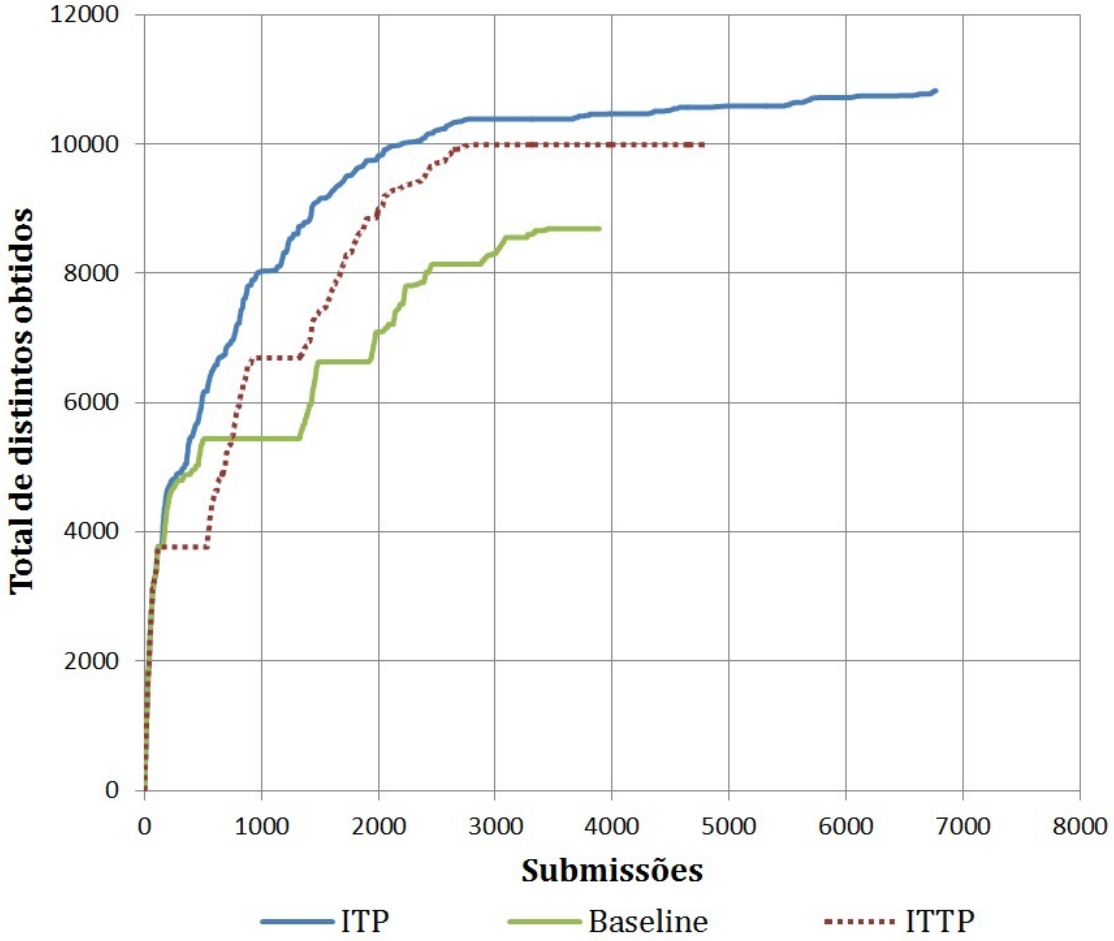

Figura 5.7: Dados distintos por submissões para o formulário 1 da base de dados experimental 2

simples. Porém na maioria dos formulários (Figuras 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.14 e 5.16) a curva amortece até parar de crescer, o que sugere que se chegou perto ou no total da base por trás do formulário.

A Figura 5.7 mostra o gráfico para o único formulário que o método ITTP teve custo de execução (TS) maior que o *baseline*. No momento que o *baseline* ultrapassa o ITTP em dados distintos é o mesmo momento que o método ITTP considera um *template* como não informativo e assim não obtém os valores contidos naquele *template* (o que não acontece com o método TP). Em outro momento o método TP considera outros dois *templates* como não informativos diferentemente do ITTP. Isso afeta na geração ou não de *templates* de ordem superior, o que faz que ao final o método ITTP tenha submetido mais instâncias apesar de ter se recuperado na cobertura. Por isso que a eficiência de execução do ITTP é menor que o *baseline* para esse formulário.

Quatro padrões diferentes foram detectados. Em alguns gráficos (como nas Figuras 5.13 e 5.9) os métodos ITTP e ITP traçam curvas semelhantes e em outros gráficos (como na Figura 5.8 e 5.14) os métodos ITTP e TP é que traçam curvas semelhantes. Para o primeiro padrão, a razão é que os distintos a mais descobertos são decorrentes da lista de poda, pois o método evita de submeter instâncias de *template* ruins e com isso fica aberto a novas possibilidades de submissão que são desconsideradas pelo método TP. Provavelmente essas novas possibilidades exploradas é que retornam esses dados. Já o

