

# Automatic Construction and Adaptation of Wrappers for Semi-Structured Web Documents

Wong Tak Lam



A Thesis Submitted in Partial Fulfilment  
of the Requirements for the Degree of  
Master of Philosophy  
in  
Systems Engineering and Engineering Management

© The Chinese University of Hong Kong  
June 2003

The Chinese University of Hong Kong holds the copyright of this thesis. Any person(s) intending to use a part or whole of the materials in the thesis in a proposed publication must seek copyright release from the Dean of the Graduate School.



A Thesis Submitted in Partial Fulfillment  
of the Requirements for the Degree of  
Master of Philosophy

iii

Department of Mechanical and Electrical Engineering

© The Chinese University of Hong Kong  
June 2004

The Chinese University of Hong Kong holds the copyright in this thesis. No part of this thesis may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the Chinese University of Hong Kong.

# 摘要

許多現存的封裝 (Wrapper) 學習方法只能夠處理結構簡單的文件。爲了處理更豐富的半結構文件如互聯網網頁，以及減少使用者的負荷，我們發展了一套新的封裝歸納提案。我們的提案運用了兩階段的自動學習工作，分別稱爲階式紀錄結構推論及抽取規則歸納，在階式紀錄結構推論中，我們會自動爲訊息來源的紀錄產生階式結構的描述，在抽取規則歸納中，對應階式紀錄結構中的每一個節點的抽取規則會被歸納。這樣的設計能夠抽取含有缺掉的屬性項，多重值的屬性項，以及沒有次序限制的屬性項。我們亦發展了一套構架去處理封裝適應的問題，我們的構架試圖使早前學習得到的封裝及早前收集得到的詞彙適應到一個先前未被看見之網站。這是一個兩階段，以及運用了多種學習範式的方法，用以消除封裝適應的挑戰。第一階段是從先前未被看見之網站當中搜尋潛在的候選訓練樣本，我們發展了一個修改了的最鄰近分類法，用以辨認出適當的文字片斷作爲潛在的候選訓練樣本。潛在的候選訓練樣本隨後會被一個文字片斷分類模型所分類，「良好」的潛在的候選訓練樣本會被視爲先前未被看見之網站的機器註釋訓練樣本。根據機器註釋訓練樣本，一個爲先前未被看見之網站而全新定造的封裝便能學習得到。我們陳述封裝歸納及封裝適應在不同領域之網站的實驗結果。

# Abstract

Many existing wrapper learning methods can only handle documents with simple structures. To handle a richer set of semi-structured documents such as Web documents and minimize the burden of users, we develop a new wrapper induction approach. Our approach employs a two-stage learning task, namely, hierarchical record structure inference and extraction rule induction. In hierarchical record structure inference, we automatically generate a representation of hierarchical structure for the records in an information source. In extraction rule induction, extraction rules are induced for each node in the hierarchical record structure. This design is able to extract records which have missing attribute items, multi-valued attribute items, and attribute items in unrestricted order. We also develop a framework to solve the wrapper adaptation problem. Our framework attempts to adapt a previously learned wrapper and previously collected lexicons to an *unseen* Web site. It is a two-stage method employing multiple learning paradigms in order to tackle the challenges in wrapper adaptation. The first stage is to seek *potential training example candidates* from the unseen Web site. A modified nearest neighbour classification model is developed for identifying appropriate text fragments as potential training example candidates. The potential training example

candidates will then be classified by a text fragment classification model in the second stage. Those “good” candidates will be considered as *machine annotated training examples* for the unseen Web site. Based on the machine annotated training examples, a new wrapper tailored to the unseen Web site can be learned. We present experimental results on wrapper induction and wrapper adaptation for different real-world Web sites of different domains.

# Acknowledgments

There are many people whom I would like to thank for their support and contributions to this research.

Firstly, I would like to express my sincere gratitude to my research advisor, Prof. Wai Lam. His constant encouragement and advice contributed a great deal in this research. I would also like to thank Prof. Youhua Chen and Prof. Chuen Chi Yang for their helpful suggestions and comments on improving the quality of my work.

Next, I would like to thank my family for their support during my research study. My parents always support me in every challenge I undertake and provide me a comfortable and warm environment. My brother and sister-in-law, Eric and Vivian always give me endless encouragement.

Thanks must go to my friends, Fan, Ivy, Rachel, Jelly, Foo, Kevin, Jian Feng, for their daily support and help. Zoe, Monkey, Cindy, No, Wai, Siu Ming always give me encouragement.

Lastly, I would dedicate my work to my best friend Natalie. She is my source of love and joy. Thanks for her valuable support and encouragement in my daily life.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Wrapper Induction for Semi-structured Web Documents . . . .	1
1.2	Adapting Wrappers to Unseen Web Sites . . . . .	6
1.3	Thesis Contributions . . . . .	7
1.4	Thesis Organization . . . . .	8
<b>2</b>	<b>Related Work</b>	<b>10</b>
2.1	Related Work on Wrapper Induction . . . . .	10
2.2	Related Work on Wrapper Adaptation . . . . .	16
<b>3</b>	<b>Automatic Construction of Hierarchical Wrappers</b>	<b>20</b>
3.1	Hierarchical Record Structure Inference . . . . .	22
3.2	Extraction Rule Induction . . . . .	30
3.3	Applying Hierarchical Wrappers . . . . .	38
<b>4</b>	<b>Experimental Results for Wrapper Induction</b>	<b>40</b>
<b>5</b>	<b>Adaptation of Wrappers for Unseen Web Sites</b>	<b>52</b>
5.1	Problem Definition . . . . .	52
5.2	Overview of Wrapper Adaptation Framework . . . . .	55
5.3	Potential Training Example Candidate Identification . . . . .	58
5.3.1	Useful Text Fragments . . . . .	58
5.3.2	Training Example Generation from the Unseen Web Site	60
5.3.3	Modified Nearest Neighbour Classification . . . . .	63

5.4	Machine Annotated Training Example Discovery and New Wrapper Learning . . . . .	64
5.4.1	Text Fragment Classification . . . . .	64
5.4.2	New Wrapper Learning . . . . .	69
<b>6</b>	<b>Case Study and Experimental Results for Wrapper Adaptation</b>	<b>71</b>
6.1	Case Study on Wrapper Adaptation . . . . .	71
6.2	Experimental Results . . . . .	73
6.2.1	Book Domain . . . . .	74
6.2.2	Consumer Electronic Appliance Domain . . . . .	79
<b>7</b>	<b>Conclusions and Future Work</b>	<b>83</b>
	<b>Bibliography</b>	<b>88</b>
<b>A</b>	<b>Detailed Performance of Wrapper Induction for Book Domain</b>	<b>95</b>
<b>B</b>	<b>Detailed Performance of Wrapper Induction for Consumer Electronic Appliance Domain</b>	<b>99</b>



# List of Figures

1.1	A sample of a Web page about book catalog . . . . .	4
1.2	An excerpt of the HTML texts for the Web page shown in Figure 1.1 . . . . .	5
3.1	The hierarchical record structure for the book information shown in Figure 1.1 . . . . .	21
3.2	A situation for “cover” . . . . .	25
3.3	The outline of the hierarchical record structure inference algorithm . . . . .	25
3.4	The outline of the funcion <i>Reduce_Nodes</i> used by the hierarchical record structure inference algorithm . . . . .	26
3.5	A sample of training example for the record structure inference algorithm . . . . .	28
3.6	An other sample of training example for the record structure inference algorithm . . . . .	28
3.7	The raw record structure constructed by the record structure inference algorithm from the training examples as shown in Figures 3.5 and 3.6 . . . . .	28
3.8	A sampe of possible resulting hierarchical record structure . . . . .	29
3.9	Different training examples from the Web site as shown in Figure 1.1 for the hierarchical record structure inference . . . . .	31
3.10	Examples of semantic classes organized in a hierarchy . . . . .	33
3.11	The outline of the extraction rule induction algorithm . . . . .	34
3.12	The <i>Generate_And_Test</i> function used by the extraction rule induction algorithm . . . . .	35

3.13	The <i>Post-Prune</i> function used by the extraction rule induction algorithm . . . . .	35
3.14	The score functions used by the rule induction algorithm . . .	37
3.15	The <i>Find_Best_N</i> function, and the <i>Find_Common_Pattern</i> function used by the rule induction algorithm . . . . .	37
3.16	A sample of extraction rule for the final price of the Web page shown in Figure 1.1 . . . . .	39
4.1	A sample of a testing Web page of S5 . . . . .	46
4.2	The hierarchical record structure for the book information in the Web sites S1, S2, and S10 . . . . .	48
4.3	A sample of a testing Web page of S18 . . . . .	50
5.1	A sample of a Web page containing book records coming from a different Web site shown in Figure 1.1 . . . . .	53
5.2	An excerpt of the HTML texts for the Web page shown in Figure 5.1 . . . . .	54
5.3	Our wrapper adaptation framework . . . . .	56
5.4	A sample of Web page about networking books . . . . .	60
5.5	An excerpt of the HTML texts for the Web page shown in Figure 5.4 . . . . .	61

# List of Tables

4.1	Information sources for experiments . . . . .	42
4.2	The number of user annotated training records for learning wrapper for each Web site . . . . .	43
4.3	A sample of a user annotated training example for the Web page shown in Figure 1.1 . . . . .	43
4.4	Performance summary of our wrapper induction approach on the Web sites containing book catalogs . . . . .	45
4.5	Performance summary of our wrapper induction approach on the Web sites containing consumer electronic appliance . . . . .	48
5.1	Normalized character-level edit distances between tokens . . . . .	68
6.1	Samples of a machine annotated training example obtained by adapting the wrapper from the Web site S8 (Figure 1.1) to the Web site shown in S7 (Figure 5.1). . . . .	72
6.2	Experimental results of applying a learned wrapper without adaptation from one Web site to extract contents from the remaining sites in the book domain. . . . .	75
6.3	Experimental results of adapting a learned wrapper from one Web site to the remaining sites in the book domain. . . . .	77
6.4	Average extraction performance on <i>title</i> , <i>author</i> , and <i>price</i> for the book domain for the cases of without adaptation and with adaptation when training examples of one particular Web site are provided. (prec. refers to precision) . . . . .	78

6.5	Experimental results of applying a learned wrapper without adaptation from one Web site to extract contents from the remaining sites in the consumer electronic appliance domain.	80
6.6	Experimental results of adapting a learned wrapper from one Web site to the remaining sites in the consumer electronic appliance domain. . . . .	81
6.7	Average extraction performance on <i>model number</i> , <i>description</i> , and <i>price</i> for the electronic appliance domain for the cases of without adaptation and with adaptation when training examples of one particular Web site are provided. (prec. refers to precision) . . . . .	82
A.1	Performance of our wrapper induction approach for each testing page in the Web sites S1 (P and R refer to precision and recall respectively.) . . . . .	95
A.2	Performance of our wrapper induction approach for each testing page in the Web sites S2 (P and R refer to precision and recall respectively.) . . . . .	95
A.3	Performance of our wrapper induction approach for each testing page in the Web sites S3 (P and R refer to precision and recall respectively.) . . . . .	96
A.4	Performance of our wrapper induction approach for each testing page in the Web sites S4 (P and R refer to precision and recall respectively.) . . . . .	96
A.5	Performance of our wrapper induction approach for each testing page in the Web sites S5 (P and R refer to precision and recall respectively.) . . . . .	96
A.6	Performance of our wrapper induction approach for each testing page in the Web sites S6 (P and R refer to precision and recall respectively.) . . . . .	96
A.7	Performance of our wrapper induction approach for each testing page in the Web sites S7 (P and R refer to precision and recall respectively.) . . . . .	97

A.8	Performance of our wrapper induction approach for each testing page in the Web sites S8 (P and R refer to precision and recall respectively.) . . . . .	97
A.9	Performance of our wrapper induction approach for each testing page in the Web sites S9 (P and R refer to precision and recall respectively.) . . . . .	97
A.10	Performance of our wrapper induction approach for each testing page in the Web sites S10 (P and R refer to precision and recall respectively.) . . . . .	98
A.11	Performance of our wrapper induction approach for each testing page in the Web sites S11 (P and R refer to precision and recall respectively.) . . . . .	98
B.1	Performance of our wrapper induction approach for each testing page in the Web sites S12 (P and R refer to precision and recall respectively.) . . . . .	99
B.2	Performance of our wrapper induction approach for each testing page in the Web sites S13 (P and R refer to precision and recall respectively.) . . . . .	100
B.3	Performance of our wrapper induction approach for each testing page in the Web sites S14 (P and R refer to precision and recall respectively.) . . . . .	100
B.4	Performance of our wrapper induction approach for each testing page in the Web sites S15 (P and R refer to precision and recall respectively.) . . . . .	100
B.5	Performance of our wrapper induction approach for each testing page in the Web sites S16 (P and R refer to precision and recall respectively.) . . . . .	101
B.6	Performance of our wrapper induction approach for each testing page in the Web sites S17 (P and R refer to precision and recall respectively.) . . . . .	101

B.7 Performance of our wrapper induction approach for each testing page in the Web sites S18 (P and R refer to precision and recall respectively.) . . . . .	102
B.8 Performance of our wrapper induction approach for each testing page in the Web sites S19 (P and R refer to precision and recall respectively.) . . . . .	102

## Chapter 1

# Introduction

## 1.1 Wrapper Induction for Semi-structured Web Documents

The trailing rise and rapid growth of World Wide Web provides a rich source of online electronic documents. Users usually access information from the Web by visual browsing and browsing. However, the amount of information and large quantity of documents on the Web is so large that it is hard for a system that can extract pertinent information from these documents. Text Extraction (TE) system and its related techniques have been used to extract information from web documents. The data extracted can be used as a database or used for other important tasks [1, 4]. The purpose of this book is to propose a wrapper induction method for extracting information from web documents. The proposed wrapper induction method is designed to reduce the amount of information extracted. For example, in the Message Determining, Deane [2] (1997), [3], [5].

# Chapter 1

## Introduction

### 1.1 Wrapper Induction for Semi-structured Web Documents

The continuous and rapid growth of World Wide Web provides a vast amount of online electronic documents. Users usually obtain information from the Web by manual browsing and keyword searching. However, the lack of automation and large quantity of imprecise data returned raise the need for a system that can extract precise and useful information automatically. Information Extraction (IE) systems aim at extracting such kind of information from text documents. The data extracted can be stored in a database or used for other intelligent tasks [16, 47, 57]. Different kinds of IE systems are proposed to extract information from different kinds of documents. One kind of IE systems is designed to extract information from natural language texts. For example, in the Message Understanding Conferences (MUCs) [14, 15],

the perpetrator names, victim names, instruments, and locations of attack are extracted from a collection of newswire articles on Latin American terrorism. The other example is the question answering task of the Text REtrieval Conference (TREC) [17]. The goal of this task is to find short phrases for answering questions from a collection of texts. Natural language processing (NLP) techniques such as syntactic parsers are usually employed for extracting information from natural language texts. Another kind of IE systems is designed to extract information from structured texts [8]. Uniform syntactic rules such as labels and mark-up tags are employed to deal with the rigid format of structured texts. Unlike natural language texts and structured texts, semi-structured texts are characterized by the fact that they are not fully well formatted and are not totally grammatically correct. HTML and XML documents are examples of semi-structured documents. Various IE systems are designed to deal with semi-structured documents [1, 5, 19, 24, 50]. A promising approach for extracting information from semi-structured documents is to make use of wrappers.

A wrapper usually consists of extraction rules or extraction patterns, which can identify the attribute items of interest. In the past, wrappers are constructed manually by human experts [33]. The manual construction of wrapper is time-consuming, tedious, and error-prone. Wrapper learning systems try to solve this problem by automatically constructing wrappers from the user provided training examples. Several wrapper learning systems [10, 12, 22, 23, 35, 36, 38, 39, 46, 50] have been proposed. They make use of machine learning techniques to discover the wrappers from the user annotated training examples.



Figure 1.1 shows a sample of a Web page containing information about book catalog<sup>1</sup>. Figure 1.2 depicts the excerpt of the HTML text document associated with the Web page. The attribute items of interest are book title, author(s), list price, final price. In this example, the last record has a book title “Foundations of Visual C++ Programming for Windows 95”, a list of authors “Paul Yao”, “Joseph Yao”, a list price “39.90” and a final price “19.99”. A user can simply provide few training examples of attribute items on a Web page via a graphical user interface. Wrapper induction aims at discovering the extraction rules of the wrapper from these training examples. The learned wrapper is able to extract precise attribute items from different Web pages of a particular Web site where the user annotated training examples have been provided.

Some of the wrapper learning systems construct single-slot extraction rules which can identify a single attribute item. However, the relationship between the attribute items in a record cannot be well represented. Some systems construct multi-slot extraction rules which can identify one or more attribute items in a record. However, these wrappers may fail when the records contain missing attribute items, multi-valued attribute items, or attribute items in unrestricted order. Stalker [46] used hierarchical record structure to represent the relationship of the attribute items. This record structure can solve some shortcomings of the single-slot and multi-slot extraction rules. However, users have to provide the hierarchical record structure to the wrapper learning system in advance.

---

<sup>1</sup>The URL associated with the Web page shown in Figure 1.1 is *www.halfpricecomputerbooks.com*.






<p>SEE LIST OF ALL CATEGORIES  <a href="#">see full category map</a></p>	<p>LIST PRICE: → <del>39.98</del> YOU SAVE: → 7.99          Our Price: → 31.96</p>	<p>FREE shipping and more than 1000 different          retail locations.</p>
<p></p>	<p><b><u>C++/C# Programmers Guide for Windows 2000</u></b>          Author: Ronald D. Reeves Published: 2001          List Price: → <del>49.98</del> You Save: → 25.00 Our          Price: → 24.99</p>	<p> <a href="#">Canadian order here!</a></p>
<p></p>	<p><b><u>C++ For Fortran Programmers</u></b>          Author: Ira Pohl Published: 1997          List Price: → <del>39.98</del> You Save: → 20.00 Our          Price: → 19.99</p>	
<p></p>	<p><b><u>C++ Object Databases : Programming With the          Odmg Standard (Object Technology Series)</u></b>          Author: David Jordan Published: 1997          List Price: → <del>39.94</del> You Save: → 19.98 Our          Price: → 19.97</p>	
<p></p>	<p><b><u>The Visual C++ 5 Programmers Reference:          Windows 95/Nt</u></b>          Author: Richard C. Leinecker Published: 1997          List Price: → <del>49.98</del> You Save: → 25.00 Our          Price: → 24.99</p>	
<p><a href="#">How to Order</a>  <a href="#">Contact Info</a>  <a href="#">Privacy Policy</a>  <a href="#">Waiting List</a>  <a href="#">FAQ/Help</a>  <a href="#">Affiliate Program</a>  <a href="#">New Arrivals</a>  <a href="#">HP/IB Bestsellers</a>  <a href="#">International FAQ</a>  <a href="#">XML Feeds</a></p>	<p><b><u>Mastering Microsoft Visual C++ 2 Programming,          with Disk</u></b>          Author: Michael Young Published: 1996          List Price: → <del>44.98</del> You Save: → 22.50 Our          Price: → 22.49</p>	
	<p><b><u>Foundations of Visual C++ Programming for          Windows 95</u></b>          Author: Paul Yao, Joseph Yao Published: 1995          List Price: → <del>39.98</del> You Save: → 20.00 Our          Price: → 19.99</p>	

Figure 1.1: A sample of a Web page about book catalog

...

```
<br> <b> <font face="Verdana, Arial, Helvetica, sans-serif" size="2" color="#666666"> <a
href="/book/1566047315"> The Visual C++ 5 Programmers Reference: Windows 95/Nt </a> </b>
<br> <b> Author: </b> Richard C. Leinecker &nbsp; &nbsp; <b> Published: </b> 1997 <br> <b>
List Price:  <strike> 49.98 </strike>
&nbsp; &nbsp; <b> You Save:  25.00
&nbsp; &nbsp; <b> Our Price:  24.99
&nbsp; &nbsp; </b> </font> <br> <br> <b> <font face="Verdana, Arial, Helvetica, sans-serif"
size="2" color="#666666"> <a href="/book/078211606X"> Mastering Microsoft Visual C++ 2
Programming, with Disk </a> </b> <br> <b> Author: </b> Michael Young &nbsp; &nbsp; <b>
Published: </b> 1996 <br> <b> List Price:  <strike> 44.98 </strike> &nbsp; &nbsp; <b> You Save:  22.50 &nbsp; &nbsp; <b> Our Price:  22.49 &nbsp; &nbsp; </b> </font> <br> <br> <b> <font
face="Verdana, Arial, Helvetica, sans-serif" size="2" color="#666666"> <a href="/book/1568843216">
Foundations of Visual C++ Programming for Windows 95 </a> </b> <br> <b> Author: </b> Paul
Yao, Joseph Yao &nbsp; &nbsp; <b> Published: </b> 1995 <br> <b> List Price:  <strike> 39.98 </strike> &nbsp; &nbsp; <b>
You Save:  20.00 &nbsp; &nbsp; <b> Our
Price:  19.99 &nbsp; &nbsp; </b>
</font> <br> <br>
...
```

Figure 1.2: An excerpt of the HTML texts for the Web page shown in Figure 1.1

## 1.2 Adapting Wrappers to Unseen Web Sites

As the layout format of Web sites changes from time to time, a previously constructed wrapper may become obsolete sooner or later. Wrapper maintenance aims at re-learning a new wrapper when the current wrapper can no longer extract correct information. Some methods [37, 42] are proposed to evaluate the validity of the wrappers. However, they can only partially solve the wrapper maintenance problem.

