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

MAURÍCIO COUTINHO MORAES

Towards Completely Automatized HTML Form Discovery on the Web

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Prof. Dr. Carlos Alberto Heuser
Advisor

Profa. Dra. Viviane Pereira Moreira
Coadvisor

Porto Alegre, March 2013

CIP – CATALOGING-IN-PUBLICATION

Moraes, Maurício Coutinho

Towards Completely Automatized HTML Form Discovery on the Web / Maurício Coutinho Moraes. – Porto Alegre: PPGC da UFRGS, 2013.

89 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2013. Advisor: Carlos Alberto Heuser; Coadvisor: Viviane Pereira Moreira.

1. Deep Web. 2. Hidden Web. 3. Crawling. 4. Domain-specific search. 5. Query form discovery. I. Heuser, Carlos Alberto. II. Moreira, Viviane Pereira. III. Título.

“If I Had More Time,
I Would Have Written You a Shorter Letter.”
— Blaise Pascal

AGRADECIMENTOS

Em primeiro lugar, à minha esposa, Fernanda, pelo apoio irrestrito. Em segundo, aos meus orientadores e ao Denilson Barbosa pelo voto de confiança que me deram ao me aceitarem em seus respectivos grupos de pesquisa quando eu ainda não tinha nada de concreto para apresentar-lhes. Por último, ao Hugo Pinto pela contribuição bastante pontual porém fundamental para a conclusão desta tese.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

DW	Deep Web
HTML	Hypertext Markup Language
HTTP	Hypertext Tranfer Protocol
PIW	Publicly Indexable Web
URL	Uniform Resource Locator

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The discovery of HTML forms is one of the main challenges in Deep Web crawling. Automatic solutions for this problem perform two main tasks. The first is locating HTML forms on the Web, which is done through the use of traditional/focused crawlers. The second is identifying which of these forms are indeed meant for querying, which also typically involves determining a domain for the underlying data source (and thus for the form as well). This problem has attracted a great deal of interest, resulting in a long list of algorithms and techniques. Some methods submit requests through the forms and then analyze the data retrieved in response, typically requiring a great deal of knowledge about the domain as well as semantic processing. Others do not employ form submission, to avoid such difficulties, although some techniques rely to some extent on semantics and domain knowledge. We offer an up-to-date review of 19 methods for the discovery of domain-specific query forms that do not involve form submission. This thesis details these methods and discusses how form discovery has become increasingly more automated over time, providing the context in which we propose a novel method to advance the current state-of-the-art in domain-specific structured HTML form discovery. The current state-of-the-art in domain-specific structured HTML form discovery consists mainly of methods that directly or indirectly depend heavily on human intervention. This thesis proposes and evaluates a method capable of discovering domain-specific structured HTML forms on the Web with very little effort from a human expert, who is required only to define the name of the domain of interest (i.e., the domain for which the discovery should be made). The forms discovered by our proposal can be directly used as training data by some form classifiers. Our experimental validation used thousands of real Web forms, divided into six domains, including a representative subset of the publicly available DeepPeep form base (DEEPPEEP, 2010; DEEPPEEP REPOSITORY, 2011). Our results show that it is feasible to mitigate the demanding manual work required by two cutting-edge form classifiers (i.e., GFC and DSFC (BARBOSA; FREIRE, 2007a)), at the cost of a relatively small loss in effectiveness.

Keywords: Deep Web, Hidden Web, Crawling, Domain-specific search, Query form discovery.

1 INTRODUCTION

This chapter provides the background necessary to develop our thesis in the following chapters. It offers a succinct itemization of our contributions and some fundamental concepts that have to be presented before proposing our contribution and discussing the consequences of it.

1.1 What is this thesis about?

This thesis is about Web form discovery. We present a survey that characterizes the current state-of-the-art in form discovery, exposing its deficiencies related to the demanding manual work required from human experts. Based on our survey, we propose a novel heuristic-based method for pre-query discovery of domain-specific structured forms on the Web.

Our final goal while proposing our method is to mitigate the demanding manual work required by the current state-of-the-art in form discovery through automatic training of form classifiers. We present substantial empirical evidence that it is feasible to automatically train at least two state-of-the-art form classifiers, at the cost of a relatively small loss in effectiveness.

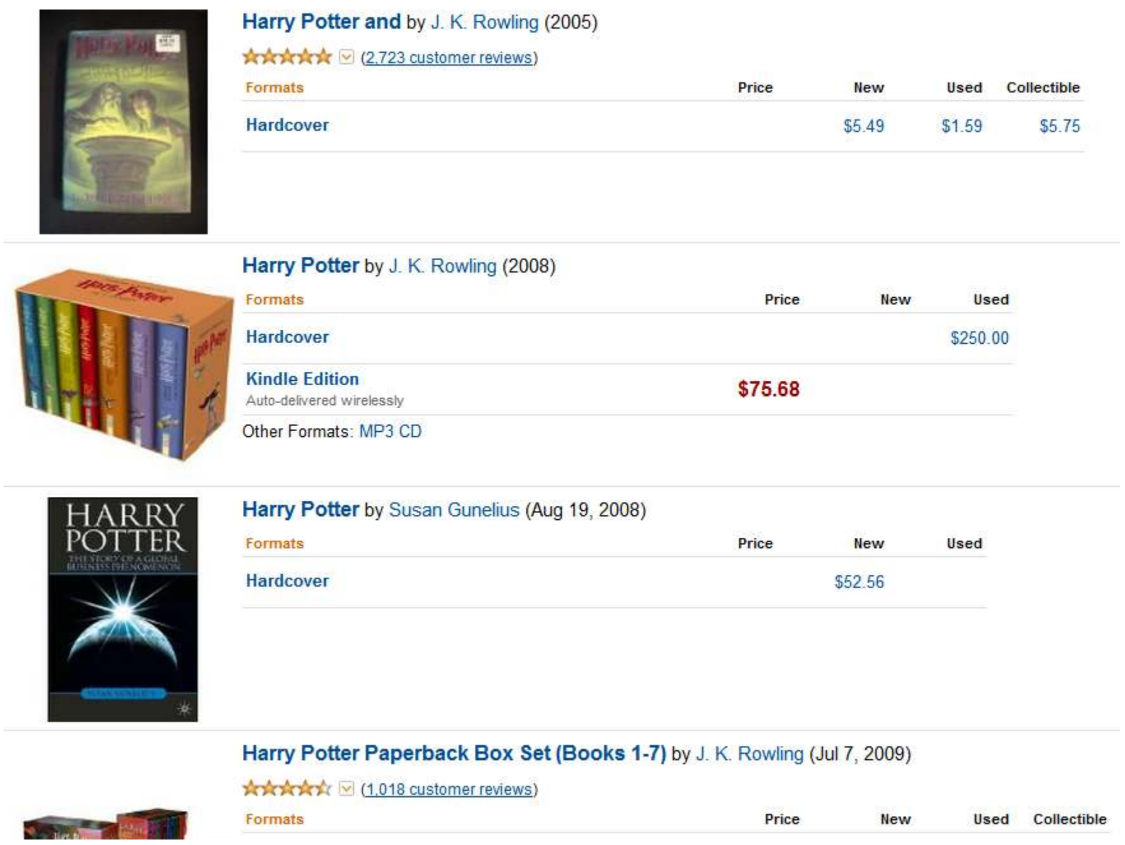
Our hypothesis is that it is possible to perform domain-specific form discovery based on implicit or tacit form construction rules. We believe that fully automatic form discovery can be achieved by means of: (i) interacting with a generic search engine in order to locate forms; and (ii) applying simple heuristics in order to identify forms.

1.2 Web Forms

A Web form is the HTML component defined by the `<form>` tag. Its main purpose is to allow the submission of manually defined data to an HTTP server through a Web browser.

There is a wealth of highly specialized data on the Web that is accessible only by Web form submission. This data covers many different domains or subjects, which can be as disparate as, for example, data about books, cars, and jobs.

To be able to access data "hidden behind" forms is necessary to know the exact location (i.e. the URL) of at least one Web page that contains a form. By doing this, the page can be reached through a Web browser, the form can be manually filled out with meaningful and correct input and then submitted. As an illustration of how data is "hidden behind" forms, Figure 1.1 shows sample data from the `Books` domain that was reached by filling out and submitting the form depicted on Figure 1.2.



Harry Potter and by J. K. Rowling (2005)

(2,723 customer reviews)

Formats	Price	New	Used	Collectible
Hardcover		$5.49	$1.59	$5.75

Harry Potter by J. K. Rowling (2008)

Formats	Price	New	Used
Hardcover			$250.00
Kindle Edition Auto-delivered wirelessly	$75.68		

Other Formats: MP3 CD

Harry Potter by Susan Gunelius (Aug 19, 2008)

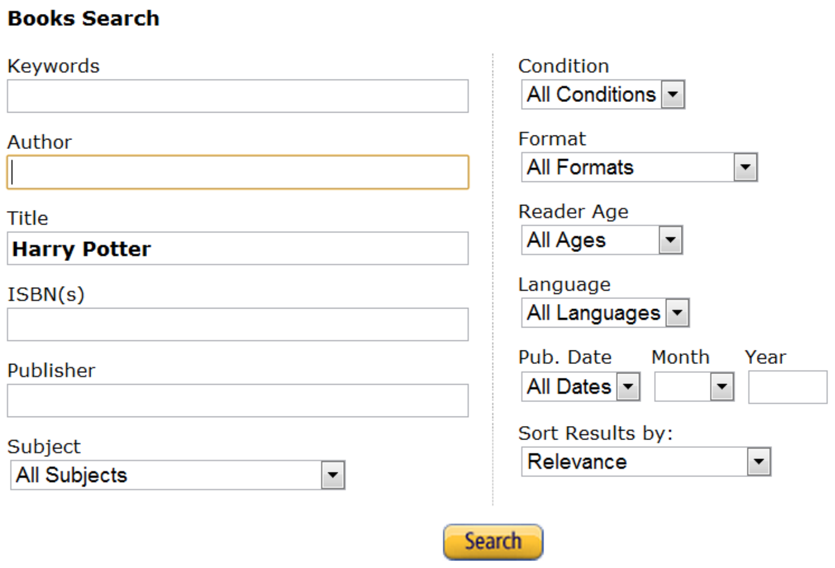
Formats	Price	New	Used
Hardcover		$52.56	

Harry Potter Paperback Box Set (Books 1-7) by J. K. Rowling (Jul 7, 2009)

(1,018 customer reviews)

Formats	Price	New	Used	Collectible

Figure 1.1: Sample data from the Books domain.



Books Search

Keywords	
Author	
Title	Harry Potter
ISBN(s)	
Publisher	
Subject	All Subjects
Condition	All Conditions
Format	All Formats
Reader Age	All Ages
Language	All Languages
Pub. Date	All Dates
Month	
Year	
Sort Results by:	Relevance

Figure 1.2: An example of form that belongs to the Books domain.

1.3 Form Discovery

The very nature of the Web — which is huge, dynamic and decentralized in a way that pages are freely and autonomously added, deleted, or modified — makes it practically impossible to manually discover at large the exact location of URLs of pages containing forms. If someone wants to do this, he or she has to rely on automatic methods for form discovery.

Automatic form discovery can be divided into two main tasks. The first one is locating forms on the Web. The second one is identifying among those forms which are indeed relevant according to arbitrary criteria. The first task is challenging because forms are sparsely distributed on the Web (i.e., there are large regions of the Web that contain few forms and small regions that contain lots of them) (BARBOSA; FREIRE, 2005, 2007b). The second task is necessary because usually the users have very specific needs that have to be fulfilled by the form discoverer. The users may not be — and frequently are not — interested in all the forms that a form locator is able to find on the Web. Irrelevant forms need to be automatically identified and filtered so that the relevant ones may be discovered. Also, the second task is challenging because forms in general are designed to be used (i.e., understood, filled out, and submitted) by humans. Typically, forms do not carry explicit machine-readable semantic information that could be used by automatic form processing tools. In fact, the only concern of most Web page designers while constructing a form is on its rendering layout as presented by a Web browser (i.e., the look-and-feel of the form when rendered or displayed by a Web browser). Therefore, automatic form discoverers can be certain only of syntactic features of forms.

1.4 Three Dimensions to the Problem of Form Discovery

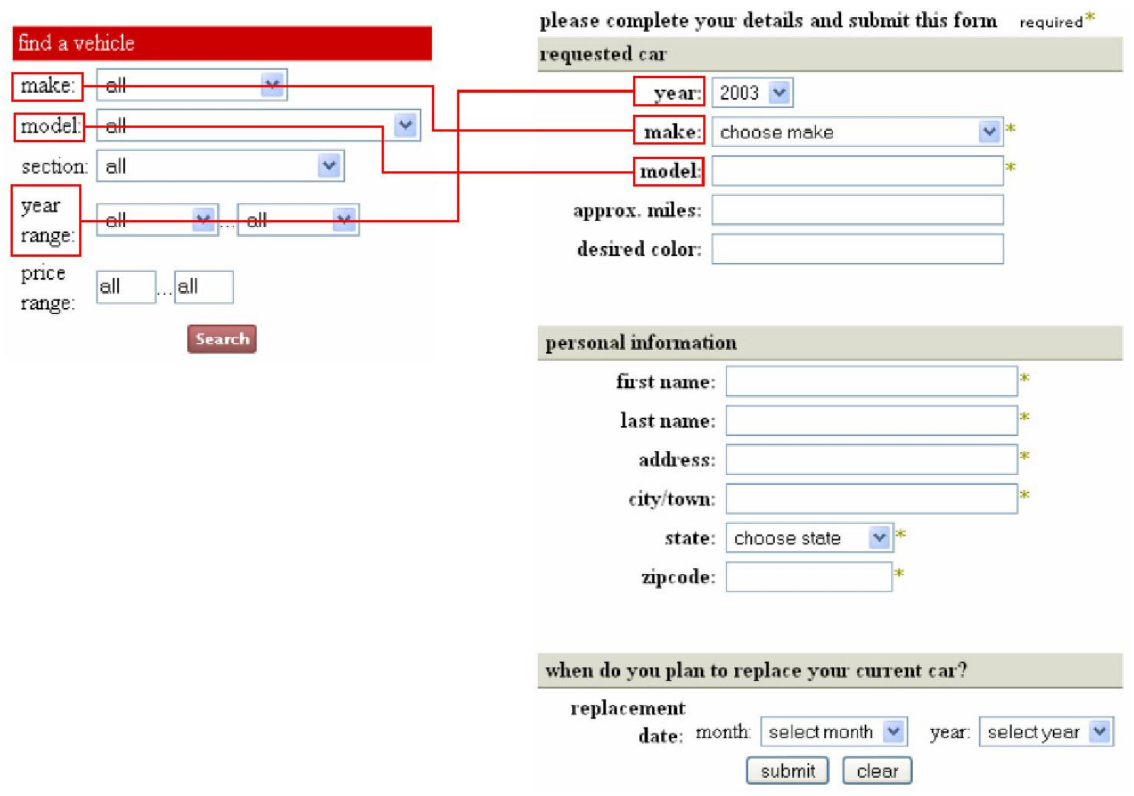
A consequence of the creative freedom of Web page designers and the lack of semantic information in forms is the existence of many disparate forms on the Web, even among the ones related to the same domain or subject (BARBOSA; FREIRE; SILVA, 2007). Such a stern reality lead us to discuss three dimensions to the problem of form discovery before properly defining the scope of this thesis, namely: (i) domain *vs* function; (ii) form structure; and (iii) pre- *vs* post-query approaches. The following subsections provide details about each one of them.

1.4.1 Domain *vs* Function

Generally, two types of form identification are relevant for form discovery, namely: by domain and by function. The first type discriminates the domain to which a form belongs (i.e., the subject that characterizes it and the data "hidden behind" it), as domains capture the purpose of a form (KUSHMERICK, 2003), such as *"searching for books"* (illustrated in Figure 1.2), *"querying a vehicle database"* (illustrated on the left side of Figure 1.3), *"finding a job"* (illustrated by all forms depicted in Figure 1.4), etc.

The second type of form identification discriminates forms based on the main kind of access to their underlying databases, regardless of the domain to which they belong. If a form is used primarily for reading from an on-line database, it is called a query form; if it is used primarily for writing on an on-line database, it is called a non-query form.

Examples of query and non-query forms are shown in Figure 1.3, which contains two forms that belong to the `Cars` domain: the one on the left is a query form because its main function is to query or search an on-line database; the one on the right is a non-


find a vehicle

make: all

model: all

section: all

year range: all ... all

price range: all ... all

Search

please complete your details and submit this form required*

requested car

year: 2003

make: choose make *

model: *

approx. miles:

desired color:

personal information

first name: *

last name: *

address: *

city/town: *

state: choose state *

zipcode: *

when do you plan to replace your current car?

replacement date: month: select month year: select year

submit clear

Figure 1.3: Examples of a query and a non-query form.

query form because its main function is to register or write data on an on-line database. The connected boxes in the referred figure show that query and non-query forms may share similar components. Independently of specific domains, general examples of non-query forms are purchase forms, forms for commenting in discussion groups, login forms, etc.

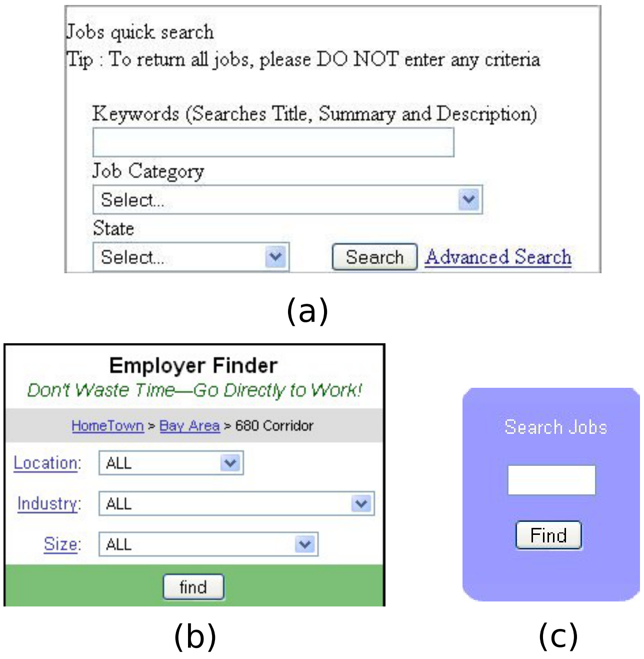
1.4.2 Form Structure

Query forms can be divided into structured forms and unstructured or keyword-based forms. The first type presents an implicit schema providing hints about the structure of the underlying data source. The second type does not.

Structured forms are composed of one or more attributes, which represent elements of the implicit schema of the form's underlying database. An attribute is composed of one or more form input fields. There are form input fields of several types (e.g., text, select, password, etc.)

Most form attributes often have labels. A label is the text directly associated with the form attribute, which explains its meaning/purpose. Through the label, the user can understand how to correctly fill a specific attribute before submitting it.

Figures 1.4(a) and 1.4(b) present examples of structured forms that belong to the `Jobs` domain. The second attribute of the form depicted in Figure 1.4(a) is labeled *"Job Category"* and the second attribute of the form depicted in Figure 1.4(b) is labeled *"Industry"*. Even though the referred figures do not show the contents of the comboboxes associated with these labels, it is reasonable to assume that both form attributes belong to the same data type, since *"Job Category"* and *"Industry"* are frequently used as syn-



Jobs quick search
Tip : To return all jobs, please DO NOT enter any criteria

Keywords (Searches Title, Summary and Description)

Job Category: Select...

State: Select...

Search | Advanced Search

(a)

Employer Finder
Don't Waste Time—Go Directly to Work!

HomeTown > Bay Area > 680 Corridor

Location: ALL

Industry: ALL

Size: ALL

find

(b)

Search Jobs

Find

(c)

Figure 1.4: Examples of forms from the `Jobs` domain.

onyms in the `Jobs` domain and indicate the kind of job that should be searched.

Form 1.4(a) presents three schema attributes: `job category`, `state`, and `keywords`. The attribute `keywords` allows keyword-based search in three different columns of the underlying database, namely: `title`, `summary`, and `description`. Form 1.4(b) presents three schema attributes: `location`, `industry`, and `size`.

In contrast, keyword-based or unstructured query forms do not present an implicit schema. This kind of form generally presents a single text box with no schema label associated to it. Figure 1.4(c) presents an example of an unstructured form that belongs to the `Jobs` domain. The string above the unique input field in the figure gives no hint about the internal structure of the underlying on-line database. Most frequently, unstructured forms present considerably less information than structured ones.

1.4.3 Pre- *vs* Post-Query Approaches

There are two major approaches for form identification (COPE; CRASWELL; HAWKING, 2003; RU; HOROWITZ, 2005), namely: (i) pre-query and (ii) post-query. The first approach deals only with the form itself and with the page that contains it. The second approach makes use of data that result from form submissions.

A post-query method has more information available to substantiate its identification decisions, but it has to handle complex issues related to form submissions (i.e., form filling and processing of form submission results). A pre-query method has less information available to substantiate its identification decisions as it does not submit forms, but it is relatively simpler.

Generally, the post-query approach is employed by techniques that identify keyword-based or unstructured forms and the pre-query approach is employed by techniques that identify structured query forms. Since the pre-query approach is less dependent on computer network resources, it is generally more lightweight than the post-query approach. Also, the pre-query approach is more polite, since it does not incur on the risk of eventually submitting garbage to a non-query form simply for the sake of identification (COPE; CRASWELL; HAWKING, 2003).

1.5 Scope

We believe it would be necessary more than one thesis to cover all facets of the referred dimensions to the problem of form identification. Consequently, we limit the scope of this thesis to a clearly defined subset of form discovery.

The scope of this thesis is predominantly focused on pre-query-based discovery of domain-specific structured forms. We are mainly interested in methods that do not require the submission of forms (i.e., that use the pre-query approach) for the discovery (i.e. location and identification) of domain-specific forms that contain implicit schemas providing hints about the structures of their underlying data sources (i.e., structured forms).

Form discovery aspects outside the scope of this thesis (i.e., domain-independent form discovery and post-query form discovery) are discussed only when their mention becomes necessary for the sake of completeness. More specifically, we note that some of the related work that we cover combine pre- and post-query approaches. For those, we discuss in detail the pre-query steps while providing an overview of the post-query processing. Also, we cover the rare related work for unstructured forms that present pre-query-based behavior.

1.6 Research Context

Form discovery is an integral part of a number of research areas (e.g., meta-queriers (GRAVANO; GARCÍA-MOLINA; TOMASIC, 1999; YU et al., 2002), on-line database directories (GALPERIN, 2005), Web information integration systems (CHANG; HE; ZHANG, 2005; HE et al., 2003), and vertical search engines (CAZOODLE, 2011; TRULIA, 2011), etc.). All of these research areas are important for advancing Web processing methods in general, but we are specifically interested in applying form discovery to Deep Web crawling (BERGMAN, 2001; RAGHAVAN; GARCIA-MOLINA, 2000; CHANG et al., 2004; MADHAVAN et al., 2008).

Deep Web crawling is the subject of a project being carried on by the database research group at the Instituto de Informática/UFRGS[1]. The main goal of this project is to create a domain-specific Deep Web crawler that is able to effectively operate requiring minimum — ideally, none at all — input from a human expert. The next subsections briefly present the concepts of Deep Web and Deep Web crawling in order to present the research context in which this thesis was created.

1.6.1 The Deep Web

Early publications[2] presented lengthy discussions about the the Deep Web[3] (DW) (BERGMAN, 2001; RAGHAVAN; GARCIA-MOLINA, 2000; CHANG et al., 2004; MADHAVAN et al., 2008). In the words of some of these authors, the DW can be defined as (SHERMAN; PRICE, 2001):

...text pages, files, or other often high-quality authoritative information

[1]*Edital MCT/CNPq 14/2010 - Universal - Faixa B (HWeb - Gestão de Dados na Web Oculta).*

[2]The first author to study the Deep Web was Dr. Jill Ellsworth in 1994 (BERGMAN, 2001).

[3]Usual synonyms for the term "Deep Web" are "Hidden Web" and "Invisible Web".

available via the World Wide Web that general-purpose search engines cannot, due to technical limitations, or will not, due to deliberate choice, add to their indices of Web pages.

Given this unexacting definition, it is hard to draw a line between the DW and the rest of the Web, also known as Publicly Indexable Web (PIW). After all, there are Web documents that could arguably be attributed to the PIW by a person A and to the DW by a person B (and vice-versa), since their judgment would depend on the available general-purpose search engines at their disposal at a given time.

There are many types of Web documents that compose the DW (e.g., images, multimedia file, PDFs, etc.). However, this thesis focus only on Web documents that result from form submissions. In other words, among all Web documents that compose the DW, we are only interested in the ones that are "hidden behind" forms. Hereafter, we will refer to the DW that is "hidden behind" forms simply as *the DW*.