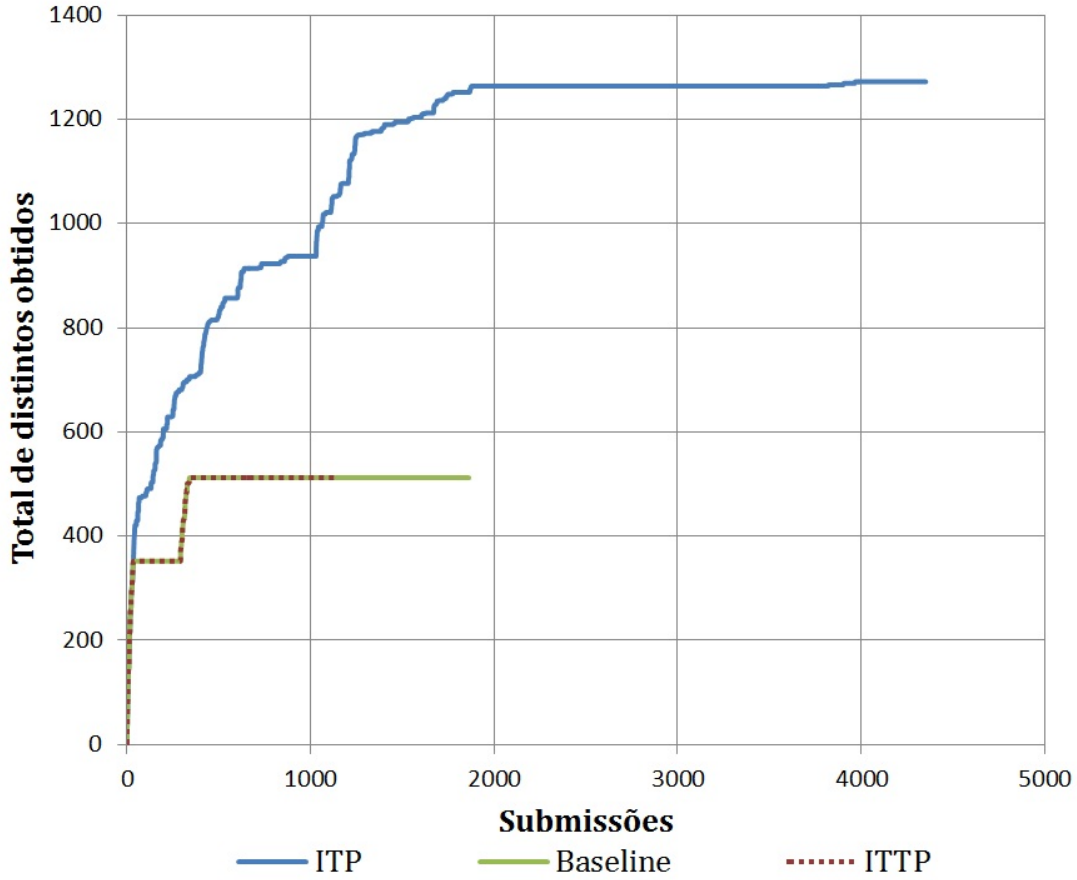

Figura 5.8: Dados distintos por submissões para o formulário 2 da base de dados experimental 2

segundo padrão ocorre pois os *templates* desconsiderados, tanto pelo ITTP como pelo TP, tinham sim alguma informação que poderia ser buscada. Quando os dois fatores ocorrem juntos, o método ITP tem a vantagem sobre os demais pelas duas circunstâncias descritas e o ITTP tem vantagem apenas pela primeira situação descrita, o que gera o terceiro padrão, onde a curva do método ITTP está entre as curvas dos métodos TP e ITP (como nos gráficos das Figuras 5.7 e 5.12). O último padrão ocorre quando os métodos obtém resultados praticamente iguais durante a execução (como no gráfico apresentado na Figura 5.16).

Outro aspecto interessante é que o método ITP tem clara vantagem sobre os outros métodos (ITTP e TP). Inclusive a busca em *templates* não informativos se provou útil. Na maioria dos formulários o método ITP rapidamente começa a adquirir mais distintos, se livrando rapidamente de pontos de estagnação (onde o processo fica sem buscar novos distintos por um tempo). Na pior das hipóteses se mantém com o mesmo desempenho que os outros métodos, normalmente para bases pequenas e no início do processo, pois para os *templates* de ordem 1 os métodos pouco se diferenciam.

O problema da eficiência do método ITP é claro também. Muitas vezes o método demora muito a parar, encerrando após diversas submissões que não retornam mais novos distintos (é o caso dos gráficos das Figuras 5.8, 5.9, 5.11, 5.12, 5.14 e 5.16). Esse fato sugere que um novo critério de parada para o método pode ser pensado. Ao invés de

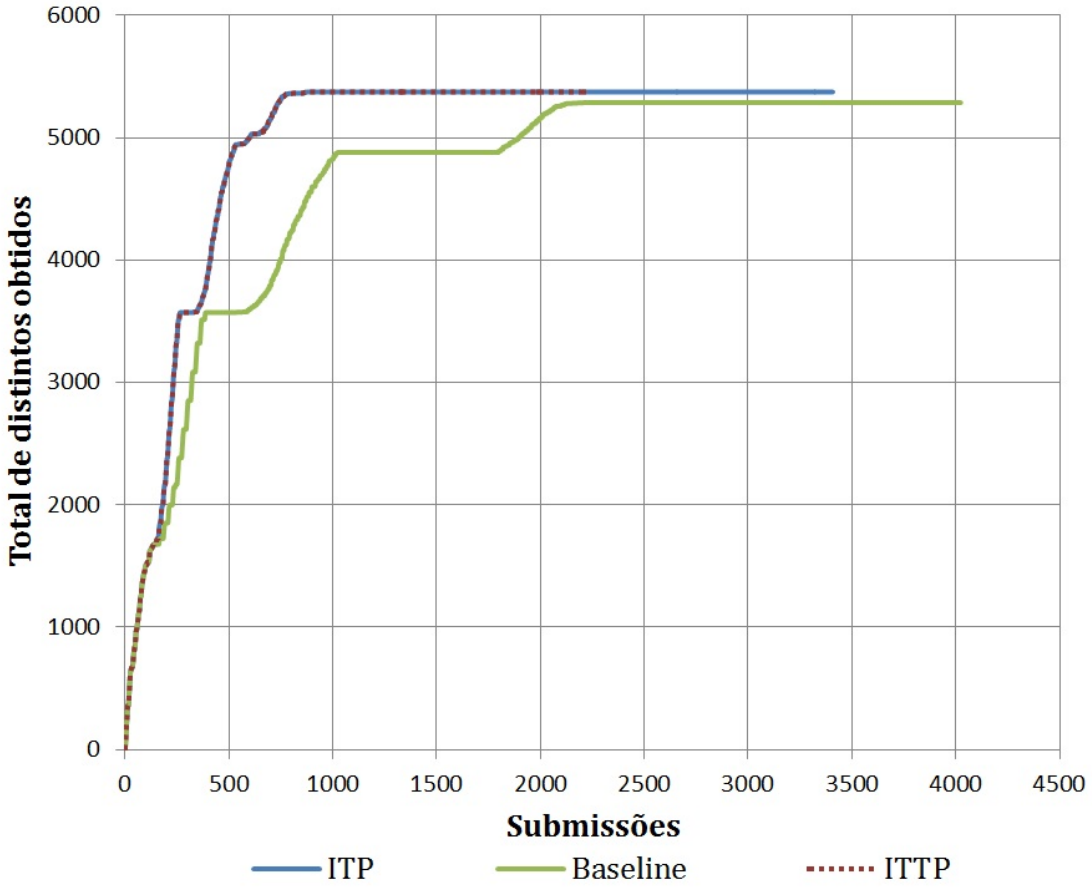

Figura 5.9: Dados distintos por submissões para o formulário 3 da base de dados experimental 2

se submeter *templates* até ordem 3 e para cada *template* se submeter no máximo k valores (MADHAVAN et al., 2008), se poderia pensar em um método dinâmico que analisasse a inclinação da curva apresentada nos gráficos. Isso fica claro pela análise dos gráficos, pois se o método ITP parasse sua execução sempre que a obtenção de novos dintintos se mostrasse estagnada por um determinado número de submissões, na maioria dos casos o método ITP obteria cobertura e eficiência melhores que as obtidas pelos outros métodos. Nos demais casos, o ITP pelo menos igualaria o desempenho dos outros métodos, ou seja, teria coberto mais com menos. Esse fato fortalece mais a importância da execução de podas de instâncias de *template* e não apenas de *templates*, que é a ideia central do presente trabalho.