When a wrapper is found to be obsolete, a new wrapper may be re-learned using previously collected training examples. Most likely, these training examples may also become invalid. Besides, the wrappers learned from a particular Web site typically cannot be applied to extract attribute items from other sites. A separate effort is required to annotate a new set of training examples in the new Web site so as to learn a new wrapper for the new site. A possible solution for solving this problem is to address the issue of the preparation of training examples. Several approaches [4, 13, 45] have been proposed to tackle the problem of preparing training examples. However, manual work is still required in these approaches. Another solution is wrapper adaptation. Wrapper adaptation aims at adapting a previously learned wrapper from a source Web site to a new, unseen target site in the same domain. This can also solve the wrapper maintenance problem. Wrapper adaptation problem has two challenges. One challenge is that the layout format of Web pages are different in different Web sites. The other challenge is that the format of the attribute items may also be different in different Web sites although they are referring to the same concept or object.

## 1.3 Thesis Contributions

We develop a framework to solve both the wrapper induction problem and the wrapper adaptation problem. Our wrapper induction approach is a two-stage learning task. The first stage is hierarchical record structure inference and the second stage is extraction rule induction. Existing approaches for wrapper induction can only handle records with simple and flat structure. This poses a limitation on the representation of the structure of the records. Muslea et al. [46] proposed a hierarchical record structure for representing the structure of the records. However, the hierarchical record structure is required to be identified by user in advance. In the first stage of our wrapper induction approach, we try to automate the identification of the record structure by using a machine learning approach. Our system tries to automatically generate a representation of hierarchical structure for the records in an Website based on the user annotated training examples. Our hierarchical record structure is a tree-like structure which models the relationship between the attribute items of a record. It allows missing attribute items, multi-valued attribute items and attribute items in unrestricted order. Based on the learned record structure, extraction rules are learned for the extraction task in the second stage. A set of extraction rules are associated with each node in the hierarchical record structure. In the rule induction process, we incorporate both lexical and semantic generalization so that more expressive rules can be learned. In most of the previous approaches, only the surrounding tokens of the target attribute items are considered when constructing the wrappers. We observe that in addition to the surrounding tokens, the semantic content

of the attribute item itself can be exploited to enrich the expressiveness of the rules.

The objective of wrapper adaptation is to adapt the previously learned wrapper to a new unseen Web site. Existing approaches for wrapper adaptation can either only partially solve the problem, or solve the problem in a semi-automatic manner. We develop a wrapper adaptation framework which can fully automate this task. The idea of our adaptation framework is to automatically seek some training examples for learning a new wrapper for the unseen Web site. Our adaptation framework is a two-stage method employing multiple learning paradigms in order to tackle the challenges in wrapper adaptation. The first stage is to seek *potential training example candidates* from the unseen Web site. In the second stage the potential training example candidates will then be classified by a text fragment classification model. Those “good” potential training example candidates will be considered as *machine annotated training examples* for the unseen Web site. From these machine annotated training examples, a new wrapper for the unseen Web site can be learned to extract information.

## 1.4 Thesis Organization

This thesis is organized as follows: Chapter 2 gives a review on the related work for wrapper induction and wrapper adaptation. Chapter 3 presents the detail of our wrapper induction approach. In Chapter 4, some experimental results of our wrapper induction approach will be presented. Chapter 5 describes the detail of our wrapper adaptation framework. Chapter 6 presents



# Chapter 2

## Related Work

This chapter presents a brief review on the related work to wrapper induction and wrapper adaptation.

### 2.1 Related Work on Wrapper Induction

Our wrapper induction approach is a two stage learning task. The first stage is to infer the hierarchical record structure, which is used to represent the relationship among the attribute items in a record. This task is related to regular or tree grammar inference problems [28, 31]. One approach to infer regular grammars is to introduce characterizable algorithms [26]. These algorithms make various assumptions on classes of languages. For example, Augluin focuses on  $k$ -reversible languages [2]. Some approaches employ heuristic algorithms [6, 43, 58]. A recent approach for inferring stochastic regular grammars of text database structure based on state-merging method has been developed [58]. Another approach has been proposed for regular



tree grammar inference from stochastic samples when structural information is available [6]. All these existing approaches are not suitable for our task since our record structure is hierarchical and allows multi-valued attribute items, missing attribute items, and attribute items arranged in unrestricted order. Moreover, these existing approaches require both positive and negative examples. In our hierarchical record structure inference, only positive examples are available.

Wrappers usually consist of extraction rules or extraction patterns to identify the attribute items of interest. The second stage of our wrapper induction approach is to induce the extraction rules for identifying the attribute items of interest. A number of different methods have been proposed to automatically induce extraction rules. Compared to the manual wrapper construction, these methods dramatically reduce both the time and effort required to build a wrapper for an information source.

One kind of extraction rules is single-slot extraction rules. Such extraction rules can identify one attribute item each time. Many wrapper induction systems can learn single-slot extraction rules. Kushmerick attempted to formalize the wrapper induction task. He proposed a system called WIEN [38] which can learn six wrapper classes using machine learning techniques. The system tries to find the common prefix and suffix of the attribute items at character level. Hence the expressiveness of the extraction rules is quite limited.

Freitag and Kushmerick introduced an approach, called BWI [23], to building a trainable information extraction system. BWI makes use of boosting technique to improve the performance of a simple machine learning al-

gorithm. It learns relatively simple contextual patterns which identify the prefixes and suffixes of the relevant text fields, and only capture the length information of the relevant text fields.

RAPPIER [5] is an inductive logic programming system which employs a specific-to-general learning algorithm for learning extraction rules. The extraction rules include constraints on the words, part-of-speech tags from a part-of-speech tagger [3] and semantic classes from WordNet [44]. Freitag developed a system called SRV [22]. It uses relational learning algorithm to generate first-order logic extraction patterns. The rule is able to incorporate orthographic features, and other information such as tokens' lengths, part-of-speech tags, semantic classes from WordNet, and link grammars. Both RAPPIER and SRV pose constraints on the length of the attribute items. If the length of the attribute items varies too much, RAPPIER and SRV may not be able to identify the attribute items precisely.

Embley et al. [21] exploited a conceptual-modeling approach to extracting structure data automatically. This approach is based on finding the record boundary using several heuristics such as highest tag count, identifiable separator, etc. [20]. A disadvantage of this approach is that an ontology, which describes the data of interest including the relationships, lexical appearance, cardinality constraints, and context keywords, is needed to be defined by expert in advance.

A system known as WAWA-IE [18] is developed for information extraction from texts based on theory refinement. It employs neural network and part-of-speech tagging to achieve the task. It generates a set of candidate extraction. Each candidate is then judged by the trained neural network and

output those candidate that exceeds a system-selected threshold. However, users' instructions are needed to be provided to the system in advance.

$(LP)^2$  [10] is a system developed by Ciravegna. It performs bottom-up generalization from the training examples to discover the extraction rules. Two kinds of extraction rules are induced by the system. One is tagging rules which is to identify the attribute items. The other one is correction rules which is to improve the precision by correcting the mistake made by of the tagging rules. It employs shallow natural language processing in the generalization process of the extraction rules. However, the large search space and the lack of intelligence make  $(LP)^2$  inefficient.

All the above systems learn single-slot extraction rules. This poses a serious limitation on the systems as the relationships between the attribute items are lost. The other kind of extraction rules is multi-slot extraction rule. Such kind of extraction rules identifies one or more attribute items simultaneously.

WHISK [50] is a system which can learn multi-slot extraction rules. It can handle documents ranging from highly structured texts as well as natural language texts. The extraction rule of WHISK uses multiple landmarks for extracting multiple attribute items. Hence the relationship between the attribute items can be retained. However, if there are missing attribute items, multi-valued attribute items, or attribute items in unrestricted order, such kind of extraction rules may fail.

Hsu, et al. developed a system known as SoftMealy [35]. They use finite-state transducer (FST) to model information extraction problems. SoftMealy can handle missing attribute items, multi-valued attribute items, and

attribute items in unrestricted order. However, in order to deal with these requirements, it needs to have training examples that include all possible combinations of the attribute items.

Multi-slot extraction rules can represent the relationship between the attribute items. However, such rules will fail if the records contain missing attribute items, multi-valued attribute items, or attribute items in unrestricted order. Moreover, all the above approaches assume that the record structure is flat. STALKER [46] is a wrapper learning method that can extract content from documents with hierarchical structure. Each slot or field of items is associated with a set of extraction rules. It uses an embedded content tree to group together the individual attribute items to assemble a multi-slot record. It can extract attribute items from documents that contain complex combinations of embedded lists and attribute items. A major disadvantage is the requirement of providing a description of the record structure of the document.

Recently, some statistical based methods are proposed for information extraction tasks. Seymore et al. [49] explored the use of HMMs to learn model structure from data and attempted to make the best use of labeled and unlabeled data. Freitag and McCallum [24] demonstrated the ability of shrinkage to improve the performance of HMMs for information extraction. Chieu and Ng [9] proposed an approach using maximum entropy classifier for extracting information from semi-structured and free texts. However, extraction rules constructed by statistical methods are often difficult for users to interpret.

Some methods focus on extracting information from tables and lists in

Web documents. Lim and Ng [41] constructed the content tree of the data contents in a given HTML table. Wang et al. [53] proposed a semantic search approach capable of extracting information from general tables. Semantic ontology allows it to read tables in the same knowledge domain with different layouts. Cohen et al. [11] developed a system called  $WL^2$  which considers several different representations of the HTML document for wrapper construction. Such representations include document-object model (DOM) level, token level, and the two-dimensional geometry visual information of tabular data. However, all of these methods only deal with structured table or list layouts.

The existing approaches for wrapper induction can only handle records with simple and flat structure. Single-slot and multi-slot extraction rules pose a serious limitation on the structure representation of the records. Muslea et al. [46] attempted to tackle the problems in single-slot and multi-slot extraction rules by modeling the records with a hierarchical record structure. As mentioned before, the record structure is required to be identified by user in advance. To reduce the human effort in wrapper induction, we develop a novel approach for automatically inferring the record structure by machine learning approach. In addition, most of the previous approaches only consider the surrounding tokens in construction of extraction rules. Our approach also captures the semantic content of the attribute item itself to enrich the expressiveness of the extraction rules.

## 2.2 Related Work on Wrapper Adaptation

As the layout format of Web sites changes from time to time, a previously constructed wrapper may become obsolete sooner or later. Wrapper maintenance aims at re-learning a new wrapper when the current wrapper can no longer extract correct information.

RAPTURE [37] is a wrapper verification system. It verifies the validity of the wrapper by performing regression testing on the data extracted by the wrapper. WebCQ [42] is designed to monitor the changes of the Web documents. Both of them can only partially solve the wrapper maintenance problem.

Lerman et al. [40] tried to tackle the wrapper maintenance problem by their DataPro algorithm. However, they assume that the format of attribute items does not change over time. This requirement poses a serious limitation for their approach.

When a wrapper is found to be obsolete, a new wrapper may be re-learned using previously collected training examples. Most likely, these training examples may also become invalid. Besides, the wrappers learned from a particular information source typically cannot be applied to extract attribute items from other sources. A separate effort is required to annotate a new set of training examples in the new Web site, so as to learn a new wrapper for that source.

A possible solution for solving this problem is to address the issue of the preparation of training examples. Muslea et al. [45] proposed an active learning technique called co-testing which asks the users to label an example

that maximizes the information conveyed to the learning system. However, it can only partially reduce human effort in preparing training examples.

Brin's DIPRE [4] tackled this problem by continuously providing some concept pairs (e.g., book title/author) to the system. DIPRE searches the documents that contain the concept pairs and learns the extraction patterns. The extraction patterns are then applied to other documents to find more training examples.

Bootstrapping algorithms [27, 48] aim at reducing the number of training examples. They initiate their training with a set of *seed words* and assume that the seed words will be present in the training data. All the above systems can only partially solve the problem. A separate effort is still required for different Web sites.

IEPAD [7] generates extraction rules by finding repeated patterns in the Web page using a data structure called PAT trees and performing multiple string alignment on the discovered repeated patterns. It requires no training example for discovering the wrappers. However, the user is required to manually select the extraction rules that contain their desired information.

ROADRUNNER [13] also attempts to solve the problem by eliminating the need for training example preparation. The idea is based on the difference and the similarity of the text content of the Web pages. However, by using either ROADRUNNER or IEPAD, user cannot obtain the semantic meaning of the extracted data, and the relationship between the extracted data.

DeLa [54] is a system developed for generating wrapper without using training examples. The idea of DeLa is to find repeated patterns in the Web page and discover a regular expression for the repeated patterns. It

also assigns labels to the extracted attribute items using several heuristics. The extraction rule of DeLa is in the form of regular expression. Such regular expression is similar to multi-slot extraction rule and hence will fail to extract attribute items in unrestricted order. The assigned labels also require human interpretation.

Wrapper adaptation aims at adapting a previously learned wrapper in an information source to a new, unseen information source in the same domain. Golgher et al. [30] tried to address the wrapper adaptation problem by a query-like approach. This approach searches the exact matching of attribute items in an unseen Web page. However, exact match of attribute items in different Web sites is ineffective.

Wong et al. developed approaches for solving the wrapper adaptation problem [55, 56]. The main idea of their approach is to automatically prepare a new set of training examples in the unseen Web sites for inducing a new wrapper. They make use of the extraction knowledge in the source site to achieve this task. However, manual intervention is still required in this method.

In summary, some of the existing methods for wrapper adaptation can either only partially solve the problem, or solve the problem in a semi-automatic manner. Some of them make use of the structure of Web pages for extracting items without providing any training example. However, the semantic meaning of the extracted attribute items requires human effort for interpretation. Although Golgher et al. [30] proposed an automatic approach to wrapper adaptation, they made some assumptions on the format of the attribute items. We develop a fully automatic approach to solving the wrapper



adaptation problem. The idea of our approach is to automatically generate training examples for learning a new wrapper for the new unseen Web site. Our system can handle attribute items with ambiguous format in different Web sites.

## Chapter 3

# Automatic Construction of Hierarchical Wrappers

Our wrapper learning framework is composed of two stages. The first stage is the Hierarchical record structure inference task. The second stage is the extraction rule induction task. Once a wrapper is learned, it can be used for information extraction for the particular Web page.

The hierarchical record structure inference task is designed to infer the hierarchical record structure of the Web documents based on the positive and negative examples given by users. Users only need to provide the positive and negative examples and by annotating them by the hierarchical structure, the system can automatically infer the hierarchical record structure. The example in Figure 2.1 depicts a sample of hierarchical record structure. The hierarchical record structure problem seems to share some similarities with the record structure inference problems. However, our user-defined record structure is more

## Chapter 3

# Automatic Construction of Hierarchical Wrappers

Our wrapper learning framework is composed of two stages. The first stage is the hierarchical record structure inference task. The second stage is the extraction rule induction task. Once a wrapper is learned after these two tasks, it can be used for information extraction for the particular Web site.

The hierarchical record structure inference task attempts to infer the hierarchical record structure of the Web documents based on the attribute item samples given by users. Users only need to specify the attribute items of interest and by annotating them in the document. The inference process automatically infers the hierarchical record structure. For example, Figure 3.1 depicts a sample of hierarchical record structure which can model the records contained in the Web page as shown in Figure 1.1. The inference problem seems to share some resemblances with context-free or tree grammar inference problems. However, our hierarchical record structure allows multi-

valued attribute items and missing attribute items. Ordinary context-free or tree grammars are inadequate for representing the hierarchical record structure. Furthermore, only positive examples are available and they are likely incomplete. To cope with these characteristics, we develop a viable record structure inference algorithm. The detailed description of this algorithm is discussed in Section 3.1.

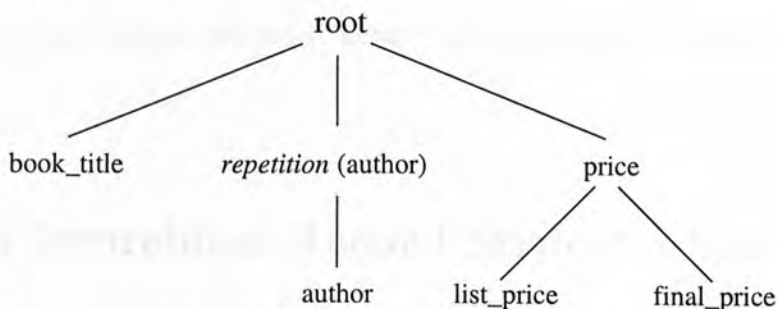


Figure 3.1: The hierarchical record structure for the book information shown in Figure 1.1

The second stage is the extraction rule induction task. A set of extraction rules are induced for each node in the hierarchical record structure. One unique characteristic of our rule induction approach is that it considers both lexical and semantic generalization in the learning process. An extraction rule mainly consists of three parts: the left pattern component, the target pattern component, and the right pattern component. The rule induction can optionally allow two kinds of knowledge: domain independent and domain specific semantic classes. Domain independent semantic classes are used to recognize common contents such as city names and country names. Domain specific knowledge is used to recognize specific content tailor-made for a particular domain. For example, in online product catalogs, there is

a specific semantic class for the attribute item “price” that stores the keywords, thesaurus, and symbols related to price like “price”, “sell”, and “buy”. The details of the extraction rule induction algorithm will be described in Section 3.2.

After this two-stage learning process completes, attribute items from other Web pages of the same Web site can be extracted by making use of the inferred hierarchical record structure and applying the learned extraction rules.

### 3.1 Hierarchical Record Structure Inference

The first stage is to infer a hierarchical record structure description for the records in a Web site. The content of the Web site is automatically downloaded and tokenized. The relationship among attribute items of interest is modeled by a hierarchical record structure. It is a tree-like structure in which the leaf nodes are single attribute items. The root node in the structure represents the whole record. An internal node in the structure represents a certain part of the content of its parent. An internal node normally comprises multiple attribute items. There is a special kind of internal node called *repetition*. The child pattern under a repetition node can be repeated zero or more times. Hence, a repetition node can model multi-valued attribute items. In principle, the hierarchical record structure can have arbitrary many levels of embedded data. There is no restriction on the order of the nodes in the structure at the same level. Besides, it allows missing attribute items in a record. An example of a hierarchical record structure is shown in Figure 3.1.

The record structure in this example contains a book title, a list of authors, and a price. The price consists of a “list price” and an “our price”. There is no restriction on the order among items under a parent node. A record can have any item missing.