At least two published surveys provided in-depth characterizations of the DW (BERGMAN, 2001; CHANG et al., 2004; MADHAVAN et al., 2008). These surveys reported several estimations for the DW, among which the following are relevant to us:

- the DW is orders of magnitude bigger than the PIW, covers a broad range of domains or subjects and is growing considerably;
- the DW is poorly indexed by general search engines[4], presents lots of semi-structured content and is highly disconnected;
- the vast majority of documents "hidden behind" query forms require no authentication to be reached;
- less than 20% of all forms are covered by DW directories, which are Web sites that organize DW resources by domain, topic or subjects;
- the vast majority of query forms can be found up to three links in depth from the home page of their respective Web sites[5].

These estimates lead us to believe that the DW is not yet satisfactorily covered by the current state-of-the-art in DW processing. The following are secondary observations derived from the referred estimated characteristics:

- the huge size of the DW imposes a heavy burden on automatic methods, requiring them to be efficient since they have to cover and process large-scale data sets;
- the DW presents some clearly defined characteristics (i.e., it is considerably structured, highly disconnected and its entry-points are located near their respective home pages) that could be exploited by automatic methods;
- authentication issues are not a practical matter for form discoverers in general, since most DW documents can be accessed anonymously;

[4]Recently, Madhavan *et al.* (MADHAVAN et al., 2008) published a method for DW processing that works with forms located by Google's crawler. Such a method augments the indexing capacities of general search engines, but still presents limitations (e.g., it indexes only pages accessible through the HTTP `get` method) that lead us to believe that the DW remains poorly indexed by general search engines.

[5]The first paper that reported some findings related to this estimated characteristic of the DW was published in 2003 by Bergholz & Chidlovskii (BERGHOLZ; CHILDLOVSKII, 2003).

1.6.2 Deep Web Crawling

The Web as a whole (i.e., the DW and its complement, the PIW) can be modeled as a graph where the nodes are the Web documents and the edges are the hyperlinks (KLEINBERG et al., 1999) present on the Web documents. The process of automatically traversing the Web graph is called Web crawling (DILIGENTI et al., 2000).

DW crawling is composed of three phases. First, forms are discovered in the PIW. Second, the discovered forms have to be submitted (NTOULAS; PZERFOS; CHO, 2005; WU et al., 2006). Finally, the results of the submissions have to be processed in a suitable manner, which is generally done by data extraction processes (HEDLEY et al., 2004; **?**).

Furthermore, each phase of DW crawling can be divided into steps. The discovery of forms involves locating and identifying them on the PIW (refer to Chapter 2 for an in-depth survey of these techniques). Form submission involves form understanding (KHARE; AN; SONG, 2010) (i.e. semantic processing and correct assigning labels to fields) and efficient form filling (WU et al., 2006). Data extraction from the results of a form submission involves data cleaning (YI; LIU; LI, 2003) and data mapping (CRESCENZI; MECCA; MERIALDO, 2002; LIDDLE et al., 2003; WANG; LOCHOVSKY, 2003; LERMAN et al., 2004; ZHAI; LIU, 2005) from Web pages to some data model in order to allow structured queries.

In this thesis, we are interested only in the first phase of DW crawling and on its two component steps (i.e., locating forms, and identifying among those which are indeed meant for submission). The other phases of DW crawling are briefly discussed here solely to offer a broader view of the subject and to prepare the reader for the discussion about DW crawlers that has place in Chapter 2.

1.6.2.1 The Second Phase of DW Crawling: Form Submission

Form submission requires the forms to be “understood” and correctly filled out so that meaningful data is sent to the data source. Filling out forms necessarily involves understanding form semantics (KHARE; AN; SONG, 2010) (e.g., automatic attribute label identification, attribute clustering, etc.), so that adequate values may be assigned to form fields. For example, the string *“john”* may be appropriate for a form attribute `name`, but a numeric value may be inappropriate for the same attribute. Understanding and reasoning about form semantics is crucial in this phase, as only correct submissions result in useful data (while incorrect submissions generally lead to error messages).

Another concern in this phase is performance and resource usage: incorrect submissions also lead to waste of computational resources (e.g., bandwidth). Efficient form submission is specially important for forms that present many text fields, since brute-force combination of different values for all text fields would easily lead to an explosion in the number of submissions.

1.6.2.2 The Third Phase of DW Crawling: Data Extraction

The third and final DW crawling phase is the extraction of information from the responses obtained at the previous phase. This phase is necessary to achieve the end-goal of DW crawling: processing and indexing data “hidden behind” the forms the were discovered in the first phase.

This phase may also provide valuable information that can be used to improve the performance of techniques involved in the previous phases. For example, one could analyze (as post-query methods do) the results of a form submission to test a hypothesis about the domain of the form or to provide data to be used in future form submission tasks.

1.7 Contributions

This section presents an overview of the two main contributions of this thesis. Briefly, the first contribution is a survey about form discovery methods. The second contribution is the definition of a method that mitigates the demanding manual work required by two cutting-edge form classifiers, at the cost of a small loss in effectiveness.

1.7.1 A Survey

Currently, there are five published in-depth surveys dedicated to the DW. Two of them are mainly about DW characterization (BERGMAN, 2001; CHANG et al., 2004). Another two discuss DW crawling techniques in general, including some discovery techniques (RU; HOROWITZ, 2005; MADHAVAN et al., 2009). Finally, the most recent survey deals specifically with *search interface understanding* (KHARE; AN; SONG, 2010). To the best of our knowledge, this thesis presents the first in-depth survey focused on domain-specific form discovery based on the pre-query identification approach. The survey we carried out as part of this thesis will be published soon (MORAES et al., 2012).

The lists of pre-query methods (e.g., (PERKOWITZ et al., 1997; DOORENBOS; ETZIONI; WELD, 1997; RAGHAVAN; GARCIA-MOLINA, 2000; BERGHOLZ; CHILDLOVSKII, 2003; COPE; CRASWELL; HAWKING, 2003; BARBOSA; FREIRE, 2007a; KUSHMERICK, 2003; HE; KUSHMERICK, 2003; XU et al., 2007; LE; CONRAD, 2009; BARBOSA; FREIRE, 2005, 2007b; LU et al., 2006; HE; TAO; CHANG, 2004, 2005; BARBOSA; FREIRE; SILVA, 2007; SONG et al., 2008; ZHAO et al., 2008; LI et al., 2010; LIN; CHEN, 2002; KABRA; LI; CHANG, 2005; SHEN et al., 2008)) and post-query methods (e.g., (IPEIROTIS; GRAVANO; SAHAMI, 2001a,b; GRAVANO; IPEIROTIS; SAHAMI, 2003; GRAVANO; GARCÍA-MOLINA; TOMASIC, 1999; HEDLEY et al., 2006; AGICHTEIN; GRAVANO, 2003; IPEIROTIS; GRAVANO, 2004)) are fairly extensive. In addition, these approaches differ substantially, justifying separate surveys to cover each area properly.

The focus of our survey is on domain-specific discovery methods that rely predominantly on pre-query methods.[6] Moreover, we cover methods for structured as well as for unstructured forms, contributing the following:

- an up-to-date overview of several works related to domain-specific discovery based on the pre-query identification approach. In the few cases where pre-query and post-query are combined to perform form identification, only the pre-query methods are detailed;
- a comprehensive classification of the surveyed methods, in which they are grouped by their end-goals;
- a discussion on how the groups of methods depend on external factors and on each other;
- the identification of a growing trend on the automation in form discovery and a subsequent discussion on our forecasts about the next events in this area.

[6]Some of the methods surveyed combine pre- and post-query approaches. For those, we discuss in detail the pre-query steps while providing an overview of the post-query processing for the sake of completeness.

1.7.2 A Novel Method

We combine and evaluate several heuristic-based methods for pre-query discovery of domain-specific structured forms on the Web. Individually, most of the heuristics evaluated in this thesis were already proposed by other authors or were strongly inspired by the works of other authors. The novel aspect presented in this thesis is the way in which we combine the referred heuristics and the purpose to which we employ them.

Our final goal while employing the heuristics is to mitigate the demanding manual work required by the current state-of-the-art in form discovery and thus to relieve the human expert from tiresome work while discovering domain-specific structured forms. We present substantial empirical evidence that it is feasible to automatically train two state-of-the-art form classifiers, at the cost of a relatively small loss in effectiveness.

1.8 Text Organization

The remainder of this thesis is organized as follows. Chapter 2 presents the related work, organizing 19 methods for form discovery into five groups.

Chapter 3 builds upon the study presented in Chapter 2 in order to identify how the groups of related work depend on each other and on the human expert. Chapter 3 also shows that there is a trend of gradual increase in automation in form discovery.

Chapter 4 presents a very simple method for locating forms on the Web. It also provides the description of some empirical tools that helped us execute the experiments that validated our proposal.

Chapter 5 presents and combines a set of heuristics for form identification by function. The experiments discussed in that chapter show that simple heuristic-based methods for form identification by function are fully capable of replacing manual training for the two experimented form classifiers.

Chapter 6 presents a set of heuristic-based methods for form identification by domain, which are built on top of the solution presented in Chapter 5. The experiments discussed in that chapter show that our heuristic-based methods for form identification by domain are able to replace manual training for the experimented form classifiers at the cost of a relatively small loss in effectiveness.

In Chapter 7 we discuss some potential caveats of our proposal and present some further experimentation with the proposed solution. The motivation of Chapter 7 is to more completely characterize our contribution for form discovery.

Finally, Chapter 8 concludes this thesis. It wraps up our proposal by briefly reviewing the previous chapters and presenting a potential future work to be built upon our research.

2 RELATED WORK

This chapter presents the works related to Web form discovery that are relevant to this thesis. To the best of our knowledge, this chapter constitutes the first in-depth study focused on domain-specific form discovery based on the pre-query identification approach[1].

The 19 surveyed works are organized in this chapter according to their primary goal into five groups:

- *DW crawlers*, which perform all three phases of DW crawling, and, consequently, the two tasks involved in form discovery;
- *form classifiers*, which need to be manually trained in order to identify forms;
- *form crawlers*, which specialize in form location, prioritizing newly-found links in order to avoid portions of the PIW that do not contain many forms;
- *form clusterers*, which group together sets of similar forms, identifying those that belong to the same domain; and
- *form rankers*, which impose a partial order to the elements of a set of forms according to some user requirements.

We adopted this goal-oriented categorization because we believe it is easier to comprehend for most readers. Table 2.1 provides a summary of the major characteristics of these groups of methods, which we discuss next.

[1]Two short studies closely related to ours were published very recently (NOOR; RASHID; RAUF, 2011; EL-GAMIL et al., 2011). They reference several of the works covered here but they lack enough details and conclusive considerations to be considered in-depth. Also, they focus on aspects of form discovery that are distinct from the ones that we focus here.

Table 2.1: Summary of major characteristics of the surveyed groups of works.

Method	**Discovery Tasks**	**Identification Types**	**Form Filling**	**Attribute Label Identification**	**Training**	**Target Form Type**
DW CRAWLERS						
Shopbot	location / identification	both	yes	yes	manual	structured
HiWE	location / identification	both	yes	yes	manual	structured
B&C	location / identification	both	yes	no	no	unstructured
FORM CLASSIFIERS						
CC&H	identification	by function	no	no	automatic	structured
HIFI	identification	both	no	no	automatic	structured
H&K	identification	by domain	no	yes	automatic	structured
X&A	identification	by domain	no	yes	automatic	structured
L&Co	identification	by domain	no	no	automatic	structured
FORM CRAWLERS						
FFC	location	by function	no	no	automatic	structured
ACHE	location	both	no	no	automatic	structured
FORM CLUSTERERS						
WISE-Cluster	identification	both	no	yes	no	structured
HT&C	identification	by domain	no	yes	no	structured
CAFC	identification	by domain	no	no	no	structured
So&A	identification	by domain	no	yes	no	structured
Z&A	identification	by domain	no	yes	no	structured
L&A	identification	by domain	no	yes	no	structured
FORM RANKERS						
L&Ch	location /identification	by domain	no	no	no	unstructured
KL&C	identification	by domain	no	yes	no	structured
Sh&A	identification	by domain	no	yes	no	structured

In short, while form crawlers aim at locating forms, form classifiers, form clusterers, and form rankers aim at identifying forms. DW crawlers aim at providing both location and identification of forms. One could say that DW crawlers subsume form crawlers and form classifiers. However, we consider form crawlers and form classifiers as separate categories since both their behavior and outcome are sufficiently different from DW crawlers.

All of the referred five groups of techniques are discussed in the next subsections. Within each group, the techniques are sorted in chronological order for the most part. Eventual breaches in the chronological order are made for easier exposition. Even though form location is a discovery task that is necessarily performed before form identification, we discuss some form identifiers (i.e., form classifiers) before some form locators (i.e., form crawlers) because, as it will be discussed later, there is a structural dependency between them (i.e., form crawlers are build upon some form classifiers).

An alternative way of classifying the methods above would be to consider the type of identification performed: (i) by domain; (ii) by function; or (iii) both by domain and by function. The majority of the current methods perform identification by domain only, while the form classifier CC&H (COPE; CRASWELL; HAWKING, 2003) (Section 2.2.1) and the form crawler FFC[2] (BARBOSA; FREIRE, 2005) (see Section 2.3.1) perform only identification by function. On the other hand, all DW crawlers (Section 2.1), one form classifier (HIFI (BARBOSA; FREIRE, 2007a), see Section 2.2.2), one form crawler (ACHE[3] (BARBOSA; FREIRE, 2007b), see Section 2.3.2) and one form clusterer (WISE-Clusterer (LU et al., 2006), Section 2.4.1) perform both types of identification.

It is worth noting that several of the surveyed techniques use off-the-shelf components from related areas as tools to achieve their goals. For example, some techniques rely on inductive learning algorithms (e.g., (COPE; CRASWELL; HAWKING, 2003; BARBOSA; FREIRE, 2007a)), others employ statistical methods (e.g., (HE; TAO; CHANG, 2004; LI et al., 2010)), or Fuzzy Logic methods (e.g., (SONG et al., 2008; ZHAO et al., 2008)). Exploring the internals of such rather complex tools is outside the scope of our study.

2.1 Deep Web Crawlers

DW crawlers are among the earliest techniques for dealing with the DW, and perform all three phases of DW crawling. We focus on their form discovery capabilities, which present the following commonalities:

- use of manually defined or automatically generated domain-specific knowledge bases that are mainly composed of sets of textual terms used for automatic query form submission and domain identification;
- collection of forms from the PIW using traditional in-site crawlers, which follow only the links that point to pages that belong to the same site; and
- combination of pre-query and post-query approaches to identify domain-specific query forms among the forms collected by the in-site crawlers.

[2]Even though the surveyed form crawlers aim primarily at form location, they depend on form identifiers as internal components; therefore, they indirectly perform form identification.

[3]Ibid.

Table 2.2: Summary of major characteristics of DW crawlers.

	Attribute Label Identi-fication	**Comple-tely Reprodu-cible**	**Knowledge-Base Construction Approach**	**Seed URL Definition for In-Site Crawling**	**Target Form Type**
ShopBot	yes	yes	manual	manual	structured
HiWE	yes	yes	manual	manual	structured
B&C	no	no	automatic	semi-automatic (it builds upon a Web Directory)	unstructured

The pre-query form identifiers employed by these methods are heuristic filters (i.e., filters that heavily rely on manually defined heuristics or rules that aim to describe common properties of forms); they discard non-query forms, generally consisting of simple rules e.g.,: *if a form presents a password field, it is not a query form*; or *if a form presents the word "search" or one of its synonyms, it is a query form*; etc. After filtering, post-query form identifiers are employed, making use of the referred knowledge bases to fill out the query forms and to identify the domains to which they belong.

The next subsections discuss each of the three domain-specific DW crawlers found in the literature. Table 2.2 summarizes their major characteristics. By "completely reproducible" we mean that the technique was described in sufficient detail to enable its implementation.

2.1.1 Perkowitz et al. (ShopBot)

ShopBot (PERKOWITZ et al., 1997; DOORENBOS; ETZIONI; WELD, 1997) is the oldest technique associated with the DW, dating back to 1997. It builds upon manually constructed knowledge bases containing sets of labeled textual descriptions of products.

Perkowitz *et al.* acknowledged that the construction of knowledge bases for ShopBot may require considerable effort from the human expert. For instance, their experiments on a single domain required a full day of work from the human expert (PERKOWITZ et al., 1997).

Since its crawler frequently finds many forms of different types in a single Web site, ShopBot performs two tasks before starting to compare products of different sites: (i) identification of query forms among all collected forms; and (ii) selection of the most promising forms to perform product comparison, among those identified within each site.

The first task is accomplished with the use of an heuristic-based filter that employs the following rule: *query forms do not present some words frequently found in non-query forms (e.g., "phone", "e-mail", "address", etc.).* They reasonably assume that forms that present such words are clearly non-query forms, since usually the kind of information associated with words like these are used for posting personal data to an HTTP Web server. Nevertheless, Perkowitz *et al.* do not discuss how these words are defined; supposedly, they have to be manually defined by the human expert.

The second task is based on an estimation of how successfully the data accessed through a query form can contribute to product comparison. This estimate involves post-query identification techniques which make use of the manually defined knowledge base.

Perkowitz *et al.* did not experiment ShopBot's discovery capabilities since their main focus was on product comparison and not on query form discovery. However, they briefly mentioned that ShopBot was able to learn vendor descriptions from all 12 DW sites used in their experiments, which were manually pointed by the human expert.

2.1.2 Raghavan & Garcia-Molina (HiWE)

Like ShopBot, HiWE (*Hi*dden *W*eb *E*xposer) (RAGHAVAN; GARCIA-MOLINA, 2000) builds upon manually constructed knowledge bases containing sets of labeled values. However, HiWE's knowledge base does not contain descriptions of products, but sets of suitable values associated with labels that are likely to be associated to the textual attributes of the query forms of the domain of interest.

HiWE uses an heuristic-based filter that distinguishes query forms from non-query ones, according to these rules: (i) *query forms present more fields than a given manually defined integer threshold*; and (ii) *query forms present all their textual form attribute labels matched by string similarity with the strings present in the knowledge base*. The goal of the first rule is to ignore generic search engine forms (e.g., Google's in-site search), frequently found in many Web sites. The goal of the second rule is to process only the forms that belong to the domain of interest.

HiWE's heuristic-based filter is more restrictive than ShopBot's. While ShopBot filters only forms that present one or more specific words, HiWE filters all forms that do not present the minimum number of fields allowed or that do not have all their attribute labels matched with the ones present in the knowledge base. This may potentially cause HiWE to discover fewer query forms than ShopBot, since it is not easy to manually build a knowledge base that contains more than just the most known domain-specific form attribute labels.

Like Perkowitz *et al.*, Raghavan & Garcia-Molina did not run detailed experiments to assess HiWE's query form discovery capabilities, because the main focus of their work was on how best to automate DW content retrieval, given the location of potential sources (RAGHAVAN; GARCIA-MOLINA, 2000). Nevertheless, they briefly mentioned that HiWE correctly identified 94 query forms among 220 forms collected during a crawl seeded with 50 manually defined URLs.

2.1.3 Bergholz & Chidlovskii (B&C)

The DW crawler designed by Bergholz & Chidlovskii (BERGHOLZ; CHILDLOVSKII, 2003) (hereafter referred to as *B&C*) has as main goal the discovery of keyword-based query forms through the interaction with a generic search engine (e.g., Google (GOOGLE SEARCH ENGINE, 2011), Yahoo! (YAHOO! WEB SEARCH ENGINE, 2011), etc.). It presents two features that HiWE and ShopBot lack: it builds upon a Web directory (GOOGLE DIRECTORY, 2011) to find the initial set of URLs (i.e., the seeds) to its in-site crawler; and, more importantly, it is capable of automatically constructing the knowledge base required to identify the domain of the collected query forms. It requires as input only URLs that point to the relevant categories from a Web directory, which are the ones that present URLs of DW sites in the domain of interest.

The authors offer an overview of B&C's architecture and a reasonable discussion about its main components; nevertheless, they do not discuss how B&C automatically builds its knowledge base. They only briefly mention that the knowledge base is composed of domain-specific relevant sets of textual terms automatically computed by the XeLDA server, an unpublished technology from Xerox (XELDA, 2011). Since the definition of textual terms is core to B&C, it is not possible to say that their method is completely reproducible.

The experiments performed by Bergholz & Chidlovskii resulted in the discovery of approximately 5000 query forms related to 14 categories from the Google Directory (GOOGLE DIRECTORY, 2011). These authors are among the first to execute experiments with large-scale discovery of domain-specific query forms.

Also, the authors, followed by Chang *et al.* (CHANG et al., 2004), found empirical evidence that query forms on the PIW are located near the home pages of their respective Web sites. More specifically, they found that the vast majority of query forms can be found up to three links in depth from the home page of a DW site. This is a strong reason to avoid extensive in-site crawling while looking for query forms, it helps save computational resources (e.g., time, bandwidth, etc.) during the crawl.[4]

2.1.4 Note about Google's Domain-Independent DW Crawler

It is worth mentioning the relatively recent DW crawler proposed by Madhavan *et al.* (MADHAVAN et al., 2008), which works with forms located by Google's crawler. It distinguishes query forms from non-query forms through the use of a sole heuristic which states: *a query form uses the HTTP Get method.* Once the query forms have been identified through the heuristic, it employs post-query techniques to reach and index the pages "hidden behind" them using Google's indexing infra-structure. This, in turn, allows the pages to be accessible by means of the Google's search engine.

This DW crawler is outside the scope of our study, however, as it is domain-independent (i.e., it tries to reach and index pages "hidden behind" the query forms from any domain). Even though the main argument for being domain-independent (i.e. data on the Web is about everything and boundaries of domains are not clearly definable) is compelling for a horizontal search engine, there is a strong tendency in the literature to support domain-awareness as key to higher quality discovery of structured forms on the Web. Time will tell if domain-independent crawlers will evolve to surpass domain-aware ones in terms of accuracy.

2.2 Form Classifiers

Form classifiers assign a form to a class defined in some preexisting taxonomy. These classifiers are built on top of inductive learning algorithms (MITCHELL, 1997; RUSSELL; NORVIG, 2009) and thus they need training in order to operate.

Training a form classifier involves providing its inductive learning algorithm with sets of manually labeled features extracted from the forms, which constitute the *training base* of the classifier. During its training, the inductive learning algorithm infers concepts from the training base; then, it becomes able to classify any given form by analyzing the same features that were targeted during training (e.g., the number of text fields, the presence of a password field, the occurrence of the word *"search"*, etc.). A comprehensive discussion about supervised learning can be found in (MITCHELL, 1997).

Training bases contain as many sets of labeled features of forms as there are classes in the preexisting taxonomy used to guide the classification process. An usual type of classification is the *binary* (or Boolean) classification. This type of classification is based on a taxonomy of only two classes: (i) `Yes` or `True`, which holds the relevant forms; and (ii) `No` or `False`, which holds the irrelevant forms. Therefore, binary classifiers require

[4]Chang, He & Zhang (CHANG; HE; ZHANG, 2005) created a crawler that exploits this empirical evidence.

Table 2.3: Summary of major characteristics of form classifiers.

	Attribute Label Identification	**Identification Types**	**Labeling Targets**
CC&H	no	by function	forms
HIFI	no	by function & by domain	forms
H&K	yes	by domain	forms & attributes
X&A	yes	by domain	forms & attributes
L&Co	no	by domain	forms

training bases containing only two sets of labeled features of forms, usually called the *positive* and *the negative training examples* (BARBOSA; FREIRE, 2007a). As shown in the next subsections, binary or Boolean classification is employed by form classifiers that perform identification by function.

It has been shown experimentally that, while the actual algorithm used to classify forms plays an important role in form classification, it does not seem to be the most important factor to achieve good classification results (COPE; CRASWELL; HAWKING, 2003; BARBOSA; FREIRE, 2007a; KUSHMERICK, 2003). Instead, what really appears to greatly influence classification results is the set of features considered by the form classifiers.

Also, the set of features determines the degree of dependency on the human expert, demanding more or less involvement from the human expert while training the classifiers. As we discuss below, different classifiers take into account very different sets of form features. Table 2.3 shows a summary of the major characteristics of form classifiers.

2.2.1 Cope, Craswell and Hawking (CC&H)

Cope, Craswell & Hawking (COPE; CRASWELL; HAWKING, 2003) designed a form classifier (hereafter referred as *CC&H*) that uses a decision tree (QUINLAN, 1986) as inductive learning algorithm to classify forms by function. CC&H targets the following features of forms to operate: (i) the `name` parameters of HTML `input` tags; (ii) the `value` parameters of HTML `input` tags; (iii) the `name` parameters of HTML `form` tags; and (iv) the set of distinct words from the HTML `action` parameters of HTML `form` tags.