O método ITP, além de obter resultados relevantes em comparação com o estado da arte, também possui a maioria das características descritas na Seção 3.2 para comparar os métodos de busca na *Web* oculta. A tabela 5.7 apresenta novamente a comparação dos métodos apresentados na Seção 3.1, sendo acrescentada mais uma linha com a avaliação do método ITP. Como se pode ver, o método só não trata do preenchimento e submissão de campos de texto tipado (característica TT).

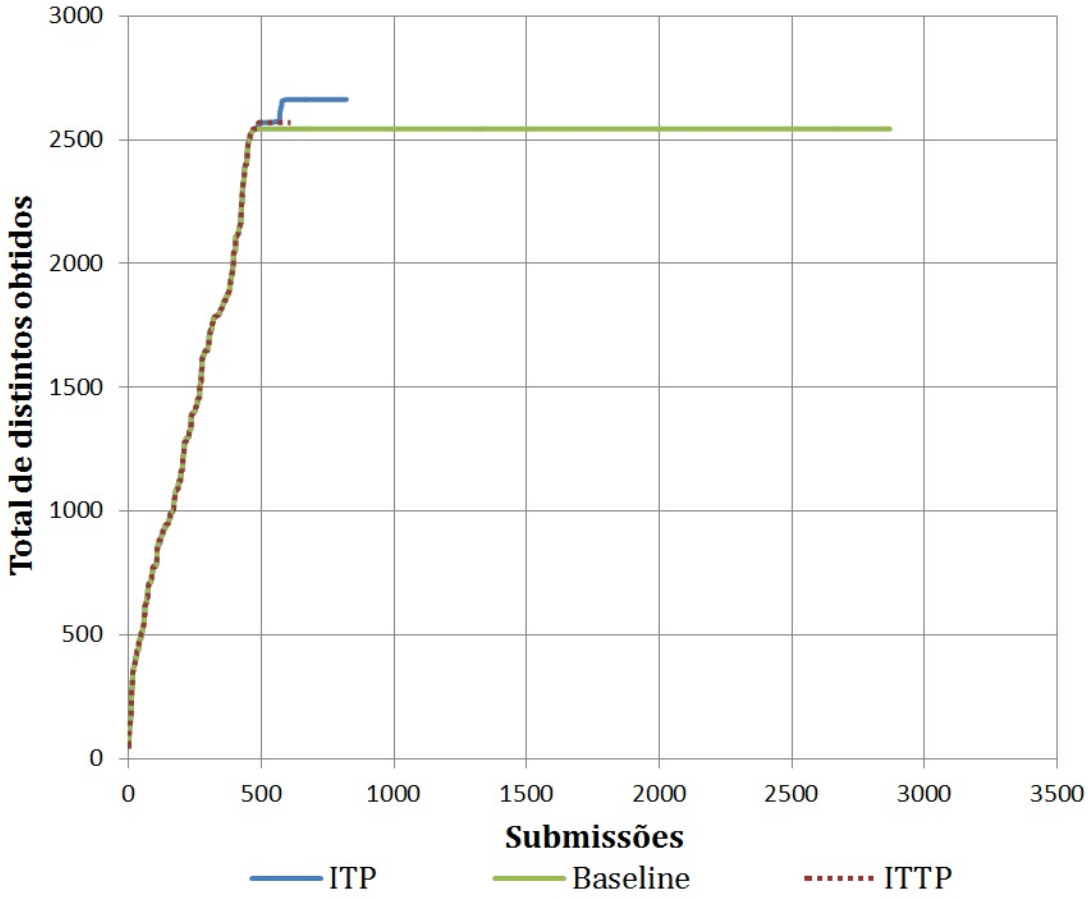

Figura 5.10: Dados distintos por submissões para o formulário 4 da base de dados experimental 2

Tabela 5.7: Comparação entre os métodos de preenchimento de formulário estudados e o ITP

Método	**Itens avaliados**							
	VC	**Se**	**TG**	**TT**	**DF**	**DC**	**Us**	**BC**
Raghavan	**sim**	**sim**	**sim**	(**sim**)	não	não	não	não
Liddle	**sim**	**sim**	não	não	**sim**	**sim**	não	não
Barbosa	não	não	**sim**	não	**sim**	**sim**	**sim**	não
Ntoulas	não	não	**sim**	não	**sim**	**sim**	não	não
Wu	não	(não)	**sim**	não	**sim**	**sim**	**sim**	não
Madhavan	**sim**	**sim**	**sim**	**sim**	**sim**	não	**sim**	não
Liu	**sim**	**sim**	**sim**	(**sim**)	não	não	não	**sim**
Jiang	não	não	**sim**	não	**sim**	**sim**	**sim**	**sim**
ITP	***sim***	***sim***	***sim***	*não*	***sim***	***sim***	***sim***	***sim***