In general, our hierarchical record structure shares some resemblances with context-free tree grammars [31]. Recall that an ordinary tree grammar is defined as a four-tuple  $G_t = (V, r, P, \zeta)$  where  $V = N \cup \Sigma$  is the grammar alphabet (nonterminal and terminals) and  $(V, r)$  denotes ranked alphabets. Productions in  $P$  are of the form  $T_i \rightarrow T_j$ , where  $T_i$  and  $T_j$  are trees.  $\zeta$  in  $T_v$  is a finite set of “starting trees” and  $T_v$  denotes the set of trees with nodes labeled by elements in  $V$ . However, our hierarchical record structure cannot be fully represented by this ordinary tree grammar. The first difference is that our hierarchical record structure allows missing alphabets. One could model it by enumerating all combinations of all nonterminal nodes with the rank ranging from 1 to  $r$ , where  $r$  is the rank of the original node. However, the grammar will be extremely complicated and messy. The second difference is that our hierarchical record structure allows multi-valued alphabets. In ordinary tree grammars, the rank of a node is finite and hence it cannot effectively model multi-valued items. Third, in our hierarchical record structure, the terminal alphabets can only appear once. Lastly, only positive training examples are available in our inference process. Gold [29] proved that regular language cannot be identified by only positive examples. In order to cope with these properties, we propose a record structure grammar to represent the hierarchical record structure.

Formally, our record structure grammar can be modeled by three-tuple

$G' = (V, P, \zeta)$  where  $V = N_o \cup N_r \cup \Sigma$  is the grammar alphabet.  $N_o$  and  $N_r$  are two kinds of nonterminals, namely, ordinary nonterminals and repetition nonterminals respectively. There are no common alphabets among  $N_o$ ,  $N_r$ , and  $\Sigma$  (i.e.,  $N_o \cap N_r \cap \Sigma = \emptyset$ ). Productions in  $P$  are of the form  $T_i \rightarrow^* T_j$ , where  $T_i$  and  $T_j$  are trees. For any tree  $T$ , there is no restriction on the order among the subtrees in  $T$ . Consider any production  $p$  of the form  $T_i \rightarrow^* T_j$  in  $P$ . It denotes the fact that productions of the form  $T_i \rightarrow T'_j$  exist such that  $T'_j$  is formed by removing any subtree in  $T_j$ . The set of repetition nonterminals,  $N_r$ , can model *repetition*. The subtree under a repetition nonterminal can repeat any number of times.

We have developed a hierarchical record structure inference algorithm tailored to our problem. Before presenting the inference algorithm, we introduce a notion, called “*cover*”. A tree/subtree  $T_i$  “cover”  $T_j$  if  $T_i$  and  $T_j$  follow the situation shown in Figure 3.2. In this figure,  $A$ ,  $B$ , and  $C$  denote different subtrees and “ $\Rightarrow$ ” denotes the tree in left hand side “cover” the tree in right hand side. It describes that a newly formed tree will cover the ordinary tree by moving any one of the subtree one level higher. Recall that a leaf node in our hierarchical record structure represents an attribute item and an internal node represents certain content of its parent. In order to extract the content of an internal node or a leaf node, the content of its parent is required to be identified in advance. Therefore, we can deduce that the most general record structure is the one without any internal node. Although this kind of record structure gives the least hierarchical information of the record, it can handle most variations in the record structure in practice. The idea of “cover” is to relax some restrictions on the hierarchical record structure in order to han-

de more variations of the records although some of the information of the internal node may be lost.

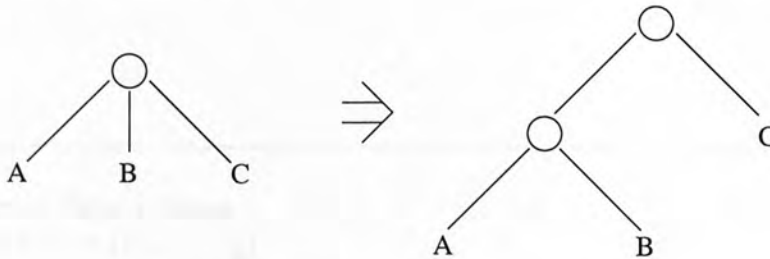


Figure 3.2: A situation for “cover”

---

```

# Function Hierarchical Structure Inference
1 Construct the raw record structure from examples
2 foreach label  $v$  of the nodes
3   if the nodes labeled with  $v$  have parents which have different
      labels  $(v'_1, \dots, v'_k)$  where  $k > 1$ 
4     Reduce_Nodes
5   end if
6   if the proportion of the number of training example that
      have more than one node labeled with  $v$  to the number of
      total training examples is greater than a threshold
7     Create a repetition node as the parent of this node
8   end foreach

```

---

Figure 3.3: The outline of the hierarchical record structure inference algorithm

Figure 3.3 shows the outline of our hierarchical record structure inference algorithm. Figure 3.4 shows the function *Reduce\_Nodes* used by the hierarchical record structure inference algorithm. We illustrate our algorithm through a simple example. Suppose there are fifteen training examples as shown in

---

```

# Function Reduce Nodes
1  foreach  $v'_i$  in  $(v'_1, \dots, v'_k)$ 
2     $n_i$  = number of training examples that have a node labeled with  $v'_i$ 
3     $f_i$  = number of training examples that have a node labeled with  $v$ , which is
        also a child of the node labeled with  $v'_i$ 
4    Estimate the probability  $Prob(v'_i)$  that a node labeled with  $v$  is a child of the node
        labeled with  $v'_i$  by calculating  $f_i/n_i$ 
5  end foreach
6   $p_j$  = the highest probability among  $p_1, \dots, p_k$ 
7  if  $p_j$  is significantly higher than all the others
8    Move all the subtrees of the nodes labeled with  $v$  to the node
        labeled with  $v$  which is also a children of the node labeled with  $v'_j$ 
9    Delete all the subtrees whose root is labeled with  $v$  and are not
        children of the nodes labeled with  $v'_j$ 
10 else if  $p_j$  is not significantly higher than the probabilities of  $N - 1$  nodes
11   Form a new node  $n'$  under the common ancestor of these  $N$  nodes
12   Move all the subtree whose root is labeled with  $v$  and are children of
        these  $N$  nodes
13   Delete all the subtrees whose root is labeled with  $v$  except
        the newly created one
14 end if

```

---

Figure 3.4: The outline of the function *Reduce\_Nodes* used by the hierarchical record structure inference algorithm



Figure 3.5 and five training examples as shown in Figure 3.6. The algorithm first will create the *raw record structure* as shown in Figure 3.7. The number within the square brackets in Figure 3.7 shows the number of training examples containing the node. In this raw record structure, two leaf nodes are labeled with  $D$  and have different parents. We refer the node  $D$  under the node  $B$  as  $D_b$  and the node  $D$  under the node  $C$  as  $D_c$ . The probability that  $D$  will be a child of  $B$  is estimated by  $15/20 = 0.75$  and the probability that  $D$  will be a child of  $C$  is estimated by  $5/20 = 0.25$ . The algorithm decides if the probability that the node  $D_b$  will appear is significantly higher than the probability that the node  $D_c$  will appear. If the probability that the node  $D_b$  will appear is significantly higher than the probability that the node  $D_c$  will appear, the subtree of  $D_c$  will be moved to the position under  $D_b$  and  $D_c$  will be removed. The resulting hierarchical record structure will be the same as the one shown in Figure 3.5. If the probability that the node  $D_b$  will appear is not significantly higher than the probability that the node  $D_c$  will appear, a new node will be formed under their common ancestor (*root*). Their subtrees will be placed under the newly formed node. Both  $D_b$  and  $D_c$  are then removed. The resulting hierarchical record structure will be same as the one shown in Figure 3.8.

Hoeffding proved that the confidence range of Bernoulli variable with probability  $p$  and observed frequency  $f$  out of  $n$  tries is given by Hoeffding bound [34]:

$$\left| p - \frac{f}{n} \right| < \sqrt{\frac{1}{2n} \log \frac{2}{t}} \text{ with probability larger than } (1 - t). \quad (3.1)$$

Based on the Hoeffding bound, the probability that a node labeled with  $v$

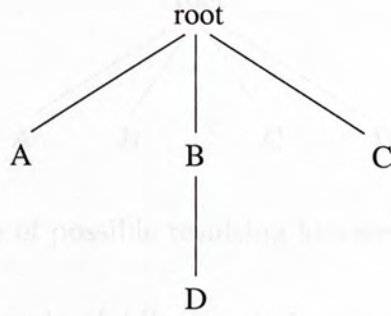


Figure 3.5: A sample of training example for the record structure inference algorithm

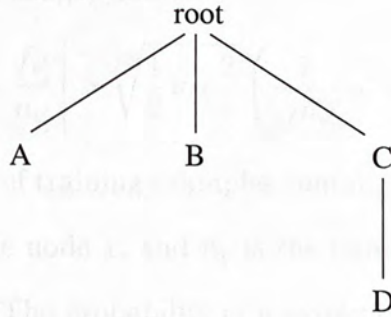


Figure 3.6: An other sample of training example for the record structure inference algorithm

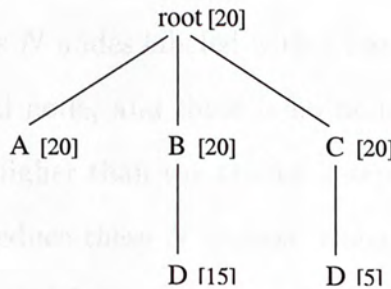


Figure 3.7: The raw record structure constructed by the record structure inference algorithm from the training examples as shown in Figures 3.5 and 3.6

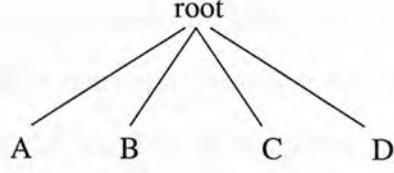


Figure 3.8: A sample of possible resulting hierarchical record structure

will be a child of the node  $v'_i$  ( $Prob(v'_i)$ ) is significantly higher than the probability that the node labeled with  $v$  will be a children of a node  $v'_j$  ( $Prob(v'_j)$ ) if the difference of the estimated probabilities is larger than the sum of their confidence range, that is:

$$\left| \frac{f_{v'_i}}{n_{v'_i}} - \frac{f_{v'_j}}{n_{v'_j}} \right| > \sqrt{\frac{1}{2} \log \frac{2}{t}} \left( \frac{1}{\sqrt{n_{v'_i}}} + \frac{1}{\sqrt{n_{v'_j}}} \right) \quad (3.2)$$

where  $f_x$  is the number of training examples containing the node labeled with  $v$  which is a child of the node  $x$ , and  $n_y$  is the number of training examples containing the node  $y$ . The probability of a wrong rejection is kept below  $2t$ . Hence, the higher the value of  $t$ , the more likely that accepting the hypothesis that  $Prob(v'_i) > Prob(v'_j)$ . Refer to the above mentioned example, if  $t$  is set to 0.2, we will accept that  $Prob(D_b) > Prob(D_c)$ .

Suppose there exists  $N$  nodes labeled with  $v$  have  $Prob(v_x)$  where  $x$  is  $N$  distinct labeled internal node, and there is no node in these  $N$  nodes with  $Prob(v_x)$  significantly higher than the others. (step 10 of the *Reduce\_Nodes* function), we have to reduce these  $N$  highest probability nodes into a single node. In our algorithm, we form a new node under the common ancestor of these  $N$  highest probability nodes. All the subtrees of these nodes will be placed under the newly created node. These  $N$  nodes are then removed. The idea of this approach is based on the idea of “cover” mentioned above. Placing

a node in a higher level of the hierarchy will cover more general sequence and the resulting hierarchical record structure will be more general.

If the proportion of the number of examples containing more than one nodes labeled with  $v$  to the total number training examples is greater than a threshold, such node labeled with  $v$  will be placed under a repetition node.

Figure 3.9 depicts four different training examples from the Web site as shown in Figure 1.1 for the hierarchical record structure inference algorithm. Some of the training examples consist of a “final price” under the “price” node, while some of them consist of a “final price” under the “root” node. Some of them contain “list price”. Some of them contain more than one “author”. Our record structure inference algorithm infers the hierarchical record structure from these training examples. The resulting hierarchical record structure for the records in the Web site as shown in Figure 1.1 is shown in Figure 3.1.

## 3.2 Extraction Rule Induction

After a hierarchical record structure is inferred in the first stage, the next stage is extraction rule induction. Recall that each node in the record structure corresponds to a certain part of the record. A set of extraction rules are learned for each node in the structure. The nodes in the structure are processed in a depth-first order. The set of rules are responsible for extracting that particular item from its parent field. If a node is a repetition, the set of extraction rules are applied to extract a field that may contain multiple items. After that, the extraction rules of the child node will be applied it-

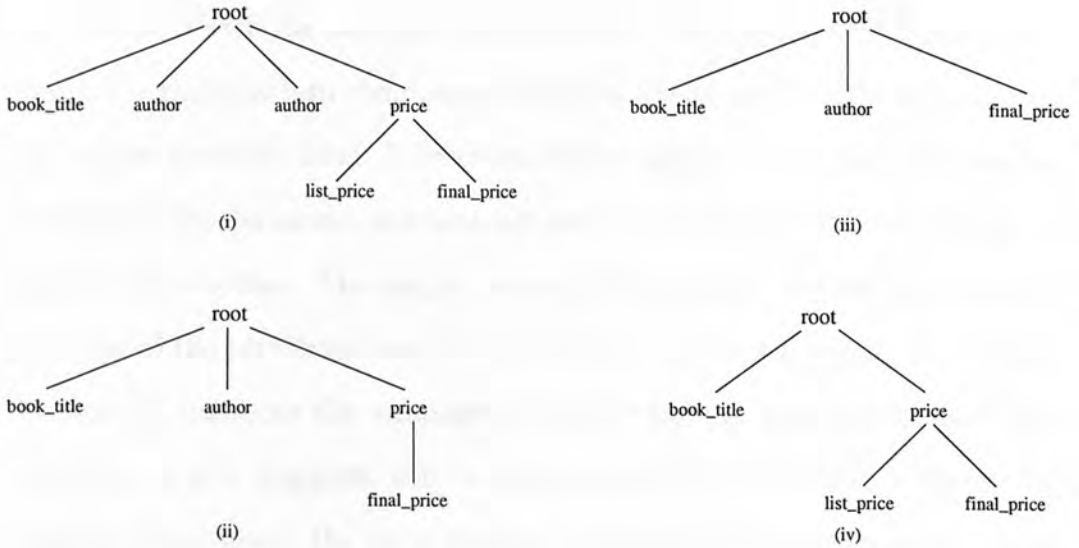


Figure 3.9: Different training examples from the Web site as shown in Figure 1.1 for the hierarchical record structure inference

eratively to extract multiple individual attribute items. Refer to Figure 3.1, there is a set of extraction rules associated with the “root” which identifies each record inside the documents. There is also a set of extraction rules associated with the node “repetition(author)” which is applied to the “root” node to identify the text fragment regarding the whole list of authors from the record. The set of extraction rules associated with “author” will be applied to the node “repetition(author)” iteratively to identify the individual author attribute items.

An extraction rule consists of three distinct components: the left pattern component, the target pattern component, and the right pattern component. The left pattern component describes a sequence of left delimiters of the target attribute item. Each delimiter can be a raw token or a semantic class. It instructs the wrapper to scan and consume the content of the document

and matches with the sequence of delimiters in the pattern description. Similarly, the right pattern component describes a sequence of right delimiters of the target attribute item. It instructs the wrapper to scan and consume the content of the document and matches with the sequence of delimiters in the pattern description. The target pattern component describes the semantic concept of the attribute item. It consists of a list of raw tokens or semantic classes. It instructs the wrapper to test if the text fragment contains the pattern. A text fragment will be extracted only if it matches with the left pattern component, the right pattern component, and contains the target component.

A raw token can be a lexical term like HTML tag, or a lexical string in the free texts. A semantic class represents a more general concept. We organize lexical terms and semantic classes in a hierarchical manner. Each token or semantic class can be generalized to another semantic class. The semantic classes can be either domain independent or domain specific. Consider the following fragment of a HTML document:

Now Sell : \$ < B > 265.95 < /B >< BR >

The semantic classes of the tokens is shown in Figure 3.10. Domain independent semantic classes include *TEXT*, *DIGIT*, *FLOAT*, *PUNCT*, *HTML\_TAG*, *HTML\_LAYOUT*, *HTML\_FONT*, and *HTML\_PARAGRAPH*. *CURRENCY* is a domain specific semantic class representing currency literals like “\$”.

To extract the price attribute item from the example, our rule induction algorithm will learn the following extraction rule:

Left pattern component: (*<CURRENCY>*, *semantic\_class*),

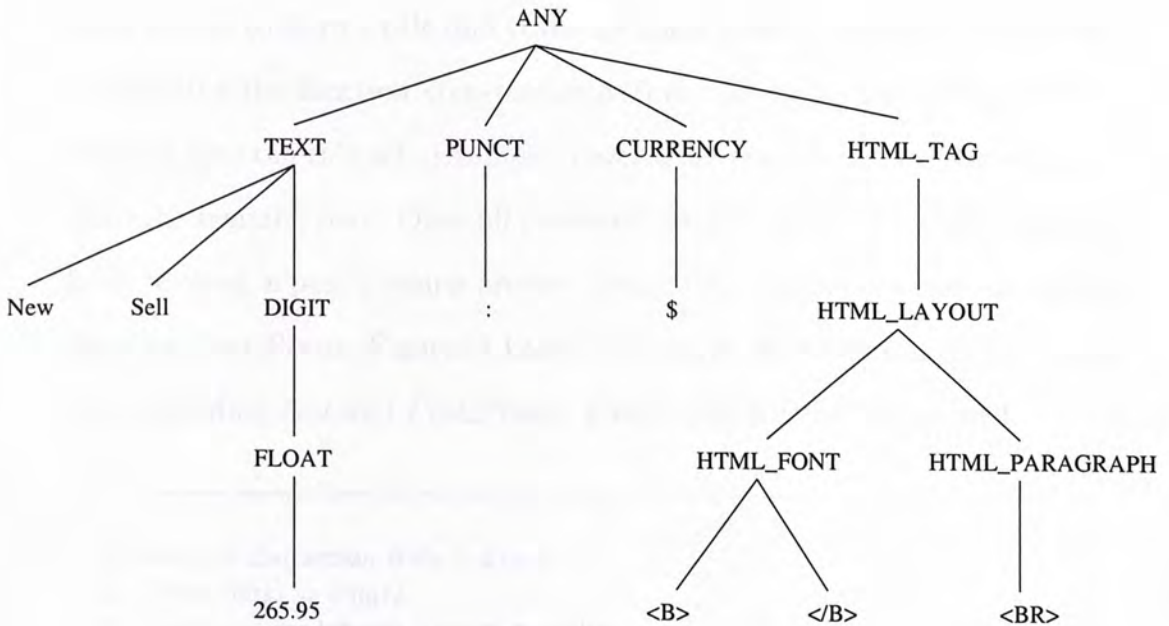


Figure 3.10: Examples of semantic classes organized in a hierarchy

$(\text{"<B>"}, \text{token})$ .

Target pattern component:  $(\text{<FLOAT>}, \text{semantic\_class})$ .

Right pattern component:  $(\text{"</B>"}, \text{token})$ ,

$(\text{"<BR>"}, \text{token})$ .

where  $(X, \text{token})$  represents that  $X$  is a token string.  $(Y, \text{semantic\_class})$  represents that  $Y$  is a semantic class.

Our rule induction algorithm discovers extraction rules based on training examples associated with a node in the hierarchical record structure. The training examples are text fragments corresponding to the content represented by the node in the record structure. Figure 3.11 presents the outline of the extraction rule induction algorithm. It is a sequential covering algorithm. When there are some uncovered positive examples in the training

pool, it tries to learn a rule that covers as many positive examples as possible by invoking the function *Generate\_And\_Test*. The best rule learned will be inserted into the rule set. Examples covered by the rule set will be removed from the training pool. Once all positive examples in the training pool have been covered, a post-pruning process is required to refine the rule set by the function *Post\_Prune*. Figures 3.12 and 3.13 depict the outline of the functions *Generate\_And\_Test* and *Post\_Prune*. Finally, the rule set is returned.