Cope, Craswell & Hawking used two sets of forms to train CC&H, each containing approximately 200 forms. One set was collected from a single top-level internet domain[5]. The other was collected from randomly selected top-level internet domains and was considered to be representative of the Web as a whole.

These sets of forms presented very distinct characteristics from each other, which resulted in completely different decision trees and, therefore, different results. The experiments show that CC&H achieved much better results for the specific collection, due to the greater variation of features present in the query forms from the Web collection.

It is worth noting that CC&H was tested with different types of supervised algorithms in order to assure that it is independent of the inductive learning algorithm used to classify forms. The results lead to the conclusion that CC&H is not tied to decision trees.

In addition, while CC&H does not perform classification by domain, its training may be regarded as domain dependent, since it takes into account textual form features that

[5]Namely, the Australian National University (ANU).

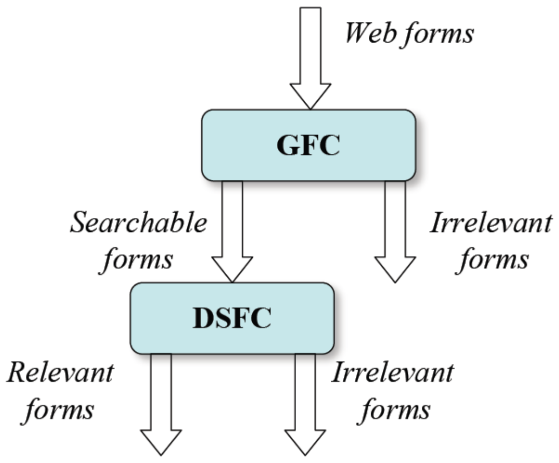


Figure 2.1: HIFI's hierarchy of classifiers.

greatly vary depending on the domain to which the form belongs (e.g., form field names and values). This observation was originally articulated by Barbosa and Freire (BARBOSA; FREIRE, 2007a).

2.2.2 HIFI

Barbosa & Freire are among the most prolific authors of discovery techniques (BARBOSA; FREIRE, 2006; BARBOSA et al., 2010).[6] Their strategy for form identification, named HIFI (*HI*erarchical *F*orm *I*dentification) (BARBOSA; FREIRE, 2007a), is currently employed by the DeepPeep search engine (DEEPPEEP, 2010), which is specialized in Web forms and currently tracks approximately 45,000 forms across seven domains.

HIFI partitions the feature space in two pieces and uses two different classifiers that are best suited to the features of each partition: (i) GFC, which performs classification by function, and (ii) DSFC, which performs classification by domain. Figure 2.1 offers a more detailed view of how GFC and DSFC are hierarchically composed: GFC sits at the top of the hierarchy and targets the most general concept (i.e., the function of a form); DSFC sits at the bottom of the hierarchy and targets the most specific concept (i.e., the domain of a form).

Such a composition of classifiers is backed up by experiments that lead to the conclusion that HIFI performs better than a monolithic classifier (i.e., a single classifier that performs classification both by function and by domain, using a single set of form features). More specifically, the results of the experiments show that the monolithic approach misclassifies a large number of relevant forms and, consequently, achieves very low recall (BARBOSA; FREIRE, 2007a).

GFC performs the same type of classification done by CC&H. However, unlike CC&H, GFC explores only structural form features. Therefore, GFC's training can be regarded as domain-independent, because structural form features do not vary substantially across domains.

The structural form features considered relevant by GFC are: (i) the number of certain HTML types of form fields; and (ii) the presence of the word *"search"* within a form.[7]

[6]Another three techniques proposed by Barbosa & Freire are surveyed in this paper, in Subsections 2.3.1, 2.3.2 and 2.4.3.

[7]Interestingly, Barbosa & Freire found that the presence of the word *"search"* within a form is the most

The choice of these features was based on empirical experiments, in which the authors observed that some structural characteristics of a form can be a good indicator as to whether the form is a query form or not.

DSFC uses the textual contents of a form to identify its domain, because it assumes that textual contents often have meta-data and data pertaining to the database (BARBOSA; FREIRE, 2007a). To justify this decision, the authors mention that form attribute names often match names of fields in the database, and selection lists often contain values that are present in the database (BARBOSA; FREIRE, 2007a). This assumption is strongly supported by previous works done by He, Tao & Chang (HE; TAO; CHANG, 2004, 2005), discussed in Subsection 2.4.2 and, in a smaller scale, by Hess & Kushmerick (KUSHMERICK, 2003; HE; KUSHMERICK, 2003), discussed in Subsection 2.2.3.

Both GFC and DSFC were experimented with several different inductive learning algorithms (e.g., Naive Bayes, C4.5, SVM, etc.) (MITCHELL, 1997; RUSSELL; NORVIG, 2009). The results show that some algorithms performed relatively better than others for both GFC and DSFC. In these experiments, GFC was trained with over 500 positive and negative examples of query forms; DSFC was trained with over 400 positive and negative examples per domain, which totalized more than 2000 manually classified forms. DSFC was tested on eight domains.

2.2.3 Hess & Kushmerick (H&K)

Motivated by the launching of Web service standards (e.g., WSDL, SOAP, UDDI, etc.) (WALSH, 2002), Hess & Kushmerick designed a strategy for form classification by domain (hereafter referred as *H&K*) as a first step toward allowing ordinary HTML query forms to be automatically accessed in the same way Web service providers are (KUSHMERICK, 2003; HE; KUSHMERICK, 2003).

The goal of H&K is not only to assign a form to a concept in a predefined domain taxonomy, but also to assign each form attribute to a concept in a data type taxonomy, where data type refers to the expected semantic category of the value of the attribute, and not to raw data types (i.e. the low-level HTML encoding issues: text, numeric, etc.).

For instance, consider two forms $f1$ and $f2$. A form attribute whose form label is *"City"* in $f1$ and a form attribute whose label is *"Town"* in $f2$ may belong to the same data type, according to an arbitrary definition by a human expert. This data type could be arbitrarily named `location`, since both attributes refer to places. Therefore, the inductive learning process employed by H&K explores not only form features, but also attribute features. Consequently, H&K's training base has to contain information about the domains of forms and about the data types of form attributes.

Compared to the training required by CC&H and HIFI, H&K's training is more laborious, since it requires the association of data types to form attributes, which is done manually. Hess & Kushmerick explicitly mentioned the effort and attention demanded from the human experts while training and testing H&K. According to them, two assistants continually refined the data type set as additional forms were examined and revisited previously inspected forms to ensure consistency. The assistants admitted that they have not identified the optimal domains and data types, instead, they have simulated a realistic scenario with a reasonably large number of heterogeneous sets of query forms.

H&K builds upon a three-layered tree-structured Bayesian Network (FRIEDMAN; GEIGER; GOLDSZMIDT, 1997) as the inductive learning algorithm used for classifica-

important generic characteristic of a query form from any domain.

tion. This algorithm computes statistics about the following features of forms: (i) the frequencies of the terms present in their textual contents; (ii) the data types of their attributes; and (iii) their domains. Operationally, the algorithm classifies forms in two steps. In the first step, the data types of the form attributes are inferred from the textual terms found in the form attribute labels. In the second step, the domain of the form is inferred from the data types of its attributes, which have been inferred in the previous step.

To run their experiments, Hess & Kushmerick used a training base composed of 129 forms, which were manually collected from the PIW. Their training base consisted of query forms from six domains, comprising 656 form attributes from 71 different data types.

2.2.4 Xu et al. (X&A)

Xu *et al.* designed a form classifier by domain (hereafter referred as *X&A*) (XU et al., 2007) as an intermediate step towards performing automatic query interface integration (CHANG; HE; ZHANG, 2005; HE et al., 2003) over the Web.

Similarly to H&K, X&A works with data types of form attributes. However, while H&K works with data types with the sole goal of implicitly identifying synonyms (i.e., two or more attributes that have the same semantic category, but that present different labels) and homonyms (i.e., two or more attributes that have different semantic categories, but that present the same label), X&A also works with meronym/holonym (i.e., *part-of*) relationships between form attributes. An example of a *part-of* relationship can be found in the `Books` domain, where a form attribute `Author` is frequently composed of two other form attributes: `first name` and `last name`.

While H&K takes into account only the data types and the textual terms of the attributes, X&A also takes into account the raw data types of form fields (e.g., `string`, `numeric`, etc.) to perform classification. According to Xu *et al.*, it is useful to consider raw data types of form fields because they provide good hints about the data type of an attribute. To illustrate this, they mention the form attribute `depart` that, in the `Airfares` domain, may have two meanings: (i) a departing date, which is frequently represented by a `date-time` raw data type; and (ii) a departing city, which is frequently represented by a `string` raw data type.

Another difference between H&K and X&A is that the latter takes into account the specificity and the data relevance of attributes in relation to a specific domain. The specificity of an attribute informs that the attribute is domain-specific *or* is used in more than one domain. The data relevance of an attribute informs that the attribute is a data holder or just an input to help the user to define some constraint on the search (e.g., some query forms from the `Airfares` domain present a checkbox field labelled as *"I am flexible"* to indicate that the flight departure time should be used as a loose filter; another example is the presence of an option labelled as *"Help"* that on some forms augments the requested data with suggestions for query refinements (KUSHMERICK, 2003)).

X&A employs the Vector Space Model (VSM) (SALTON; WONG; YANG, 1975) to perform form classification, in which each form is defined as a vector of form features and associated weights. The weights are calculated by a novel measure inspired by TF-IDF (*T*erm *F*requency - *I*nverse *D*ocument *F*requency) (BAEZA-YATES; RIBEIRO-NETO et al., 1999), which takes into account the data types of the attributes of the forms. The similarity of two query forms is defined by the cosine distance (SALTON; WONG; YANG, 1975) among their respective vectors in the VSM.

Similarly to H&K, X&A's training is laborious. In fact, we estimate that building the training base for X&A is even more difficult than building the training base for H&K, because the human expert is required to identify more features (i.e., attribute relationships, data types, and specificity/data relevance of attributes). Xu *et al.* argue that X&A's training requires from the human expert only a few dozen instances from each domain of interest to achieve good performance. Nevertheless, they did not comment on the effort required from the human expert in training X&A to perform their experiments, as Hess & Kushmerick openly did. X&A was experimented with 260 forms from four domains.

2.2.5 Le & Conrad (L&Co)

Le & Conrad (LE; CONRAD, 2009) studied and measured the impact of various methods of textual features selection (SEBASTIANI, 2002) to be employed by classifiers by domain. They noticed that all previous domain-specific classifiers indistinctly took into account all textual contents of forms, while it was already known (i.e., accepted as true by empirical evidence found in previous research) that some textual terms characterize some domains much better than others (e.g., the term *"ISBN"* characterizes the `Books` domain better than the term *"publication year"*).

Le & Conrad approached form classification by employing a technique for feature selection (e.g., CHI (YANG; PEDERSEN, 1997), Information Gain (YANG; PEDERSEN, 1997), Bi-normal separation (FORMAN, 2003), etc.) together with a Gaussian classifier (RASMUSSEN, 2004). Their experiments led them to the conclusion that textual feature selection achieves higher classification performance compared to classification that does not select textual features. They performed their experiments with 431 query forms from eight domains.

Also, Le & Conrad created a novel textual feature selection method (referred hereafter as *L&Co*) with the goal of improving the performance of existing form classifiers in the domains that present schema overlap (i.e., domains that share a number of form attributes). Examples of these domains are `Movies` and `Music`, which share, for example, the attributes `title` and `author`, as pointed out by He, Tao & Chang (HE; TAO; CHANG, 2004) and Barbosa & Freire (BARBOSA; FREIRE, 2007a).

2.3 Form Crawlers

Form crawlers are focused crawlers (MENCZER et al., 2001; WANG et al., 2008; CHAKRABARTI; BERG; DOM, 1999) whose goal is to find forms on the PIW. They typically perform a broad search on the Web, discovering new links and prioritizing them according to their probabilities of leading to pages that contain forms.

Due to the size of the PIW, as any other focused crawler, the main concern of form crawlers while prioritizing newly-found links is efficiency, by avoiding portions of the Web that present only few forms. Therefore, the biggest challenge faced by form crawlers is to automatically learn how to correctly prioritize links in order to be efficient.

The only authors to approach the location of query forms on the Web as a focused crawling task were Barbosa & Freire (BARBOSA; FREIRE, 2005, 2007b). They addressed many of the issues related to form crawling, in special the creation of logic capable of calculating the importance of a link not only in regard to its immediate benefit (i.e., its ability to directly lead to a page containing a form), but also in regard to its delayed benefit (i.e., its ability to indirectly or eventually lead to a page containing a form).

The work by Barbosa & Freire was presented in two papers, which reported different evolutionary stages of their approach for form crawling. Each of these papers is discussed

Table 2.4: Summary of major characteristics of form crawlers.

	Adaptive Crawling Behavior	**Identification Types**
FFC	no	by function
ACHE	yes	by function & by domain

in the next subsections. Table 2.4 shows a summary of their major characteristics.

It is worth mentioning that the crawler employed by Chang, He & Zhang (CHANG; HE; ZHANG, 2005) is not focused, since it does not prioritize newly discovered links according to their estimates of leading to relevant pages (MENCZER et al., 2001; WANG et al., 2008; CHAKRABARTI; BERG; DOM, 1999); it is a traditional crawler that exploits the fact that the vast majority of query forms on the PIW are located near the home pages of their respective Web sites (BERGHOLZ; CHILDLOVSKII, 2003; CHANG et al., 2004).

2.3.1 FFC

FFC was designed with the main goal of efficiently locating domain-specific query forms on the PIW (BARBOSA; FREIRE, 2005). Besides the decentralized dynamic nature of the Web, the authors justify the need for a broad search over the PIW with the observation that it is difficult to define an accurate schema description for the domain of interest. Even for a well-defined domain, it is hard to specify an accurate schema that is able to describe all or at least many forms among the relevant ones, because there is great variation in the structure and the vocabulary of the forms; an overly strict definition would miss relevant forms and an overly wide definition would hit irrelevant forms.

FFC combines ideas from two previously proposed focused crawlers that are ineffective when directly applied to form crawling, because they were designed to tackle different problems: one that takes into account only the features of the pages it crawls (CHAKRABARTI; BERG; DOM, 1999); and another that uses reinforcement learning to effectively crawl for pages that belong to sparse domains, taking into account not only features of the pages it crawls but also features of the URL paths that lead to the crawled pages (RENNIE; MCCALLUM, 1999). FFC inspects the contents of the crawled pages to guide the crawl and to focus the search on a specific domain. In order to further focus the search, it also inspects the URL paths that lead to the crawled pages that contain relevant forms.[8]

FFC builds upon three supervised classifiers: the page, the link, and the form classifiers. The page and the link classifiers, which build upon Naive Bayes textual classifiers (MITCHELL, 1997), are used by the crawler to guide its search. The page classifier identifies pages that belong to the domain of interest. The goals of the link classifier are two-fold: identify links that may eventually lead to pages presenting query forms; and assign a numeric relevance to each URL found during the crawl. This classifier takes into account three features of links: (i) their anchors; (ii) their URLs; and (iii) the text around them. The form classifier is used to filter out non-query forms collected during the crawl and it is very similar to HIFI's GFC (see Section 2.2.2).

Barbosa & Freire tested FFC on three domains. The measure used to assess its effec-

[8]The process of URL path building employed by FFC is not done from scratch. Instead of explicitly building the Web graph through repeated crawls of selected sites, FFC builds an approximation of the Web graph based on the backward crawling facilities provided by some search engines (BHARAT et al., 1998).

tiveness was the number of *distinct* query forms it retrieves as a function of the number of pages visited, regardless of the domains of the retrieved forms. The results of the experiments show that FFC is able to collect thousands of different forms, using as seeds some domain-related URLs from the Google Directory (GOOGLE DIRECTORY, 2011). Despite the number of forms collected, FFC achieves low precision in domain-specific query form discovery (BARBOSA; FREIRE, 2007a).

2.3.2 ACHE

ACHE (BARBOSA; FREIRE, 2007b) was designed to overcome some of the limitations presented by FFC. In relation to FFC, the main improvements brought by ACHE are on the following aspects of the crawling process: the way the link classifier acquires knowledge to guide the crawler; and the kind of form classification performed. More specifically, ACHE differs from FFC because:

- it presents the capacity of automatically adapting the behaviour of its link classifier during the crawl, automatically (re-)learning to estimate the distance (i.e., the length of the path) between a link and a target page containing a relevant form;[9] and
- it employs HIFI as form classifier, instead of GFC.

ACHE's ability to adapt its link classifier during the crawl is desirable for two reasons. First, it considerably relieves the human expert from performing heavy manual tuning. Second, it allows the exploration of areas of the Web that are unknown to the classifier (i.e., that were not well represented during its training), correcting eventual biases (MITCHELL, 1980, 1997) introduced during its training. Moreover, using HIFI instead of GFC narrows the focus of the crawler, since HIFI performs form classification both by domain and by function, while GFC performs form classification by function only. Therefore, HIFI leads ACHE to better identify relevant query forms.

Barbosa & Freire experimented with ACHE in eight domains. The measure used to assess its effectiveness was the number of *distinct relevant* query forms it retrieves as a function of the number of pages visited. Like FFC, ACHE was able to find large numbers of query forms, but with a higher precision regarding the domain of interest. The authors observed that not only forms are sparsely distributed on the Web, but also that they present large variations in density across different domains. Some domains used in their experiments presented up to 19 times as many forms as other domains.

Like HIFI, ACHE is being employed by the DeepPeep search engine (DEEPPEEP, 2010). It is possible to download anonymously and for free a form base automatically discovered by ACHE, containing thousands of forms from seven domains (DEEPPEEP REPOSITORY, 2011): `Airfares`, `Autos`, `Biology`, `Books`, `Hotels`, `Jobs`, and `Car Rental`.

2.4 Form Clusterers

Form clusterers group together query forms according to the domains to which they belong. In other words, they group together query forms that correspond to similar online databases. Given a set of n forms from k domains, an error-free form clusterer would produce as output k groups of forms, where each group contains only forms from the same domain. The decisions made by form clusterers are based on similarity (or proximity)

[9] ACHE's link classifier explores the same link features considered relevant by FFC.

Table 2.5: Summary of major characteristics of form clusterers.

	Attribute Label Identification	**Clearly Depend on the Human Expert**	**Form Features Employed**	**Identification Types**
WISE-Cluster	yes	no	ad-hoc features	by domain & by function
HT&C	yes	no	text	by domain
CAFC	no	no	text & web graph context	by domain
So&A	yes	yes	text	by domain
Z&A	yes	no	text	by domain
L&A	yes	no	text & form fields	by domain

measures that quantify how similar (or distinct) two forms are. As it is shown in the next subsections, similarity measures vary greatly from one form clusterer to another.

Unlike form classifiers, form clusterers rely on unsupervised machine learning algorithms (XU; WUNSCH D., 2005; BERKHIN, 2006); therefore, they require no training at all. On the other hand, in general, clusterers require setup or tuning of many input parameters, whose number and function relate to specific details (e.g., weights, thresholds, etc.) of the clustering approaches being employed and whose best values are frequently defined on a case-by-case basis. Even though it may be possible to setup many of these input parameters by some automatic means (e.g., through parameter estimation with genetic algorithms), the authors of form clusterers normally do not discuss how the values of such input parameters were defined in their experiments; generally, it is reasonable to assume that these values were manually defined, something that may lead to poor results if not done very carefully.

The next subsections discuss the six form clusterers found in the literature. Table 2.5 shows a summary of their major characteristics.

2.4.1 Peng et al. (WISE-Cluster)

Peng *et al.* proposed WISE-Cluster (LU et al., 2006) as a means to reach the goal of building a system which supports unified access to multiple e-commerce query forms (i.e., a meta-search engine for e-commerce Web sites).

Before clustering, WISE-Cluster filters the input forms in order to identify those which are in fact query forms. Similarly to ShopBot, HiWE, and B&C, WISE-Cluster uses an heuristic-based filtering approach, according to which query forms are expected to present: (i) an HTML `action` attribute; (ii) a submission button containing some manually defined special textual terms (e.g., *"find"*, *"search"*, *"query"* e *"quote"*); and (iii) at least a certain manually defined number of specific HTML fields.[10] This filter is assumed to block all non-query forms.

Since the meta-search engine pursued by Peng *et al.* is restricted to e-commerce query forms, domains are defined in relation to the types of products they commercialize (e.g., books, digital cameras, etc.). Therefore, WISE-Cluster explores the following ad-hoc product-oriented features:

[10]This heuristic takes into account the counting and the combined presence of various types of form fields, such as `text`, `select` and `checkbox`.

- the number of links and images in the pages that contain the query forms;
- the prices found in the pages that contain the query forms;[11]
- form attribute labels, default values in form fields, and the textual contents of the page that contain the form.

The clustering algorithm used by WISE-Cluster consists of two phases. In the first phase, which makes use of the number of links and images, the query forms are clustered into two groups, one for tangible products (e.g., books, music CDs, etc.) and another for intangible products (e.g., insurance policies). In the second phase, each group is further clustered using prices, labels, fields default values, and textual contents of the forms and their respective pages. The similarity measure used by the clustering algorithm is a weighted sum of functions that make direct use of the referred form features to compare query forms.

2.4.2 He, Tao & Chang (HT&C)

He, Tao & Chang proposed a form clusterer (hereafter named *HT&C*) that characterizes query forms by the labels of their attributes (HE; TAO; CHANG, 2004, 2005). The authors observerd that form attributes are discriminative of the domains to which their forms belong. For example, the attributes `make` and `model` are discriminative of the `Automobiles` domain, because they frequently appear in forms from this domain but seldom in forms from other domains.

The authors computed form attribute frequencies for different domains (i.e., the aggregate occurrences of an attribute across many query forms that belong to the same domain) and observed that each domain contains a dominant range of attributes, mainly distinctive from other domains.[12]

Also, the authors found empirically that the aggregate schema of query forms in the same domain tends to converge at a relatively small vocabulary with respect to the growth of the number of query forms. In other words, their experiments show that the domain-specific vocabulary growth rate decreases rapidly with respect to the increase of the number of query forms. This observation indicates that query forms from the same domain share some concerted vocabulary of attributes.

These observations led the authors to hypothesize about the existence of a hidden schema model of query forms from a specific domain, which probabilistically generates query form schemas from a finite vocabulary of form attributes.[13] In turn, this hypothesis led them to create a novel model-based form clustering algorithm, which seeks to maximize the statistical heterogeneity of form attributes among form clusters. Contrary to WISE-Cluster, which relies on ad-hoc cluster similarity measures (e.g., the number of image/links, the representative prices, etc.), HT&C takes principled hypothesis testing in statistics to evaluate if multiple clusters of data are generated from homogeneous distributions.

The experiments performed by He, Tao & Chang used approximately 500 query forms from eight domains: `Airfares`, `Musics`, `Movies`, `Books`, `Automobiles`, `Car`

[11]Prices are identified by a simple heuristic according to which a price is any number prefixed with a national currency symbol, such as Dollar $, for instance.

[12]However, as pointed out by Barbosa & Freire (BARBOSA; FREIRE, 2007a), some domains may present some degree of attribute overlap with other domains (e.g., `Movies` and `Musics` share a considerable number of form attributes).

[13]By *schema*, they simply mean an unordered set of form attributes.

`rentals` and `Hotels`. This form collection, named TEL-8, is part of the UIUC's reference form base (UIUC WEB INTEGRATION REPOSITORY, 2003), which is widely-known and frequently referenced by many related works.

2.4.3 CAFC

Barbosa, Freire & Silva proposed a clustering framework named CAFC (*C*ontext *A*ware *F*orm *C*lustering) (BARBOSA; FREIRE; SILVA, 2007), which presents a family of techniques that take into account different combinations of the following features:

- the textual contents of the query forms;
- the textual contents of the pages in which the query forms are found;
- the Webgraph (KLEINBERG et al., 1999) context for query forms, which is the link structure of the pages in which the forms are found on the Web.

Exploiting the Webgraph context makes CAFC dependable on a crawler or some mechanism able to help it build the Webgraph context for query forms. The authors prototyped CAFC using the back-link search mechanism (BHARAT et al., 1998) provided by some search engines.[14]

CAFC employs a small adaptation of the TF-IDF weighting scheme (BAEZA-YATES; RIBEIRO-NETO et al., 1999), taking into account textual term location. By doing so, different weights are applied to textual terms located at different places on a page or a form, assigning higher weights to terms that are supposedly good discriminators of the domain of a form. For instance, when processing a query form textual content, database schema can be considered more important than database contents by the assignment of lower weights to text located inside HTML `option` tags, since frequently HTML `option` tags present text related to the contents of a database.

Besides text similarity, CAFC explores a particular similarity notion that can be obtained from the link structure around form pages: the existence of common ancestors. The rationale is straightforward: if a set of pages share a common back-link, they are probably related. In other words, the existence of a hub page that points to a set of pages serves as an indication that these pages may be related.