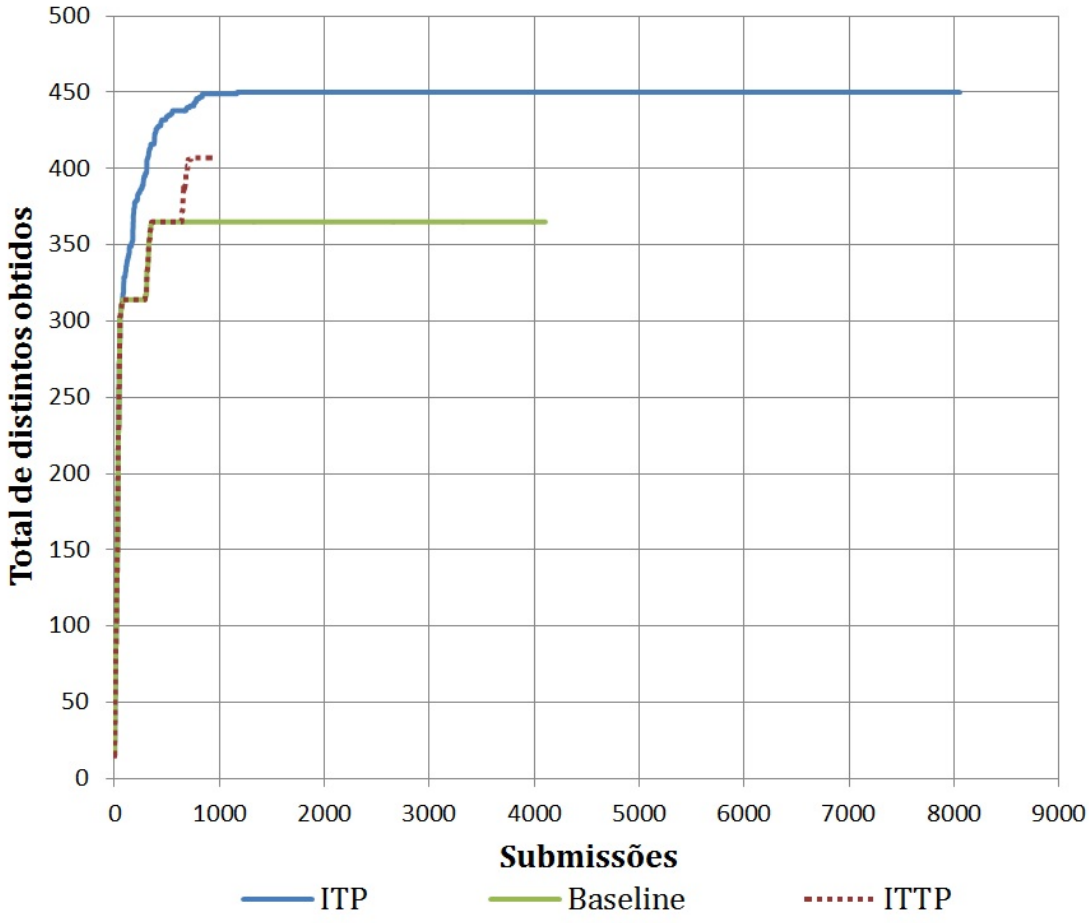

Figura 5.11: Dados distintos por submissões para o formulário 5 da base de dados experimental 2

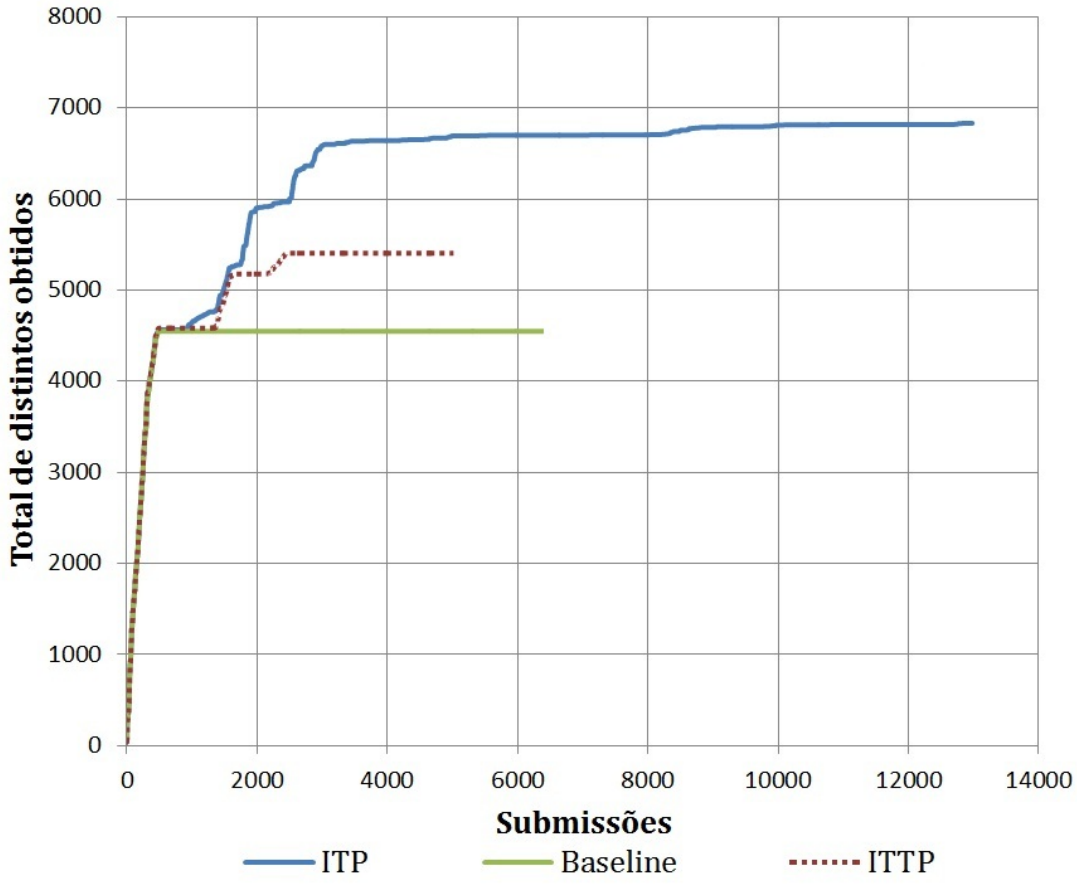

Figura 5.12: Dados distintos por submissões para o formulário 6 da base de dados experimental 2