---

```

# Function Extraction Rule Learning
1  {Rule_Sets} = empty.
2  {Inst} = user-labeled training samples
3  Struct = Hierarchical Record Structure
4  foreach node n in Struct
5      {Inst_n} = training instances for node n in {Inst}
6      Rule_Set_n = rule set of node n in Struct = empty
7      while {Inst_n} is not empty
8          Rule = Generate_And_Test({Inst_n}
9          {Rule_Set_n} = {Rule_Set_n} ∪ Rule
10         Remove_Covered_Instances({Inst_n})
11     Post_Prune({Rule_Set_n}
12     add Rule_Set_n to {Rule_Sets}
13 return {Rule_Sets}

```

---

Figure 3.11: The outline of the extraction rule induction algorithm

In each iteration, our algorithm tries to select the shortest example that is not covered by the current rule set as the *seed*. Then a left pattern will be generated by scanning  $w$  tokens before the seed. After that, we generate all the combinations of the token strings, as well as their semantic classes. Each



---

```

# Function Generate_And_Test({Inst_n})
1 seed = the shortest instance in {Inst_n}
2 {left_tokens} = w tokens before the seed
3 {Left_Candidates} = all combinations of the token strings and
                       their semantic classes in {left_tokens}
4 foreach candidate in {Left_Candidates}
5     Score_Left_Candidate(candidate, {Inst_n})
6 {Best_Left} = Find_Best_N({Left_Candidates})
7 {right_tokens} = w tokens after the seed
8 {Right_Candidates} = all combinations of the token strings and
                       their semantic classes in {right_tokens}
9 foreach candidate in {Right_Candidates}
10    Score_Right_Candidate(candidate, {Inst_n})
11 {Best_Right} = Find_Best_N({Right_Candidates})
12 {Candidate_Rules} = {Best_Left} × {Best_Right}
13 foreach candidate_rule in {Candidate_Rules}
14    Score_Rule(candidate_rule, {Inst_n})
15 best_rule = the best candidate_rule in {Candidate_Rules}
16 Find_Common_Pattern(best_rule, {Inst_n})
17 return the best_rule

```

---

Figure 3.12: The *Generate\_And\_Test* function used by the extraction rule induction algorithm

---

```

# Function Post_Prune({Rule_Set})
1 foreach rule in {Rule_Set}
2     if precision of rule is less than a pre-defined threshold  $\alpha$ 
3         remove rule from {Rule_Set}
4 sort the rules in descending of precision

```

---

Figure 3.13: The *Post\_Prune* function used by the extraction rule induction algorithm

combination acts as a candidate for the left pattern component of the rule. These candidates will be scored by the function *Score\_Left\_Candidate*. Next,  $N_1$  best left pattern candidates will be selected by the function *Find\_Best\_N*. Then,  $N_2$  best right pattern candidates will be selected in a similar manner. The *Candidate\_Rules* will be generated by forming combinations of the  $N_1$  best left pattern candidates and  $N_2$  best right pattern candidates. The best rule will be obtained after each candidate rule in the *Candidate\_Rules* is tested. After that, the patterns in the content of training examples covered by the best rule will be found by the function *Find\_Common\_Pattern*. These patterns will become the target pattern component of the best rule. Finally, the best rule will be returned by the function and the rule will be added into the rule set.

In the function *Post\_Prune*, each rule in the rule set will apply to the training examples individually. Their precisions will be calculated. If the precision is less than a pre-defined threshold  $\alpha$ , the rule will be removed from the rule set. The rules in the rule set will be sorted in descending order of their precisions.

Figures 3.14 and 3.15 describe some functions used by the extraction rule induction algorithm. In the *Score\_Left\_Candidate* function, the left pattern component candidate tries to “tag” the start positions of the attribute item. In the *Score\_Right\_Candidate* function, the right pattern component candidate tries to “tag” the end position of the attribute item. In the *Score\_Rule* function, the candidate rule tries to “tag” the start and end positions of the attribute item. The number of correct and wrong “tag” will be calculated. The score is computed by dividing the number of wrong “tag” by the

---

```

# Function Score_Left_Candidate(left_pattern, inst_n)
1 apply left_pattern in {inst_n} to "tag" the start position
2 count the number of correct, and wrong "tag"
3 return score = wrong/correct

# Function Score_Right_Candidate(right_pattern, inst_n)
1 apply right_pattern in {inst_n} to "tag" the end position
2 count the number of correct, and wrong "tag"
3 return score = wrong/correct

# Function Score_Rule(candidate_rule, inst_n)
1 apply candidate_rule in {inst_n} to "tag" the start and end positions
2 count the number of correct, and wrong "tag"
3 return score = wrong/correct

```

---

Figure 3.14: The score functions used by the rule induction algorithm

---

```

# Function Find_Best_N(candidates)
1 return the N best candidates with
  1. lowest value of score
  2. highest number of correct tag
  3. most specific patterns

# Function Find_Common_Pattern(best_rule, inst_n)
1 return the semantic classes which are:
  1. contained by most of the training examples covered by the best_rule
  2. specific semantic classes

```

---

Figure 3.15: The *Find\_Best\_N* function, and the *Find\_Common\_Pattern* function used by the rule induction algorithm

number of correct “tag”. The smaller the *score*, the better the candidate is. In the *Find\_Best\_N* function,  $N$  best candidates are returned. A candidate will be preferred if: 1) It has a smaller value of score; 2) It has a higher number of correct “tag”; 3) It is a more specific pattern component or rule. A pattern is said to be more specific than other patterns if 1) it consists of more token strings, or 2) the semantic classes are more specific. A semantic class is said to be more specific if the semantic class is located at a lower level of the semantic class hierarchy. For example, as shown in Figure 3.10, the semantic class “HTML\_FONT” is more specific than the semantic class “HTML\_LAYOUT”. In the *Find\_Common\_Pattern* function, the training examples covered by the best rule will be tested if they contain some common patterns. We consider patterns such as *DIGIT*, *FLOAT*, or some domain specific semantic classes like *CURRENCY*. If the proportion that the training examples containing one or more common pattern exceeds a threshold  $\beta$ , these common patterns will be returned. Figure 3.16 depicts one of the extraction rule learned by our extraction rule induction algorithm for the final price of the Web page as shown in Figure 1.1.

### 3.3 Applying Hierarchical Wrappers

In order to extract the attribute items of interest from a Web page or a document, our wrapper makes use of the inferred hierarchical record structure and the sets of learned extraction rules. The document is first broken down into a sequence of tokens. Based on the inferred hierarchical record structure, the induced extraction rules for the root node in the structure are applied to

Left pattern component: (*“Our”, token*),  
                                   (*“Price”, token*),  
                                   (*“.”, token*),  
                                   (*<HTML\_IMG\_TAG>, semantic\_class*).

Target pattern component: (*<FLOAT>, semantic\_class*).

Right pattern component: (*“&nbsp;”, token*),  
                                   (*“&nbsp;”, token*),  
                                   (*“</b>”, token*),  
                                   (*<HTML\_FONT\_TAG>, semantic\_class*).

Figure 3.16: A sample of extraction rule for the final price of the Web page shown in Figure 1.1

the sequence of tokens to obtain each record. Then the content corresponding to each internal node is extracted. If the node is a repetition node, the extraction rules of the repetition node will be applied to extract a text fragment. After that, the extraction rules of the child node will be applied to this fragment to extract multiple individual attribute items. Each attribute item is extracted independently among its siblings in the structure. Finally, we group together the individual attribute items to assemble multi-slot records.

## Chapter 4

# Experimental Results for Wrapper Induction

In order to demonstrate the effectiveness of our wrapper induction approach, we have conducted extensive experiments to extract items from a variety of real-world Web sites containing online book catalogs and consumer electronic appliance catalogs. Table 4.1 depicts the Web site name and the URL address used in our experiment. T1 to T3, S1 to S11 are book catalogs. The attribute items of interest are title, author, and price of the books. T1 to T3 are used for parameter tuning. S12 to S19 are consumer electronic appliance catalogs. The attribute items of interest are the model number, description, and price of the products. For each Web site, we have manually collected a number of Web pages and all records from each page for evaluation purpose. The total number of Web pages and the total number of records collected are also depicted in Table 4.1. Some of these Web sites contain records with hierarchical structure. Most of them contain records with missing attribute

items, multi-valued attribute items or attribute items in unrestricted order.

For each Web site, we randomly selected one Web page and used the annotated records in this page for training examples. The remaining records in other Web pages of the Web site are reserved for evaluating the extraction performance. Table 4.2 depicts the number of user annotated training records for learning a wrapper for each Web site. Table 4.3 shows a sample for training record for the Web site S8. Based on the user annotated training examples, a wrapper is induced for each Web site. In order to measure the extraction performance, the answers extracted by the system will be compared with the correct answers. We use two metrics, namely, precision and recall, which are widely used in information retrieval tasks, to evaluate the extraction performance. Their definitions are as follows:

**Definition 4.1** *Precision* is defined as the number of attribute instances for which the system correctly identifies divided by the total number of attribute instances it extracts.

**Definition 4.2** *Recall* is defined as the number of attribute instances for which the system correctly identifies divided by the total number of actual attribute instances.

In the parameter tuning process described below, we also make use of an evaluation metric called F-measure [51] which is defined as follows:

$$\text{F-measure} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \quad (4.1)$$

In our wrapper induction approach, three parameters are needed to be determined in advance. The first parameter is the window size  $w$  in the

	Web site (URL)	Number of pages	Number of records
T1	1Bookstreet.com ( <a href="http://www.1bookstreet.com">http://www.1bookstreet.com</a> )	5	125
T2	DigitalGuru Technical Bookshops ( <a href="http://www.digitalguru.com">http://www.digitalguru.com</a> )	17	102
T3	Jim's Computer Books ( <a href="http://www.vstore.com/cgi-bin/pagegen/vstorecomputers/jimsbooks/">http://www.vstore.com/cgi-bin/pagegen/vstorecomputers/jimsbooks/</a> )	7	139
S1	Amazon.com ( <a href="http://www.amazon.com">http://www.amazon.com</a> )	5	125
S2	Barnes & Noble.com ( <a href="http://www.barnesandnoble.com">http://www.barnesandnoble.com</a> )	5	120
S3	BookCloseouts.com ( <a href="http://www.bookcloseouts.com">http://www.bookcloseouts.com</a> )	3	112
S4	Powell's Books ( <a href="http://www.powells.com">http://www.powells.com</a> )	5	100
S5	WordsWorth Books ( <a href="http://www.wordsworth.com">http://www.wordsworth.com</a> )	10	125
S6	bookpool.com ( <a href="http://www.bookpool.com">http://www.bookpool.com</a> )	5	124
S7	half.com ( <a href="http://half.ebay.com">http://half.ebay.com</a> )	6	120
S8	Half Price Computer Books ( <a href="http://www.halfpricecomputerbooks.com">http://www.halfpricecomputerbooks.com</a> )	5	100
S9	Discount-PCBooks.com ( <a href="http://www.discount-pcbooks.com">http://www.discount-pcbooks.com</a> )	14	110
S10	mmistore.com ( <a href="http://www.mmistore.com">http://www.mmistore.com</a> )	11	110
S11	eCampus.com ( <a href="http://www.ecampus.com">http://www.ecampus.com</a> )	11	110
S12	1-888Camcorder.com ( <a href="http://store.yahoo.com/1888camcorder">http://store.yahoo.com/1888camcorder</a> )	10	100
S13	American eSuperstore.com ( <a href="http://store.yahoo.com/americanesuperstore">http://store.yahoo.com/americanesuperstore</a> )	10	100
S14	220Appliances.com ( <a href="http://www.220appliances.com">http://www.220appliances.com</a> )	8	113
S15	Circuit City ( <a href="http://www.circuitcity.com">http://www.circuitcity.com</a> )	6	120
S16	Etronics.com ( <a href="http://www.etrronics.com">http://www.etrronics.com</a> )	12	107
S17	DVD Overseas Electronics ( <a href="http://www.dvdoverseas.com">http://www.dvdoverseas.com</a> )	13	110
S18	Cambridge SoundWorks ( <a href="http://www.hifi.com">http://www.hifi.com</a> )	12	157
S19	BestBuy.com ( <a href="http://www.bestbuy.com">http://www.bestbuy.com</a> )	4	123

Table 4.1: Information sources for experiments



Web site label	Number of user annotated training records	Web site label	Number of user annotated training records	Web site label	Number of user annotated training records
T1	10	S1	10	S12	10
T2	6	S2	10	S13	10
T3	10	S3	10	S14	10
		S4	10	S15	10
		S5	10	S16	8
		S6	10	S17	10
		S7	10	S18	10
		S8	10	S19	10
		S9	8		
		S10	10		
		S11	10		

Table 4.2: The number of user annotated training records for learning wrapper for each Web site

Attribute item	Field value
<b>Book Title:</b>	Foundations of Visual C++ Programming for Windows 95
<b>Author:</b>	Paul Yao
<b>Author:</b>	Joseph Yao
<b>Final Price:</b>	19.99

Table 4.3: A sample of a user annotated training example for the Web page shown in Figure 1.1

*Generate\_And\_Test* function described in Section 3.2 in Chapter 3. The second parameter is the threshold  $\alpha$  used in the *Post\_Prune* function described in Section 3.2 in Chapter 3. The third parameter is the threshold  $\beta$  used in the *Find\_Common\_Pattern* function described in Section 3.2 in Chapter 3. In order to determine the values of the parameters, we randomly chose three Web sites, labeled as T1, T2, and T3, for tuning the parameters. We exhaustively conducted experiments for these three Web sites with different parameter values. A wrapper is automatically generated for each of the Web sites using the user annotated training examples. The wrapper is then used to extract records from other Web pages in the same Web sites. The average of the F-measure, which is defined as the combination of precision and recall, is used for the evaluation of the parameter settings. We selected the parameter setting that achieves the highest performance. Then this set of parameters will be used in all the remaining testing sites in our experiments. The parameters selected were  $w = 4$ ,  $\alpha = 0.1$ , and  $\beta = 0.8$ .

Table 4.4 shows the performance of our wrapper induction approach on the Web sites containing book catalogs. It depicts the precision (P) and recall (R) of each attribute item of interest of all records in a particular Web site. The result illustrates that our wrapper induction approach is very effective. The overall average of both precision and recall are over 94%. Detailed performance of each page in each Web site is depicted in Appendix A. Some of the Web sites such as S3, S7, S8, and S9 contain records with relatively simple record structure. Our wrapper induction approach can handle these Web site very effectively. The precision and recall for extracting attribute items from these Web sites are close to, or even reach 100%. Some of the Web

	Web site	<i>Title</i>		<i>Author</i>		<i>Price</i>	
		P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
S1	Amazon.com	97.0	97.0	97.0	98.0	97.0	98.0
S2	Barnes & Nobel.com	90.6	100.0	100.0	66.7	97.9	100.0
S3	BookCloseouts.com	100.0	100.0	100.0	100.0	100.0	100.0
S4	Powell's Books	99.0	99.0	80.2	97.0	100.0	100.0
S5	WordsWorth Books	100.0	100.0	97.8	100.0	75.6	75.6
S6	bookpool.com	100.0	99.0	100.0	99.0	100.0	99.0
S7	half.com	100.0	100.0	94.0	100.0	97.0	97.0
S8	Half Price Computer Books	100.0	100.0	100.0	100.0	91.3	91.3
S9	Discount-PCBooks.com	100.0	100.0	100.0	100.0	100.0	100.0
S10	mmistore.com	100.0	100.0	100.0	100.0	100.0	100.0
S11	eCampus.com	96.0	96.0	100.0	98.0	100.0	100.0
	Average	97.0	99.2	95.8	96.2	94.8	95.0

Table 4.4: Performance summary of our wrapper induction approach on the Web sites containing book catalogs

sites, for example S1, S2, and S10, contain book records with one or more authors. A sample of the hierarchical record structure inferred by the system for these Web sites is shown in Figure 4.2. Our wrapper induction can still handle these Web sites effectively. The precision of our wrapper induction approach on the extraction of attribute items are over 90% in most of the cases.

S5 shows a less satisfactory performance on the attribute item price. Figure 4.1 shows one of the testing HTML pages of S5. The price of the first record is “44.95”. The following shows the corresponding excerpt of the HTML text:

```
... <br>Our Price: $44.95 <br><i><A HREF="/searches/
isbnsearch.asp?isbn=0132870797&sessionID=ww74255237011">Read ...
```

The price of the third record is “35.96”. The following shows the correspond-

shipped immediately

---

**Author Events**  
We have them!

---

**The Writer's Desk**  
What authors are saying about themselves and their books

---

**Weekly Contest**  
Try your hand at our weekly first lines contest.

---

**The Independent Bestseller List**  
Our bestseller list

---

**Customer Service Desk**  
Do you discount, how do you ship, is this secure, and everything else.

---

**WorldWide Shipping**  
you name it...

---

**Send us feedback**

---

**Contact WordsWorth**

---

**Cgi Programming With Java --**  
Cornell, Gary / Paperback / 12/1/02  
Our Price: \$44.95  
[\*Read more about this title...\*](#)

**Foundations of Programming Languages : Design and Implementation --**  
Rocsta, Seyed / Paperback / 12/1/02  
Our Price: \$71.95  
[\*Read more about this title...\*](#)

**Tuomas J. Lukka's Object-Oriented Programming in Perl --**  
Lukka, Tuomas J./Lukka, T. J. / Paperback / 12/1/02  
Our Price: \$35.96 ~ You Save: \$4.00 (10.00%)  
[\*Read more about this title...\*](#)

**Client/Server Programming With Javabeans --**  
Orfali, Robert/Harkey, Dan / Paperback / 11/1/02  
Our Price: \$54.99  
[\*Read more about this title...\*](#)

**Inferno Programming With Limbo --**  
Stanley-Marbell, Philip / Paperback / 10/1/02  
Our Price: \$59.99  
[\*Read more about this title...\*](#)

**Visual Basic .Net Internet Programming --**  
Franklin, Carl / Paperback / 10/1/02  
Our Price: \$45.00 ~ You Save: \$5.00 (10.00%)

Figure 4.1: A sample of a testing Web page of S5

ing excerpt of the HTML text:

... <br>Our Price: \$35.96 ~ <NOBR><font color=#990033> You ...

In our experiment, we only randomly used few user annotated records as training examples for inducing the wrapper. The training set we used for S5 only contained training examples having the format similar to the first record in Figure 4.1. The following shows the extraction rule induced for the attribute item price of S5:

Left pattern component: (*Our*, *token*),

(*"Price"*, *token*),

(*":"*, *token*),

(*"\$"*, *token*).

Target pattern component: (<*FLOAT*>, *semantic\_class*).

Right pattern component: (<*br*>, *token*),

(<*i*>, *token*).

(<*HTML\_LINK*>, *semantic\_class*),

(<*Read*>, *token*).

Therefore, it cannot correctly extract the attribute item price if the records are formatted as the format of the third record in Figure 4.1. However, we believe that our wrapper induction approach can achieve a more satisfactory performance by providing more training examples to the system.