Among other things, Barbosa, Freire & Silva learned from their experiments that the exploration of the Webgraph context of the pages that contain the forms is effective for clustering unstructured forms. This observation is remarkable, since form clusterers in general are mainly text oriented and, consequently, not very effective while clustering unstructured forms as they do not usually present enough textual terms to be processed.

2.4.4 Song et al. (So&A)

The form clusterer proposed by Song *et al.* (SONG et al., 2008) (hereafter named as *So&A*) is able to represent the semantics of the attributes of the forms to be clustered. According to the authors, no previous form clusterer was able to employ semantics to allow clustering decisions because all previous works were engaged in clustering through the computation of the cosine distance (BAEZA-YATES; RIBEIRO-NETO et al., 1999) in the VSM (SALTON; WONG; YANG, 1975), which is not appropriate to process synonyms and homonyms.

[14]It would be interesting to integrate CAFC with FFC. CAFC would help the user to find out which query forms among the ones collected by FFC really belong to the domain of interest.

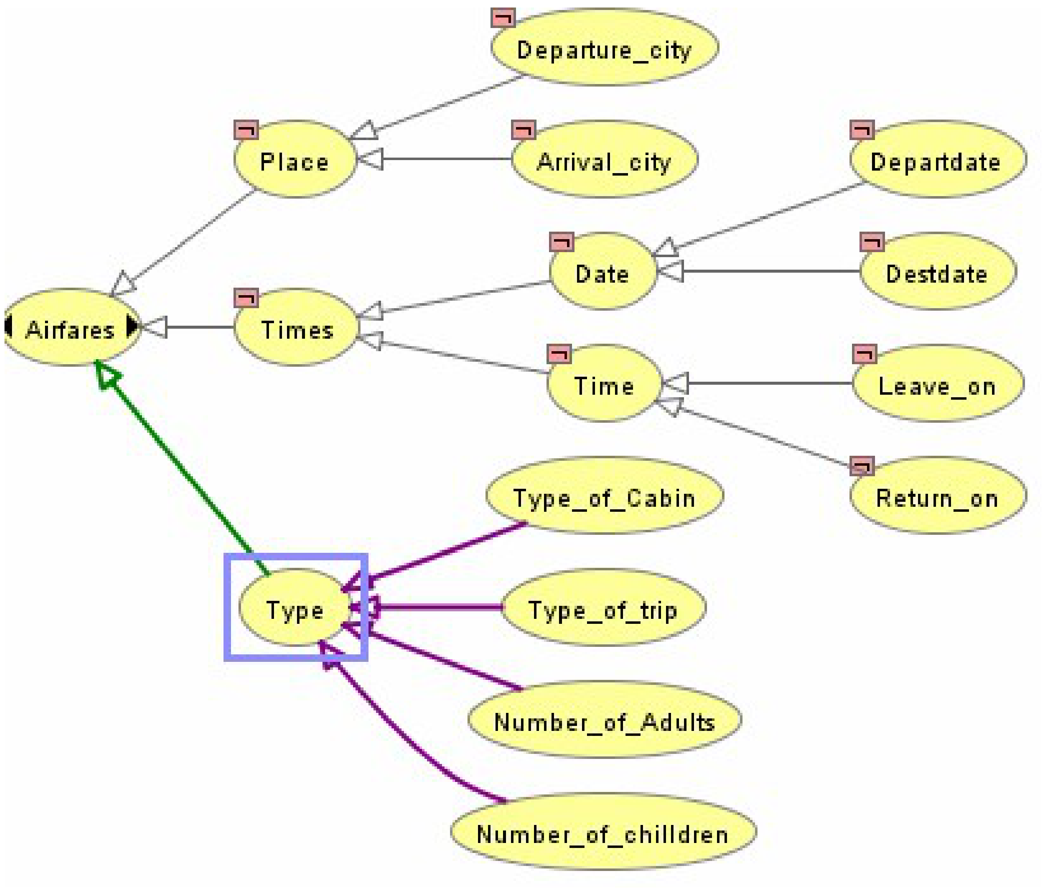


Figure 2.2: An ontology of the `Airfares` domain.

So&A calculates the similarity of two query forms using Fuzzy Logic (ZIMMERMANN, 2001). It clusters the forms using a novel machine learning algorithm which is a hybrid of the algorithms K-Means (BERKHIN, 2006) and Particle Swarm Optimization (MERWE; ENGELBRECHT, 2003).

So&A approaches semantic clustering by the use of ontologies, being based on previous research on ontology-based document clustering, which, however, were not designed to handle query forms and, therefore, are not effective to perform form clustering directly. So&A is able to cluster a set of forms F if it is parameterized with n ontologies representative of the n domains to which the forms in F belong.

The authors argue that the construction of the ontologies required by So&A does not demand much effort from the human expert, because ontologies for modeling query forms are normally very simple, for three reasons. First, there is only one kind of semantic relation present in such ontologies: the *a-kind-of* semantic relation. Second, generally these ontologies are not deep. Finally, the human expert can use tools that semi-automate the process of creating an ontology, such as OntoBuilder (GAL; MODICA; JAMIL, 2004) and Protege (NOY; FERGERSON; MUSEN, 2000). An illustrative example of an ontology representative of the `Airfares` domain is shown in Figure 2.2, where it is possible to visualize many concepts referring to airfares (e.g., places, departure dates, types of flights, etc.)

Song *et al.* do not comment about the effort required from the human expert while building the ontologies used in their experiments, but they provide some statistics about the employed ontologies. The numbers show that there were 95.5 concepts per ontology on average, leading to the impression that it may be considerably difficult to create and manage these ontologies even with the help of semi-automatic tools.

2.4.5 Zhao et al. (Z&A)

Zhao *et al.* (ZHAO et al., 2008) proposed a graph-based approach for form clustering (hereafter referred as *Z&A*), in which the nodes are the query forms to be clustered and the edges are fuzzy degrees of similarities among the nodes. The authors justify their approach by stating that traditional clustering techniques are based on binary logic, which cannot handle the inherent uncertainty in Web modeling.

Z&A clusters a set of forms using a Fuzzy Logic (BEZDEK, 1973) algorithm and a novel similarity measure, which computes the similarity between two forms F_1 and F_2 through the weighted sum of:

- the normalized number of form attribute labels shared by $F1$ and $F2$;
- the cosine measures (BAEZA-YATES; RIBEIRO-NETO et al., 1999) on the different VSMs (SALTON; WONG; YANG, 1975), which employ different combinations of features about the attribute labels and to the default values of the fields found on the forms to be clustered.

2.4.6 Li et al. (L&A)

Li *et al.* designed a statistical form clusterer (hereafter referred as *L&A*), which is mainly based on the identification of co-occurrences of textual terms and on form structure. L&A employs a novel textual feature extraction algorithm, named *F*orm *I*nformation *E*xtraction (FIE) (LI et al., 2010).

Similar to CAFC, FIE regards as relevant the location of terms present in a query form (e.g., terms located in labels, in checkboxes, in selects, etc.). This behavior of FIE is justified by empirical evidence that the location of textual terms provides good hints about how to cluster forms, since it facilitates the identification of term co-occurrences. Once the form features are extracted, L&A calculates similarities and clusters query forms through a method inspired by data-mining techniques for finding frequent itemsets (PASQUIER et al., 1999).

2.5 Form Rankers

Form rankers aim at solving the problem of identifying the query forms that are most relevant to a given user requirement. Given a set of forms F and a set of textual terms T, a form ranker assigns relevance degrees or priorities to each one of the elements of F, in relation to T.

Like form classifiers, form rankers allow the human experts to clearly define the domain of forms in which they are interested. Nevertheless, they do not demand training, since the human experts define the domain of interest through T. Table 2.6 shows a summary of their major characteristics.

2.5.1 Lin & Chen (L&Ch)

The system proposed by Lin & Chen (LIN; CHEN, 2002) (hereafter referred as *L&Ch*) presents the characteristics of both an unstructured form ranker and a DW crawler. We focus only on its form ranking characteristics because its main contributions come from

Table 2.6: Summary of major characteristics of form rankers.

	Attribute Label Identification	**Completely Reproducible**	**Discovery Tasks**	**Target Form Type**
L&Ch	no	yes	location / identification	unstructured
KL&C	yes	yes	identification	structured
Sh&A	yes	no	identification	structured

its ability to choose which forms are suitable to satisfy the users' requirements. Since L&Ch employs a traditional crawler to collect unstructured forms from the Web and fills out these forms with values manually provided by users, it cannot be stated that L&Ch innovates as a DW crawler.

Basically, what L&Ch does is to collect keyword-based forms from the Web and to redirect user-defined queries to the most appropriate (i.e., the ones likely to give the best answers) among them. More specifically, L&Ch works in four steps:

1. Before L&Ch accepts any user queries, the PIW is crawled using a traditional crawler that collects all forms it can find. No filtering or selection criteria is applied in this step, which means that all forms (i.e., query and non-query forms) are collected, regardless their domain. The forms are collected together with their textual descriptions, which include: meta-tags and titles in their pages; texts in the anchors of the links that point to their pages (i.e., back-links anchors); and their textual contents.

2. L&Ch pre-processes the user-defined input query supplementing it with keywords/phrases that can be helpful to determine which query forms to select in the next steps. While pre-processing user queries, L&Ch sends them to a PIW search engine and, through the application of rules discovery techniques (MANNILA; TOIVONEN; INKERI VERKAMO, 1997), finds words and phrases that appear often within the results returned by the search engine.

3. The most appropriate query forms to answer the supplemented query are selected. L&Ch performs this operation by ranking the forms using a keyword/phrase matching approach. In this matching task, each keyword/phrase previously generated is matched against the textual descriptions of the forms collected by the crawler and the number of matches is used to rank the forms. Nevertheless, Lin & Chen do not provide much detail on how L&Ch performs keyword/phrase matching in order to rank forms, nor do they mention how many forms among the top-ranked ones are selected by L&Ch.

4. The user-defined query is sent to the most appropriate query forms. The results of the submissions are simply redirected to the user. Similar to what happens with B&C, discussed in Subsection 2.1.3, it is not clear how L&Ch identifies the keyword-based forms at this point, since no filtering is applied to the crawled forms.

The experiments performed by the authors show that about 15% of the forms selected by L&Ch are in fact non-query forms. A possible cause of such a high error-rate is the lack of form filtering before the query form selection process.

It is worth mentioning some examples of the type of queries used by Lin & Chen in their experiments: *"3d architecture"*, *"Laser vision surgery"* and *"Sports radio stations"*.

As shown in the next subsection, these queries, suitable for a form ranker like L&Ch, which ranks only keyword-based forms, differ considerably from the ones used by form rankers that target structured forms.

2.5.2 Kabra, Li & Chang (KL&C)

Kabra, Li & Chang proposed a graph-based form ranker (hereafter referred as *KL&C*) that targets structured query forms (KABRA; LI; CHANG, 2005). Different from L&Ch, KL&C accepts input queries composed of user-defined form attribute labels. These labels are expected to be found in the query forms that the user wants to find. For instance, if the user is interested in finding query forms that belong to the `Airfares` domain, he or she could use the following input query: *("from", "to", "arrival date", "departure date")*, since these labels are frequently found in the referred domain.

The authors observed experimentally that: (i) relevant form attributes occur in relevant query forms; and (ii) form attributes that are relevant to some domain co-occur with other form attributes relevant to that domain very often. The behavior of KL&C is based on these assumptions, and on the observations made by He, Tao & Chang (see Section 2.4.2), which hypothesized that each domain is characterized by a set of form attributes, mainly distinctive from other domains.

To rank forms, KL&C executes the following steps:

1. Models a set of forms F to be ranked as a form attribute co-occurrence graph, in which the nodes are the form attribute labels found in all elements of F and the edges are weighted values in the interval [0,1], which correspond to the co-occurrence of the labels.

2. Locates in the graph the nodes that contain the labels given as input and attributes maximum relevance (i.e., 1) to these nodes.

3. Employs a novel interactive algorithm that propagates form attribute label relevances taking into account attribute co-occurrences, represented in the graph by the values previously assigned to the edges of the graph.

4. Computes the relevances of each element f of F based on the relevances of the attributes labels present on f.

Kabra, Li & Chang experimented with eight domains, reaching two interesting conclusions. The first one is that KL&C is robust regarding the incompleteness or impreciseness in the user queries. This ability is desirable because users may not be aware of all the attributes, or even all the best attributes to characterize the domain of interest. According to the experiments, KL&C achieves similar results without regard to the domain-specific variation of attributes present in the input queries. As long as the input queries *present at least some of the most relevant attributes of the domain of interest*, the results are stable. The second conclusion is that KL&C is able to correlate synonymous attribute labels, which are labels that describe the same attributes but present different syntax (e.g., *"to"*, *"destination"*, *"destination city"* and *"to city"* refer to the same data type in the the `Airfares` domain). This ability is important because labels used for the same attribute data type across different query forms are often just syntactic variations or synonyms.

2.5.3 Shen et al. (Sh&A)

The form ranker proposed by Shen *et al.* (SHEN et al., 2008) (referred hereafter as *Sh&A*) follows the same rationale used in KL&C. It ranks structured query forms, accepting as input the same sort of user query and building upon the same empirical observations about form attribute relevances and co-occurrences between relevant form attributes. Nevertheless, Sh&A approaches form ranking by the use of a novel statistical algorithm inspired by the statistical data mining technique named FP-Growth (HAN et al., 2002; HAN; PEI; YIN, 2000).

Besides attribute frequency and co-occurrences of form attributes, Sh&A takes into account user preferences over form attributes, which are based on the history of query form submissions. The authors believe that user preferences over form attributes is a good factor to determine the relative relevance of form attributes that present the same frequency and co-occur frequently. They use the following example to defend their rationale: according to the frequency and co-occurrences of form attributes observed in the `Books` domain the attributes `author` and `title` have nearly the same relevance; but, through user preference comparison between the attributes `author` and `title`, it can be assumed that `title` is more relevant, because users are usually more interested in it.

Nevertheless, Shen *et al.* do not discuss ways to automatically identify user preferences. Also, it is not clear how they took user preferences into account while performing their experiments. Apparently, user preferences were completely ignored in their experiments. This lack of discussion lead us to the conclude that their method is not completely reproducible.

2.6 Other Works

Some related work do not present any advance to the state-of-the-art in Web form discovery, as we understood them. We will mention them here with the sole purpose of completeness:

- Gong *et al.*'s DW crawler (GONG; ZHANG; LIU, 2006), which presents HiWE's goal (recall Section 2.1.2) and ShopBot's behavior (recall Section 2.1.1);
- Wang *et al.*'s focused crawler (WANG et al., 2008, 2011), which replicates many of the ideas already proposed by Barbosa & Freire (recall Sections 2.3.1, 2.3.2 and 2.2.2);
- Zuo *et al.*'s Web form similarity measure (ZUO et al., 2010) based on WordNet (FELLBAUM, 1998), which is not enough explained to be completely understood;
- Marin-Castro *et al.*'s form classifier (MARIN-CASTRO; SOSA-SOSA; LOPEZ-AREVALO, 2011), which behaves similarly to HIFI (recall Section 2.2.2).

3 MOTIVATION

Chapter 2 characterized the current state-of-the art in domain-specific form discovery based on the pre-query identification approach. It provided the basis for presenting and justifying the main motivation for this thesis, which is detailed in this chapter.

3.1 External Dependencies of Form Discovery Techniques

An external dependency is something (or someone) outside a system, on which it relies/depends. Generally, a system has no control over its external dependencies. It has to adapt to the limitations/constraints imposed by its external dependencies. Therefore, it is likely that the more external dependencies a system has, the harder it is to automatize it.[1]

This section identifies the external dependencies of the discovery techniques that were surveyed in Chapter 2. We believe this discussion is valuable for identifying the automation capabilities of the studied groups of works, which ground the discussion that is going to be elaborated in Section 3.2.

3.1.1 Dependency on Attribute Label Identification

As mentioned in Chapter 1, it is difficult to process HTML forms because they are intended to be used/consumed by humans. Automatic processing of HTML forms is hard since there is no standard layout of components and there is a vast range of possible layout patterns (KHARE; AN; SONG, 2010). Many discovery techniques depend on form attribute label identification (NGUYEN; NGUYEN; FREIRE, 2008; HE et al., 2007; ZHANG; HE; CHANG, 2004) to distinguish form attribute labels from explanatory/helper form textual contents. For example, in Figure 1.4 (a), the string *"Job Category"* is an attribute label, which gives meaning to the first combobox of the form. In the same figure, the string *"Jobs quick search"* is an explanatory or helper textual content, which does not relate directly to one of the form attributes, but helps the user to understand and/or use better the form. Form attribute label identification is an integral part of Web interface semantic understanding, which is discussed in-depth elsewhere (KHARE; AN; SONG, 2010).

As a general rule, discovery techniques that do not depend on form attribute label identification are easier to automate, because they treat form textual contents as simple bags-of-words. The first authors to observe the issues associated to this dependency were Barbosa, Freire & Silva (BARBOSA; FREIRE; SILVA, 2007), which stated that the effectiveness of form identification techniques is highly dependent on the ability to identify

[1]By *"automatic"*, we mean *"operating with minimal human intervention or external control"*.

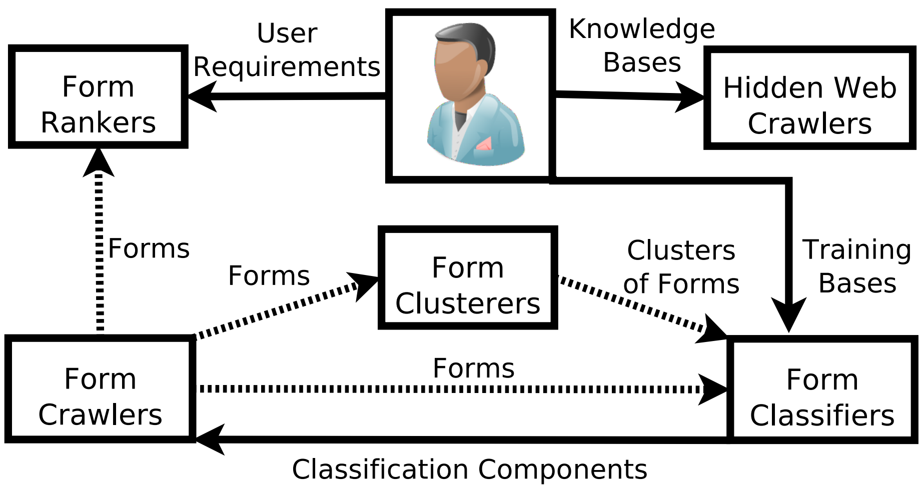


Figure 3.1: Dependency of discovery techniques on the human expert and on each other.

descriptive labels for form attributes, a task that is hard to automate. Eleven of the discovery techniques surveyed in Chapter 2 depend on form attribute label identification (i.e., ShopBot, HiWE, H&K, X&A, WISE-Cluster, HT&C, So&A, Z&A, L&A, KL&C and Sh&A); the other eight do not (i.e., B&C, CC&H, HIFI, L&Co, FFC, ACHE, CAFC and L&Ch).

3.1.2 Dependency on the Human Expert and on other Groups of Techniques

Another external dependency presented by discovery techniques is in relation to the human expert. With the exception of two techniques, B&C and L&Ch, all others directly or indirectly depend on significant human intervention. Direct dependence on the human expert occurs when a technique requires the human expert to provide some input. On the other hand, indirect dependence occurs when a technique depends on another technique which in turn directly or indirectly depends on the human expert. Figure 3.1 shows how the surveyed groups of techniques depend on the human expert and on each other. The arrows on the figure denote the dependencies, pointing to the dependent on the relations.

DW crawlers that work with structured forms require manually built knowledge bases. These knowledge bases are hard to create in an automatic way, since they have to contain details about the domain of interest that are difficult to automatically define. A good evidence of this difficulty is that the authors of ShopBot and HiWE used manually built knowledge bases in their experiments and do not mention ways to automatically create them. It is worth noticing that HiWE presents some strategies to acquire data and automatically improve its knowledge base during its crawl. Nevertheless, it still requires an initial knowledge base to start its operation.

Differently, the DW crawler B&C requires from the human expert only the manual identification of some relevant Google Directory categories, being able to automatically build its own knowledge base from the Web, through the interaction with a PIW search engine. However, B&C works solely with keyword-based forms, building its solution upon a proprietary technology, whose details are not published and, therefore, cannot be reproduced.

Form classifiers require manually labeled training bases, whose construction demand considerable effort from the human expert. HIFI and CC&H require training bases containing dozens to hundreds of forms. Compared to HIFI and CC&H, H&K and X&A

require only a small number of forms to be trained, but they demand a much more detailed labeling process, which amounts to a very laborious training task. H&K requires manual labeling of form attributes. In addition to that, X&A requires manual identification of the semantic relations between form attributes, and the specificity and relevance of a form attribute regarding a specific domain.

Additionally, for practical reasons, form classifiers can be used to classify the output of form clusterers or form crawlers, in order to try to improve the process of form identification executed by these techniques. Therefore, it can be said that form classifiers may present some sort of dependency on form clusterers or form crawlers, if we consider the form identification process as a whole. Nevertheless, nothing forbids a form classifier to directly process forms that were not clustered nor automatically collected by a form crawler. These optional dependencies are denoted in Figure 3.1 by dashed arrows.

Form crawlers are built upon form classifiers, which they use as internal components. Therefore, it can be stated that form crawlers are at least as dependent on the human expert as form classifiers are. Besides, form crawlers depend on other supervised classifiers (i.e., page and link classifiers), too. This additional dependency is weaker in ACHE than in FFC, since ACHE's link classifier is able to automatically learn how to adjust its behavior during the crawl; however, ACHE still heavily depends on the human expert.

Form clusterers and form rankers accept as input a set of forms. Since they do not address the issues associated with the task of locating forms on the Web and the manual gathering of forms on the Web is a hard task, it can be said that they present a practical (but optional) dependency on form crawlers. Besides, form rankers require the human expert to define a set of requirements to guide the ranking process.

3.1.3 Dependency on Publicly Available Form Bases

The existence of publicly available form bases, such as the ones built by the researchers of the Meta-querier (TEL-8) (UIUC WEB INTEGRATION REPOSITORY, 2003) and the DeepPeep (DEEPPEEP, 2010; DEEPPEEP REPOSITORY, 2011) projects, surely alleviates the work required from the human expert while experimenting and/or parameterizing discovery techniques. In fact, the vast majority of the experiments performed by the authors of the surveyed techniques employed the TEL-8 form base.

However, such bases are composed of query forms from a limited predefined set of domains. Also, these bases supposedly contain query forms written only in English. They are of no avail if experiments deal with other languages or within domains not available in these bases (e.g., the `Real Estate` domain). Furthermore, one cannot be certain that these knowledge bases will remain publicly available forever.

3.2 Increase in Automation: an Evolutionary Trend in Domain-specific Structured Query Forms Discovery

Comparisons among techniques belonging to distinct groups are not suitable as most of them are cooperative rather than competitive. For example, form classifiers could be used to improve form clustering results or form rankers could be used to find out the most relevant forms among the ones collected by a form crawler. It would only be meaningful to compare intra-group techniques. Below we provide a discussion (backed-up by Table 3.1) on the identification of an evolutionary trend in form discovery: increase in automation.

Table 3.1: Chronology of remarkable events in form discovery.

Year	**Remarkable Events in Form Discovery**
1997	– ShopBot is the first DW crawler.
2001	– HiWE is the first DW crawler able to automatically improve its own knowledge-base.
2002	– L&Ch is the first ranker of unstructured forms.
2003	– B&C is the first automatic DW crawler (for unstructured forms only); – CC&H and H&K are the first form classifiers; – H&K is also the first form identifier to work with semantics.
2004	– HT&C and WISE-Cluster are the first form clusterers.
2005	– FFC is the first form crawler; – KL&C is the first ranker of structured forms.
2007	– HIFI and CAFC are the first form identifiers to circumvent form attribute label identification; – ACHE is the first adaptive form crawler.
2008	– So&A constitutes an effort toward semi-automatic semantic form identification.
2009	– L&Co improves identification of forms from domains that present overlapping schemas.

3.2.1 Past and Present

ShopBot and HiWE, the first DW crawlers (and the pioneer form discoverers), are relatively limited in their ability to locate and identify forms for three reasons: (i) their in-site crawlers make use of small sets of manually-defined seed URLs; (ii) they heavily rely on manually-defined knowledge bases and heuristics; and (iii) they depend on form attribute label identification. The first reason severely restrains their ability of locating forms on the Web, since they are restricted to the initial set of seed URLs. The second reason restrains their scalability, since it is hard to provide accurate descriptions of the huge variations in text and structure that can be found in Web forms. The third reason characterizes an undesirable external dependency that adds complexity to automation, as discussed in Section 3.1.1.