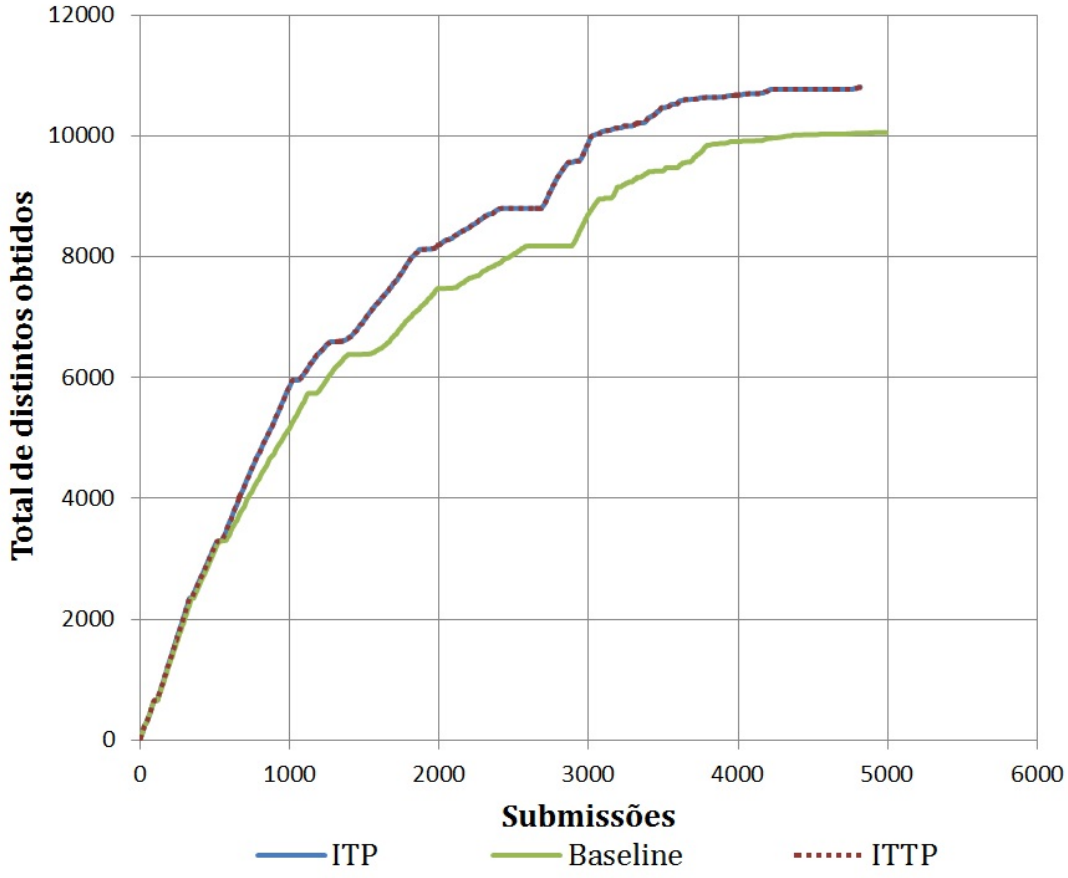

Figura 5.13: Dados distintos por submissões para o formulário 7 da base de dados experimental 2

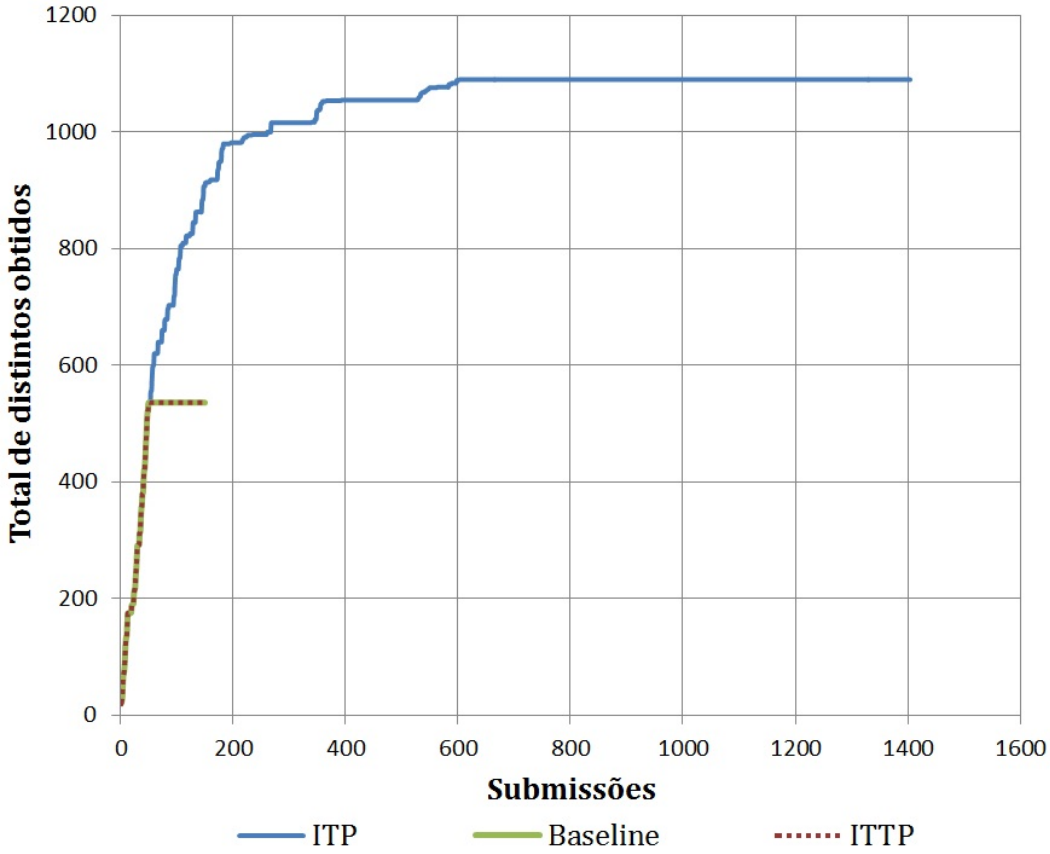

Figura 5.14: Dados distintos por submissões para o formulário 8 da base de dados experimental 2