Table 4.5 shows the performance of our wrapper induction approach on the Web sites containing consumer electronic appliance catalogs. It depicts the precision (P) and recall (R) of each attribute item of interest of all records in a particular Web site. The result illustrates that our wrapper induction

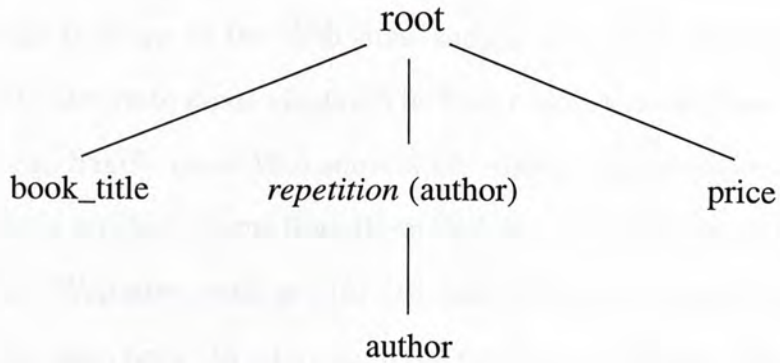


Figure 4.2: The hierarchical record structure for the book information in the Web sites S1, S2, and S10

	Web site	<i>Model number</i>		<i>Description</i>		<i>Price</i>	
		P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
S12	1-888Camcorder.com	100.0	100.0	100.0	100.0	100.0	100.0
S13	American eSuperstore.com	100.0	97.8	93.3	93.3	100.0	100.0
S14	220Appliances.com	100.0	100.0	100.0	100.0	100.0	100.0
S15	Circuit City	96.0	95.0	100.0	99.0	100.0	98.4
S16	Etronics.com	99.0	100.0	90.0	97.8	98.0	99.0
S17	DVD Overseas Electronics	100.0	100.0	96.0	100.0	100.0	100.0
S18	Cambridge SoundWorks	100.0	87.1	52.9	86.3	100.0	87.1
S19	BestBuy.com	100.0	97.9	82.0	52.1	100.0	96.5
	Average	99.4	97.2	89.3	91.1	99.8	97.6

Table 4.5: Performance summary of our wrapper induction approach on the Web sites containing consumer electronic appliance

approach is very effective. The overall average of both precision and recall are over 89%. Detailed performance of each page in each Web site is depicted in Appendix B. Some of the Web sites, such as S12, S13, and S14, contain records with attribute items displayed in fixed order. Our wrapper induction approach can handle these Web sources effectively. The precision and recall for extracting attribute items from these Web sites are close to, or even reach 100%. Some Web sites, such as S15, S16, and S19 contain records with missing attribute item price. In addition to the price attribute item, some of these Web sites contain similar information like the “list price” or “save” as well. Still, our wrapper induction approach performs very well on extracting precise contents from these Web sources with extraction performance exceeding 90% in most of the cases.

The extraction performance for the attribute item description in S18 and S19 is less satisfactory. The reason for the extraction performance for S19 is similar to that for S5 in the book domain. Some records in the testing set of the experiment have a different format from the records in the training set. Figure 4.3 shows a testing HTML page of S18 in our experiment. The correct description for the first record in Figure 4.3 is:

“The perfect blend of high-performance video and audio combined with 6-disc convenience in one surprisingly slim package.”

However, the induced wrapper incorrectly extracts the following text fragment as description:

“NEW! The perfect blend of high-performance video and audio combined with 6-disc convenience in one surprisingly slim package.”

Similar extractions also occur in extracting the description of the third and

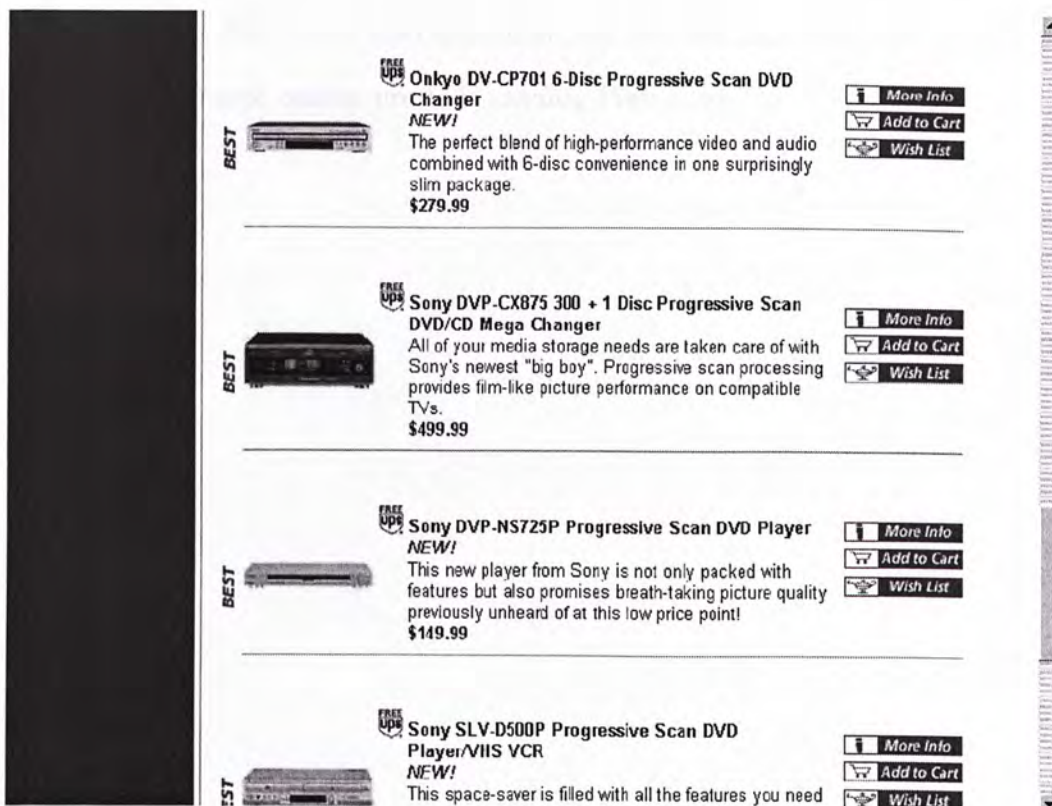


Figure 4.3: A sample of a testing Web page of S18

forth records in Figure 4.3. The reason for such incorrect extraction is that some of the training examples in the training set have the format similar to the second record in Figure 4.3. The left pattern component of the extraction rule mis-locates the left boundary of the description and includes additional tokens in the extracted data. However, the extracted data still contains the content of the description.

The encouraging result of our wrapper induction shows that our wrapper induction approach is capable of extracting content from Web pages with a flat or hierarchical structure. It can handle records which have missing attribute items, multi-valued attribute items, and attribute items in unre-



stricted order. The result also demonstrates that our approach can handle a range of real-world online product catalog Web sites.

## Chapter 5

# Adaptation of Wrappers for Unseen Web Sites

### 5.1 Problem Definition

Most of the wrapper induction techniques are able to automatically learn a wrapper from a few user annotated training examples. The learned wrapper is able to effectively extract precise information from arbitrary Web pages located in the same Web site where the user annotated training examples have been provided. However, the learned wrapper cannot be applied to extract information from other new unseen Web sites even in the same domain. A separate effort is required to prepare a new set of user annotated training examples in order to learn a new wrapper for the new Web site.

For instance, Figure 5.1 shows another Web page containing a book catalog. Figure 5.2 depicts the excerpt of the HTML text of a new Web page.

---

<sup>1</sup>The URL associated with the Web page shown in Figure 5.1 is <http://www.amazon.com>.

# Chapter 5

## Adaptation of Wrappers for Unseen Web Sites

### 5.1 Problem Definition

Most of the wrapper induction techniques are able to automatically learn a wrapper from a few user annotated training examples. The learned wrapper is able to effectively extract precise information from different Web pages located in the same Web site where the user annotated training examples have been provided. However, the learned wrapper cannot be applied to extract information from other new unseen Web sites even in the same domain. A separate effort is required to prepare a new set of user annotated training examples, in order to learn a new wrapper for the new Web site.

For instance, Figure 5.1 shows another Web page containing a book catalog<sup>1</sup>. Figure 5.2 depicts the excerpt of the HTML text document associated

---

<sup>1</sup>The URL associated with the Web page shown in Figure 5.1 is *www.half.com*.

The screenshot shows a list of six books for sale, each with a small thumbnail image of the book cover to its left. The books are separated by horizontal lines. The first book is 'C how to Program: Introducing C++ and Java 3RD BK&CDR (with CD-ROM)' by Harvey M. Deitel and Paul J. Deitel, available as a paperback from 2000 for \$2.99 (85% off). The second is 'C++: How to Program' by Harvey M. Deitel, available as a paperback textbook from 1994 for \$1.99 (96% off). The third is 'Data Structures and Program Design in C++' by Robert L. Kruse and Alexander J. Ryba, available as a paperback from 1998 for \$39.00 (46% off). The fourth is 'Program Development & Design Using C++' available as a paperback textbook from 2000 for \$34.00 (51% off). The fifth is 'Programming with C++: Program Design Including Data Structures' by Davendar Malik and D. S. Malik, available as a paperback textbook from 2002 for \$45.00 (35% off). The sixth is 'C++ Programming: From Problem Analysis to Program Design' by Davendar Malik and D. S. Malik, available as a paperback from 2002 for \$32.99 (51% off).

Book Title	Author(s)	Format	Year	Price	Savings
<b>C how to Program: Introducing C++ and Java 3RD BK&amp;CDR (with CD-ROM)</b>	Harvey M. Deitel, Paul J. Deitel	Paperback	2000	\$2.99	85%
<b>C++: How to Program</b>	Harvey M. Deitel	Paperback Textbook	1994	\$1.99	96%
<b>Data Structures and Program Design in C++</b>	Robert L. Kruse, Alexander J. Ryba	Paperback	1998	\$39.00	46%
<b>Program Development &amp; Design Using C++</b>		Paperback Textbook	2000	\$34.00	51%
<b>Programming with C++: Program Design Including Data Structures</b>	Davendar Malik, D. S. Malik	Paperback Textbook	2002	\$45.00	35%
<b>C++ Programming: From Problem Analysis to Program Design</b>	Davendar Malik, D. S. Malik	Paperback	2002	\$32.99	51%

Figure 5.1: A sample of a Web page containing book records coming from a different Web site shown in Figure 1.1

```

...
<TD WIDTH="100<FONT FACE="Verdana, Geneva, Arial" size="2"> <A
HREF="http://half.ebay.com/cat/buy/prod.cgi?cpid=5254706&domain_id=1856&meta_id=1"> <B>
Program Development & Design Using C++ </B> </A> &nbsp; <BR> <FONT
FACE="Verdana, Geneva, Arial" SIZE="2"> &nbsp; &nbsp; <a
href="http://half.ebay.com/cat/buy/prod.cgi?cpid=5254706&domain_id=1856&meta_id=1"> Paperback
Textbook, 2000 </a> &nbsp; &nbsp; - Buy it for <a
href="http://half.ebay.com/cat/buy/prod.cgi?cpid=5254706&domain_id=1856&meta_id=1"> <font
color="#CC0000"> $34.00 </font> </a> (Save 51</TD> </TR> <TR> <TD COLSPAN=3>
<TABLE WIDTH="100<TD WIDTH="100</TD> </TR> </TABLE> </TD> </TR> <TR> <TD
WIDTH=50 VALIGN=TOP> <A
HREF="http://half.ebay.com/cat/buy/prod.cgi?cpid=1108738972&domain_id=1856&meta_id=1"> <IMG
BORDER=0 SRC="http://art.half.ebay.com/prod70/1581141.jpeg" WIDTH=56 HEIGHT=70
ALIGN=ABSMIDDLE> </A> </TD> <TD WIDTH="100<FONT FACE="Verdana, Geneva, Arial"
size="2"> <A
HREF="http://half.ebay.com/cat/buy/prod.cgi?cpid=1108738972&domain_id=1856&meta_id=1"> <B>
Programming with C++: Program Design Including Data Structures </B> </A> &nbsp; &nbsp;
Davendar Malik, D. S. Malik <BR> <FONT FACE="Verdana, Geneva, Arial" SIZE="2">
&nbsp; &nbsp; &nbsp; &nbsp; &nbsp; <a
href="http://half.ebay.com/cat/buy/prod.cgi?cpid=1108738972&domain_id=1856&meta_id=1">
Paperback Textbook, 2002 </a> &nbsp; &nbsp; - Buy it for <a
href="http://half.ebay.com/cat/buy/prod.cgi?cpid=1108738972&domain_id=1856&meta_id=1"> <font
color="#CC0000"> $45.00 </font> </a> (Save 35</TD> </TR>
...

```

Figure 5.2: An excerpt of the HTML texts for the Web page shown in Figure 5.1

with the Web page. The Web pages in Figures 5.1 and 1.1 are collected from different Web sites. Both Web pages are about “programming books”. Although the wrapper learned from Figure 1.1 is able to extract book records from Web pages in the same site very effectively, it cannot be applied to extract information from the Web page shown in Figure 5.1. In order to extract information in Figure 5.1, the user has to provide a new set of user annotated training examples. Then a new wrapper tailored to the Web site shown in Figure 5.1 can be learned.

Wrapper adaptation aims at adapting a previously learned wrapper of a particular source Web site to a new unseen target Web site in the same domain. For example, through wrapper adaptation, the previously learned wrapper from the Web page shown in Figure 1.1 can be utilized and a new wrapper can be discovered for extracting attribute items in the Web page shown in Figure 5.1. This capability can also help solving the wrapper maintenance problem. In this paper, we present our framework for tackling the wrapper adaptation.

## **5.2 Overview of Wrapper Adaptation Framework**

Figure 5.3 depicts our wrapper adaptation framework. The idea of our approach for wrapper adaptation is to first automatically identify some machine annotated training examples in the unseen Web site. The machine annotated training examples, corresponding to certain text fragments from Web pages

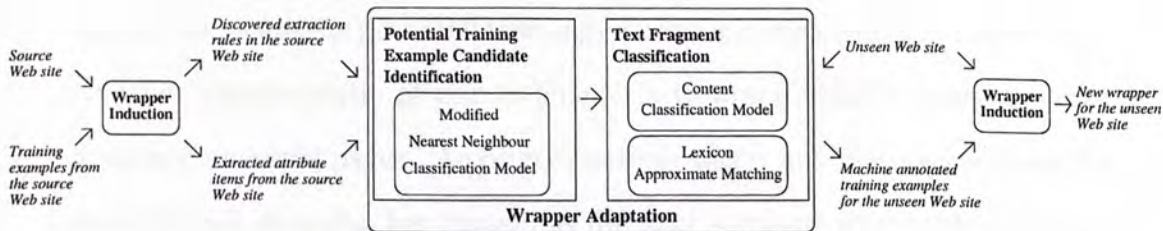


Figure 5.3: Our wrapper adaptation framework

in the unseen site, are automatically annotated by our wrapper adaptation approach. To achieve this, we make use of the previously discovered wrapper and the extracted attribute items from the source Web site. After machine annotated training examples are obtained, a new wrapper can be learned to extract information for the unseen Web site.

Our wrapper adaptation framework is a two-stage method employing multiple learning paradigms. In the first stage, a modified nearest neighbour algorithm is developed to seek potential training example candidates in the unseen Web site. In the second stage, a text fragment classification model is proposed to classify the potential training example candidates. Those “good” potential training example candidates will be selected as machine annotated training examples for the unseen Web site. From these machine annotated training examples, a new wrapper for the unseen Web site can be learned to extract information.

The objective of the first stage is to seek some potential training example candidates from the unseen Web site. We make use of the extraction rules of the wrapper previously discovered in the source Web site. Recall that the target pattern component of the extraction rules captures the semantic class of the attribute items of interest. This semantic information will be

utilized to locate the potential training example candidates in the unseen site. Another characteristic of our technique is to automatically generate some auxiliary example pages. Auxiliary example pages are Web pages from the same unseen Web site, but containing different attribute item content. Under the same Web site, we observe that generally the text fragments regarding the attribute items in different pages are different, while the text fragments regarding the layout format are similar in different pages. Based on this observation, a modified nearest neighbor classification model is developed to identify the appropriate text fragments for potential training example candidates. One of the challenges in wrapper adaptation problem is that Web pages in different Web sites have different layout format. In this stage, the layout format of the Web pages is analyzed in a site-independent manner with the modified nearest neighbour classification model.

In the second stage, a text fragment classification model is employed to classify the “good” potential training example candidates. The attribute items previously extracted from the source Web site embody rich knowledge about the content of the attribute items. The idea of text fragment classification is to use a classification model to capture the characteristics of the attribute items. This model consists of two components. The first component is a content classification model. This content classification model considers several features to characterize the content of the attribute items. In the second component, we develop a lexicon approximate matching technique. The previously extracted attribute items from the source Web site can be treated as a large lexicon of the corresponding attribute items. This lexicon provides another useful clue for locating “good” potential training example

candidates. Those selected “good” potential training example candidates will become machine annotated training examples for the unseen Web site. This text fragment classification model is proposed to tackle the challenge of different layout format of attribute items in different Web sites.

After obtaining the machine annotated training examples, a new wrapper for the unseen Web site can be discovered by our wrapper induction approach.

## 5.3 Potential Training Example Candidate Identification

Recall that in the first stage of our wrapper adaptation framework, the objective is to seek potential training example candidates. We introduce a concept called *useful text fragment*.

### 5.3.1 Useful Text Fragments

A Web page can be regarded as a sequence of text tokens. A token can be a word, number, punctuation, date, HTML tag, specific ASCII character, or some domain specific contents such as manufacture names.

**Definition 5.1** We define a *segment* to be a sequence of continuous tokens in a Web page not containing any HTML tag, and is immediately before and after tokens belonging to the semantic class *delimiter*. The delimiter semantic class contains user defined tokens such as HTML tag, punctuation, specific ASCII characters, or some domain specific contents.



**Definition 5.2** We define a *seed* to be a segment of a Web page which contains certain semantic meaning.

We can generate some text fragments by extending the seeds forward and backward. A parameter  $T$  is used to control the extending window size.

**Definition 5.3** For a seed  $s$ ,  $Pre-Seed(s)$  is defined as the set of the positions of the  $i$ -th token immediately before  $s$ , where  $i = 1, \dots, T$ .

**Definition 5.4** For a seed  $s$ ,  $After-Seed(s)$  is defined as the set of the positions of the  $i$ -th token immediately after  $s$ , where  $i = 1, \dots, T$ .

**Definition 5.5** For a seed  $s$ ,  $Extended-Seed(s)$  is defined as the set of text fragments whose starting position and ending position are indicated by the cross product of  $Pre-Seed(s)$  and  $After-Seed(s)$ .

For instance, consider a seed  $c_0$  in a Web page and suppose  $T$  is set to 3. We get  $\{Pre-Seed(c_0)\} = \{s_1, s_2, s_3\}$  where  $s_i$  is the position of the  $i$ -th token immediately before  $c_0$ , and  $\{After-Seed(c_0)\} = \{e_1, e_2, e_3\}$  where  $s_j$  is the position of the  $j$ -th token immediately after  $c_0$ . Then  $\{Extended-Seed(c_0)\}$  becomes the set of text fragments whose starting position and ending position is indicated by the set as follows:

$$\{(s_1, e_1), (s_1, e_2), (s_1, e_3), (s_2, e_1), (s_2, e_2), (s_2, e_3), (s_3, e_1), (s_3, e_2), (s_3, e_3)\}.$$

We define the *useful text fragment (UTF)* of a Web page as follows:

**Definition 5.6** For a Web page  $P$ , we define  $UTF(P)$  as:

$$UTF(P) = \cup_s \{Extended-Seed(s)\}$$

where  $s$  are all the seeds found in  $P$ .

### 5.3.2 Training Example Generation from the Unseen Web Site



Figure 5.4: A sample of Web page about networking books

Our wrapper adaptation will automatically annotate some machine annotated training examples in one of the Web pages in the unseen Web site. We call the Web page where the machine annotated training examples will be automatically annotated as *main example page*. Relative to a main example page, auxiliary example pages are Web pages from the same Web site, but containing different attribute item content. Generally, under a particular Web site, the text fragments regarding the attribute item content are

```

...
</TD> <TD WIDTH="100<FONT FACE="Verdana, Geneva, Arial" size="2"> <A
HREF="http://half.ebay.com/cat/buy/prod.cgi?cpid=3051974&domain_id=1856&meta_id=1"> <B>
The Terror Network </B> </A> &nbsp; &nbsp; Claire Sterling <BR> <FONT
FACE="Verdana, Geneva, Arial" SIZE="2"> &nbsp; &nbsp; <a
href="http://half.ebay.com/cat/buy/prod.cgi?cpid=3051974&domain_id=1856&meta_id=1"> Paperback,
1985 </a> &nbsp; &nbsp; - Buy it for <a
href="http://half.ebay.com/cat/buy/prod.cgi?cpid=3051974&domain_id=1856&meta_id=1"> <font
color="#CC0000"> $0.75 </font> </a> (Save 81</TD> </TR> <TR> <TD COLSPAN=3>
<TABLE WIDTH="100<TD WIDTH="100</TD> </TR> </TABLE> </TD> </TR> <TR> <TD
WIDTH=50 VALIGN=TOP> <A
HREF="http://half.ebay.com/cat/buy/prod.cgi?cpid=5019677&domain_id=1856&meta_id=1"> <IMG
BORDER=0 SRC="http://art.half.ebay.com/prod70/1770674.jpeg" WIDTH=51 HEIGHT=70
ALIGN=ABSMIDDLE> </A> </TD> <TD WIDTH="100<FONT FACE="Verdana, Geneva, Arial"
size="2"> <A
HREF="http://half.ebay.com/cat/buy/prod.cgi?cpid=5019677&domain_id=1856&meta_id=1"> <B>
Network Performance Baseline </B> </A> &nbsp; &nbsp; <BR> <FONT
FACE="Verdana, Geneva, Arial" SIZE="2"> &nbsp; &nbsp; <a
href="http://half.ebay.com/cat/buy/prod.cgi?cpid=5019677&domain_id=1856&meta_id=1"> Hardcover,
2000 </a> &nbsp; &nbsp; - Buy it for <a
href="http://half.ebay.com/cat/buy/prod.cgi?cpid=5019677&domain_id=1856&meta_id=1"> <font
color="#CC0000"> $6.44 </font> </a> (Save 87</TD> </TR>
...