L&Ch and B&C, which appeared just a few years after the first DW crawlers, are more automated because they automatically learn from scratch domain-specific knowledge from the Web. Unfortunately, L&Ch and B&C are unsuitable to work with structured forms.

Relatively to DW crawlers, the development of all other groups of works (i.e., form crawlers, form classifiers, form clusterers and form rankers) brought more automation to the discovery of structured query forms, since their algorithms internally/implicitly learn/acquire domain-specific knowledge directly from the forms they process. Consequently, they expect from the human expert sets of (un)labeled forms to operate rather than detailed sets of domain-specific labeled values.

H&K was the first form identifier that took the semantics of form attributes into account. So&A followed H&K's lead. HIFI and CAFC were the first structured form identification techniques to avoid the need for form attribute label identification. After them, came L&Co.

These facts evidence a growing trend in the automation of structured form discovery techniques. Discovery techniques are noticeably evolving from manual to completely unsupervised. In the early years, form location methods did not scale, lots of detailed definitions were demanded from the human expert, and form attribute label identification was a must for techniques that exploited renderizable textual contents of forms. Gradually, though, this reality changed: (i) form crawlers were introduced, adding scalability to the location of forms on the Web; (ii) (un)supervised algorithms were employed, freeing the human expert from knowing the domain of interest in detail; and (iii) the dependency on form attribute label identification was circumvented.

3.2.2 Brief Note About the Domains Being Covered by Form Discoverers

We are aware of only one limitation imposed on domains covered by domain-specific form discoverers: the forms that belong to them should be clearly and easily identifiable among random forms by a human expert. Thus, the forms of any domain restrained by this limitation have potential for being automatically discovered on the Web.

Several are the domains used in the experiments of the surveyed works (e.g., `Softwares` (PERKOWITZ et al., 1997), `CD Products` (PERKOWITZ et al., 1997), `Colleges` (KUSHMERICK, 2003), `Stocks` (KUSHMERICK, 2003), `Biology` (DEEPPEEP REPOSITORY, 2011)), but there is a relatively small set of domains that have been used more frequently than others (i.e., `Airfares`, `Autos`, `Books`, `Car Rentals`, `Hotels`, `Jobs`, `Movies` and `Music`). Three are the main reasons for this fact. Firstly, they are very common (i.e., their respective forms are abundant) on the Web. Secondly, they present different degrees of similarities among themselves: some of them contain resemblant forms (e.g., `Airfares` and `Hotels`), while others contain forms that are quite disparate (e.g., `Airfares` and `Jobs`). Finally, they were sampled in a widely known form base named TEL-8 (as mentioned in Section 3.1.3), which is handy for form discovery experimentation.

3.2.3 Future Forecast

Table 3.1 shows a clear trend towards increasing the use of statistical machine learning techniques over time, as opposed to other forms of artificial intelligence, such as rule-based methods. The choice for machine learning is perhaps an obvious one, given the highly heterogeneous environment which is the Web. It is hard to conceive a manually defined rule-based system that can cope with the vast, complex and volatile Web environment. It is thus expected to see even more sophisticated machine learning techniques being developed and used to tackle the form discovery problem. However, as observed by many authors (recall, e.g., Section 2.2), the accuracy of machine learning tools depends greatly on choosing the right set of features to represent the objects of interest (the forms in this case), which is an inherently empirical process. Such explorations can start with a careful study of which features contribute the most to the result of the machine learning algorithm (WITTEN; FRANK; HALL, 2011).

The holy grail in statistical machine learning is to have learning without any training or supervision from a human. The area of reinforcement learning (MITCHELL, 1997) deals with this problem, and proposes a generic framework for a fully autonomous learning: we need to formalize the space of actions that the agent can perform, and define a reward function that provides feedback to the learner. If the learner makes a good decision, the reward function must return a positive outcome; conversely, when an incorrect decision is made, the function must penalize the learner with a negative reward. Reinforcement learning, however, is not a solved area. In fact, computationally defining the space of actions to be explored by the learner and the reward function are often very hard to do in practice. In the context of form discovery, as in most practical cases, the feature space for the learner is essentially infinite – bringing the challenge of how to explore it effectively and efficiently. One could conceive a reward function that rewards the learners proportionally to the number of new data items that they extract after each step in its exploration. However, this brings several difficulties, such as determining whether or not the results of each extraction contain valuable data, which is far from trivial. Nevertheless, it seems clear that ever more sophisticated machine learning techniques will be deployed for form discovery.

Another trend identifiable in Table 3.1 points towards the reduction or elimination of the dependence on human expertise that is still required in the state-of-the-art, as discussed in Section 3.1.2. This is one area that has received a great deal of attention from the artificial intelligence research community as well, as machine learning gains traction as the tool of choice for complex data analysis problems. One effective tool in this area, which has been under-explored in the current crop of tools, is combining different classifiers, in an attempt to complement their strengths (WITTEN; FRANK; HALL, 2011). Another option is to use co-training (BLUM; MITCHELL, 1998), which is a semi-supervised method where labeled and un-labeled data are mixed together and classified independently using different features; the resulting classifier is obtained by carefully combining the individual predictions. Co-training could be employed, for instance, to forms classified independently by function and by domain. Another technique that might prove valuable in this context is boosting (SCHAPIRE, 2003), in which several “rough” predictions are combined, producing an accurate prediction.

Schema.org (COLLECTION OF SCHEMAS FOR SEARCH PROVIDERS, 2011), a recent development led by major commercial search engines Google (GOOGLE SEARCH ENGINE, 2011), Bing (MICROSOFT WEB SEARCH ENGINE, 2011), and Yahoo! (YAHOO! WEB SEARCH ENGINE, 2011), aims at improving search results by allowing Web masters to annotate the content from their sites with semantic information that is harvested by these search engines. Schema.org already contains a fairly rich type hierarchy defining the most common kinds of objects on the Web, such as people, organizations, events, products, and locations. Web masters can annotate their HTML pages with the corresponding types, and are also free to suggest new types to be added to the repository. Schema.org is a voluntary effort, and does not define a standard. Nevertheless, if adopted widely, it holds the promise to greatly help in understanding the semantics of the content in the Web pages (and forms as well). One contentious issue with Schema.org is their choice of departing from W3C standards; instead, the search engines decided to propose their own microdata format for this initiative.

3.3 The Proposal

The current state-of-the-art evidently lacks methods for domain-specific structured form discovery that require little intervention from the human expert in order to operate. In this thesis, we present our effort to mitigate this deficit. Our final goal is to create a method for domain-specific structured form discovery that depends as little as possible on the human expert. More specifically, we propose a method that require from the human expert only the name of the domain of interest as input, being able to create training bases for two state-of-the-art form classifiers in a completelly automatic fashion. As far as we know, this thesis constitutes the first effort in this direction[2].

We empirically validate our proposal of automatically creating training bases for form classifiers through the comparison of two scenarios: (i) one in which two state-of-the-art form classifiers are trained with examples manually defined; and (ii) other in which the same form classifiers are trained with examples automatically defined by our heuristics. The results of our experiments show that our method is indeed capable of automatically training the experimented form classifiers at the cost of a relatively small loss in performance.

[2]It could be objected that being based on heuristics is actually a negative point of our proposal but we are unaware of a better way to approach automatic training, since heuristics carry in their own definitions all the knowledge they need to operate and therefore, do not have any dependences on the human expert.

4 EMPIRICAL SETUP: FORM BASES AND QUALITY MEASURES

This chapter presents our solution for locating forms on the Web in order to achieve our final goal in this thesis. It also presents the auxiliar tools that we used in order to validate our proposal through the experiments described in the following chapters.

4.1 The Proposed Method for Form Location

The form locator proposed by Bergholz & Chidloviskii (BERGHOLZ; CHILDLOVSKII, 2003) inspired our own solution for the location of forms on the Web. The main difference between their solution and ours is that theirs uses a Web Directory as the source of relevant URLs to crawl, while ours uses a general-purpose Web search engine (e.g., Google, Yahoo!, Bing, etc.). Like theirs, our form locator performs limited-depth in-site crawling[1], since there are empirical evidence that the vast majority of query forms can be found up to three links in depth from the home page of a Web site[2] (BERGHOLZ; CHILDLOVSKII, 2003; CHANG et al., 2004).

Notice that we are not necessarily interested in a scalable and/or high-performance form locator, since our goal is to create a method for domain-specific structured form discovery that depends as little as possible on the human expert in order to operate. Therefore, we assume that we can build our proposal on top of a simplistic form locator, whose main function is solely to locate on the Web a few hundreds of forms while requiring very little intervention from the human expert. We call our form locator a simplistic one because it relies entirely on a general-purpose Web search engine, which actually performs the hard work of matching HTML Web pages to a user-defined query. This kind of form location provides us with an inexpensive and easy way to reach hundreds of forms, which amount to a sufficient quantity for automatically creating form classifier training bases.

The input of our form locator is a small set of manually defined keywords that describe the domain of interest. This input is sent directly as a query to a general-purpose Web search engine and the resulting URLs are used as seed URLs (i.e., the initial set of URLs) for our limited-depth in-site crawler, which collects some of the forms it can find in a Web site.

Our fundamental assumption while locating forms this way is that general-purpose Web search engines are able to return a significant number of relevant URLs (i.e., URLs

[1]In-site crawlers are restricted to (i.e., follow only) the links that point to pages that belong to the same site.

[2]See (CHANG; HE; ZHANG, 2005) for another example of form locator that executes limited-depth in-site crawling.

Table 4.1: The forms located by our proposal.

Domain	Description	Located Forms	Query Forms	On-topic Forms
`Airfares`	flight availability	502	353	240
`Autos`	automobiles for sale	504	286	155
`Books`	books, ebooks, papers and magazines	504	344	161
`Car Rentals`	cars for rental	501	318	183
`Hotels`	hotel rooms availability	501	398	280
`Jobs`	job offerings	503	331	282

pointing to Web sites that contain relevant forms) when queried with keywords that properly describe the domain of interest. By significant number of relevant URLs, we mean enough to allow effective automatic form identification by the methods that we are going to discuss in the following chapters.

4.2 Experiments

To simplify our prototype, we assumed that the URLs returned by the search engine pointed directly to the home page of their respective Web sites, so that we have a point of reference to measure the depth of the crawl and use it as stop condition for the crawler operation. To avoid locating unstructured forms, which are outside the scope of this thesis, only one form among those collected in a Web site may be composed of a single input field. Our empirical observations pointed out that a single input field is the main characteristic presented by unstructured forms in general. Also, to avoid the discovery process to become biased by Web sites that present many similar forms, our crawler does not collect more than five forms per Web site.

Through the interaction with the Yahoo! search engine (YAHOO! BOSS API, 2011), our prototype located approximately 3000 forms from six domains, which amounted to approximately 500 forms per domain. Table 4.1 provides the details about the experimented domains, their descriptions, and their respective located forms, which were manually classified by function and by domain.

The keywords sent to the search engine were literally the names of the domains of interest, exactly as they are shown in Table 4.1 (i.e., *"airfares"* for the `Airfares` domain, *"autos"* for the `Autos` domain, and so on.). The Query Forms column indicates the number of query forms found among the ones located for each domain. The On-topic Forms column indicates the number of query and non-query forms written in English that belong to their respective domains.

Notice that we considered as relevant only the forms that were written in English. Forms from the domains of interest but written in other languages were considered misidentified. Manual validation was performed using Mozilla Firefox configured to automatically determine the character encoding of a given HTML page and, in case of failure, to use the default character encoding ISO-8859-1. The forms described in Table 4.1 were used as material for the experiments that will be discussed in the following two chapters.

Table 4.2: Descriptive statistics of the publicly available form base of the DeepPeep project.

Domain	**Total Forms**	**Distinct Forms**	**Sample Size**	**Query Forms (Error)**	**On-topic Forms (Error)**
`Airfares`	3,641	1,955	1,307	1,264 (3.3%)	1,240 (5.2%)
`Autos`	9,972	7,565	2,111	1,846 (12.6%)	956 (54.8%)
`Books`	2,035	1,809	1,305	1,266 (3.0%)	1,130 (13.5%)
`Car Rentals`	2,515	1,995	1,316	1,140 (13.4%)	906 (32.2%)
`Hotels`	20,303	13,417	2,361	2,172 (8,1%)	1,490 (36.9%)
`Jobs`	14,958	11,573	2,261	1,896 (16.2%)	1,088 (51.9%)

4.3 Auxiliar Experimentation Tools

This section presents two experimentation tools that helped us to empirically validate our thesis, as described in the following chapters. One tool is a subset of a huge form base collected independently by the form discoverer of the DeepPeep project (DEEPPEEP, 2010), which provided us with reference data for method comparison. The other tool is a set of metrics widely known by researchers of Information Retrieval, which can be used as means to compare the effectiveness of different form discovery methods.

4.3.1 A Third-party Form Base

We built another form set from a manually classified random sample of the freely available DeepPeep Project form base (DEEPPEEP REPOSITORY, 2011; DEEPPEEP, 2010), which was automatically built by the DeepPeep form discoverer (BARBOSA; FREIRE, 2007b). For convenience, hereafter we refer to the former (i.e., the form base described in Table 4.1) as *the small test set* and to the latter (i.e., our manually classified random sample of the DeepPeep form base) as *the big test set.* To build the big test set, we filtered the DeepPeep form base for the removal of replicas, as replicas were identified through two *renderizable properties* of forms (i.e., properties that can be rendered by a web browser). More specifically, we considered replicas the forms that present: (i) identical *renderizable texts* (i.e., the form itself minus its HTML tags); and (ii) the same amounts of *renderizable fields* (i.e., all fields except the hidden ones). There is no point in allowing replicas in the test base, since it is likely that the more disparate the forms of the test base, the more accurately the effectiveness of the form identifier tested against it is measured.

The big test set is detailed in Table 4.2. The Total Forms column indicates the total number of forms present in the DeepPeep base as it was downloaded from one of its authors Web site (DEEPPEEP REPOSITORY, 2011). The Distinct Forms column indicates the number of distinct (i.e., non-replicated) forms found on the same base. The Sample Size column indicates the sizes of the random samples, which were built according to a confidence level of 95% and a confidence interval of 2%. The semantics of the columns Query Forms and On-topic Forms are the same of Table 4.1.

Notice that the form discoverer of the DeepPeep Project is supposed to discover only domain-specific query forms (BARBOSA et al., 2010). In other words, no non-query form was expected to be found in the big test set. Also, among the forms attributed to a

Table 4.3: Structure of a confusion matrix.

	Predicted pos.	Predicted neg.
Actual pos.	True Pos.(TP)	False Neg.(FN)
Actual neg.	False Pos.(FP)	True Neg.(TN)

domain in the big test set, only the ones that actually belong to that domain were expected to be found. Therefore, since we found both non-query forms and forms that do not belong to their respective domains in the big test set, it was possible to attribute an error-rate to the DeepPeep form discoverer with respect to both identifications by function and by domain. These error rates are shown in Table 4.2 between parentheses in the cells of the columns Query Forms and On-topic Forms, respectively.

We are convinced that the errors in the identification peformed by DeepPeep's discoverer were due to two main reasons. First, it had to identify forms from domains that present vocabulary overlap (i.e., they share words of their characteristic vocabularies (BARBOSA; FREIRE, 2007a), as it will be discussed in more details in Chaper 6). Second, it had to identify lots of forms that actually belonged to the domain of interest, but that were written in a language other than English[3][4].

4.3.2 Quality Measures

The measures employed in our experiments were the same used in the evaluation of several related works (BARBOSA; FREIRE, 2007a; COPE; CRASWELL; HAWKING, 2003; KUSHMERICK, 2003; LE; CONRAD, 2009; XU et al., 2007). These measures are based on a confusion matrix (see Table 4.3), which represents the relationship between actual and predicted classification. The measures are the following:

$$Precision(P) = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall(R) = \frac{TP}{TP + FN} \tag{4.2}$$

$$F1 = \frac{2 * Recall * Precision}{Recall + Precision} \tag{4.3}$$

$$Accuracy(A) = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.4}$$

$$Specificity(S) = \frac{TN}{TN + FP} \tag{4.5}$$

Accuracy alone is a suitable measure when the input to the classifier contains similar proportions of positive and negative examples, which is not the case in the experiments

[3]Many forms from the domains of interest were written in German, Spanish, French, and Japanese.

[4]As occurred for the small test set, the manual validation of the big test set was performed using Mozilla Firefox configured to automatically determine the character encoding of a given HTML page and, in case of failure, to use the default character encoding ISO-8859-1.

we performed in this thesis. *Recall* shows the number of relevant items retrieved as a fraction of all relevant items. *Precision* shows the number of relevant retrieved items as a fraction of all items predicted as positive. *Specificity* shows the proportion of actual irrelevant items predicted as irrelevant. Finally, *F1* is the harmonic mean between *recall* and *precision*. The perfect classification would result in *F1* equal to 1. Since our test bases do not contain similar proportions of positive and negative examples, we believe that *F1* is the most important measure for us.

5 IDENTIFYING FORMS BY FUNCTION

This chapter presents and combines a set of heuristics for form identification by function. The heuristics presented here are not novel, as they were previously employed by several other authors. The novel aspect of this thesis in this regard is the way we combine these heuristics, providing a more complete understanding about their importance for the process of form identification by function. Other authors used them isolatedly as secondary means for the process of form identification, hardly measuring their importance. On the contrary, we use them as the core of our solution, measuring their importance when isolated and when combined.

5.1 Relevant Form Features

Inspired by previous work (BERGHOLZ; CHILDLOVSKII, 2003; DOORENBOS; ETZIONI; WELD, 1997; RAGHAVAN; GARCIA-MOLINA, 2000; LU et al., 2006; GONG; ZHANG; LIU, 2006), our heuristics took into account the following three form features:

1. *The type of the HTTP submission method.* According to the HTTP specification, the `get` submission method should be used for requesting read-only operations from an HTTP server (HTTP, 2011), while other types – among which `put` is the most commonly used – should be used for requesting write operations. Since the main characteristic of a query form is that it allows the execution of read-only operations (i.e., queries or searches) in an on-line database, the observance of the HTTP submission method arguably constitutes valuable information for form identification by function[1]. This feature was previously explored by Madhavan *et al.* (MADHAVAN et al., 2008);

2. *The presence/absence of password fields*, which are most usually employed for authentication. Since authentication and queries on on-line databases do not conceptually mix (i.e. in concept, they are completely unrelated distinct operations), it is hard to think about realistic scenarios in which password fields could be used to query on-line databases. The presence/absence of password fields was previously explored heuristically by Bergholz & Chidloviskii (BERGHOLZ; CHILDLOVSKII, 2003);

3. *The presence/absence of the word "search" in the body of a form.* The observance of this feature for the purpose of query form identification is not as intu-

[1]Though the experiments presented in this chapter show empirical evidence that HTML form designers do not care much about this usage guideline.

ituive as the previous ones, since there are no conceptual ties between the presence/absence of the word *"search"* and query forms. Nevertheless, other authors empirically observed that this feature is a strongly indicative of query forms (BARBOSA; FREIRE, 2007a; COPE; CRASWELL; HAWKING, 2003). This feature was previously explored heuristically by Peng *et al.* (LU et al., 2006)[2].

5.2 The Proposed Method

The three form features previously listed provide us with the basis for the definition of our seven heuristics for form identification by function. Three of these heuristics take into account a single form feature; the other four combine more than one form feature using the Boolean operator `AND`. The seven heuristics are defined below. They identify any given form as a query form if it:

- $(heuristic\#1)$ uses the HTTP `get` submission method;
- $(heuristic\#2)$ does not present a password field;
- $(heuristic\#3)$ presents the word *"search"*;
- $(heuristic\#4)$ is identified as a query form by heuristics #1 and #2;
- $(heuristic\#5)$ is identified as a query form by heuristics #1 and #3;
- $(heuristic\#6)$ is identified as a query form by heuristics #2 and #3;
- $(heuristic\#7)$ is identified as a query form by heuristics #1, #2 and #3.

5.3 Experiments

This section presents our experiments using the referred seven heuristics for form identification by function. First, we identify the most effective among them. Then we use the most effective heuristic to train a state-of-the-art form classifier by function and compare automatic *vs* manual training.

5.3.1 Measuring Effectiveness

In order to measure the effectiveness of the seven heuristics previously defined, we ran each one of them directly against the small and the big test sets. The results of these runs are shown in Tables 5.1 and 5.2, respectively.

The reported numbers show heuristics #2, #3 and #6 as the most effective for form identification by function. Such results constitute empirical evidence that the second and the third of the experimented form features are considerably more effective to query form identification by function than the first one. Consequently, these results led us to induce that HTML form designers do not care much about the correct usage of the HTTP `get` submission method. Since heuristic #6 was the one that presented the best $F1$ results for the small test set, we used it in all subsequent experiments that relied on our method for query form identification.

[2] Peng *et al.*'s method detect the presence of the word *"search"* inside form submission buttons, instead of detecting it in the whole body of a form.

Table 5.1: Heuristic-based identification by function against the small test set.

Heuristic	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
#1	0.535	0.924	0.678	0.909	0.657
#2	**0.989**	**0.722**	**0.835**	**0.218**	**0.737**
#3	**0.882**	**0.884**	**0.883**	**0.763**	**0.843**
#4	0.534	0.928	0.678	0.915	0.659
#5	0.497	0.976	0.659	0.975	0.653
#6	**0.873**	**0.914**	**0.893**	**0.831**	**0.860**
#7	0.4965	0.9767	0.6583	0.975	0.653

Table 5.2: Heuristic-based identification by function against the big test set.

Heuristic	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
#1	0.409	0.982	0.578	0.935	0.462
#2	**0.988**	**0.907**	**0.946**	**0.110**	**0.899**
#3	**0.858**	**0.959**	**0.906**	**0.677**	**0.840**
#4	0.408	0.983	0.577	0.939	0.462
#5	0.365	0.992	0.534	0.974	0.427
#6	**0.8497**	**0.9680**	**0.905**	**0.750**	**0.839**
#7	0.3650	0.9928	0.5338	0.976	0.427

5.3.2 Comparing Manual and Automatic Training for GFC

In order to show that our heuristic-based method for form identification by function (i.e. query form identification using heuristic #6) achieves our final goal of mitigating the manual work required from the human expert while discovering forms, we performed experiments with the cutting-edge form classifier GFC (BARBOSA; FREIRE, 2007a).

Like other form classifiers, GFC demands training with dozens to hundreds of manually labeled forms. Its training can be regarded as domain-independent, because it explores only structural form features which do not vary substantially across domains. The structural form features considered relevant by GFC are: (i) the number of certain HTML types of form fields (e.g., `text`, `select`, `radio`, etc.); and (ii) the presence of the word *"search"* within a form. We implemented GFC according to the specification given by its authors using the decision tree algorithm available in the widely known and freely distributed Weka library (HOLMES; DONKIN; WITTEN, 1994).

In our experiments, GFC's effectiveness was measured in two scenarios. In the first scenario, we manually trained GFC with random query forms from all domains of the big test set. We used 220 positive examples and 220 negative examples to train GFC in each execution, which are amounts similar to the ones used in the experiments done by its authors (BARBOSA; FREIRE, 2007a). The results reported for this scenario represent the average of 100 executions using such training configuration, where the forms of the training set were excluded from the test set.

In the second scenario, GFC was trained with the forms located by our form locator and identified by heuristic #6. In other words, in the second scenario, the forms of the small test set identified as query forms by heuristic #6 were used as positive examples and the other forms of the same test set were used as negative examples to train GFC.

Both scenarios were tested against the big test set, which, as described in Table 4.2, is composed of 9584 query forms and 1077 non-query forms. The results are shown

Table 5.3: GFC's effectiveness against the big test set.

Scenario	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
First	0.907	0.986	0.945	0.867	0.904
Second (trained for `Airfares`)	0.858	0.959	0.906	0.677	0.840
Second (trained for `Autos`)	0.855	0.963	0.906	0.713	0.841
Second (trained for `Books`)	0.851	0.967	0.905	0.744	0.840
Second (trained for `Car Rentals`)	0.858	0.959	0.906	0.677	0.840
Second (trained for `Hotels`)	0.849	0.968	0.905	0.750	0.839
Second (trained for `Jobs`)	0.849	0.968	0.905	0.750	0.839

in Table 5.3. The second scenario was tested in each one of the domains detailed in Table 4.1, since they provide different automatically created training bases for GFC (i.e., one training base for each set of forms associated by our form locator to each domain described in Table 4.1). The similar results obtained in the two referred scenarios provide empirical evidence that GFC keeps most of its effectiveness when trained by our heuristic-based method, relatively to when it receives manual training: the losses summed up to approximately 0.04 points in the F1 measure.