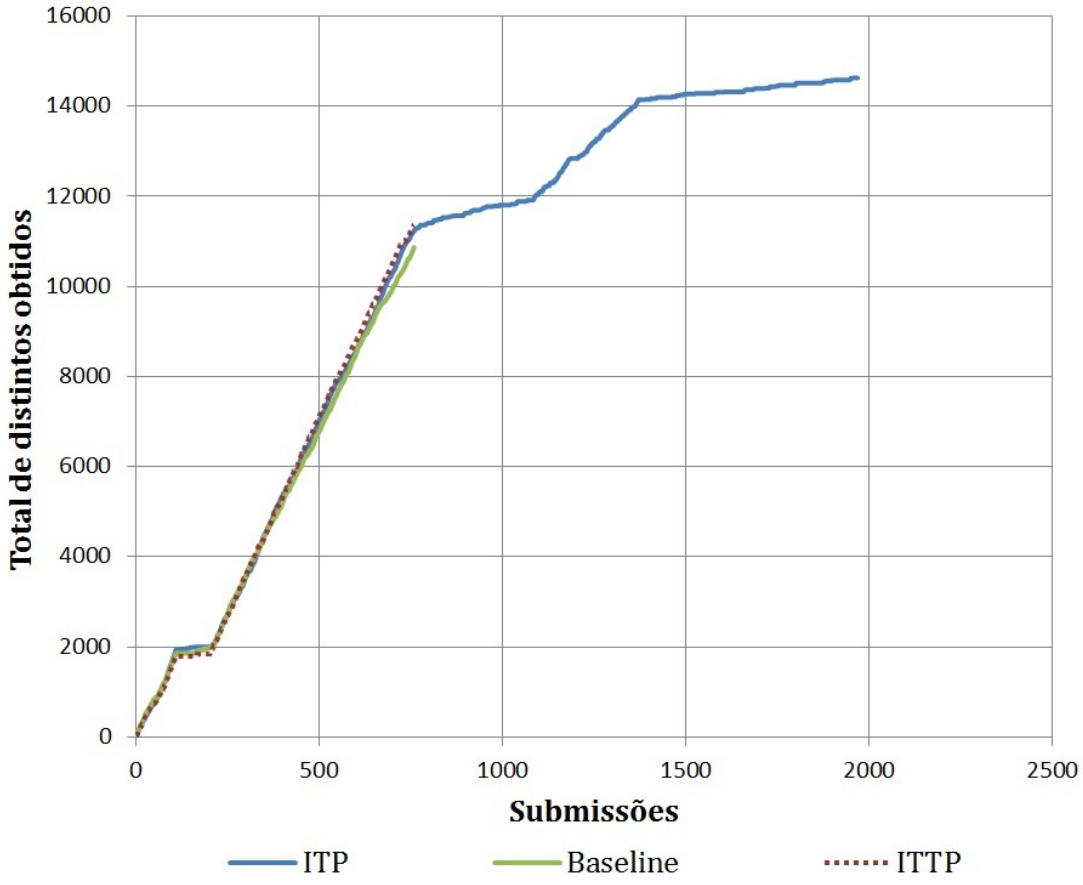

Figura 5.15: Dados distintos por submissões para o formulário 9 da base de dados experimental 2

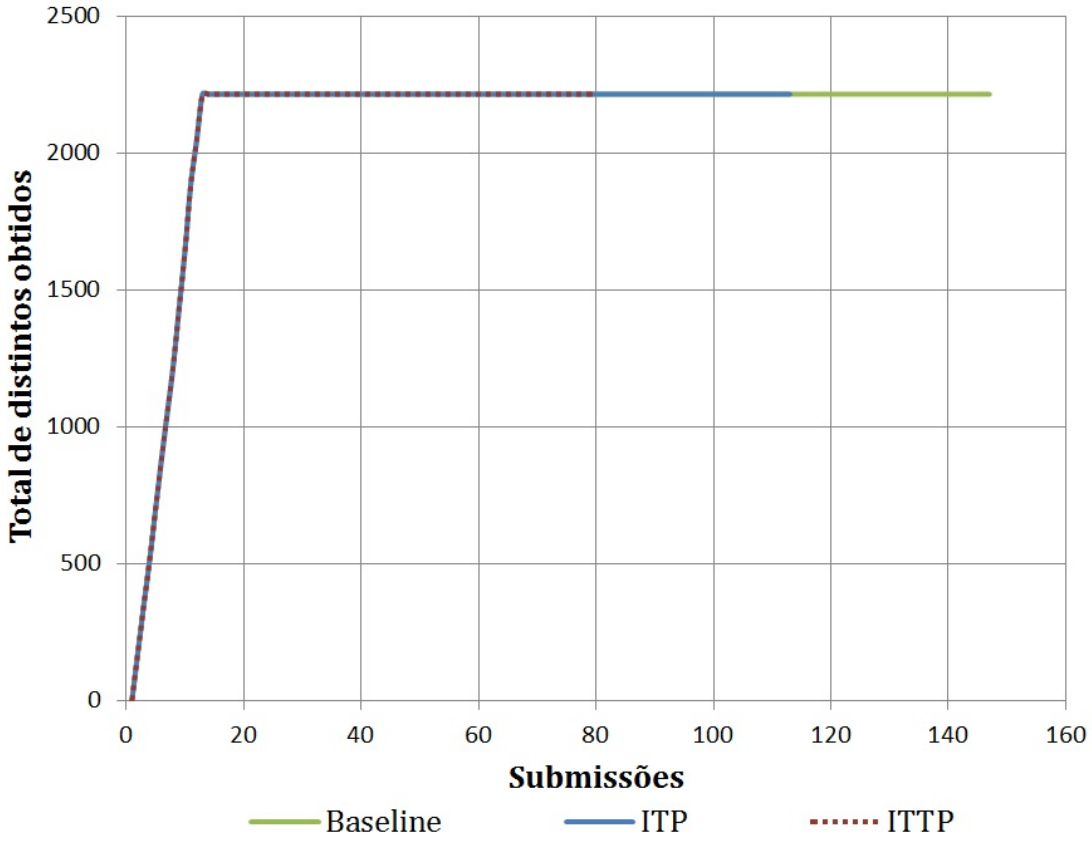

Figura 5.16: Dados distintos por submissões para o formulário 10 da base de dados experimental 2

6 CONCLUSÕES

A porção da *Web* apenas acessível através da submissão de formulários, como interfaces de busca, representa grande parte da informação estruturada disponível na *Web*. Essa parte da *Web* é chamada de *Web* oculta. Além de possuir um tamanho bem maior que a *Web* indexável através de *links*, a *Web* oculta possui conteúdo de melhor qualidade (BERGMAN, 2001). Contudo a busca nesta parte da *Web* oculta agrega um grande desafio e tem motivado um grande esforço de pesquisa. Os estudos e métodos apresentados estão longe de ser unanimidade e a ciência e busca na *Web* oculta estão na sua "infância" (OLSTON; NAJORK, 2010).

O grande problema é o processamento e a correta submissão dos formulários, pois esses consistem de interfaces desenvolvidas para o entendimento humano, apresentando diversas variáveis e ampla diversidade. Os trabalhos estudados (Seção 3.1) que propõem métodos para a busca na *Web* oculta apresentam algumas limitações que dificultam a busca automática pelo conteúdo da *Web* oculta de forma horizontal (na maioria das páginas). Alguns necessitam da intervenção do usuário (RAGHAVAN; GARCIA-MOLINA, 2000; LIDDLE et al., 2003; NTOULAS; ZERFOS; CHO, 2005; LIU et al., 2009). Outros tratam apenas de campos de um tipo, como de texto genérico (BARBOSA; FREIRE, 2004; NTOULAS; ZERFOS; CHO, 2005; JIANG et al., 2009) ou de seleção (LIDDLE et al., 2003). Existem métodos que necessitam de conhecimento prévio, como informações de domínio (RAGHAVAN; GARCIA-MOLINA, 2000; MADHAVAN et al., 2008; LIU et al., 2009). Há ainda os que consideram a submissão de formulários ou preenchendo um campo por vez, sem combiná-los (BARBOSA; FREIRE, 2004; NTOULAS; ZERFOS; CHO, 2005; WU et al., 2006; JIANG et al., 2009), ou considerando sempre todos os campos para a submissão (RAGHAVAN; GARCIA-MOLINA, 2000), poucos trabalhos consideram todas possibilidades de preenchimento. Desses métodos, o que apresenta a melhor solução é o de (MADHAVAN et al., 2008) e é por isso que esse método foi escolhido como *baseline*.