```

Figure 5.5: An excerpt of the HTML texts for the Web page shown in Figure 5.4

different, while the text fragments regarding the layout format are similar in different Web pages. This observation gives a good indication for locating the attribute items.

Auxiliary example pages can be automatically obtained from different pages easily in a Web site. One typical method is to supply different keywords or queries automatically to the search engine provided by the Web site. For instance, consider the book catalog associated with the Web page shown in Figure 5.1. This Web page is generated by supplying the keyword “PROGRAM” to the search engine provided by the Web site. Suppose a different keyword such as “NETWORK” is supplied to the search engine, a new Web page as shown in Figure 5.4 is returned. Only a few keywords are needed for a domain and they can be easily chosen in advance. The Web page in Figure 5.4 can be regarded as an auxiliary example page relative to the Web page in Figure 5.1. The text content of the auxiliary example pages provides very useful clues for seeking appropriate text fragments related to the attribute item of interest. Figures 5.2 and 5.5 show the excerpt of the HTML text document associated with the Web page shown in Figures 5.1 and 5.4 respectively. The bolded text fragments are related to the attribute items of interest, while the remaining text fragments are related to the format layout. The text fragments related to attribute items are very different in different Web pages, whereas the text fragments related to the format layout are very similar.

We provide some formal definitions of main example page and auxiliary example page as follows:

**Definition 5.7** We define *main example page*,  $M$ , as one of the Web pages in the unseen Web site where the machine annotated training examples will be obtained.

**Definition 5.8** We define *auxiliary example page*,  $A(M)$ , as a Web page under the same Web source as the main example page  $M$ , but it contains different attribute item content from those in the main example page  $M$ .

Based on the properties of the main example page and auxiliary example pages, a modified nearest neighbour classification model is developed to identify the potential training example candidates.

### 5.3.3 Modified Nearest Neighbour Classification

Recall that the target pattern component of the previously discovered extraction rules from the source Web site contains the semantic class of the attribute item. From the main example page  $M$  of the unseen target Web site, we can obtain the set  $UTF(M)$  based on the semantic class of the attribute items. From an auxiliary example page  $A(M)$ , we can also obtain the set  $UTF(A(M))$ . As mentioned in the previous subsection, the text fragments regarding the attribute items in the main example page are less likely to appear in the auxiliary example page, while the text fragments regarding the layout format will probably appear in both of the main example page and the auxiliary example page. Hence, all the elements in  $UTF(A(M))$  are treated as negative instances relative to the text fragment regarding the attribute items in the main example page  $M$ .

**Definition 5.9** Suppose we have two text fragments  $t_1$  and  $t_2$  with  $m$  and  $n$  tokens respectively. We define the similarity between the two text fragments  $t_1$  and  $t_2$ ,  $s(t_1, t_2)$ , as:

$$s(t_1, t_2) = \frac{k}{\max\{m, n\}} \quad (5.1)$$

where  $k$  is the number of tokens in  $t_1$  matched with the tokens in  $t_2$ .

The goal of our modified classification model to classify the potential training examples from  $UTF(M)$ . To achieve this task, for each element in  $UTF(M)$ , we first find its nearest neighbour in  $UTF(A(M))$  based on our defined similarity measure. If the similarity between the element in  $UTF(M)$  and its nearest neighbour in  $UTF(A(M))$  exceeds a threshold,  $\theta$ , it will be classified as negative instance. On the other hand, if the similarity is below  $\theta$ , it will be classified as a potential training example candidate.

Once the potential training example candidates for an attribute item are identified, they are processed by a text fragment classification model in the second stage to classify those candidates that likely become the machine annotated training examples for the unseen Web site.

## 5.4 Machine Annotated Training Example Discovery and New Wrapper Learning

### 5.4.1 Text Fragment Classification

Text fragment classification is designed to classify “good” candidates for subsequent learning process. The text fragment classification model consists of

two major components, namely, content classification and lexicon approximate matching.

### **Content Classification**

We identify some features for characterizing the content of the attribute item. A classification model can then be learned to classify the “good” potential training example candidates. The features used are as follows:

- $F_1$ : the number of characters in the content
- $F_2$ : the number of tokens in the content
- $F_3$ : the average number of characters per token
- $F_4$ : the proportion of the number of digit number to the number of tokens
- $F_5$ : the proportion of the number of floating point number to the number of tokens
- $F_6$ : the proportion of the number of alphabet to the number of characters
- $F_7$ : the proportion of the number of upper case characters to the number of characters
- $F_8$ : the proportion of the number of lower case characters to the number of characters
- $F_9$ : the proportion of the number of punctuation to the number of characters

- $F_{10}$ : the proportion of the number of HTML tags to the number of tokens
- $F_{11}$ : the proportion of the number of tokens starting with capital letter to the number of tokens
- $F_{12}$ : whether the content starts with a capital letter

These features are chosen because they can effectively characterize the format of the attribute items. For example, a book title usually starts with a capital letter and contains less proportion of HTML tags. Some of the features are also used in [37]. With the above feature design, a classification model can be learned from a set of training examples. The content classification model will return a score,  $f_1$ , which indicates the degree of confidence being “good” potential training example candidates.  $f_1$  will be normalized to a value between 0 and 1.

The content classification model is learned from a set of training examples composed of a set of *positive attribute content examples* and *negative attribute content examples*. The set of positive attribute content examples are those manual annotations obtained from the main example page,  $M_s$ , of the source site  $S$ . Then we obtain the  $UTF(M_s)$  based on the semantic class of the target pattern component of the extraction rules for the source site  $S$ . Those elements in  $UTF(M_s)$  which are not in the set of positive attribute content examples are collected to become the negative attribute content examples. Next, the values of the features  $F_i$  ( $1 \leq i \leq 12$ ) of each positive and negative attribute content example can be computed. As a result, a set of training examples for the content classification is prepared. To learn the content



classification model, we employ Support Vector Machines [52] to achieve this task.

### Lexicon Approximate Matching

For each attribute item in the same domain, we maintain a lexicon by storing the previously automatically collected attribute items from the source site. For example, there is a lexicon containing entries for the model number attribute in the electronic appliance domain. This previously discovered lexicon can help determine good training examples for an unseen site. Precisely, the content of a lexicon can be taken into consideration for classifying the potential training example candidates. French et al. [25] discussed the effectiveness of approximate word matching in information retrieval. We pose the problem of classifying potential training example candidates as a lexicon approximate matching task. We make use of edit distance [32] to handle this task. Basically, our lexicon approximate matching algorithm is a two-level matching algorithm. At the lower level, we compute the character-level edit distance of a given pair of tokens. At the upper level, we compute the token-level edit distance of a given pair of text fragments. We will illustrate our algorithm by an example.

Suppose we obtain a potential training example candidate of model number “*PANASONIC DVDCV52*” and a particular entry “*PAN DVDRV32K*” in the lexicon. (Actually these two model numbers are obtained from two different Web sites in our electronic appliance domain experiment. They refer to the same brand of product, but different model number.) At the lower level, we compute the character-level edit distance between two tokens with

the cost of insertion, deletion, and modification of a character all equal to one. Then the character-level edit distances computed are normalized by the longest length of the tokens. For example, the normalized character-level edit distance between “*PAN*” and “*PANASONIC*” is 0.667. Table 5.1 shows the character-level edit distances between the tokens in this example.

	PAN	DVDRV32K
PANASONIC	0.667	1.000
DVDCV52	1.000	0.375

Table 5.1: Normalized character-level edit distances between tokens

At the upper level, we compute the token-level edit distance between a potential training example candidate and a lexicon entry, with the cost of insertion and deletion of a token equal to one, and the cost of modification of a token equal to the character-level edit distance between the tokens. The token-level edit distance obtained is then normalized by the largest number of tokens among the training example candidate and the lexicon entry. For instance, the normalized token-level edit distance between “*PANASONIC DVDCV52*” and “*PAN DVDRV32K*” is 0.521.

Both of the character-level and token-level edit distance can be computed efficiently by dynamic programming. We describe briefly the calculation of the token-level edit distance. Suppose a potential training example candidate  $c$  consists of a sequence of tokens  $c_1, \dots, c_m$ . Let the set of previously discovered lexicons be  $l^1, l^2, \dots$ , where each  $l^i$  is represented by a sequence of tokens  $l_1^i, \dots, l_n^i$ . Then the token-level edit distance  $D(c, l^i)$  between  $c$  and each  $l^i$  is computed by dynamic programming with the following recursive

equation:

$$D_{m,n}(c, l^i) = \min \left\{ \begin{array}{l} d(c_m, l_n^i) + D_{m-1, n-1}(c, l^i), \\ 1 + D_{m, n-1}(c, l^i), \\ 1 + D_{m-1, n}(C, l^i) \end{array} \right\} \quad (5.2)$$

where  $D_{m,0}(c, l^i) = m$ ,  $D_{0,n}(c, l^i) = n$ , and  $d(p, q)$  is the normalized character-level edit distance between token  $p$  and  $q$ .

The score,  $f_2$ , of a potential training example candidate is then computed as follows:

$$f_2 = \max_i \{D'(c, l^i)\} \quad (5.3)$$

where  $D'(c, l^i) = 1 - D_{m,n}(c, l^i) / \max\{m, n\}$ .

In the text fragment classification, the score from content classification and lexicon approximate matching will be computed. The final score  $Score(c)$  of each potential training example candidate  $c$  is given by:

$$Score(c) = wf_1 + (1 - w)f_2 \quad (5.4)$$

where  $f_1$  and  $f_2$  are the score obtained in content classification and lexicon approximate matching respectively;  $w$  is a parameter controlling the weight of the content classification and lexicon approximate matching and  $0 < w < 1$ .

### 5.4.2 New Wrapper Learning

After the scores of the potential training example candidates are computed, our framework will select “good” candidates as machine annotated training examples for the unseen site. The  $N$  best potential training example

candidates will be selected as the machine annotated training examples for wrapper induction for the unseen site.

Recall that our hierarchical record structure models the whole record in a tree-like structure. The record consists of different attribute items. The machine annotated training examples obtained have not grouped in a record. We adopt the discovery of repeated pattern approach [7] to discover the record boundary and group the candidates into records. This method can automatically identify the repeated pattern in a Web page, by making use of PAT trees. The repeated pattern will be considered to determine if it contains useful information. Candidates within two repeated patterns are then grouped to the same record. The records will become the training examples for the unseen site. Users could optionally scrutinize discovered training examples to improve the quality of the training examples. However, in our experiment, we did not conduct manual intervention and the adaptation was conducted in a fully automatic way. A new tailor-made wrapper to the unseen site can be learned by our wrapper induction approach. The newly learned wrapper can then be applied to the remaining pages in the unseen target Web site.

## Chapter 6

# Case Study and Experimental Results for Wrapper Adaptation

### 6.1 Case Study on Wrapper Adaptation

Chapter 4 demonstrates the performance of our wrapper induction approach. For example, the wrapper learned from S8 (Figure 1.1) by our system can extract records from the Web pages in the same site very effectively. However, if we apply the learned wrapper directly to extract records from S7 (Figure 5.1), it cannot extract any record. We apply our wrapper adaptation framework to tackle this problem. After the first stage, some machine annotated training examples for S7, such as the two samples shown in Table 6.1, are automatically obtained. The last column of Table 6.1 shows the score of the attribute item calculated by our wrapper adaptation framework.

	Attribute item	Field value	Score
Example 1	<b>Book Title:</b>	Programming with C++: Program Design Including Data Structures	0.63
	<b>Final Price:</b>	45.00	1.00
Example 2	<b>Author:</b>	Steve Heller	0.60
	<b>Final Price:</b>	4.99	1.00

Table 6.1: Samples of a machine annotated training example obtained by adapting the wrapper from the Web site S8 (Figure 1.1) to the Web site shown in S7 (Figure 5.1).

Example 1 has a book title with score 0.63 and a final price with score 1.0. Example 2 has an author with score 0.60 and a final price with score 1.0. Users can optionally scrutinize the machine annotated training examples to improve the quality of the training examples. In this case study, we did not conduct manual intervention and the adaptation was conducted in a fully automatic way. We applied our wrapper induction approach to learn a wrapper for S7 from the machine annotated training examples. Although some of the machine annotated training examples are incomplete and contain missing attribute items, our wrapper induction approach can still learn the hierarchical record structure and extraction rules from incomplete examples.

The newly learned wrapper was then applied to Web pages within the Web site S7. We obtained very promising results: the precision and recall for title are 100.0% and 95.0% respectively; the precision and recall for author are 94.3% and 73.5% respectively; the precision and recall for final price are 100.0% and 92.5% respectively. Hence, the extraction performance with

applying our wrapper adaptation approach is much better than the extraction performance without adaptation.

## 6.2 Experimental Results

In order to demonstrate the effectiveness of our wrapper adaptation approach, we have conducted extensive experiments to extract items from a variety of real-world Web sites containing online book catalogs and consumer electronic appliance catalogs as shown in Table 4.1.

For each domain, we conducted two sets of experiments. The first set of experiments is to simply apply the hierarchical record structure and extraction rules learned from one particular Web site without adaptation to all other sites for information extraction. This experiment can be treated as a baseline for our wrapper adaptation approach. The second set of experiments is to adapt the hierarchical record structure and extraction rules learned from one particular Web site to other sites automatically by our wrapper adaptation approach.

In our framework, three parameters are needed to be determined in advance. The first parameter is the threshold  $\theta$  in the modified nearest neighbour classification model described in Section 5.3.3 in Chapter 5. The second parameter is the weight  $w$  in the text fragment classification model described in Section 5.4.1 in Chapter 5. The last parameter is the  $N$  in the  $N$  best potential training example candidates as described in Section 5.4.2 in Chapter 5. In order to determine the values of the parameters, we randomly chose three Web sites, labeled as T1, T2, and T3, for tuning the parameters. We

exhaustively conducted experiments for these three Web sites with different parameter values. The wrapper discovered from one particular Web site is adapted to the remaining two sites. The average of the F-measure is used for the evaluation of the parameter settings<sup>1</sup>. We selected the parameter setting that achieves the highest performance. Then this set of parameters will be used in all the remaining testing Web sites in our experiments. The parameters selected were  $\theta = 0.5$ ,  $w = 0.3$ , and  $N = 5$ .

### 6.2.1 Book Domain

Table 6.2 shows the results of the the first set of experiments for the book domain. The  $i^{th}$  row in Table 6.2 represents an experiment of extracting contents from all the Web sites by using the wrapper learned from the site labeled as  $S_i$ . Each cell in Table 6.2 is divided into two sub-columns and three sub-rows. The three sub-rows represent the extraction performance of the attribute items, namely, title, author, and price of the books respectively. The two sub-columns represent the precision (P) and recall (R) for extracting the attribute items of interest respectively. These results are obtained by simply applying the learned wrapper from one Web site to the remaining sites without adaptation.

The diagonal cells of Table 6.2 shows the capability of extracting information from Web pages originating from the same Web site. This is, in fact, the experimental result of our wrapper induction and is also shown in Table 4.4. Other cells in Table 6.2 represent the results of extracting information with-

---

<sup>1</sup>F-measure is defined in Equation 4.1.



	S1		S2		S3		S4		S5		S6		S7		S8		S9		S10		S11	
	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R
S1	97.0	97.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	97.0	98.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	97.0	98.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
S2	0.0	0.0	90.6	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	100.0	67.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	97.9	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
S3	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
S4	0.0	0.0	0.0	0.0	0.0	0.0	99.0	99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	80.2	97.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
S5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	97.8	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	75.6	75.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
S6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
S7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	94.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	97.0	97.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
S8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	91.3	91.3	0.0	0.0	0.0	0.0	0.0	0.0
S9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	84.3	100.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	84.3	100.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	84.3	84.3	0.0	0.0	0.0	0.0
S10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0
S11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	96.0	96.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	98.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0

Table 6.2: Experimental results of applying a learned wrapper without adaptation from one Web site to extract contents from the remaining sites in the book domain.

out applying our wrapper adaptation framework. The result indicates that no learned wrapper from a particular Web site can be applied to extract attribute items from other sites.

Table 6.3 shows the results of the the second set of experiment for the book domain. These results are obtained by adapting a learned wrapper from one Web site to the remaining sites using our wrapper adaptation approach. The result indicates that the extraction performance is very satisfactory.

Table 6.4 summarizes the average extraction performance on title, author, and price respectively for the cases of without adaptation and with adaptation when training examples of one particular Web site are provided. The first column shows the Web sites where training examples are given. Each row summarizes the results obtained by using the learned wrapper of the Web site in the first column and applying to all other sites for extraction. Essentially, it summarizes the results in Tables 6.2 and 6.3. The result indicates that the extraction rules of a particular Web site cannot be directly applied to others without adaptation. After applying our wrapper adaptation approach, the wrapper learned from a particular Web site can adapt to other sites. The results show that our wrapper adaptation approach achieves a very promising performance especially compared with the performance obtained without adaptation. S10 cannot extract the price attribute from other Web sites with and without adaptation. The reason is that the price attribute of the records in S10 contains items with a totally different format.