6 IDENTIFYING FORMS BY DOMAIN

In this chapter, we elaborate and evaluate an heuristic-based method for form identification by domain that is built upon the form identifier by function presented in Chapter 5. Inspired by previous work (PERKOWITZ et al., 1997; RAGHAVAN; GARCIA-MOLINA, 2000; BERGHOLZ; CHILDLOVSKII, 2003; BARBOSA; FREIRE, 2007a; KUSHMERICK, 2003; XU et al., 2007; LE; CONRAD, 2009; LU et al., 2006; HE; TAO; CHANG, 2004; BARBOSA; FREIRE; SILVA, 2007), we regard form textual contents as the sole form feature to be explored, since they normally reflect both the schema and the contents of the underlying data-sources, providing important hints about their respective domains.

6.1 Basic Behavior of the Proposed Method

Our fundamental assumption is that the vocabularies that characterize the labels of the attributes of domain-specific structured forms: (i) tend to converge at relatively small size, and (ii) are discriminative of their domains. In other words, we assume that there are small vocabularies that characterizes the attribute labels frequently found in domain-specific forms. This assumption is backed-up by the work of He, Tao & Chang (HE; TAO; CHANG, 2004, 2005), which counted attribute occurrences in several domains. They found empirical evidence that, in general, domains contain dominant sets of attributes, which are distinctive from other domains.

As an example of the existence of domain-specific vocabularies, it can be stated that it is common to find some words like *“model”*, *“price”* and *“make”*, among others, in both query and non-query forms of the `Autos` domain, but these words are not usually found in the renderizable texts of forms of other domains. Nevertheless, these words may eventually be characteristic of other domains, too. If this happens, it is said that these domains present vocabulary overlap (BARBOSA; FREIRE, 2007a). Three domains among the ones experimented in this thesis present vocabulary overlap: `Airfares`, `Car Rentals` and `Hotels`. As the results of our experiments show, is is not difficult for a form domain identifier to confuse forms that belong to these domains.

Based on our fundamental assumptions, we elaborated two complementary heuristics for form identification by domain. The first heuristic is able to recognize at least some of the words that characterize the attribute labels frequently present in the forms of the domain of interest, by means of measuring the occurrences of the renderizable words present in domain-specific query forms and selecting the most frequent ones.

Algorithmically, the main steps of the first heuristic are detailed as follows, where α is a manually defined threshold given as input parameter whose value belongs to the interval $[0, 1]$ (i.e., $\alpha >= 0$ and $\alpha <= 1$):

1. Locate a set of forms from the domain of interest using the form locator presented in Chapter 4;
2. Using the method presented in Chapter 5 (i.e., using heuristic #6 for form identification by function), identify the query forms among the ones located in step #1;
3. Compute the occurrences of the words present in the renderizable texts of the query forms identified in step #2;
4. Return the set of words that occur in at least α (percentual) of the query forms identified in step #2, which constitutes the vocabulary that characterizes the domain of interest (i.e. the domain related to the word given as input to the form locator in step #1).

Once the vocabulary that characterizes the domain of interest is defined by the described algorithm, our second heuristic for form identification by domain is used to identify any given form as belonging to the desired domain or not. The rationale of our second heuristic is straightforward: a form is considered as belonging to the domain of interest if it contains at least N of the words present in the vocabulary that characterizes the domain of interest; where N is a manually defined input parameter.

It is worth mentioning that the referred two heuristics are independent of form attribute label identification techniques because they process the renderizable texts of the forms as bags-of-words (see Section 6.1.1.1 for illustrative examples). Since form attribute label identification techniques are hard to automate (recall Section 3.1.1), this characteristic is desirable for an automatic form identification method like ours, which aims at mitigating the manual involvement required by the current state-of-the-art in form discovery.

6.1.1 Experiments

This section presents our experiments using the referred two heuristics for form identification by domain. First, we measure their effectiveness directly against our test bases. Then we use them to train a state-of-the-art form classifier by domain and compare manual *vs* manual training. In all experiments described in this thesis regarding form identification by domain, all words of the renderizable texts of all forms were stemmed before having their non-alpha-ascii characters removed.

6.1.1.1 Discovering Vocabularies

For an illustrative purpose, Table 6.1 shows the vocabularies that our first heuristic for form identification by domain associated to each domain when it processed the small test set. Some of the words appear truncated or misspelled because, as it was stated before, all words were stemmed before having their non-alpha-ascii characters removed. Notice that the domains `Airfares`, `Car Rentals` and `Hotels` present vocabulary overlap, as they have at least two words in common: *"citi"* and *"date"*[1]. As some of the results presented in the following sections show, this appears to be a major source of confusion for text-based form identifiers.

6.1.1.2 Observing and Defining α and N

In order to observe the impact of different values of α and N on the effectiveness of the proposed method for form identification by domain, we run it many times against the

[1] `Airfares` and `Hotels` also share the word *"adult"*.

Table 6.1: Vocabularies that characterize the experimented domains, as defined by our first heuristic for form identification by domain.

Domain	**Vocabulary**
`Airfares`	*"flight"*, *"depart"*, *"return"*, *"children"*, *"adult"*, *"travel"*, *"date"*, *"citi"*
`Autos`	*"model"*, *"price"*, *"make"*, *"toyota"*
`Books`	*"author"*, *"titl"*, *"book"*
`Car Rentals`	*"car"*, *"airport"*, *"locat"*, *"pickup"*, *"pm"*, *"date"*, *"citi"*
`Hotels`	*"checkin"*, *"adult"*, *"hotel"*, *"date"*, *"citi"*, *"room"*
`Jobs`	*"locat"*, *"manag"*, *"servic"*, *"keyword"*, *"engin"*, *"job"*, *"educ"*, *"sale"*

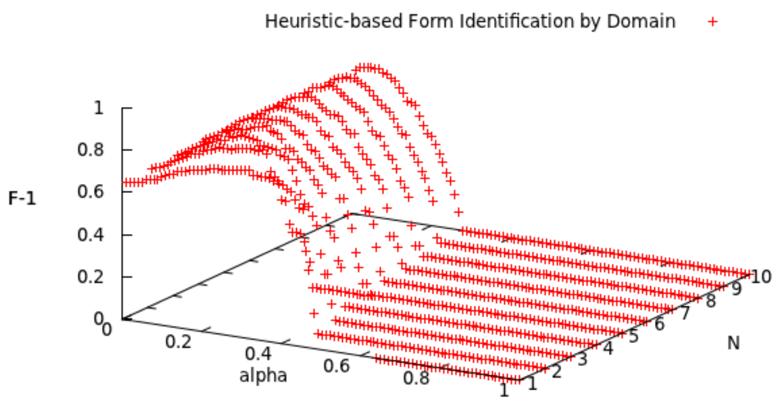


Figure 6.1: Average F1 results for different values of α and N.

forms of the small test set, each time using a distinct combination of values for α and N, as shown in Figure 6.1.

The values associated with the $F1$ axis in Figure 6.1 represent the average of the $F1$ results computed for all domains of the small test set. In order to eliminate misleading observations caused by the 3-dimensional representation of the plot, the most effective combinations of α and N values are detailed in Table 6.2.

The reported results show that our two heuristics for form identification by domain are more effective when associated to a relatively small range of α (i.e., between 0.00 and approximately 0.30). Also, N apparently does not have much impact on the behavior of the method. Even though α and N are meant to be manually setup with specific values for different domains so that the best possible results are achieved, the reported numbers constitute empirical evidence that there are general values for α and N that could be defined beforehand so that our form identification method achieves acceptable results (i.e., $F1 >= 0.7$) for any domain.

Based on these results and observations, we defined $\alpha = 0.29$ and $N = 2$ as the input values to be used in all subsequent experiments for form identification by domain in all domains. Table 6.3 reports the results achieved against the small test set using this specific configuration of input parameters.

Table 6.2: Top F1 results for different combinations of α and N.

α	N	**Top $F1$ value**
0.31	1	0.7960
0.29	**2**	**0.8271**
0.17	3	0.8096
0.16	4	0.7890
0.16	5	0.7599
0.08	6	0.7506
0.08	7	0.7431
0.07	8	0.7368
0.06	9	0.7139
0.05	10	0.7054

Table 6.3: Heuristic-based identification by domain against the small test set using $\alpha = 0.29$ and $N = 2$.

Domain	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
`Airfares`	0.875	0.758	0.812	0.744	0.806
`Autos`	0.812	0.969	0.884	0.988	0.934
`Books`	0.677	0.844	0.751	0.941	0.857
`Car Rentals`	0.956	0.697	0.806	0.761	0.832
`Hotels`	0.828	0.885	0.856	0.864	0.844
`Jobs`	0.829	0.847	0.838	0.809	0.821

6.1.1.3 Measuring Effectiveness

Our second heuristic was measured against the on-topic forms of the big test set, using the vocabularies detailed in Table 6.1. Notice that all forms used in this experiment are guaranteed by previous manual identification to belong to their respective domains. Therefore, we measured the effectiveness of the second heuristic employing not only the standard IR measures (see Table 6.4) used in the previous experiments, but also the identification error rates (see Table 6.5) associated to each experimented domain.

The lines of Table 6.5 present the error rates of the identifications based on the vocabularies that characterize the domains named in the first column of each line. The columns present the error rates of the identifications of the on-topic forms of the big test set related to the domains named in the first line of each column. More concretely, Table 6.5 shows that our second heuristic misclassified: (i) 2.5% of the forms of the `Airfares` domain when it used the vocabulary of the `Airfares` domain; (ii) 6.4% of the forms of the

Table 6.4: Heuristic-based identification by domain against the big test set.

Domain	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
`Airfares`	0.974	0.385	0.552	0.653	0.711
`Autos`	0.878	0.896	0.887	0.983	0.968
`Books`	0.940	0.966	0.953	0.993	0.984
`Car Rentals`	0.953	0.310	0.468	0.674	0.711
`Hotels`	0.786	0.406	0.535	0.678	0.702
`Jobs`	0.938	0.678	0.787	0.915	0.918

Table 6.5: Error rates of the heuristic-based identification by domain against the big test set.

	Airfares	Autos	Books	Car Rentals	Hotels	Jobs
Airfares	2.5%	6.4%	13.3%	64.9%	55.5%	27.7%
Autos	0.5%	12.1%	1.3%	5.6%	0.8%	0.9%
Books	0.5%	0.2%	5.9%	0.1%	0.2%	2.2%
Car Rentals	70.4%	19.1%	5.3%	4.6%	28.4%	34.9%
Hotels	77.4%	5.9%	5.3%	47.4%	21.3%	18.6%
Jobs	1.6%	11.1%	17.4%	9.1%	5.1%	6.1%

`Autos` domain when it used the vocabulary of the `Airfares` domain; (iii) 0.5% of the forms of the `Airfares` domain when it used the vocabulary of the `Autos` domain; and so on.

Notice that the results reported for the big test set were considerably worse than the ones reported for the small test set for the domains that present vocabulary overlap. These numbers constitute empirical evidence that, while the second heuristic is able to reasonably (i.e., $F1 >= 0.7$) identify the forms actually employed by the first heuristic, it may be not able to directly identify random forms from the Web, like a full-fledge form classifier does. Notice the poor results associated to the domains that present vocabulary overlap (i.e., `Airfares`, `Car Rentals` and `Hotels`).

It is also important to notice that error rates on the domain of interest represent the amount of false positives, and the error rates on the other domains represent the amount of false negatives. Since the error rates result from the simple presence/absence of the automatically identified relevant words, it is perfectly reasonable to eventually find smaller error rates on domains that are not the ones of interest. Therefore, there is no reason to expect error rates to be always smaller on the domains of interest than in any other experimented domain.

6.1.1.4 Comparing Manual and Automatic Training for DSFC

In order to show that the proposed heuristic-based method for form identification by domain achieve our final goal of mitigating the demanding manual work required from the human expert while discovering forms, we performed experiments with the cutting-edge form classifier DSFC (BARBOSA; FREIRE, 2007a). Like other form classifiers, DSFC demands training with dozens to hundreds of manually labeled forms.

DSFC makes decisions based on the textual contents of forms, which vary substantially across domains. Therefore, unlike GFC (see Section 5.3.2), its training can be regarded as domain-dependent. We implemented DSFC according to the specification given by its authors using the SMO Machine Learning algorithm available in the widely known and freely distributed Weka library (HOLMES; DONKIN; WITTEN, 1994). DSFC's effectiveness was measured in two scenarios for form identification by domain. Both scenarios were tested against the forms of the big test set.

In the first scenario, the training set was composed of: (i) positive examples that belong to the domain of interest, which was present in the test set; and (ii) negative examples that belong to domains that were completely absent in the test set. The upper halves of Tables 6.6 and 6.7 report the effectiveness of manually trained DSFC in this scenario. The numbers represent the average of ten runs executed for each domain. In each run, 220

Table 6.6: DSFC's results in the first scenario for form identification by domain.

Manual Training					
Domain	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
`Airfares`	0.991	0.438	0.608	0.767	0.801
`Autos`	0.988	0.483	0.649	0.868	0.882
`Books`	0.982	0.591	0.738	0.891	0.903
`Car Rentals`	0.988	0.279	0.435	0.703	0.732
`Hotels`	0.982	0.333	0.497	0.531	0.618
`Jobs`	0.973	0.479	0.642	0.839	0.857
Automatic Training					
Domain	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
`Airfares`	0.954	0.426	0.589	0.713	0.757
`Autos`	0.902	0.824	0.861	0.968	0.959
`Books`	0.940	0.891	0.915	0.977	0.971
`Car Rentals`	0.933	0.290	0.443	0.649	0.687
`Hotels`	0.778	0.376	0.507	0.638	0.669
`Jobs`	0.924	0.464	0.618	0.797	0.817

random forms from the domain of interest and 220 random forms from the `Biology` domain present in the DeepPeep form base (but absent in the big test set) composed the positive and negative training examples, respectively[2].

The lower halves of Tables 6.6 and 6.7 report the effectiveness of automatically trained DSFC in this scenario. The positive and negative examples of forms employed to train DSFC were the forms of the small test set, as they were classified by our method for form identification by domain. More concretely, for each domain of interest, the positive training examples were the forms of the small test set that our method identified as belonging to the domain of interest and the negative examples were the forms that our method identified as not belonging to the domain of interest.

Even though automatic training led DSFC to perform reasonably in two domains (i.e., `Autos` and `Books`), both manual and automatic trainings led DSFC to achieve unsatisfactory results (i.e., $F1 < 0.7$), specially in those domains that present vocabulary overlap. This is evidence that even a full-fledged form classifier like DSFC cannot be effective if not trained very carefully, with training examples that belong to all the domains of the forms that have to be identified. This is probably one of the main causes of the high error rates presented by DeepPeep's discoverer detailed in Table 4.2.

In the second scenario for experimenting DSFC, the training set was composed of: (i) positive examples that belong to the domain of interest, which was present in the test set; and (ii) negative examples that belong to all the domains that were present in the test set. The upper halves of Tables 6.8 and 6.9 report the effectiveness of manually trained DSFC in this scenario. The numbers represent the average of ten runs executed for each domain. In each run, 220 random forms from the domain of interest and 45 (i.e., 220/5) random forms from each one of the other five domains present in the big test set composed the positive and negative training examples, respectively.

The lower halves of Tables 6.8 and 6.9 report the effectiveness of automatically trained DSFC in this scenario. The positive and negative examples of forms used to train DSFC

[2]Barbosa & Freire used similar amounts of forms for each domain to train DSFC in their experiments (BARBOSA; FREIRE, 2007a).

Table 6.7: DSFC's error rates in the first scenario for form identification by domain.

Manual Training						
	Airfares	Autos	Books	Car Rentals	Hotels	Jobs
Airfares	0.8%	2.7%	1.7%	65.3%	39.2%	6.6%
Autos	7.5%	1.1%	6.3%	40.1%	4.8%	15.3%
Books	4.4%	5.6%	1.7%	8.6%	5.7%	31.7%
Car Rentals	72.3%	21.0%	1.5%	1.1%	30.3%	16.8%
Hotels	93.0%	22.8%	11.2%	78.3%	1.7%	26.0%
Jobs	20.5%	11.2%	11.5%	31.3%	9.5%	2.6%
Automatic Training						
	Airfares	Autos	Books	Car Rentals	Hotels	Jobs
Airfares	4.5%	6.5%	8.4%	54.5%	47.6%	21.1%
Autos	1.1%	9.7%	2.9%	4.4%	2.8%	4.7%
Books	0.5%	0.2%	5.9%	0.1%	0.2%	10.5%
Car Rentals	71.1%	16.6%	11.1%	6.6%	35.5%	33.9%
Hotels	76.6%	6.5%	9.4%	66.4%	22.1%	18.3%
Jobs	4.0%	12.6%	26.2%	53.7%	13.7%	7.5%

Table 6.8: DSFC's results in the second scenario for form identification by domain.

Manual Training					
Domain	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
Airfares	0.971	0.898	0.933	0.978	0.977
Autos	0.993	0.937	0.964	0.991	0.991
Books	0.991	0.981	0.986	0.996	0.996
Car Rentals	0.970	0.805	0.880	0.971	0.971
Hotels	0.973	0.870	0.919	0.964	0.965
Jobs	0.975	0.930	0.951	0.988	0.986
Automatic Training					
Domain	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
Airfares	0.869	0.567	0.686	0.852	0.855
Autos	0.941	0.955	0.948	0.992	0.985
Books	0.984	0.978	0.981	0.995	0.993
Car Rentals	0.777	0.509	0.615	0.884	0.870
Hotels	0.698	0.570	0.627	0.852	0.818
Jobs	0.966	0.945	0.955	0.989	0.985

Table 6.9: DSFC's error rates in the second scenario for form identification by domain.

Manual Training						
	Airfares	Autos	Books	Car Rentals	Hotels	Jobs
Airfares	2.8%	0.0%	0.3%	7.3%	2.7%	0.4%
Autos	0.1%	0.6%	0.1%	1.9%	0.8%	1.4%
Books	0.0%	0.1%	0.8%	0.0%	0.0%	1.4%
Car Rentals	6.6%	2.3%	0.2%	2.9%	2.6%	1.8%
Hotels	0.8%	0.9%	0.3%	3.9%	2.6%	2.8%
Jobs	0.3%	0.7%	2.7%	1.0%	0.9%	2.5%
Automatic Training						
	Airfares	Autos	Books	Car Rentals	Hotels	Jobs
Airfares	13.0%	0.6%	1.9%	32.8%	29.5%	5.3%
Autos	0.0%	5.8%	0.2%	3.0%	0.2%	0.5%
Books	0.0%	1.2%	1.5%	0.0%	0.0%	1.0%
Car Rentals	24.2%	12.3%	1.6%	22.2%	13.8%	3.2%
Hotels	47.9%	0.1%	1.4%	16.7%	30.1%	1.8%
Jobs	0.1%	1.2%	2.5%	0.6%	0.8%	3.3%

were the ones of the small test set, as they were classified by our method for form identification by domain. For each domain of interest, the positive training examples used were the forms that the method identified as belonging to the domain of interest and the negative examples used were the forms that the method identified as belonging to one of the other five domains present in the small test set. For each domain that composed the negative examples, the number of the training forms was $PF/5$, where PF was the number of positive training examples. The numbers show that manual training is considerably more effective than automatic training in this scenario, because automatic training leads DSFC to unsatisfactory results (i.e., $F1 < 0.7$) related to domains that present vocabulary overlap.

6.2 Improved Behavior of the Proposed Method

The numbers presented in Section 6.1.1.4 evidence that the automatic training made available by our method was not a match to manual training in the second scenario. Therefore, we decided to improve our method through the use of a simple heuristic-based form ranker strongly inspired by the work of Kabra, Li & Chang (KABRA; LI; CHANG, 2005).

6.2.1 Heuristic-based Ranking

Instead of training DSFC with all the forms discovered by our method, as we did in Section 6.1.1.4, we trained it using only the top-F most relevant among them, as defined by the ranker. To the ranker, form relevance is a function of the presence of the top-W most frequent words as defined by our first heuristic for form identification by domain. The rationale of the ranking-based training is straightforward: the more relevant words a form presents in its renderizable text, the more relevant it is to train DSFC. According to this rationale, for instance, the form illustrated in Figure 1.4(a) would be considered as

Table 6.10: DSFC's results in the first scenario for form identification by domain (with ranking).

Manual Training					
Domain	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
`Airfares`	0.991	0.438	0.608	0.767	0.801
`Autos`	0.988	0.483	0.649	0.868	0.882
`Books`	0.982	0.591	0.738	0.891	0.903
`Car Rentals`	0.988	0.279	0.435	0.703	0.732
`Hotels`	0.982	0.333	0.497	0.531	0.618
`Jobs`	0.973	0.479	0.642	0.839	0.857
Automatic Training					
Domain	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
`Airfares`	0.867	0.759	0.809	0.938	0.925
`Autos`	0.910	0.826	0.866	0.969	0.960
`Books`	0.937	0.907	0.922	0.980	0.973
`Car Rentals`	0.772	0.433	0.555	0.845	0.835
`Hotels`	0.590	0.660	0.623	0.915	0.844
`Jobs`	0.700	0.938	0.802	0.991	0.944

more relevant than the one illustrated in Figure 1.4(b) for the `Jobs` domain when ranked according to the vocabulary attributed to the same domain, as presented in Table 6.1, because the former form presents more relevant words (i.e., *"job"* and *"keyword"*) than the latter (i.e., *"location"*)[3].

6.2.2 Experiments

We experimented our ranking-based automatic training for DSFC using arbitrarily defined different values for W (i.e., the number of words present in the vocabulary defined by the first heuristic for form identification by domain parameterized with $\alpha = 0.00$) and F (i.e., the number of domain-specific forms to be used in DSFC's training). The best results (see the lower halves of Tables 6.10 and 6.11) for the first scenario described in Section 6.1.1.4 were achieved with $W = 5$, $F = 100$, and 100 negative examples for each run. Automatic training in this scenario led to better results when compared to manual training because it was guaranteed in manual training that the negative examples of the training base would not be present in the test set. Such guaranteed was not present in automatic training. The only guaranteed in automatic training is that no effort was done during training in order to learn about all the domains present in the test set.

Notice that our ranker allows to rank only the forms associated to a specific domain, since it employs domain-specific vocabularies. Consequently, there was no way to attribute relevance to the forms that composed the negative examples in the first scenario. The results show that the use of ranking lead to more effective automatic training, arguably better than manual training on the same scenario. The numbers from manual training are replicated from Tables 6.6 and 6.7.

The best results for the second scenario (see see the lower halves of Tables 6.12

[3]This illustrative comparison is purposefully incomplete in order to give a didactic example of our ranking mechanism. A complete comparison would take into account the complete renderizable texts of the referred forms, including the contents of their drop boxes.

Table 6.11: DSFC's error rates in the first scenario for form identification by domain (with ranking).

Manual Training						
	Airfares	Autos	Books	Car Rentals	Hotels	Jobs
Airfares	0.8%	2.7%	1.7%	65.3%	39.2%	6.6%
Autos	7.5%	1.1%	6.3%	40.1%	4.8%	15.3%
Books	4.4%	5.6%	1.7%	8.6%	5.7%	31.7%
Car Rentals	72.3%	21.0%	1.5%	1.1%	30.3%	16.8%
Hotels	93.0%	22.8%	11.2%	78.3%	1.7%	26.0%
Jobs	20.5%	11.2%	11.5%	31.3%	9.5%	2.6%
Automatic Training						
	Airfares	Autos	Books	Car Rentals	Hotels	Jobs
Airfares	13.2%	0.1%	0.9%	14.2%	12.3%	1.5%
Autos	1.1%	8.9%	2.2%	4.1%	2.8%	5.6%
Books	0.3%	0.1%	6.2%	0.1%	0.0%	9.3%
Car Rentals	5.5%	1.6%	1.7%	22.7%	8.6%	5.9%
Hotels	27.9%	0.4%	1.5%	7.1%	40.9%	1.6%
Jobs	0.4%	0.5%	2.4%	1.1%	0.1%	29.9%

and 6.13) were achieved with $W = 5$, $F = 150$ for the positive examples, and $F = 100$ for each domain that composed the negative examples, totalizing 500 negative training examples for each run. The results show that the use of ranking lead to a more effective automatic training. Automatic training in the second scenario, however, led to results that were effective ($F1 >= 0.8$) but poorer than manual training for the domains that present vocabulary overlap: the losses summed up to approximately 0.11 points in F1 measure. The numbers from manual training are replicated from Tables 6.6 and 6.7.

Table 6.12: DSFC's results in the second scenario for form identification by domain (with ranking).

Manual Training					
Domain	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
Airfares	0.971	0.898	0.933	0.978	0.977
Autos	0.993	0.937	0.964	0.991	0.991
Books	0.991	0.981	0.986	0.996	0.996
Car Rentals	0.970	0.805	0.880	0.971	0.971
Hotels	0.973	0.870	0.919	0.964	0.965
Jobs	0.975	0.930	0.951	0.988	0.986
Automatic Training					
Domain	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
Airfares	0.867	0.790	0.827	0.948	0.933
Autos	0.884	0.964	0.922	0.994	0.979
Books	0.986	0.978	0.982	0.995	0.994
Car Rentals	0.843	0.835	0.839	0.974	0.957
Hotels	0.710	0.918	0.801	0.982	0.922
Jobs	0.927	0.992	0.958	0.998	0.987

Table 6.13: DSFC's error rates in the second scenario for form identification by domain (with ranking).