De forma a contribuir com a pesquisa na área, o presente trabalho debate a forma como os valores para submissão (instâncias de *template*) podem ser escolhidos. Primeiramente, foi definida uma arquitetura (Seção 4.2) para busca na *Web* oculta. Ela recebe como parâmetro um formulário e extrai os campos e valores, após ela executa as primeiras submissões e gera valores para campos texto quando for necessário, por fim ela retorna todas instâncias de *template* a serem indexadas, que retornam todos os registros distintos obtidos. A arquitetura é genérica e pode ser estendida facilmente. Dentro dessa arquitetura, no módulo de *Geração, submissão e avaliação das instâncias de template* foi discutido as principais contribuições do trabalho.

Primeiramente, são apresentadas quatro formas de se escolher um subgrupo de instâncias de um mesmo *template*, chamada de seleção estática (Seção 4.4). Os experimentos

mostraram a cobertura e eficiência de execução em 10 formulários. As técnicas apresentadas (*k-first*, *k-random*, *k-linear*, *k-all values*) obtiveram desempenho semelhante, e, portanto qualquer uma pode ser utilizada. Os estudos sobre a seleção estática foram apresentados no SBBD 2011 (KANTORSKI; MORAES; HEUSER, 2012, 2011).

A principal contribuição do trabalho consiste na forma de seleção dinâmica apresentada, o método ITP. Esse método consiste da avaliação de todas as instâncias de *template* submetidas, elas são classificadas em informativas ou não informativas. Uma instância de *template* é considerada não informativa caso a página retorne poucos dados, nenhum resultado ou uma página de erro. O método aprende informações importantes à medida que identifica instâncias não informativas. Esse conhecimento obtido permite que o método evite submeter uma série de instâncias também não informativas, ou seja, indesejadas. Os estudos sobre a seleção dinâmica foram apresentados na conferência ADBIS 2012 (KANTORSKI et al., 2012).

O método ITP pode ser agregado em diversos métodos, como foi feito com o método de Madhavan. Para os experimentos se considerou os seguintes métodos: ITP, que avalia as instâncias de *template*, o método ITTP, que avalia tanto os *templates* como as instâncias de *template* e o método TP (MADHAVAN et al., 2008) que avalia apenas os *templates*. Foram realizados experimentos em dez formulários. Nos resultados obtidos para a cobertura, os dois métodos apresentados (ITP e ITTP) foram melhores que o *baseline* (TP), principalmente o método ITP. Na eficiência de execução o método ITTP foi o melhor, mas o *baseline* levou ligeira vantagem sobre o ITP. E, na eficiência de indexação o método ITTP novamente foi o melhor deixando o ITP em segundo lugar.

As heurísticas utilizadas para avaliar uma instância de *template* foram consideradas boas, pois os experimentos mostraram que para *templates* de ordem um e dois, em torno de 60% das instâncias foram consideradas não informativas, para o método ITP. Já para o ITTP o percentual obtido foi 45%. Esses dados sugerem que essas heurísticas são suficientes para se decidir a informatividade de uma instância de *template*. Os métodos têm um nível alto de aprendizado com essas heurísticas, conseguindo prever e evitar de submeter um grande número de instâncias de *template*.

A análise da cobertura obtida pelo número de submissões mostrou que o método ITP foi sempre melhor, buscando mais rápido os dados da base. A principal lacuna do ITP é o seu custo, porém esse pode ser evitado. É necessário um critério de parada que identifique quando o método atinge uma cobertura tal que a maioria dos dados retornados são duplicatas. Tal critério de parada iria tornar o método melhor que os outros métodos em todas as métricas apresentadas. Cobrindo um percentual da base maior com um número de submissões menor. A obtenção de um critério de parada que considere esse amortecimento na taxa de obtenção de novos distintos é deixado para um possível trabalho futuro.

Outros possíveis trabalhos futuros são o adequado tratamento de todos os campos de texto. O correto preenchimento de campos de texto tipados constitui um enorme desafio de pesquisa, as poucas soluções existentes recaem em conhecimento prévio a cerca de alguns domínios. Essas soluções são limitadas para a busca na *Web* de forma horizontal.

O relacionamento entre os campos também apresentam inúmeros desafios para a pesquisa da *Web* oculta. A identificação de campos obrigatórios dificilmente é abordada, porém pode ser feita, por exemplo, através da análise de poucas submissões ao formulário de diferentes *templates* antes do processo. Já a identificação e tratamento de campos correlacionados (por exemplo: “preço máximo” e “preço mínimo” ou “data de partida” e “data de chegada”) consiste de uma tarefa difícil e ainda sem solução nas buscas horizontais.

Métodos de **extração de dados estruturados** e principalmente de **entendimento de formulários** podem ser utilizados para a criação de técnicas que proponham soluções para as lacunas de pesquisa apresentadas. Com isto, o aperfeiçoamento desses métodos é vital para o desenvolvimento e aprimoramento dos motores de busca na *Web* oculta.

REFERÊNCIAS

ALI, A.; MEEK, C. **Predictive models of form filling**. [S.l.]: Microsoft Research, Tech. Rep. MSR-TR-2009-1, 2009.

ÁLVAREZ, M. et al. Crawling the content hidden behind web forms. **Computational Science and Its Applications–ICCSA 2007**, [S.l.], p.322–333, 2007.

BARBOSA, L.; FREIRE, J. Siphoning hidden-web data through keyword-based interfaces. In: XIX BRAZILIAN SYMPOSIUM ON DATABASES. **Proceedings...** [S.l.: s.n.], 2004. p.309–321.

BARBOSA, L.; FREIRE, J. An adaptive crawler for locating hidden-web entry points. In: WORLD WIDE WEB, 16. **Proceedings...** [S.l.: s.n.], 2007. p.441–450.

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

CALLAN, J.; CONNELL, M. Query-based sampling of text databases. **ACM Transactions on Information Systems (TOIS)**, [S.l.], v.19, n.2, p.97–130, 2001.

CHANG, C. et al. A survey of web information extraction systems. **IEEE transactions on knowledge and data engineering**, [S.l.], v.18, n.10, p.1411, 2006.