	S1		S2		S3		S4		S5		S6		S7		S8		S9		S10		S11	
	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R
S1	-	-	100.0	100.0	27.5	100.0	44.0	99.2	11.3	100.0	98.4	50.0	100.0	95.0	100.0	90.0	85.4	87.2	16.0	100.0	0.0	0.0
	-	-	100.0	73.3	43.8	94.6	7.7	12.0	15.2	69.4	86.1	50.0	95.7	58.4	100.0	90.0	42.7	87.2	50.2	100.0	41.5	64.5
	-	-	36.3	99.2	0.0	0.0	100.0	100.0	78.0	78.0	100.0	50.0	100.0	92.5	33.3	90.0	0.0	0.0	0.0	0.0	9.4	100.0
S2	0.0	0.0	-	-	0.0	0.0	87.3	99.2	23.9	100.0	68.9	50.0	100.0	95.0	64.7	33.0	85.4	87.2	0.0	0.0	0.2	3.6
	85.3	98.4	-	-	99.1	100.0	7.7	12.0	20.8	74.5	25.8	20.2	95.7	58.4	100.0	90.0	85.4	87.2	0.0	0.0	48.0	98.2
	3.6	99.2	-	-	100.0	100.0	100.0	100.0	78.0	78.0	100.0	50.0	100.0	92.5	33.3	90.0	0.0	0.0	0.0	0.0	9.2	100.0
S3	0.0	0.0	38.3	90.0	-	-	98.4	99.2	100.0	100.0	0.0	0.0	100.0	95.0	100.0	33.0	83.7	87.2	34.4	100.0	0.0	0.0
	10.5	74.2	43.4	74.2	-	-	0.0	0.0	22.5	100.0	0.0	0.0	0.0	0.0	100.0	90.0	80.4	87.2	0.0	0.0	41.0	64.5
	4.3	99.2	35.9	97.5	-	-	0.0	0.0	78.0	78.0	100.0	50.0	100.0	92.5	33.3	90.0	0.0	0.0	0.0	0.0	9.3	100.0
S4	45.9	98.4	0.0	0.0	50.0	100.0	-	-	10.5	100.0	33.3	0.8	100.0	95.0	100.0	90.0	80.4	87.2	1.2	90.0	6.9	96.4
	0.4	26.6	100.0	73.3	0.0	0.0	-	-	0.0	0.0	100.0	50.0	94.3	73.5	100.0	90.0	42.7	87.2	1.7	100.0	25.4	98.2
	3.9	99.2	36.3	99.2	100.0	100.0	-	-	78.0	78.0	100.0	50.0	100.0	92.5	33.3	90.0	0.0	0.0	0.0	0.0	9.4	100.0
S5	0.0	0.0	100.0	100.0	50.0	100.0	76.5	99.2	-	-	82.7	50.0	100.0	0.8	89.1	90.0	0.0	0.0	2.4	80.9	18.0	96.4
	8.9	74.2	0.0	0.0	0.0	0.0	8.6	95.2	-	-	0.0	0.0	83.1	86.7	100.0	90.0	80.4	87.2	8.0	100.0	46.8	98.2
	4.3	99.2	36.3	99.2	0.0	0.0	100.0	100.0	-	-	100.0	50.0	100.0	92.5	33.3	90.0	0.0	0.0	0.0	0.0	9.1	100.0
S6	48.8	97.6	100.0	100.0	27.1	79.5	20.0	0.8	0.0	0.0	-	-	100.0	95.0	100.0	90.0	42.7	87.2	100.0	100.0	0.0	0.0
	85.3	98.4	100.0	15.8	35.1	96.4	7.7	12.0	13.3	69.4	-	-	93.9	94.7	100.0	90.0	85.4	87.2	37.5	100.0	67.1	98.2
	5.0	99.2	36.3	99.2	100.0	100.0	100.0	100.0	78.0	78.0	-	-	100.0	92.5	33.3	90.0	0.0	0.0	0.0	0.0	9.1	100.0
S7	8.9	98.4	75.2	73.3	27.1	79.5	99.2	99.2	36.6	100.0	0.0	0.0	-	-	100.0	33.0	80.4	87.2	1.4	90.0	77.9	96.4
	84.1	98.4	100.0	73.3	35.1	96.4	0.0	0.0	98.0	100.0	46.3	50.0	-	-	100.0	90.0	0.0	0.0	76.9	100.0	0.0	0.0
	6.6	99.2	98.3	100.0	0.0	0.0	0.0	0.0	78.0	78.0	100.0	50.0	-	-	33.3	90.0	0.0	0.0	0.0	0.0	10.4	100.0
S8	0.0	0.8	100.0	73.3	27.7	79.5	99.2	99.2	11.8	100.0	0.0	0.0	100.0	95.0	-	-	80.4	87.2	15.8	100.0	0.0	0.0
	14.4	74.2	100.0	65.0	89.9	95.5	0.0	0.0	21.1	93.9	100.0	50.0	94.3	73.5	-	-	85.4	87.2	58.5	100.0	41.5	64.5
	4.4	99.2	36.3	99.2	100.0	100.0	0.0	0.0	78.0	78.0	100.0	50.0	100.0	92.5	-	-	0.0	0.0	0.0	0.0	9.4	100.0
S9	1.5	24.8	50.0	100.0	36.8	79.5	97.6	99.2	33.3	100.0	0.0	0.0	100.0	95.0	100.0	90.0	-	-	1.2	80.9	8.3	96.4
	50.0	74.2	98.9	73.3	100.0	100.0	0.0	0.0	97.3	74.5	100.0	50.0	95.7	58.4	100.0	90.0	-	-	1.5	100.0	45.8	89.1
	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	78.0	78.0	100.0	50.0	100.0	92.5	33.3	90.0	-	-	0.0	0.0	0.0	0.0
S10	14.0	98.4	100.0	73.3	36.8	79.5	97.6	99.2	0.0	0.0	98.4	50.0	100.0	95.0	100.0	90.0	85.5	100.0	-	-	34.5	96.4
	58.6	99.2	98.7	65.0	100.0	100.0	0.0	0.0	5.6	90.8	0.0	0.0	93.2	48.7	100.0	90.0	23.4	100.0	-	-	41.5	64.5
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-	-	0.0
S11	11.3	97.6	97.6	100.0	100.0	100.0	35.8	99.2	10.9	100.0	98.4	50.0	66.3	95.0	100.0	33.0	0.0	0.0	27.6	100.0	-	-
	64.2	98.4	0.0	0.0	100.0	98.2	1.6	95.2	4.5	100.0	71.4	20.2	0.0	0.0	100.0	90.0	80.4	87.2	97.3	100.0	-	-
	4.3	99.2	36.3	99.2	100.0	100.0	100.0	100.0	78.0	78.0	100.0	50.0	100.0	92.5	33.3	90.0	0.0	0.0	0.0	0.0	-	-

Table 6.3: Experimental results of adapting a learned wrapper from one Web site to the remaining sites in the book domain.

### 6.3.3 Consumer Electronic Appliances Domain

Table 6.4 shows the results of the the first set of experiments for the consumer electronic appliance domain. The F<sub>1</sub> for the title, author, and price is the average of extracting contents from all the Web sites in the training set and from the site labeled as S<sub>1</sub>. Each cell in the

Source	<i>Title</i>				<i>Author</i>				<i>Price</i>			
	Without Adaptation		With Adaptation		Without Adaptation		With Adaptation		Without Adaptation		With Adaptation	
	prec.	recall	prec.	recall	prec.	recall	prec.	recall	prec.	recall	prec.	recall
S1	0.0	0.0	58.3	82.1	0.0	0.0	58.3	69.9	0.0	0.0	45.7	61.0
S2	0.0	0.0	43.0	46.8	0.0	0.0	56.8	63.9	0.0	0.0	52.4	71.0
S3	0.0	0.0	55.5	60.4	0.0	0.0	29.8	49.0	0.0	0.0	36.1	60.7
S4	0.0	0.0	42.8	75.8	0.0	0.0	46.5	59.9	0.0	0.0	46.1	70.9
S5	0.0	0.0	51.9	61.7	0.0	0.0	33.6	63.2	0.0	0.0	38.3	63.1
S6	0.0	0.0	53.9	65.0	0.0	0.0	62.5	76.2	0.0	0.0	46.2	75.9
S7	0.0	0.0	50.7	75.7	0.0	0.0	54.0	60.8	0.0	0.0	32.7	51.7
S8	0.0	0.0	43.5	63.5	0.0	0.0	60.5	70.4	0.0	0.0	42.8	61.9
S9	0.0	0.0	42.9	76.6	0.0	0.0	68.9	71.0	0.0	0.0	41.1	41.1
S10	0.0	0.0	66.7	78.2	0.0	0.0	52.1	65.8	0.0	0.0	0.0	0.0
S11	0.0	0.0	54.8	77.5	0.0	0.0	51.9	68.9	0.0	0.0	55.2	70.9

Table 6.4: Average extraction performance on *title*, *author*, and *price* for the book domain for the cases of without adaptation and with adaptation when training examples of one particular Web site are provided. (prec. refers to precision)

## 6.2.2 Consumer Electronic Appliance Domain

Table 6.5 shows the results of the the first set of experiments for the consumer electronic appliance domain. The  $i^{th}$  row in Table 6.5 represents an experiment of extracting contents from all the Web sites by using the wrapper learned from the site labeled as  $S_i$ . Each cell in Table 6.5 is divided into two sub-columns and three sub-rows. The three sub-rows represent the extraction performance of the attribute items, namely, model number, description, and price of the products respectively. The two sub-columns represent the precision (P) and recall (R) for extracting the attribute items of interest respectively. These results are obtained by simply applying the learned wrapper from one particular Web site to the remaining sites without adaptation.

The diagonal cells of Table 6.5 shows the capability of extracting information from Web pages originating from the same Web site. This is, in fact, the experimental result of our wrapper induction and is also shown in Table 4.5. Other cells in Table 6.5 represent the results of extracting information without applying our wrapper adaptation framework. The result indicates that no learned wrapper from a particular Web site can be applied to extract attribute items from other sites except S12 and S13. Since the layout format of S12 and S13 are very similar<sup>2</sup>, the wrapper learned from one of these Web sites can apply to each other. However, it is not true for other Web sites. In general, the extraction rules of a particular Web site cannot be applied to

---

<sup>2</sup>S12 and S13 use the same display template provided by www.yahoo.com for their layout format. This is also indicated by their URL's as shown in Table 4.1.

	S12		S13		S14		S15		S16		S17		S18		S19	
	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R
S12	100.0	100.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	100.0	100.0	93.3	93.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	100.0	100.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
S13	100.0	100.0	100.0	97.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	100.0	100.0	93.3	93.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	100.0	100.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
S14	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
S15	0.0	0.0	0.0	0.0	0.0	0.0	96.0	95.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	100.0	99.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	100.0	98.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
S16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	99.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	90.0	97.8	0.0	0.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	98.0	99.0	0.0	0.0	0.0	0.0	0.0	0.0
S17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	96.0	100.0	0.0	0.0	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0	0.0	0.0
S18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	87.1	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	52.9	86.3	0.0	0.0
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	87.1	0.0	0.0
S19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	97.9
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	82.0	52.1
	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	96.5

Table 6.5: Experimental results of applying a learned wrapper without adaptation from one Web site to extract contents from the remaining sites in the consumer electronic appliance domain.

others.

Table 6.6 shows the results of the the second set of experiment for the electronic domain. These results are obtained by adapting a learned wrapper from one Web site to the remaining sites using our wrapper adaptation approach. The result indicates that the extraction performance is very satisfactory.

	S12		S13		S14		S15		S16		S17		S18		S19	
	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R
S12	-	-	0	0	64.9	100	0	0	99.1	100	98	88.2	76	88.5	0	0
	-	-	0	0	0	0	0	0	90.7	98	0	0	0	0	0	0
	-	-	100	100	100	100	0	0	98.1	99.1	100	88.2	57.9	100	52.5	98.5
S13	70.9	100	-	-	91.9	100	0	0	99.1	100	85.8	88.2	25.9	41.4	0	0
	0	0	-	-	0	0	24.6	100	90.7	98	72.1	87.7	0.3	20.4	0	0
	100	100	-	-	100	100	65.8	100	98.1	99.1	100	88.2	57.9	100	52.5	98.5
S14	80	100	0	0	-	-	0	0	99.1	100	89.8	88.2	9.4	47.8	0	0
	0	0	0	0	-	-	0	0	90.7	98	41	87.7	0	0	0	0
	100	100	100	100	-	-	100	100	98.1	99.1	100	88.2	57.9	100	52.5	96.9
S15	0	0	0	0	0	0	-	-	99.1	100	0	0	1.4	100	0	0
	0	0	0	0	0	0	-	-	90.7	98	82.3	87.7	0	0	0	0
	100	100	100	100	100	100	-	-	98.1	99.1	100	88.2	57.9	100	52.5	96.9
S16	99	100	0	0	97.4	100	9.4	99.2	-	-	89.8	88.2	17.4	88.5	0	0
	0	0	0	0	51.7	92.9	0	0	-	-	43.5	87.7	0	0	0	0
	100	100	100	100	100	100	65.8	100	-	-	100	88.2	57.9	100	52.5	98.5
S17	100	100	0	0	89.7	100	0	0	99.1	100	-	-	59.1	88.5	0	0
	0	0	0	0	48.2	92.9	8.5	100	90.7	98	-	-	0	0	0	0
	100	100	100	100	100	100	65.8	100	98.1	99.1	-	-	57.9	100	52.5	98.5
S18	50	100	0	0	94.2	100	0	0	99.1	100	44.1	88.2	-	-	0	0
	0	0	0	0	0	0	0	0	90.7	98	40.8	87.7	-	-	0	0
	100	100	100	100	100	100	65.8	100	98.1	99.1	100	88.2	-	-	52.9	98.5
S19	0	0	32.7	98	0	0	0	0	99.1	100	0	0	0	0	-	-
	0	0	0	0	99.1	92.9	0	0	90.7	98	84.5	87.7	10.9	47.8	-	-
	100	100	100	100	100	100	65.8	100	98.1	99.1	100	88.2	57.9	100	-	-

Table 6.6: Experimental results of adapting a learned wrapper from one Web site to the remaining sites in the consumer electronic appliance domain.

Table 6.7 summarizes the average extraction performance on model number, description, and price respectively for the cases of without adaptation and with adaptation when training examples of one particular Web site are

Source	<i>Model number</i>				<i>Description</i>				<i>Price</i>			
	Without Adaptation		With Adaptation		Without Adaptation		With Adaptation		Without Adaptation		With Adaptation	
	prec.	recall	prec.	recall	prec.	recall	prec.	recall	prec.	recall	prec.	recall
S12	14.3	14.0	48.8	55.1	13.3	13.3	13.3	15.0	14.3	14.3	73.4	84.1
S13	14.3	14.3	53.3	61.1	14.3	14.3	26.1	42.5	14.3	14.3	82.5	98.0
S14	0.0	0.0	40.0	49.5	0.0	0.0	18.4	26.0	0.0	0.0	87.3	97.8
S15	0.0	0.0	14.0	27.8	0.0	0.0	24.0	25.8	0.0	0.0	87.3	97.8
S16	0.0	0.0	45.0	68.9	0.0	0.0	13.2	25.1	0.0	0.0	82.8	98.2
S17	0.0	0.0	49.7	56.7	0.0	0.0	20.5	40.4	0.0	0.0	82.5	99.7
S18	0.0	0.0	41.1	56.7	0.0	0.0	18.3	25.8	0.0	0.0	88.5	98.0
S19	0.0	0.0	18.3	27.5	0.0	0.0	39.6	45.3	0.0	0.0	89.1	98.2

Table 6.7: Average extraction performance on *model number*, *description*, and *price* for the electronic appliance domain for the cases of without adaptation and with adaptation when training examples of one particular Web site are provided. (prec. refers to precision)

provided. The meaning of the precision and recall figures is the same as Table 6.4. The result indicates that the extraction of the attribute items fails without adaptation in all cases except S12 and S13. After applying our wrapper adaptation approach, the wrapper learned from a particular Web site can adapt to other sites. The results show that our wrapper adaptation approach achieves a very satisfactory extraction performance for the attribute items model number and price. The extraction performances for the attribute item model number are not impressive for S15 and S19. The reason is that the model number in S15 is particularly short while the model number in S19 is particularly long. The extraction performance on attribute description is fair. This is due to the fact that the content of the description in different Web sites is quite different and involves a large portion of natural language.



## Chapter 7

# Conclusions and Future Work

We have developed a new wrapper induction approach for precise information extraction from semi-structured Web documents. Our approach can handle a richer set of Web pages. In order to minimize the user burden and enrich the expressiveness of extraction rules, our approach employs a two-stage learning task, namely, hierarchical record structure inference and extraction rule induction. In hierarchical record structure inference, a representation of hierarchical structures for the records in an information source is generated automatically. The design of hierarchical record structure can handle records which have missing attribute items, multi-valued attribute items, and items in unrestricted order. Compared with single-slot and multi-slot extraction rule, hierarchical record structure gives richer information about the records. Our record structure inference algorithm can also automatically infer the correct hierarchical record structure based on the statistics of the training examples. This reduce the human work in identifying the hierarchical record structure.

For extraction rule induction in the second stage, the extraction rules associated with each node in the hierarchical record structure are induced automatically. We have incorporated both lexical and semantic generalization to enrich the expressiveness of the extraction rules. Both domain independent and domain specific semantic classes are considered. These semantic classes are organized in a hierarchy. This organization improves the generalization process in extraction rule induction. Our extraction rule considers the semantic content of the attribute items. Such representation can enrich the expressiveness and improve the accuracy of the extraction rule. We have conducted experiments on a range of real-world Web sites in two different domains. The experimental results show the generality of our approach. Our approach is capable of extracting records in a wide range of Web sites in different domains effectively.

We have also developed a framework for wrapper adaptation. Wrapper adaptation aims at adapting a previously learned wrapper to an unseen target Web site. To achieve this goal, we propose a wrapper adaptation framework which can automatically seek good training examples for the unseen Web site. It is a two-stage method employing multiple learning paradigms in order to tackle the challenges in wrapper adaptation. The first stage is to seek *potential training example candidates* from the unseen Web site. We make use of extraction rules previously discovered from a particular site to seek potential candidates of training examples for the unseen site. A main example page and some auxiliary example pages are automatically generated for analysis. A modified nearest neighbour classification model is developed for identifying appropriate text fragments as potential training example candi-

dates. This design shows that auxiliary example pages provide very useful information for information extraction in Web pages. Based on the information from auxiliary example pages, the modified nearest neighbour model is able to classify the potential training example candidates by using machine learning technique effectively.

The potential training example candidates will then be classified by a text fragment classification model in the second stage. The idea is to use a classification model to capture the characteristics of the attribute items of interest. Those “good” candidates will be considered as *machine annotated training examples* for the unseen Web site. The text fragment classification model consists of two components. One component is the content classification model. Several kinds of features are considered to characterize the content of the attribute items. The other component is the lexicon approximate matching. The attribute items previously extracted in the source sites can be viewed as a set of lexicons. This lexicon provides an additional clue for selecting the machine annotated training examples. An approximate matching algorithm is developed to utilize this lexicon. The design of these two components can handle the ambiguity in the format of the attribute items in different Web sites. Based on the machine annotated training examples, a new wrapper tailored to the unseen Web site can be learned. We have conducted experiments on a number of Web sites in two domains. The experimental results demonstrate that our wrapper adaptation framework achieves encouraging results. The learning paradigms incorporated in our framework can effectively seek training examples for learning a new wrapper for the new unseen Web sites automatically. In today’s web-oriented infrastructure, our wrapper

adaptation approach can be applied to automatically integrate information from different Web sites. Our approach can effectively reduce human effort in the construction of wrappers for information extraction.

Further research can be explored to improve the effectiveness. One direction is to make use of some natural language processing (NLP) techniques to handle the free text portion of a Web page. From the experiment, it can be observed that the extraction performance for the attribute items having high portion of free text, such as the attribute item description in the consumer electronic appliance domain, is not as good as the extraction performance for other attribute items. The reason is that it is difficult to capture the semantic content of this kind of attribute items. NLP techniques, such as part-of-speech tagging and lexical semantic tagging may be useful in extraction of this kind of attribute items. However Web pages are semi-structured documents and not natural language documents, NLP techniques cannot be applied to Web pages directly. Certain modifications should be made in order to incorporate NLP techniques in our information extraction approach.

Our system currently considers domain dependent semantic classes in the extraction rule induction. Another direction of our future work is to incorporate more background knowledge. Very often, users may already have some background information or knowledge about the domain. For example, user may have knowledge about the length information of the attribute items and the information of the key attribute item. We hope to develop a mechanism that can consider such information in the wrapper induction or wrapper adaptation process. The mechanism must be robust enough so that users can incorporate their domain knowledge into the system easily. A

rule-based system may be worthy for investigation.

## Bibliography

- [1] B. Adelberg. NoDoSE: a tool for semi-automatically extracting unstructured and semistructured data from text databases. In *SIGMOD 1994 Proceedings ACM SIGMOD International Conference on Management of Data*, pages 283-284, June 1993.
- [2] D. Angluin. Inference of recognizable languages. *Journal of ACM*, 29(3):744-765, July 1982.
- [3] E. Brill. Some advances in rule-based parsing of speech. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 722-727, July 1991.
- [4] S. Brin. Extracting patterns and relations from the World Wide Web. *Proceedings of the International World Wide Web Conference*, pages 172-183, June 1998.
- [5] M. Califf and R. Mooney. Rule-based extraction of relationships from text for information extraction. In *Proceedings of the Thirtieth Annual Conference on Artificial Intelligence*, pages 112-121, July 1999.
- [6] R. Carrasco, J. Oncina, and J. Alcaraz. On the complexity of regular tree languages. *Mathematics*, 2017, 2017, 2017.
- [7] C. H. Chang and N. C. Lu. Pattern-based information extraction and pattern discovery. In *Proceedings of the International Conference on World Wide Web*, pages 481-491, May 2000.