Manual Training						
	Airfares	Autos	Books	Car Rentals	Hotels	Jobs
Airfares	2.8%	0.0%	0.3%	7.3%	2.7%	0.4%
Autos	0.1%	0.6%	0.1%	1.9%	0.8%	1.4%
Books	0.0%	0.1%	0.8%	0.0%	0.0%	1.4%
Car Rentals	6.6%	2.3%	0.2%	2.9%	2.6%	1.8%
Hotels	0.8%	0.9%	0.3%	3.9%	2.6%	2.8%
Jobs	0.3%	0.7%	2.7%	1.0%	0.9%	2.5%
Automatic Training						
	Airfares	Autos	Books	Car Rentals	Hotels	Jobs
Airfares	13.2%	0.7%	0.2%	6.8%	13.9%	0.5%
Autos	0.0%	11.5%	0.0%	1.4%	0.5%	0.8%
Books	0.0%	0.8%	13.2%	0.0%	0.0%	1.4%
Car Rentals	7.8%	2.8%	0.0%	15.6%	13.4%	0.4%
Hotels	5.0%	0.0%	0.0%	2.4%	28.9%	0.7%
Jobs	0.0%	0.0%	0.2%	0.0%	0.2%	7.2%

7 FURTHER DISCUSSION

This chapter tries to answer four questions that were raised during the development of our proposal. It provides interesting additional information that more completely characterizes our contribution for form discovery.

7.1 What is the minimum number of relevant forms that have to be located through the search engine so that the proposal presents its best results?

The proposal relies on the ability of the employed search engine in returning a significant number of relevant URLs (i.e., URLs pointing to Web sites that contain relevant forms) in response to a keyword-based query consisting of descriptive keywords for the domain of interest. Since forms are sparsely distributed in the Web (BARBOSA; FREIRE, 2005, 2007b), it is reasonable to expect that there is a considerable number of domains for which search engines in general may not be able to respond with a significant number of relevant URLs.

We simulated such a pessimistic scenario by restricting the number of relevant forms used in the first scenario of the experiment discussed in Section 6.1.1.4. For each domain, we executed several slightly modified executions of that experiment: the first one used only 10 relevant forms, the second one used 20, the third one used 30, and so on, up to the total number of relevant forms located by our form locator. In all executions the number of irrelevant forms was left intact (i.e., the amount of irrelevant forms used in each run was the total number of irrelevant forms located by the form locator).

The results showed that our method for form identification by domain achieves its best results when at least approximately 150 relevant forms are found among the 500 forms located per domain. This is empirical evidence that our discoverer works reasonably well in the domains where it can locate through the search engine at least approximately one relevant form for each three located forms.

7.2 What is the impact on the effectiveness of the proposal when different keywords are sent to the search engine?

Arguably, the effectiveness of our approach is highly dependent on the queries sent to the search engine. In order to asses how the effectiveness of our approach vary depending on the keywords sent to the search engine, we performed two experiments (as discussed below) that were executed in the first scenario for form identification by domain described in Section 6.1.1.4.

The first experiment assessed the impact of the use of certain “special terms” as additional keywords to be sent to the search engine together the domain name. Here, special terms are words supposed to give a hint to the search engine that Web sites that present query forms are desired. The experimented special terms were: *“online”* and *“search”*. The results are shown in Table 7.1. The first row for each domain repeats the results from Section 6.1.1.4, which were obtained using just the name of the domain as the query sent to the search engine. The numbers show that the use of the experimented special terms do not consistently affect the effectiveness of our method for form identification by domain: in most cases, the results are similar, but in some cases the proposal becomes completely ineffective.

In the second experiment, we sent to the search engine different descriptive keywords for the experimented domains. Table 7.2 shows the manually defined keywords sent to the search engine and their respective results. The first row for each domain repeats the results from the first scenario of Section 6.1.1.4. The `Car Rentals` domain was not experimented because we could not define reasonable alternative keywords for it. The results show that different descriptive keywords lead our method to achieve significantly different results.

7.3 What if we take into account only the textual contents near the renderizable fields of forms?

During manual classification of the forms that compose the small and the big test sets, it was noticed that many forms present textual contents not necessarily related to their attributes. Take, for instance, the forms depicted in Figures 1.4(a) and 1.4(b): they present textual “headers” that are not related to any specific attribute. Even though these “additional” texts may contain important hints about the domains to which their forms belong, the study that provides basis for our proposal (HE; TAO; CHANG, 2004, 2005) is restricted to the labels of attributes and does not necessarily take the whole renderizable texts of forms into account. Therefore, we decided to measure the effectiveness of our method taking into account only a specific subset of the renderizable texts of the forms: the texts that are located near the renderizable fields of their forms. In this experiment, we delimited the texts located near the renderizable fields of their forms as the sets of words that go from the fifth word before the first renderizable field to the fifth word after the last renderizable field.

Similarly to the experiments reported in Section 6.1.1.2, we measured the impact of different values of α and N on the effectiveness of this method. The best results (i.e., $(R) = 0.822$, $(P) = 0.839$, $F1 = 0.831$, $(S) = 0.880$ and $(A) = 0.855$) against the small test set were achieved with $\alpha = 0.23$ and $N = 2$. The results for the first scenario of form identification by domain described in Section 6.1.1.4 are shown in Table 7.3. As the numbers show, restricting the method to use only the texts that are located near the renderizable fields of their forms did not significantly improve the effectiveness of automatic training for DSFC.

Table 7.1: Results related to the addition of special terms to the keywords sent to the search engine.

Keywords	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
Airfares					
airfares	0.954	0.426	0.589	0.713	0.757
online airfares	0.000	0.000	0.000	1.000	0.817
search airfares	0.000	0.000	0.000	1.000	0.817
search online airfares	0.558	0.897	0.688	0.985	0.907
Autos					
autos	0.902	0.824	0.861	0.968	0.959
online autos	0.941	0.834	0.884	0.969	0.965
search autos	0.832	0.924	0.876	0.989	0.967
search online autos	0.000	0.000	0.000	1.000	0.860
Books					
books	0.940	0.891	0.915	0.977	0.971
online books	0.000	0.000	0.000	1.000	0.833
search books	0.959	0.965	0.962	0.993	0.987
search online books	0.000	0.000	0.000	1.000	0.833
Car Rentals					
car rentals	0.933	0.290	0.443	0.649	0.687
online car rentals	0.903	0.266	0.411	0.618	0.656
search car rentals	0.386	0.261	0.311	0.831	0.772
search online car rentals	0.330	0.216	0.261	0.816	0.751
Hotels					
hotels	0.778	0.376	0.507	0.638	0.669
online hotels	0.728	0.375	0.495	0.661	0.676
search hotels	0.761	0.381	0.508	0.654	0.677
search online hotels	0.713	0.350	0.470	0.629	0.648
Jobs					
jobs	0.924	0.464	0.618	0.797	0.817
online jobs	0.916	0.517	0.661	0.837	0.849
search jobs	0.943	0.366	0.527	0.688	0.729
search online jobs	0.915	0.630	0.746	0.897	0.900

Table 7.2: Results of different sets of domain-specific descriptive keywords sent to the search engine.

Keywords	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
Airfares					
airfares	0.954	0.426	0.589	0.713	0.757
air travels	0.000	0.000	0.000	1.000	0.817
air flights	0.984	0.342	0.507	0.577	0.651
airfares air travels	0.000	0.000	0.000	1.000	0.817
airfares air flights	0.987	0.318	0.481	0.528	0.612
air travels flights	0.513	0.904	0.655	0.987	0.901
airfares air travels flights	0.522	0.895	0.659	0.986	0.901
Autos					
autos	0.902	0.824	0.861	0.968	0.959
cars	0.936	0.837	0.884	0.970	0.965
vehicles	0.000	0.000	0.000	1.000	0.860
autos cars	0.869	0.853	0.861	0.975	0.961
autos vehicles	0.000	0.000	0.000	1.000	0.860
cars vehicles	0.000	0.000	0.000	1.000	0.860
autos cars vehicles	0.000	0.000	0.000	1.000	0.860
Books					
books	0.940	0.891	0.915	0.977	0.971
bookstores	0.314	0.765	0.445	0.980	0.869
ebooks	0.000	0.000	0.000	1.000	0.833
books bookstores	0.952	0.960	0.956	0.992	0.985
books ebooks	0.000	0.000	0.000	1.000	0.833
ebooks bookstores	0.000	0.000	0.000	1.000	0.833
books ebooks bookstores	0.000	0.000	0.000	1.000	0.833
Hotels					
hotels	0.778	0.376	0.507	0.638	0.669
rooms	0.000	0.000	0.000	1.000	0.781
reservations	0.529	0.330	0.406	0.699	0.662
hotels rooms	0.722	0.377	0.495	0.666	0.678
hotels reservations	0.644	0.344	0.448	0.656	0.653
rooms reservations	0.000	0.000	0.000	1.000	0.781
hotels rooms reservations	0.769	0.389	0.517	0.662	0.685
Jobs					
jobs	0.924	0.464	0.618	0.797	0.817
employment	0.000	0.000	0.000	1.000	0.839
careers	0.650	0.577	0.611	0.909	0.867
jobs employment	0.860	0.658	0.745	0.914	0.906
jobs careers	0.829	0.650	0.729	0.914	0.901
employment careers	0.000	0.000	0.000	1.000	0.839
jobs employment careers	0.848	0.693	0.763	0.928	0.915

Table 7.3: Automatically trained DSFC's results in the first scenario for form identification by domain (taking into account only the texts near the renderizable fields of forms).

Domain	**(R)**	**(P)**	**F1**	**(S)**	**(A)**
`Airfares`	0.934	0.461	0.617	0.756	0.788
`Autos`	0.918	0.807	0.859	0.964	0.958
`Books`	0.916	0.950	0.933	0.990	0.978
`Car Rentals`	0.945	0.290	0.444	0.644	0.684
`Hotels`	0.817	0.396	0.533	0.651	0.687
`Jobs`	0.876	0.515	0.649	0.842	0.848

Table 7.4: Vocabularies that characterize the non-query forms of the experimented domains.

Domain	**Words**
`Airfares`	*"email"*
`Autos`	*"email"*
`Books`	*"email"*
`Car Rentals`	*"email"*
`Hotels`	*"email"*
`Jobs`	*"email"*, *"address"*, *"password"*

7.4 Is it possible to identify non-query forms by their textual contents?

Perkowitz *et al.* (PERKOWITZ et al., 1997; DOORENBOS; ETZIONI; WELD, 1997) employ a small set of manually defined words to perform form identification by function. They reasonably assume that forms that present some specific words (e.g., *"phone"*, *"e-mail"*, *"address"*, etc.) are clearly non-query forms, since usually the kind of information associated with words like these are used for posting personal data to an HTTP Web server.

In order to find out if we are able to identify non-query forms by their textual contents using our heuristic-based approach, we elaborated a small adaptation of our method for form identification by domain and applied it to form identification by function. More specifically, the adaptation occurs on the first heuristic for form identification by domain: the frequencies of words present in the renderizable texts of non-query forms are computed instead of the frequencies of words present in the renderizable texts of query forms. Such an adaptation has place in the second step of the first heuristic's algorithm presented in Section 6.1: non-query forms have to be identified instead of the query ones.

Similarly to the experiments reported in Section 6.1.1.2, we measured the impact of different values of α and N on the effectiveness of the method. The best results (i.e., $(R) = 0.730$, $(P) = 0.875$ and $F1 = 0.7969$, $(S) = 0.949$ and $(A) = 0.878$) against the small test set were achieved with a few values of α (i.e., values near 0.40) and $N = 1$. The results show that the use of textual contents to perform form identification by function is less effective than the method proposed in Chapter 5. For illustrative purposes, Table 7.4 shows the vocabularies that the adapted heuristic associated to each domain when it processed the small test set using the referred values of α and N.

8 CONCLUSION

The holy grail in Web processing in general is to have automatic methods that operate without any training or supervision from a human expert. Structured form discovery is no exception to this and, therefore, is gradually evolving from manual to completely unsupervised. Nevertheless, the current state-of-the-art in form discovery, directly or indirectly, still relies on significant human intervention. Consequently, novel methods that mitigate the demanding manual work required by the current state-of-the-art in form discovery are extremely welcomed.

In this context, the contribution of this thesis is two-folded. First, we offered as a survey an up-to-date overview of 19 published form discovery techniques. We were particularly interested in the aspects concerning the discovery of domain-specific query forms that employ the pre-query identification approach (i.e., that do not involve form submission).

Our survey presented a complete review of the literature related to the topic of form discovery, offering to the reader an insightful synthesis of findings of individual related works. More specifically, we analyzed how the existing groups of techniques depend on external factors such as label identification, human assistance, and freely downloadable off-the-shelf form bases. Furthermore, we identified how these groups of techniques depend on each other. Regarding this analysis, we conclude that, except from two techniques for the discovery of keyword-based or unstructured query forms, all surveyed techniques directly or indirectly still depend significantly on the human expert, despite the fact that there is a trend of gradual increase in automation in form discovery. Additionally, we forecasted some potential next steps in the evolution of form discovery techniques, which hopefully may be inspirational to other researchers (e.g., PhD students eventually interested in advancing the state-of-the-art in form discovery).

The second contribution of this thesis is the proposal and evaluation of a group of heuristic-based domain-specific structured form discovery methods that require almost no involvement from the human expert in order to operate, pushing the state-of-the-art in form discovery towards the evolutionary goal of complete unsupervised execution. The only manual input needed by our methods is the definition of a few keywords that appropriately describe the domain of interest (i.e., the domain in which the form discovery should occur).

Our experiments ran on six domains and employed real Web data composed of thousands of forms, including a representative subset of the publicly and freely available DeepPeep form base (DEEPPEEP, 2010; DEEPPEEP REPOSITORY, 2011). The results of the experiments constitute substantial empirical evidence that it is feasible to mitigate the demanding manual work required by two cutting-edge form classifiers (i.e., GFC and DSFC (BARBOSA; FREIRE, 2007a)), at the cost of a relatively small loss in effectiveness.

Our form locator is able to locate relevant forms from all the experimented domains, providing enough data for our heuristic-based form identification methods. Seven heuristics for form identification by function were experimented. Among them, we selected one that is able to effectively replace manual training for the form classifier GFC. Also, two collaborative heuristics for form identification by domain were experimented in considerably different testing scenarios. When combined to a simple novel form ranker, these two heuristics are able to effectively replace manual training for form classifier DSFC.

While manual training is arguably more effective when DSFC is correctly trained (i.e., during its training, it learns about all the domains of the forms it has to classify), automatic training (using ranking) leads to satisfactory results (i.e., $F1 >= 0.8$) requiring much less involvement from the human expert. Besides, automatic training (using ranking) can be considerably more effective than manual training when manual training is not able to correctly prepare DSFC to recognize all the domains of the forms it has to classify.

We believe that our proposal potentially constitutes the first step towards the inception of a fully unsupervised domain-specific DW crawler. Such a crawler constitutes a possible **future work** to be built upon this thesis. Our hypothesis is that the domain-knowledge necessary for such a crawler to operate can be unsupervisedly inferred from the Web in two ways: (i) *incrementally*, by automatically shifting the use of heuristic-based techniques to the use of more sophisticated techniques able to learn from automatic training; and (ii) *synergistically*, by allowing different methods of the different phases of DW crawling to interact and contribute to each other, through a shared domain-knowledge base that is automatically built entirely from scratch. In other words, we envision a DW crawler that is able to infer knowledge from the Web requiring minimum human help, in the same spirit that we carried out the experiments in Chapters 5 and 6, where GFC and DSFC were automatically trained by our proposal. The referred fully unsupervised domain-specific DW crawler could provide basis for a more dynamic definition for the best values of parameters that were employed as constants in the experiments described in Chapter6 (i.e., α, N, W, and F).

Detailed processing strategies of such a DW crawler are yet to be defined for each DW crawling phase. Nevertheless, we believe that for form discovery, the proposal of this thesis would constitute a good starting point. For form filling, form submission with default values, as proposed by Liddle et al. (LIDDLE et al., 2003), could be used as the initial approach. And finally, the starting approach for data extraction from results of form submissions would be HTML pages that present easy-to-extract labeled values inside data tables, as proposed by Wang & Lochovsky (WANG; LOCHOVSKY, 2003). After the execution of these methods as model approaches for initial processing, further domain-knowledge inference would use existing DW techniques (or inspire the creation of new ones), but adapted to take domain-knowledge previously inferred into account.

Another way in which we could improve our proposal is to apply active learning techniques to improve training. Active learning could fundament a better selection process for the forms identified by our heuristic-based method. That way, it hopefully could be employed in order to achieve better form discovery results.

REFERENCES

AGICHTEIN, E.; GRAVANO, L. Querying text databases for efficient information extraction. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 19., 2003. **Proceedings...** [S.l.: s.n.], 2003. p.113 – 124.

BAEZA-YATES, R.; RIBEIRO-NETO, B. et al. **Modern information retrieval**. [S.l.]: Addison-Wesley Reading, MA, 1999.

BARBOSA, L.; FREIRE, J. **Searching for hidden-web databases**. 2005. 1–6p. v.5.

BARBOSA, L.; FREIRE, J. Automatically constructing collections of online database directories. In: ACM INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 15., 2006, New York, NY, USA. **Proceedings...** ACM, 2006. p.796–797. (CIKM '06).

BARBOSA, L.; FREIRE, J. Combining classifiers to identify online databases. In: WWW '07, 2007, New York. **Anais...** ACM, 2007. p.431–440.

BARBOSA, L.; FREIRE, J. An adaptive crawler for locating hidden-Web entry points. In: WWW '07, 2007, New York. **Anais...** ACM, 2007. p.441–450.

BARBOSA, L.; FREIRE, J.; SILVA, A. Organizing Hidden-Web Databases by Clustering Visible Web Documents. In: DATA ENGINEERING, 2007. ICDE 2007. IEEE 23RD INTERNATIONAL CONFERENCE ON, 2007. **Anais...** [S.l.: s.n.], 2007. p.326 –335.

BARBOSA, L.; NGUYEN, H.; NGUYEN, T.; PINNAMANENI, R.; FREIRE, J. Creating and exploring web form repositories. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2010., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p.1175–1178. (SIGMOD '10).

BERGHOLZ, A.; CHILDLOVSKII, B. Crawling for domain-specific hidden Web resources. In: FOURTH INTERNATIONAL CONFERENCE ON WEB INFORMATION SYSTEMS ENGINEERING, 2003. **Proceedings...** [S.l.: s.n.], 2003. p.125 – 133.

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

BERKHIN, P. A survey of clustering data mining techniques. **Grouping Multidimensional Data**, [S.l.], p.25–71, 2006.

BEZDEK, J. C. Cluster Validity with Fuzzy Sets. **Journal of Cybernetics**, [S.l.], v.3, n.3, p.58–73, 1973.

BHARAT, K.; BRODER, A.; HENZINGER, M.; KUMAR, P.; VENKATASUBRAMANIAN, S. The Connectivity Server: fast access to linkage information on the web. **Computer Networks and ISDN Systems**, [S.l.], v.30, n.1 - 7, p.469 – 477, 1998.

BLUM, A.; MITCHELL, T. Combining labeled and unlabeled data with co-training. In: COMPUTATIONAL LEARNING THEORY, 1998, New York, NY, USA. **Proceedings...** ACM, 1998. p.92–100. (COLT' 98).

CAZOODLE. Acesso em: jan. 2011, disponível em: `http://www.cazoodle.com/`.

CHAKRABARTI, S.; BERG, M. van den; DOM, B. Focused crawling: a new approach to topic-specific web resource discovery. **Computer Networks**, [S.l.], v.31, n.11 - 16, p.1623 – 1640, 1999.

CHANG, K. C.-C.; HE, B.; LI, C.; PATEL, M.; ZHANG, Z. Structured databases on the web: observations and implications. **SIGMOD Rec.**, New York, NY, USA, v.33, n.3, p.61–70, Sept. 2004.

CHANG, K.; HE, B.; ZHANG, Z. Toward large scale integration: building a metaquerier over databases on the web. In: CIDR, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.44–55.

COLLECTION of Schemas for Search Providers. Acesso em: dez. 2011, disponível em: `http://www.schema.org/`.

COPE, J.; CRASWELL, N.; HAWKING, D. Automated discovery of search interfaces on the web. In: AUSTRALASIAN DATABASE CONFERENCE - VOLUME 17, 14., 2003, Darlinghurst, Australia, Australia. **Proceedings...** Australian Computer Society: Inc., 2003. p.181–189. (ADC '03).

CRESCENZI, V.; MECCA, G.; MERIALDO, P. RoadRunner: automatic data extraction from data-intensive web sites. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2002., 2002, New York, NY, USA. **Proceedings...** ACM, 2002. p.624–624. (SIGMOD '02).

DEEPPEEP repository. Acesso em: jan. 2011, disponível em: `http://www.cs.utah.edu/~lbarbosa/forms/forms.tar.gz`.

DEEPPEEP. Acesso em: out. 2009, disponível em: `http://www.deeppeep.org/`.

DILIGENTI, M.; COETZEE, F.; LAWRENCE, S.; GILES, C. L.; GORI, M. Focused Crawling Using Context Graphs. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 26., 2000, San Francisco, CA, USA. **Proceedings...** Morgan Kaufmann Publishers Inc., 2000. p.527–534. (VLDB '00).

DOORENBOS, R. B.; ETZIONI, O.; WELD, D. S. A scalable comparison-shopping agent for the World-Wide Web. In: AUTONOMOUS AGENTS, 1997, New York, NY, USA. **Proceedings...** ACM, 1997. p.39–48. (AGENTS '97).

EL-GAMIL, B. R.; WINIWARTER, W.; BOZIC, B.; WAHL, H. Deep web integrated systems: current achievements and open issues. In: INTERNATIONAL CONFERENCE ON INFORMATION INTEGRATION AND WEB-BASED APPLICATIONS AND SERVICES, 13., 2011, New York, NY, USA. **Proceedings...** ACM, 2011. p.447–450. (iiWAS '11).

FELLBAUM, C. (Ed.). **WordNet**: an electronic lexical database. [S.l.]: MIT Press, 1998.

FORMAN, G. An extensive empirical study of feature selection metrics for text classification. **J. Mach. Learn. Res.**, [S.l.], v.3, p.1289–1305, Mar. 2003.

FRIEDMAN, N.; GEIGER, D.; GOLDSZMIDT, M. Bayesian Network Classifiers. **Machine Learning**, [S.l.], v.29, p.131–163, 1997.

GAL, A.; MODICA, G.; JAMIL, H. OntoBuilder: fully automatic extraction and consolidation of ontologies from web sources. In: DATA ENGINEERING, 2004. PROCEEDINGS. 20TH INTERNATIONAL CONFERENCE ON, 2004. **Anais...** [S.l.: s.n.], 2004. p.853.

GALPERIN, M. Y. The Molecular Biology Database Collection: 2005 update. **Nucleic Acids Research**, [S.l.], v.33, n.suppl 1, p.D5–D24, 2005.

GONG, Z.; ZHANG, J.; LIU, Q. Hidden-Web Database Exploration. In: INTELLIGENT SYSTEMS DESIGN AND APPLICATIONS, 2006. ISDA '06. SIXTH INTERNATIONAL CONFERENCE ON, 2006. **Anais...** [S.l.: s.n.], 2006. v.2, p.838 –843.

GOOGLE Directory. Acesso em: jan. 2011, disponível em: `http://directory.google.com/`.

GOOGLE Search Engine. Acesso em: jan. 2011, disponível em: `http://www.google.com/`.

GRAVANO, L.; GARCÍA-MOLINA, H.; TOMASIC, A. GlOSS: text-source discovery over the internet. **ACM Trans. Database Syst.**, New York, NY, USA, v.24, n.2, p.229–264, June 1999.

GRAVANO, L.; IPEIROTIS, P. G.; SAHAMI, M. QProber: a system for automatic classification of hidden-web databases. **ACM Trans. Inf. Syst.**, New York, NY, USA, v.21, n.1, p.1–41, Jan. 2003.

HAN, J.; PEI, J.; YIN, Y. Mining frequent patterns without candidate generation. **SIGMOD Rec.**, New York, NY, USA, v.29, n.2, p.1–12, May 2000.

HAN, J.; WANG, J.; LU, Y.; TZVETKOV, P. Mining top-k frequent closed patterns without minimum support. In: IEEE INTERNATIONAL CONFERENCE ON DATA MINING, 2002. **Proceedings...** [S.l.: s.n.], 2002. p.211 – 218.