CHANG, K. et al. Structured databases on the web: observations and implications. **ACM SIGMOD Record**, [S.l.], v.33, n.3, p.61–70, 2004.

FLORESCU, D.; LEVY, A.; MENDELZON, A. Database techniques for the World-Wide Web: a survey. **SIGMOD Rec.**, New York, NY, USA, v.27, p.59–74, September 1998.

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

HONG, J.; SIEW, E.; EGERTON, S. Information extraction for search engines using fast heuristic techniques. **Data & Knowledge Engineering**, [S.l.], v.69, n.2, p.169–196, 2010.

JACOBS, I. **URIs, Addressability, and the use of HTTP GET and POST**. Disponível em: <http://www.w3.org/2001/tag/doc/whenToUseGet.html>. Acessado em 10 de maio 2013.

JIANG, L. et al. Learning Deep Web Crawling with Diverse Features. In: IEEE/WIC/ACM INTERNATIONAL JOINT CONFERENCE ON WEB INTELLIGENCE AND INTELLIGENT AGENT TECHNOLOGY-VOLUME 01, 2009. **Proceedings...** [S.l.: s.n.], 2009. p.572–575.

JIANG, L. et al. Efficient deep web crawling using reinforcement learning. **Advances in Knowledge Discovery and Data Mining**, [S.l.], p.428–439, 2010.

KANTORSKI, G.; MORAES, T.; HEUSER, C. Seleção de Valores para Preenchimento Automático de Formulários Web. In: XXVI SBBD. **Proceedings...** [S.l.: s.n.], 2011. p.139–146.

KANTORSKI, G.; MORAES, T.; HEUSER, C. **Strategies for Automatically Filling Web Forms**. [S.l.]: UFRGS - Instituto de Informática, 2012. Technical Report. (RP367-PPGC).

KANTORSKI, G. Z. et al. Choosing Values for Text Fields in Web Forms. In: ADVANCES IN DATABASES AND INFORMATION SYSTEMS. **Anais...** [S.l.: s.n.], 2012. v.186, p.125.

KHARE, R.; AN, Y.; SONG, I. Understanding deep web search interfaces: a survey. **ACM SIGMOD Record**, [S.l.], v.39, n.1, p.33–40, 2010.

KRISTJANSSON, T. et al. Interactive information extraction with constrained conditional random fields. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE. **Proceedings...** [S.l.: s.n.], 2004. p.412–418.

LAWRENCE, S.; GILES, C. Searching the world wide web. **Science**, [S.l.], v.280, n.5360, p.98–100, 1998.

LI, L.; LIU, Y.; OBREGON, A. Visual segmentation-based data record extraction from web documents. In: INFORMATION REUSE AND INTEGRATION, 2007. IRI 2007. IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2007. p.502–507.

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

LIU, B.; GROSSMAN, R.; ZHAI, Y. Mining data records in Web pages. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING. **Proceedings...** [S.l.: s.n.], 2003. p.601–606.

LIU, J. et al. Crawling deep web content through query forms. **Proceedings of WEBIST2009, Lisbon Portugal**, [S.l.], p.634–642, 2009.

MADHAVAN, J. et al. Web-scale data integration: you can only afford to pay as you go. In: CIDR. **Proceedings...** [S.l.: s.n.], 2007. p.342–350.

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

MADHAVAN, J. et al. Harnessing the Deep Web: present and future. **arXiv preprint arXiv:0909.1785**, [S.l.], 2009.

MADHAVAN, J. et al. **Determining keyword for a form page**. US Patent 20,130,031,083.

MADHAVAN, J.; HALEVY, A.; KO, D. **Analyzing a Form Page for Indexing**. US Patent 20,130,031,503.

MORAES, M. C. et al. Pre-Query Discovery of Domain-specific Query Forms: a survey. **IEEE Transactions on Knowledge and Data Engineering**, [S.l.], 2012.

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

NTOULAS, A.; ZERFOS, P.; CHO, J. Downloading hidden web content. **Technicalreport, UCLA**, [S.l.], 2004.

NTOULAS, A.; ZERFOS, P.; CHO, J. Downloading textual hidden web content through keyword queries. In: ACM/IEEE-CS JOINT CONFERENCE ON DIGITAL LIBRARIES, 5. **Proceedings...** [S.l.: s.n.], 2005. p.100–109.

OLSTON, C.; NAJORK, M. Web crawling. **Foundations and Trends in Information Retrieval**, [S.l.], v.4, n.3, p.175–246, 2010.

RAGHAVAN, S.; GARCIA-MOLINA, H. **Crawling the Hidden Web**. [S.l.]: Stanford InfoLab, 2000. Technical Report. (2000-36).

RAGHAVAN, S.; GARCIA-MOLINA, H. Crawling the hidden web. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES. **Proceedings...** [S.l.: s.n.], 2001. p.129–138.

RAJARAMAN, A.; SAGIV, Y.; ULLMAN, J. Answering queries using templates with binding patterns. In: ACM SIGACT-SIGMOD-SIGART SYMPOSIUM ON PRINCIPLES OF DATABASE SYSTEMS. **Proceedings...** [S.l.: s.n.], 1995. p.105–112.

SALTON, G.; MCGILL, M. J. **Introduction to modern information retrieval**. [S.l.]: McGraw-Hill College, 1983.

SHESTAKOV, D.; BHOWMICK, S.; LIM, E. DEQUE: querying the deep web. **Data & Knowledge Engineering**, [S.l.], v.52, n.3, p.273–311, 2005.

Tjin-Kam-Jet, K. T. T. E.; Trieschnigg, R. B.; Hiemstra, D. **Deep web search**: an overview and roadmap. Enschede: Centre for Telematics and Information Technology, University of Twente, 2011. Technical Report. (TR-CTIT-12-32).

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

WANG, Y.; LU, J.; CHEN, J. Crawling deep web using a new set covering algorithm. **Advanced Data Mining and Applications**, [S.l.], p.326–337, 2009.

WU, P. et al. Query selection techniques for efficient crawling of structured web sources. In: DATA ENGINEERING, 2006. ICDE'06. PROCEEDINGS OF THE 22ND INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2006. p.47–47.

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

YAMAMOTO, M.; CHURCH, K. W. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. **Computational Linguistics**, [S.l.], v.27, n.1, p.1–30, 2001.

ZHAI, Y.; LIU, B. Web data extraction based on partial tree alignment. In: WORLD WIDE WEB, 14. **Proceedings...** [S.l.: s.n.], 2005. p.76–85.

ZHANG, Z.; HE, B.; CHANG, K. Understanding web query interfaces: best-effort parsing with hidden syntax. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.107–118.