# Bibliography

- [1] B. Adelberg. NoDoSE—a tool for semi-automatically extracting structured and semistructured data from text documents. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 283–294, June 1998.
- [2] D. Angluin. Inference of reversible language. *Journal of the ACM*, 29(3):741–765, July 1982.
- [3] E. Brill. Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 722–727, July 1994.
- [4] S. Brin. Extracting patterns and relations from the World Wide Web. In *Proceedings of the International Workshop on the Web and Databases*, pages 172–183, June 1998.
- [5] M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 328–334, July 1999.
- [6] R. Carrasco, J. Oncina, and J. Calera-Rubio. Stochastic inference of regular tree language. *Machine Learning*, 44(1/2):185–197, July 2001.
- [7] C. H. Chang and S. C. Lui. IEPAD: information extraction based on pattern discovery. In *Proceedings of the Tenth International Conference on World Wide Web*, pages 681–688, May 2001.

- [8] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of the Sixteenth Meeting of the Information Processing Society of Japan*, pages 7–18, October 1994.
- [9] H. L. Chieu and H. T. Ng. A maximum entropy approach to information extraction from semi-structured and free text. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 786–791, July 2002.
- [10] F. Ciravegna.  $(LP)^2$  an adaptive algorithm for information extraction from Web-related texts. In *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining held in conjunction with the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1251–1256, August 2001.
- [11] W. W. Cohen, M. Hurst, and L. S. Jensen. A flexible learning system for wrapping tables and lists in HTML documents. In *Proceedings of the Eleventh International World Wide Web Conference*, pages 232–241, May 2002.
- [12] W. W. Cohen, M. Hurst, and Lee. Jensen. A flexible learning system for wrapping tables and lists in html documents. In *Proceedings of the Eleventh International World Wide Web Conference*, pages 232–241, May 2002.
- [13] V. Crescenzi, G. Mecca, and P. Merialdo. ROADRUNNER: Towards automatic data extraction from large web sites. In *Proceedings of the 27th Conference on Very Large Data Bases*, pages 109–118, September 2001.
- [14] Defense Advanced Research Projects Agency. *Proceedings of the Sixth DARPA Message Understanding Conference (MUC-6)*. Morgan Kaufmann Publisher, Inc., 1995.

- [15] Defense Advanced Research Projects Agency. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. Morgan Kaufmann Publisher, Inc., 1998.
- [16] R. B. Doorenbos, O. Etzioni, and D. S. Weld. A scalable comparison-shopping agent for the World-Wide Web. In *Proceedings of the First International Conference on Autonomous Agents*, pages 39–48, February 1997.
- [17] E. M. Voorhees and L. P. Buckland. *The Eleventh Text REtrieval Conference (TREC 2002)*. National Institute of Standards and Technology, November 2002.
- [18] T. Eliassi-Rad and J. Shavlik. A theory-refinement approach to information extraction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 130–137, June 2001.
- [19] D. W. Embley, D. Campbell, R. Smith, and S. Liddle. A conceptual-modeling approach to extracting data from the Web. In *Proceedings of the 17th International Conference on Conceptual Modeling (ER'98)*, pages 78–91, November 1998.
- [20] D. W. Embley, Y. S. Jiang, and Y. K. Ng. Record-boundary discovery in Web documents. In *Proceedings of 1999 ACM SIGMOD International Conference on Management of Data*, pages 467–478, June 1999.
- [21] D. W. Embley, D. M., Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y. K. Ng, and R. D. Smith. Conceptual-model-based data extraction from multiple-record Web. *Data and Knowledge Engineering*, 31(3):227–251, November 1999.
- [22] D. Freitag. Information extraction from HTML: Application of a general machine learning approach. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 517–523, July 1998.



- [23] D. Freitag and N. Kushmerick. Boosted wrapper induction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 577–583, July 2000.
- [24] D. Freitag and A. McCallum. Information extraction with HMMs and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 31–36, July 1999.
- [25] J. C. French, A. L. Powell, and E. Schulman. Applications of approximate word matching in information retrieval. In *Proceedings of the Sixth International Conference on Information and Knowledge Management*, pages 9–15, November 1997.
- [26] P. Garcia and E. Vidal. Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):920–925, September 1990.
- [27] R. Ghani and R. Jones. A comparison of efficacy and assumptions of bootstrapping algorithms for training information extraction systems. In *Proceedings of the Workshop on Linguistic Knowledge Acquisition and Representation: Bootstrapping Annotated Language Data held in conjunction with the Third International Conference on Language Resources and Evaluation*, June 2002.
- [28] T. Goan, N. Benson, and O. Etzioni. A grammar inference algorithm for the World Wide Web. In *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access*, pages 41–48, March 1996.
- [29] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
- [30] P. B. Golgher and A. S. da Silva. Bootstrapping for example-based data extraction. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 371–378, November 2001.

- [31] R. C. Gonzalez and M. G. Thomason. *Syntactic pattern recognition - An introduction*. Addison-Wesley Publishing Company, Inc., 1978.
- [32] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge, 1997.
- [33] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo. Extracting semistructured information from the Web. In *Workshop on Management of Semistructured Data*, 1997.
- [34] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, 1963.
- [35] C. Hsu and M. Dung. Generating finite-state transducers for semi-structured data extraction from the Web. *Journal of Information Systems, Special Issue on Semistructured Data*, 23(8):521–538, November 1998.
- [36] C. Knoblock, S. Minton, J. Ambite, N. Ashish, J. Modi, I. Muslea, A. Philpot, and S. Tejada. Modeling Web sources for information integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 211–218, July 1998.
- [37] N. Kushmerick. Regression testing for wrapper maintenance. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 74–79, July 1999.
- [38] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1/2):15–68, April 2000.
- [39] N. Kushmerick and B. Thomas. Adaptive information extraction: Core technologies for information agents. In *Intelligent Information Agents R&D In Europe: An AgentLink Perspective*, pages 79–103, 2002.
- [40] K. Lerman and S. Minton. Learning the common structure of data. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 609–614, August 2000.

- [41] S. J. Lim and Y. K. Ng. An automated approach for retrieving hierarchical data from HTML tables. In *Proceedings of the Eighth International Conference on Information and Knowledge Management*, pages 466–474, November 1999.
- [42] L. Liu, C. Pu, and W. Tang. WebCQ - Detecting and delivering information changes on the Web. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, pages 512–519, November 2000.
- [43] L. Miclet. *Structural methods in pattern recognition*. Springer-Verlag, 1986.
- [44] G. Miller. Wordnet: A lexical database for english. *Communication of the ACM*, 38(11):29–41, November 1995.
- [45] I. Muslea, S. Minton, and C. Knoblock. Selective sampling with redundant views. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 621–626, August 2000.
- [46] I. Muslea, S. Minton, and C. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114, March 2001.
- [47] U. Y. Nahm and R. J. Mooney. Using information extraction to aid the discovery of prediction rules from text. In *KDD-2000 Workshop on Text Mining*, August 2000.
- [48] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 1044–1049, July 1999.
- [49] K. Seymore, A. McCallum, and R. Rosenfeld. Learning hidden markov model structure for information extraction. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 37–42, July 1999.

- [50] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1/3):233–272, February 1999.
- [51] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.
- [52] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.
- [53] H. L. Wang, S. H. Wu, I. C. Wang, C. L. Sung, W. L. Hsu, and W. K. Shih. Semantic search on internet tabular information extraction for answering queries. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, pages 243–249, November 2000.
- [54] J. Wang and F. H. Lochovsky. Data extraction and label assignment for Web databases. In *Proceedings of the Twelfth International World Wide Web Conference*, pages 187–196, May 2003.
- [55] T. L. Wong and W. Lam. Adapting information extraction knowledge for unseen Web sites. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 506–513, December 2002.
- [56] T. L. Wong, W. Lam, and W. Wang. Beyond supervised learning of wrappers for extracting information from unseen Web sites. In *Proceedings of the Fourth International Conference on Intelligent Data Engineering and Automated Learning*, March 2003.
- [57] J. Y. Yang, E. S. Lee, and J. G. Choi. A shopping agent that automatically constructs wrappers for semi-structured online vendors. In *The Second International Conference on Intelligent Data Engineering and Automated Learning*, pages 368–373, December 2000.
- [58] M. Young-Lai and F. W. M. Tompa. Stochastic grammatical inference of text database structure. *Machine Learning*, 40(2):111–137, August 2000.

# Appendix A

## Detailed Performance of Wrapper Induction for Book Domain

Page label	<i>Title</i>		<i>Author</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	92.0	92.0	92.0	95.8	92.0	95.8
P3	96.0	96.0	96.0	96.0	96.0	96.0
P4	100.0	100.0	100.0	100.0	100.0	100.0
P5	100.0	100.0	100.0	100.0	100.0	100.0

Table A.1: Performance of our wrapper induction approach for each testing page in the Web sites S1 (P and R refer to precision and recall respectively.)

Page label	<i>Title</i>		<i>Author</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	82.8	100.0	100.0	66.7	100.0	100.0
P3	88.9	100.0	100.0	79.2	100.0	100.0
P4	92.3	100.0	100.0	58.3	100.0	100.0
P5	100.0	100.0	100.0	62.5	91.7	100.0

Table A.2: Performance of our wrapper induction approach for each testing page in the Web sites S2 (P and R refer to precision and recall respectively.)

Page label	<i>Title</i>		<i>Author</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	100.0	100.0	100.0	100.0
P3	100.0	100.0	100.0	100.0	100.0	100.0

Table A.3: Performance of our wrapper induction approach for each testing page in the Web sites S3 (P and R refer to precision and recall respectively.)

Page label	<i>Title</i>		<i>Author</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	56.8	100.0	100.0	100.0
P3	100.0	100.0	96.0	96.0	100.0	100.0
P4	100.0	100.0	88.9	96.0	100.0	100.0
P5	96.0	96.0	96.0	96.0	100.0	100.0

Table A.4: Performance of our wrapper induction approach for each testing page in the Web sites S4 (P and R refer to precision and recall respectively.)

Page label	<i>Title</i>		<i>Author</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	100.0	100.0	70.0	70.0
P3	100.0	100.0	100.0	100.0	70.0	70.0
P4	100.0	100.0	100.0	100.0	90.0	90.0
P5	100.0	100.0	100.0	100.0	50.0	50.0
P6	100.0	100.0	100.0	100.0	80.0	80.0
P7	100.0	100.0	100.0	100.0	80.0	80.0
P8	100.0	100.0	80.0	100.0	70.0	70.0
P9	100.0	100.0	100.0	100.0	80.0	80.0
P10	100.0	100.0	100.0	100.0	90.0	90.0

Table A.5: Performance of our wrapper induction approach for each testing page in the Web sites S5 (P and R refer to precision and recall respectively.)

Page label	<i>Title</i>		<i>Author</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	100.0	100.0	100.0	100.0
P3	100.0	100.0	100.0	100.0	100.0	100.0
P4	100.0	100.0	100.0	100.0	100.0	100.0
P5	100.0	96.0	100.0	96.0	100.0	96.0

Table A.6: Performance of our wrapper induction approach for each testing page in the Web sites S6 (P and R refer to precision and recall respectively.)

Page label	<i>Title</i>		<i>Author</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	90.0	100.0	95.0	95.0
P3	100.0	100.0	95.0	100.0	95.0	95.0
P4	100.0	100.0	90.0	100.0	100.0	100.0
P5	100.0	100.0	95.0	100.0	100.0	100.0
P6	100.0	100.0	100.0	100.0	95.0	95.0

Table A.7: Performance of our wrapper induction approach for each testing page in the Web sites S7 (P and R refer to precision and recall respectively.)

Page label	<i>Title</i>		<i>Author</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	100.0	100.0	85.0	85.0
P3	100.0	100.0	100.0	100.0	100.0	100.0
P4	100.0	100.0	100.0	100.0	100.0	100.0
P5	100.0	100.0	100.0	100.0	80.0	80.0

Table A.8: Performance of our wrapper induction approach for each testing page in the Web sites S8 (P and R refer to precision and recall respectively.)

Page label	<i>Title</i>		<i>Author</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	100.0	100.0	100.0	100.0
P3	100.0	100.0	100.0	100.0	100.0	100.0
P4	100.0	100.0	100.0	100.0	100.0	100.0
P5	100.0	100.0	100.0	100.0	100.0	100.0
P6	100.0	100.0	100.0	100.0	100.0	100.0
P7	100.0	100.0	100.0	100.0	100.0	100.0
P8	100.0	100.0	100.0	100.0	100.0	100.0
P9	100.0	100.0	100.0	100.0	100.0	100.0
P10	100.0	100.0	100.0	100.0	100.0	100.0
P11	100.0	100.0	100.0	100.0	100.0	100.0
P12	100.0	100.0	100.0	100.0	100.0	100.0
P13	100.0	100.0	100.0	100.0	100.0	100.0
P14	100.0	100.0	100.0	100.0	100.0	100.0

Table A.9: Performance of our wrapper induction approach for each testing page in the Web sites S9 (P and R refer to precision and recall respectively.)

Page label	<i>Title</i>		<i>Author</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	100.0	100.0	100.0	100.0
P3	100.0	100.0	100.0	100.0	100.0	100.0
P4	100.0	100.0	100.0	100.0	100.0	100.0
P5	100.0	100.0	100.0	100.0	100.0	100.0
P6	100.0	100.0	100.0	100.0	100.0	100.0
P7	100.0	100.0	100.0	100.0	100.0	100.0
P8	100.0	100.0	100.0	100.0	100.0	100.0
P9	100.0	100.0	100.0	100.0	100.0	100.0
P10	100.0	100.0	100.0	100.0	100.0	100.0
P11	100.0	100.0	100.0	100.0	100.0	100.0

Table A.10: Performance of our wrapper induction approach for each testing page in the Web sites S10 (P and R refer to precision and recall respectively.)

Page label	<i>Title</i>		<i>Author</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	100.0	90.0	100.0	100.0
P3	100.0	100.0	100.0	100.0	100.0	100.0
P4	100.0	100.0	100.0	100.0	100.0	100.0
P5	100.0	100.0	100.0	100.0	100.0	100.0
P6	100.0	100.0	100.0	100.0	100.0	100.0
P7	100.0	100.0	100.0	100.0	100.0	100.0
P8	100.0	100.0	100.0	100.0	100.0	100.0
P9	100.0	100.0	100.0	100.0	100.0	100.0
P10	70.0	70.0	100.0	90.0	100.0	100.0
P11	90.0	90.0	100.0	100.0	100.0	100.0

Table A.11: Performance of our wrapper induction approach for each testing page in the Web sites S11 (P and R refer to precision and recall respectively.)



## Appendix B

# Detailed Performance of Wrapper Induction for Consumer Electronic Appliance Domain

Page label	<i>Model Number</i>		<i>Description</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	100.0	100.0	100.0	100.0
P3	100.0	100.0	100.0	100.0	100.0	100.0
P4	100.0	100.0	100.0	100.0	100.0	100.0
P5	100.0	100.0	100.0	100.0	100.0	100.0
P6	100.0	100.0	100.0	100.0	100.0	100.0
P7	100.0	100.0	100.0	100.0	100.0	100.0
P8	100.0	100.0	100.0	100.0	100.0	100.0
P9	100.0	100.0	100.0	100.0	100.0	100.0
P10	100.0	100.0	100.0	100.0	100.0	100.0

Table B.1: Performance of our wrapper induction approach for each testing page in the Web sites S12 (P and R refer to precision and recall respectively.)

Page label	<i>Model Number</i>		<i>Description</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	90.0	90.0	100.0	100.0
P3	100.0	100.0	100.0	100.0	100.0	100.0
P4	100.0	80.0	100.0	100.0	100.0	100.0
P5	100.0	100.0	50.0	50.0	100.0	100.0
P6	100.0	100.0	100.0	100.0	100.0	100.0
P7	100.0	100.0	100.0	100.0	100.0	100.0
P8	100.0	100.0	100.0	100.0	100.0	100.0
P9	100.0	100.0	100.0	100.0	100.0	100.0
P10	100.0	100.0	100.0	100.0	100.0	100.0

Table B.2: Performance of our wrapper induction approach for each testing page in the Web sites S13 (P and R refer to precision and recall respectively.)

Page label	<i>Model Number</i>		<i>Description</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	100.0	100.0	100.0	100.0
P3	100.0	100.0	100.0	100.0	100.0	100.0
P4	100.0	100.0	100.0	100.0	100.0	100.0
P5	100.0	100.0	100.0	100.0	100.0	100.0
P6	100.0	100.0	100.0	100.0	100.0	100.0
P7	100.0	100.0	100.0	100.0	100.0	100.0
P8	100.0	100.0	100.0	100.0	100.0	100.0

Table B.3: Performance of our wrapper induction approach for each testing page in the Web sites S14 (P and R refer to precision and recall respectively.)

Page label	<i>Model Number</i>		<i>Description</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	100.0	100.0	100.0	100.0
P3	100.0	95.0	100.0	95.0	100.0	94.1
P4	100.0	100.0	100.0	100.0	100.0	100.0
P5	90.0	90.0	100.0	100.0	100.0	100.0
P6	90.0	90.0	100.0	100.0	100.0	100.0

Table B.4: Performance of our wrapper induction approach for each testing page in the Web sites S15 (P and R refer to precision and recall respectively.)

Page label	<i>Model Number</i>		<i>Description</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	90.0	100.0	90.0	90.0
P3	100.0	100.0	100.0	100.0	100.0	100.0
P4	100.0	100.0	83.3	100.0	100.0	100.0
P5	100.0	100.0	100.0	100.0	100.0	100.0
P6	100.0	100.0	90.0	90.0	100.0	100.0
P7	100.0	100.0	80.0	100.0	100.0	100.0
P8	100.0	100.0	100.0	100.0	100.0	100.0
P9	90.0	100.0	90.0	100.0	90.0	100.0
P10	100.0	100.0	70.0	87.5	100.0	100.0
P11	100.0	100.0	100.0	100.0	100.0	100.0
P12	100.0	100.0	90.0	100.0	100.0	100.0

Table B.5: Performance of our wrapper induction approach for each testing page in the Web sites S16 (P and R refer to precision and recall respectively.)

Page label	<i>Model Number</i>		<i>Description</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	90.0	100.0	100.0	100.0
P3	100.0	100.0	100.0	100.0	100.0	100.0
P4	100.0	100.0	66.7	100.0	100.0	100.0
P5	100.0	100.0	100.0	100.0	100.0	100.0
P6	100.0	100.0	100.0	100.0	100.0	100.0
P7	100.0	100.0	100.0	100.0	100.0	100.0
P8	100.0	100.0	100.0	100.0	100.0	100.0
P9	100.0	100.0	100.0	100.0	100.0	100.0
P10	100.0	100.0	100.0	100.0	100.0	100.0
P11	100.0	100.0	100.0	100.0	100.0	100.0
P12	100.0	100.0	100.0	100.0	100.0	100.0
P13	100.0	100.0	100.0	100.0	100.0	100.0

Table B.6: Performance of our wrapper induction approach for each testing page in the Web sites S17 (P and R refer to precision and recall respectively.)

Page label	<i>Model Number</i>		<i>Description</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	55.6	35.7	55.6	100.0	55.6
P3	100.0	41.7	35.7	41.7	100.0	41.7
P4	100.0	100.0	62.2	100.0	100.0	100.0
P5	100.0	100.0	59.3	94.1	100.0	100.0
P6	100.0	50.0	33.3	50.0	100.0	50.0
P7	100.0	81.8	47.4	81.8	100.0	81.8
P8	100.0	100.0	54.5	100.0	100.0	100.0
P9	100.0	100.0	46.2	100.0	100.0	100.0
P10	100.0	100.0	52.6	100.0	100.0	100.0
P11	100.0	83.3	50.0	83.3	100.0	83.3
P12	100.0	100.0	66.7	100.0	100.0	100.0

Table B.7: Performance of our wrapper induction approach for each testing page in the Web sites S18 (P and R refer to precision and recall respectively.)

Page label	<i>Model Number</i>		<i>Description</i>		<i>Price</i>	
	P (%)	R (%)	P (%)	R (%)	P (%)	R (%)
P2	100.0	100.0	47.4	47.4	100.0	100.0
P3	100.0	97.1	0.0	0.0	100.0	96.7
P4	100.0	97.7	97.6	95.3	100.0	95.8

Table B.8: Performance of our wrapper induction approach for each testing page in the Web sites S19 (P and R refer to precision and recall respectively.)



CUHK Libraries



004076662