HE, A.; KUSHMERICK, N. Learning to Attach Semantic Metadata to Web Services. In: FENSEL, D.; SYCARA, K.; MYLOPOULOS, J. (Ed.). **The Semantic Web - ISWC 2003**. [S.l.]: Springer Berlin Heidelberg, 2003. p.258–273. (Lecture Notes in Computer Science, v.2870).

HE, B.; TAO, T.; CHANG, K.-C. Clustering Structured Web Sources: a schema-based, model-differentiation approach. In: **Current Trends in Database Technology - EDBT 2004 Workshops**. [S.l.]: Springer Berlin Heidelberg, 2005. p.536–546. (Lecture Notes in Computer Science, v.3268).

HE, B.; TAO, T.; CHANG, K. C.-C. Organizing structured web sources by query schemas: a clustering approach. In: ACM INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 2004, New York, NY, USA. **Proceedings...** ACM, 2004. p.22–31. (CIKM '04).

HE, H.; MENG, W.; LU, Y.; YU, C.; WU, Z. Towards Deeper Understanding of the Search Interfaces of the Deep Web. **World Wide Web**, [S.l.], v.10, p.133–155, 2007.

HE, H.; MENG, W.; YU, C.; WU, Z. Wise-integrator: an automatic integrator of web search interfaces for e-commerce. In: VERY LARGE DATA BASES - VOLUME 29, 29., 2003. **Proceedings...** VLDB Endowment, 2003. p.357–368. (VLDB '2003).

HEDLEY, Y. L.; YOUNAS, M.; JAMES, A.; SANDERSON, M. Query-related data extraction of hidden web documents. In: ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 27., 2004. **Proceedings...** ACM, 2004. p.558–559. (SIGIR '04).

HEDLEY, Y.-L.; YOUNAS, M.; JAMES, A.; SANDERSON, M. Sampling, information extraction and summarisation of Hidden Web databases. **Data and Knowledge Engineering**, [S.l.], v.59, n.2, p.213 – 230, 2006. Including: Sixth ACM International Workshop on Web Information and Data Management Sixth ACM International Workshop on Web Information and Data Management, in conjunction with the 13th International Conference on Information and Knowledge Management (CIKM 2003).

HOLMES, G.; DONKIN, A.; WITTEN, I. WEKA: a machine learning workbench. In: SECOND AUSTRALIAN AND NEW ZEALAND CONFERENCE ON INTELLIGENT INFORMATION SYSTEMS, 1994., 1994. **Proceedings...** [S.l.: s.n.], 1994. p.357 –361.

HTTP. Acesso em: jan. 2011, disponível em: `http://www.w3.org/Protocols/`.

IPEIROTIS, P. G.; GRAVANO, L. When one sample is not enough: improving text database selection using shrinkage. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2004., 2004, New York, NY, USA. **Proceedings...** ACM, 2004. p.767–778. (SIGMOD '04).

IPEIROTIS, P. G.; GRAVANO, L.; SAHAMI, M. Probe, count, and classify: categorizing hidden web databases. **SIGMOD Rec.**, New York, NY, USA, v.30, n.2, p.67–78, May 2001.

IPEIROTIS, P.; GRAVANO, L.; SAHAMI, M. Automatic Classification of Text Databases through Query Probing. In: GOOS, G.; HARTMANIS, J.; LEEUWEN, J.; SUCIU, D.; VOSSEN, G. (Ed.). **The World Wide Web and Databases**. [S.l.]: Springer Berlin Heidelberg, 2001. p.245–255. (Lecture Notes in Computer Science, v.1997).

KABRA, G.; LI, C.; CHANG, K. Query Routing: finding ways in the maze of the deep-web. In: INTERNATIONAL WORKSHOP ON CHALLENGES IN WEB INFORMATION RETRIEVAL AND INTEGRATION, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.64 – 73.

KHARE, R.; AN, Y.; SONG, I.-Y. Understanding deep web search interfaces: a survey. **SIGMOD Rec.**, New York, NY, USA, v.39, p.33–40, September 2010.

KLEINBERG, J. M.; KUMAR, R.; RAGHAVAN, P.; RAJAGOPALAN, S.; TOMKINS, A. S. The web as a graph: measurements, models, and methods. In: COCOON'99: PROC. OF THE 5TH ANNUAL INTERNATIONAL CONFERENCE ON COMPUTING AND COMBINATORICS, 1999, Berlin, Heidelberg. **Anais...** Springer-Verlag, 1999. p.1–17.

KUSHMERICK, N. Learning to Invoke Web Forms. In: MEERSMAN, R.; TARI, Z.; SCHMIDT, D. (Ed.). **On The Move to Meaningful Internet Systems 2003**: coopis, doa, and odbase. [S.l.]: Springer Berlin Heidelberg, 2003. p.997–1013. (Lecture Notes in Computer Science, v.2888).

LE, H.; CONRAD, S. **Classifying Structured Web Sources Using Aggressive Feature Selection**. 2009.

LERMAN, K.; GETOOR, L.; MINTON, S.; KNOBLOCK, C. Using the structure of Web sites for automatic segmentation of tables. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2004., 2004, New York, NY, USA. **Proceedings...** ACM, 2004. p.119–130. (SIGMOD '04).

LI, Y.; WANG, S.; SHEN, D.; NIE, T.; YU, G. Domain-Independent Classification for Deep Web Interfaces. In: CHEN, L.; TANG, C.; YANG, J.; GAO, Y. (Ed.). **Web-Age Information Management**. [S.l.]: Springer Berlin Heidelberg, 2010. p.453–458. (Lecture Notes in Computer Science, v.6184).

LIDDLE, S.; EMBLEY, D.; SCOTT, D.; YAU, S. Extracting Data behind Web Forms. In: OLIVE, A.; YOSHIKAWA, M.; YU, E. (Ed.). **Advanced Conceptual Modeling Techniques**. [S.l.]: Springer Berlin Heidelberg, 2003. p.402–413. (Lecture Notes in Computer Science, v.2784).

LIN, K.-I.; CHEN, H. Automatic information discovery from the "invisible Web". In: INFORMATION TECHNOLOGY: CODING AND COMPUTING, 2002. PROCEEDINGS. INTERNATIONAL CONFERENCE ON, 2002. **Anais...** [S.l.: s.n.], 2002. p.332 – 337.

LU, Y.; HE, H.; PENG, Q.; MENG, W.; YU, C. Clustering e-commerce search engines based on their search interface pages using WISE-Cluster. **Data and Knowledge Engineering**, [S.l.], v.59, n.2, p.231 – 246, 2006. Including: Sixth ACM International Workshop on Web Information and Data Management Sixth ACM International Workshop on Web Information and Data Management, in conjunction with the 13th International Conference on Information and Knowledge Management (CIKM 2003).

MADHAVAN, J.; AFANASIEV, L.; ANTOVA, L.; HALEVY, A. Y. Harnessing the Deep Web: present and future. **CoRR**, [S.l.], v.abs/0909.1785, 2009.

MADHAVAN, J.; KO, D.; KOT, L.; GANAPATHY, V.; RASMUSSEN, A.; HALEVY, A. Google's Deep Web crawl. **Proc. VLDB Endow.**, [S.l.], v.1, n.2, p.1241–1252, Aug. 2008.

MANNILA, H.; TOIVONEN, H.; INKERI VERKAMO, A. Discovery of Frequent Episodes in Event Sequences. **Data Mining and Knowledge Discovery**, [S.l.], v.1, p.259–289, 1997.

MARIN-CASTRO, H.; SOSA-SOSA, V.; LOPEZ-AREVALO, I. Automatic Identification of Web Query Interfaces. In: BATYRSHIN, I.; SIDOROV, G. (Ed.). **Advances in Soft Computing**. [S.l.]: Springer Berlin Heidelberg, 2011. p.297–306. (Lecture Notes in Computer Science, v.7095).

MENCZER, F.; PANT, G.; SRINIVASAN, P.; RUIZ, M. E. Evaluating topic-driven web crawlers. In: ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 24., 2001, New York, NY, USA. **Proceedings. . .** ACM, 2001. p.241–249. (SIGIR '01).

MERWE, D. van der; ENGELBRECHT, A. Data clustering using particle swarm optimization. In: EVOLUTIONARY COMPUTATION, 2003. CEC '03. THE 2003 CONGRESS ON, 2003. **Anais. . .** [S.l.: s.n.], 2003. v.1, p.215 – 220 Vol.1.

MICROSOFT Web Search Engine. Acesso em: dez. 2011, disponível em: `http://www.bing.com/`.

MITCHELL, T. The need for biases in learning generalizations. **Readings in machine learning**, [S.l.], p.184–191, 1980.

MITCHELL, T. M. **Machine learning**. [S.l.]: McGraw-Hill, 1997. I-XVII, 1-414p. (McGraw Hill series in computer science).

MORAES, M. C.; HEUSER, C. A.; MOREIRA, V. P.; BARBOSA, D. Pre-Query Discovery of Domain-specific Query Forms: a survey. **IEEE Transactions on Knowledge and Data Engineering**, Los Alamitos, CA, USA, v.99, n.PrePrints, 2012.

NGUYEN, H.; NGUYEN, T.; FREIRE, J. Learning to extract form labels. **Proc. VLDB Endow.**, [S.l.], v.1, n.1, p.684–694, Aug. 2008.

NOOR, U.; RASHID, Z.; RAUF, A. Article: a survey of automatic deep web classification techniques. **International Journal of Computer Applications**, [S.l.], v.19, n.6, p.43–50, April 2011. Published by Foundation of Computer Science.

NOY, N.; FERGERSON, R.; MUSEN, M. The Knowledge Model of Protege-2000: combining interoperability and flexibility. In: DIENG, R.; CORBY, O. (Ed.). **Knowledge Engineering and Knowledge Management Methods, Models, and Tools**. [S.l.]: Springer Berlin Heidelberg, 2000. p.17–32. (Lecture Notes in Computer Science, v.1937).

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

PASQUIER, N.; BASTIDE, Y.; TAOUIL, R.; LAKHAL, L. Discovering Frequent Closed Itemsets for Association Rules. In: INTERNATIONAL CONFERENCE ON DATABASE THEORY, 7., 1999, London, UK. **Proceedings. . .** Springer-Verlag, 1999. p.398–416. (ICDT '99).

PERKOWITZ, M.; DOORENBOS, R.; ETZIONI, O.; WELD, D. Learning to Understand Information on the Internet: an example-based approach. **Journal of Intelligent Information Systems**, [S.l.], v.8, p.133–153, 1997.

QUINLAN, J. Induction of decision trees. **Machine Learning**, [S.l.], v.1, p.81–106, 1986.

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

RASMUSSEN, C. Gaussian Processes in Machine Learning. In: **Advanced Lectures on Machine Learning**. [S.l.]: Springer Berlin / Heidelberg, 2004. p.63–71. (Lecture Notes in Computer Science, v.3176).

RENNIE, J.; MCCALLUM, A. Using reinforcement learning to spider the web efficiently. In: MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-, 1999. **Anais...** [S.l.: s.n.], 1999. p.335–343.

RU, Y.; HOROWITZ, E. Indexing the invisible web: a survey. **Online Information Review**, [S.l.], v.29, n.3, p.249–265, 2005.

RUSSELL, S.; NORVIG, P. **Artificial intelligence**: a modern approach. [S.l.]: Prentice hall, 2009.

SALTON, G.; WONG, A.; YANG, C. S. A vector space model for automatic indexing. **Commun. ACM**, New York, NY, USA, v.18, n.11, p.613–620, Nov. 1975.

SCHAPIRE, R. E. The Boosting Approach to Machine Learning: an overview. In: DENISON, D. D.; HANSEN, M. H.; HOLMES, C.; MALLICK, B.; YU, B. (Ed.). **Nonlinear Estimation and Classification**. [S.l.]: Springer, 2003.

SEBASTIANI, F. Machine learning in automated text categorization. **ACM Comput. Surv.**, New York, NY, USA, v.34, p.1–47, March 2002.

SHEN, D.; LI, M.; YU, G.; KOU, Y.; NIE, T. Efficient Top-k Data Sources Ranking for Query on Deep Web. In: BAILEY, J.; MAIER, D.; SCHEWE, K.-D.; THALHEIM, B.; WANG, X. (Ed.). **Web Information Systems Engineering - WISE 2008**. [S.l.]: Springer Berlin Heidelberg, 2008. p.321–336. (Lecture Notes in Computer Science, v.5175).

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

SONG, L.; MA, J.; YAN, P.; LIAN, L.; ZHANG, D. Clustering Deep Web Databases Semantically. In: LI, H.; LIU, T.; MA, W.-Y.; SAKAI, T.; WONG, K.-F.; ZHOU, G. (Ed.). **Information Retrieval Technology**. [S.l.]: Springer Berlin Heidelberg, 2008. p.365–376. (Lecture Notes in Computer Science, v.4993).

TRULIA. Acesso em: jan. 2011, disponível em: `http://www.trulia.com/`.

UIUC web integration repository. Acesso em: out. 2009, disponível em: `http://metaquerier.cs.uiuc.edu/repository`.

WALSH, A. E. (Ed.). **Uddi, Soap, and Wsdl**: the web services specification reference book. [S.l.]: Prentice Hall Professional Technical Reference, 2002.

WANG, J.; LOCHOVSKY, F. H. Data extraction and label assignment for web databases. In: WORLD WIDE WEB, 12., 2003, New York, NY, USA. **Proceedings...** ACM, 2003. p.187–196. (WWW '03).

WANG, Y.; LI, H.; ZUO, W.; HE, F.; WANG, X.; CHEN, K. Research on discovering deep web entries. **Computer Science and Information Systems**, [S.l.], v.8, n.3, p.779–799, 2011.

WANG, Y.; YANG, J.-M.; LAI, W.; CAI, R.; ZHANG, L.; MA, W.-Y. Exploring traversal strategy for web forum crawling. In: ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 31., 2008, New York, NY, USA. **Proceedings...** ACM, 2008. p.459–466. (SIGIR '08).

WANG, Y.; ZUO, W.; PENG, T.; HE, F. Domain-Specific Deep Web Sources Discovery. In: NATURAL COMPUTATION, 2008. ICNC '08. FOURTH INTERNATIONAL CONFERENCE ON, 2008. **Anais...** [S.l.: s.n.], 2008. v.5, p.202 –206.

WITTEN, I. H.; FRANK, E.; HALL, M. A. **Data Mining**: practical machine learning tools and techniques. 3.ed. Burlington, MA: Morgan Kaufmann, 2011.

WU, P.; WEN, J.-R.; LIU, H.; MA, W.-Y. Query Selection Techniques for Efficient Crawling of Structured Web Sources. In: DATA ENGINEERING, 2006. ICDE '06. PROCEEDINGS OF THE 22ND INTERNATIONAL CONFERENCE ON, 2006. **Anais...** [S.l.: s.n.], 2006. p.47.

XELDA. Acesso em: jan. 2011, disponível em: `http://www.xrce.xerox.com/Research-Development/Historical-projects/XeLDA/%28language%29/eng-GB`.

XU, H.; ZHANG, C.; HAO, X.; HU, Y. A Machine Learning Approach Classification of Deep Web Sources. In: FUZZY SYSTEMS AND KNOWLEDGE DISCOVERY, 2007. FSKD 2007. FOURTH INTERNATIONAL CONFERENCE ON, 2007. **Anais...** [S.l.: s.n.], 2007. v.4, p.561 –565.

XU, R.; WUNSCH D., I. Survey of clustering algorithms. **Neural Networks, IEEE Transactions on**, [S.l.], v.16, n.3, p.645 –678, may 2005.

YAHOO! BOSS API. Acesso em: jan. 2011, disponível em: `http://developer.yahoo.com/search/boss/`.

YAHOO! Web Search Engine. Acesso em: jan. 2011, disponível em: `http://www.yahoo.com/`.

YANG, Y.; PEDERSEN, J. A comparative study on feature selection in text categorization. In: MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-, 1997. **Anais...** [S.l.: s.n.], 1997. p.412–420.

YI, L.; LIU, B.; LI, X. Eliminating noisy information in Web pages for data mining. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 2003, New York, NY, USA. **Proceedings...** ACM, 2003. p.296–305. (KDD '03).

YU, C.; LIU, K.-L.; MENG, W.; WU, Z.; RISHE, N. A methodology to retrieve text documents from multiple databases. **Knowledge and Data Engineering, IEEE Transactions on**, [S.l.], v.14, n.6, p.1347 – 1361, nov/dec 2002.

ZHAI, Y.; LIU, B. Web data extraction based on partial tree alignment. In: WORLD WIDE WEB, 14., 2005, New York, NY, USA. **Proceedings...** ACM, 2005. p.76–85. (WWW '05).

ZHANG, Z.; HE, B.; CHANG, K. C.-C. Understanding Web query interfaces: best-effort parsing with hidden syntax. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2004., 2004, New York, NY, USA. **Proceedings...** ACM, 2004. p.107–118. (SIGMOD '04).

ZHAO, P.; HUANG, L.; FANG, W.; CUI, Z. Organizing Structured Deep Web by Clustering Query Interfaces Link Graph. In: TANG, C.; LING, C.; ZHOU, X.; CERCONE, N.; LI, X. (Ed.). **Advanced Data Mining and Applications**. [S.l.]: Springer Berlin Heidelberg, 2008. p.683–690. (Lecture Notes in Computer Science, v.5139).

ZIMMERMANN, H. **Fuzzy set theory–and its applications**. [S.l.]: Springer Netherlands, 2001.

ZUO, W.; WANG, Y.; WANG, X.; ZHANG, D.; PENG, T. Automatic classification of Deep Web databases based on centroid and WordNet. **Journal of Computational Information Systems**, [S.l.], v.6, n.1, p.63–70, 2010.

APPENDIX A RESUMO DA TESE

O assunto desta tese é a descoberta de formulários HTML, uma tarefa que pode ser dividida em duas etapas: (i) a localização e (ii) a identificação de formulários na Web. A localização de formulários trata do encontro de páginas HTML que contenham formulários. Por sua vez, a identificação de formulários trata da seleção de quais formulários localizados previamente são de fato relevantes de acordo com algum critério definido pelos usuários consumidores do sistema de descoberta de formulários.

A etapa de localização é desafiadora pois formulários HTML seguem uma distribuição esparsa na Web, o que significa que pequenas regiões da Web possuem muitos formulários enquanto grandes regiões possuem muito poucos. Em função disso, um localizador de formulários deve manter o foco nas pequenas regiões de interesse e evitar o processamento das grandes regiões irrelevantes da Web.

A etapa de identificação, por sua vez, é desafiadora pois formulários em geral não seguem regras claras de formação. Sua construção via-de-regra é feita manualmente por humanos para "consumo" humano por meio de renderização em um software navegador (i.e. um *browser*). Devido à ausência de regras explícitas de formação de formulários HTML, até mesmo formulários com funções semelhantes e relativos a um mesmo domínio não são facilmente identificados por métodos automáticos.

As várias técnicas encontradas na literatura definem algumas dimensões para o problema da descoberta de formulários: (i) domínio *vs.* função, (ii) estrutura de formulários e (iii) abordagens pré-consulta *vs.* pós-consulta. A primeira diz respeito aos dois tipos de identificação: se um formulário é de consulta (i.e. se sua função principal é buscar informações) ou não. A segunda diz respeito à característica dos formulários de fornecer pistas a respeito do esquema das bases de dados aos quais eles permitem acesso. Por fim, a terceira diz respeito ao uso da submissão de formulários na identificação dos mesmos.

O escopo desta tese está restrito à descoberta de formulários estruturados de domínio-específico através de abordagens do tipo pré-consulta. Ou seja, esta tese trata apenas da descoberta de formulários que fornecem pistas a respeito do esquema das bases de dados aos quais eles permitem acesso; tais formulários devem obrigatoriamente relacionar-se a um determinado assunto de interesse; e, por fim, a descoberta desses formulários não deve envolver submissões de dados.

A motivação desta tese é aumentar a automatização do crawling na Web Profunda. A Web Profunda é o complemento das páginas que são normalmente processadas por motores de busca tradicionais (e.g. Google, Yahoo!). A Web Profunda é composta por diversos tipos de hiperdocumentos, como, por exemplo, páginas sensíveis ao contexto, ou que requerem autenticação, ou documentos com formatos proprietários. Uma parte bastante relevante da Web Profunda é a composta por páginas que demandam a submissão de formulários para serem acessadas (i.e. páginas "escondidas atrás" de formulários de consulta). São essas as páginas que são buscadas por um crawler da Web Profunda, que pode ter sua tarefa dividida em três etapas: (i) descobrir, (ii) preencher e (iii) tratar o resultado da submissão de formulários. Por focar na primeira etapa do crawling na Web Profunda, pode-se dizer que esta tese colabora com o avanço do estado-da-arte do crawling na Web Profunda como um todo.

São 19 os trabalhos relacionados à esta tese, que podem ser agrupados em cinco grupos: (i) os Crawlers da Web Profunda, (ii) os Classificadores de Formulários, (iii) os Agrupadores de Formulários, (iv) os Crawlers de Formulários e, finalmente, (v) os Ranqueadores de Formulários. Os Crawlers da Web Profunda representam as técnicas mais antigas de descoberta de formulários e executam as três etapas do crawling na Web Profunda mencionadas anteriormente. Os Classificadores de Formulários, dado um treinamento manual, identificam formulários de acordo com a sua função ou domínio. Os Agrupadores de Formulários recebem como entrada um conjunto de formulários e entregam como saída grupos de formulários semelhantes entre si, em função do domínio a que supostamente pertencem. Os Crawlers de Formulários especilizaram-se na localização de formulários, focando nas pequenas regiões da Web densas em formulários e ignorando as outras regiões. Por fim, os Ranqueadores de Formulários recebem como entrada um conjunto de formulário e requisitos de usuários e entregam como saídam os formulários ordenados em ordem decrescente, de acordo com os requisitos de usuários.

Um estudo detalhado de eventos ocorridos ao longo dos últimos 15 anos mostra que há uma clara tendência de aumento contínuo da automatização das técnicas de descoberta de formulários. No entanto, uma análise de dependências dos diversos grupos de trabalhos relacionados entre si e a fatores externos mostra que ainda há uma forte dependência das técnicas, direta ou indiretamente, em relação ao especialista humano. Dentro desse contexto, esta tese é apresentada como o próximo passo na identificada tendência em direção à completa automatização da descoberta de formulários.

A proposta contida aqui resume-se basicamente a eliminar o treinamento manual de classificadores de formulários. O atual estado-da-arte em classificação de formulários requer o rotulamento manual de dezenas a centenas de formulários por domínio de interesse, o que constitui, no mínimo, uma tarefa repetitiva e cansativa. O método de treinamento contido nesta tese requer do usuário especialista apenas a definição do nome do domínio de interesse (e.g. *"hotéis"*, *"carros"*, etc).

O método proposto é basedo em heurísticas e validado empiricamente. Em uma visão geral, o método segue as seguintes etapas: (i) o usário define um conjunto pequeno de palavras-chave que define o domínio de interesse (i.e. o nome do domínio); (ii) o sistema interage com um motor de busca e obtém um conjunto de URLs a serem "crawleadas"; (iii) alguns dos formulários encontrados durante o crawling são coletados; (iv) os formulários coletados são identificados por meio de heurísticas; finalmente, (v) os formulários identificados são utilizados para treinar um classificador de formulários.

Nossos experimentos fazem uso de uma amostra aleatória com milhares de formulários da base DeepPeep, que foi construída pelos pesquisadores da Universidade de Utah, líderes na criação de técnicas relacionadas à descoberta de formulários na Web. Além disso, variados cenários de teste foram explorados, com o intuito de observar o comportamento do estado-da-arte em classificação de formulários quando treinado manualmente e quando treinado automaticamente usando a nossa proposta. Complementarmente, pontos minuciosos foram explorados, medidos e ponderados a fim de tentar compreender e caracterizar em detalhes o comportamento da solução proposta.

Resumidamente, as principais conclusões a que chegamos a partir dos nossos experimentos foram as seguintes:

- É possível treinar automaticamente os classificadores de formulários GFC e DSFC usando o método proposto;
- O método proposto depende da qualidade dos nomes de domínio utilizados;
- O método proposto depende da capacidade do motor de busca de retornar URLs relevantes para a descoberta de formulários a ser executada.

Dado que os resultados dos experimentos foram favoráveis ao objetivo da nossa proposta, podemos afirmar que as contribuições desta tese são duas:

- Um survey de técnicas relacionadas à descoberta de formulários, agrupadas por seu objetivo final e comportamento. O survey evidencia um conjunto de eventos que caracteriza inequivocamente uma tendência de aumento da automatização das técnicas de descoberta de formulários ao longo dos últimos 15 anos;
- Um método automático de treinamento para classficadores de formulários que requer do especialista humano apenas a definição do nome do domínio de interesse.