

Wrapster: Semi-Automatic Wrapper Generation for Semi-Structured Websites

by

Gabriel Zaccak

Submitted to the Department of Electrical Engineering and Computer
Science

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2007

© Massachusetts Institute of Technology 2007. All rights reserved.

Author
Department of Electrical Engineering and Computer Science
May 23, 2007

Certified by
Boris Katz
Principal Research Scientist
Thesis Supervisor

Accepted by
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Wrapster: Semi-Automatic Wrapper Generation for Semi-Structured Websites

by

Gabriel Zaccak

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2007, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Many information sources on the Web are semi-structured; hence there is an opportunity for automatic tools to process and extract their information for easy access through a uniform interface language. Wrapper generation is the creation of wrappers which contains scripts that extract and integrate data from data sources, mostly from Web data sources due to the large amount of data available on the World Wide Web. Despite ongoing efforts to automate the process of wrapper generation, wrappers frequently break due to formatting and layout changes in data sources.

This thesis presents Wrapster, a new system that semi-automatically generates wrappers for semi-structured Web sources, improves wrapper robustness, and eliminates the need for programming skills and, to a large extent, the process of script creation. Wrapster's novel component is the repairing module that constantly checks if any wrapper script has failed and repairs the failing wrapper's script using stored extracted instances. In addition, Wrapster provides an interactive Web user interface to control the wrapper generation process, edit the generated wrappers, and test their scripts. Wrapster is being tested on the START Question Answering system; however, it is a generic tool to be used by any QA system that uses the Web as its knowledge base.

Thesis Supervisor: Boris Katz

Title: Principal Research Scientist

Acknowledgments

With the deepest gratitude I wish to thank my advisor, Dr. Boris Katz, for his inspirational ideas, insightful feedback, and constructive criticism.

I would also like to acknowledge and express my gratitude to my colleagues and friends Wisam Dakka, Fadi Biadisy, Luke Zettlemoyer, Ali Mohammad, Federico Mora, Yuan Shen, Sue Felshin, Raj Singh, Igor Malioutov, Gregory Marton, and Gary Borchardt for dedicating their time to discuss new ideas, proofreading my thesis and providing constructive feedback. Without your support and comments some of the ideas in this thesis would not have come to fruition.

I am grateful to my academic advisor, Prof. Randall Davis for supporting me and guiding me since my arrival to CSAIL.

I also would like to thank my previous mentors Prof. Michael Elhadad, and Moisei Rabinovitch for teaching me and giving me the opportunities to learn, and Sofia Yoshimoto for her support and encouragement through the difficult times at MIT.

I am most indebted to my family. Thank you for your concern, encouragement, and support.

This is dedicated to you!

Contents

1	Introduction	13
1.1	Motivation	13
1.2	Applications	17
1.2.1	Price Engines	17
1.2.2	Question Answering	18
1.2.3	News Tracker	19
1.2.4	Wrapper Generation for End-Users	21
1.3	Basic System Outline and Assumptions	21
1.3.1	Components	21
1.4	Thesis Overview	24
2	Related Work	25
2.1	Wrapper Generation	25
2.1.1	Wrapper Induction	25
2.1.2	Automatic Wrapper Generation	27
2.1.3	Interactive Wrapper Generation	28
2.1.4	Wrapper Generation for End-Users	28
2.2	Tree Data Structures and Algorithms	29
2.2.1	Tree Edit Distance	30
2.3	Classification	31
3	Template Creation	33
3.1	Tree Mapping	33

3.2	Region Identification	34
3.3	Region Clustering	37
3.4	Classification	38
3.4.1	Features	38
3.5	Annotator Tool	39
3.5.1	Automatic Annotation	40
4	Wrapper Induction	45
4.1	Script Generation	45
4.1.1	Pattern Induction	46
4.1.2	Region Focuses	46
5	Semantic Annotation	47
6	Wrapster User Interface	51
7	Repair Module	55
8	Experiments and Results	59
8.1	Classification	60
8.2	Comprehensive Evaluation	60
9	Conclusion and Future Work	65
9.1	Contributions	65
9.2	Future Work	66

List of Figures

1-1	<i>Garden State</i> Web page from IMDB site	15
1-2	<i>300</i> movie Web page from IMDB site	16
1-3	Froogle’s result page for Larry Wasserman’s <i>All of Statistics</i> book. . .	18
1-4	START’s answer for the question “What is the capital of India?” . .	20
3-1	Tree mapping of <i>Fahrenheit 9/11</i> movie Web page aligned with <i>Garden State</i> movie Web page from IMDB site	35
3-2	Screen shot of <i>Garden State</i> movie Web page from IMDB site	36
3-3	Annotator tool screen shot of the List data page. Selected options appear at the top and all selected training data items appear below; the first two items can be seen here.	41
3-4	Annotator tool screen shot of the List data page, zooming on one training data item.	42
3-5	Annotator tool print-screen of the Upload page.	42
3-6	Annotator tool DTD of training data XML input format.	43
3-7	Annotator tool screen shot of the Manual annotate data page. The above training sample is a comparison between a “Trivia” and an “Awards” region. The Annotator Tool decided that those regions do not match since the classification score is -1.86 outside of the threshold range.	44
3-8	Annotator tool screen shot of the Batch automatic annotate request page	44

5-1	United States population region extract from The World Factbook Web site (a) and the corresponding HTML source (b).	48
5-2	<i>The Good Shepherd</i> movie director region extract from the IMDb Web site (a) and the corresponding HTML source (b).	48
6-1	Wrapster web user interface main page	52
6-2	IMDb <i>Sean Connery</i> Web page opened using Wrapster HTML proxy	52
6-3	Wrapster Web user interface editing the IMDb wrapper	53
7-1	Screen shot of <i>Garden State</i> movie Web page from 2006 IMDB site .	56
7-2	Screen shot of <i>Garden State</i> movie Web page from 2007 IMDB site .	56
7-3	<i>Garden State</i> movie HTML code from the IMDb site (a) 2006 layout (b) 2007 layout.	57

List of Tables

1.1	Wrapster input examples for IMDb and Weather Channel.	22
8.1	Evaluation dataset.	59
8.2	SVM training data	60
8.3	SVM feature sets and performance on all training data	61
8.4	SVM feature sets and performance on IMDB movie training data . .	61
8.5	Wrapper generation content discovery	62
8.6	Wrapper generation whole evaluation on the updated Web sites on April 23rd	62

Chapter 1

Introduction

Semi-structured resources constitute many of the available information sources, especially on the Web and this trend is likely to increase. Information extraction from semi-structured data uses wrappers. A wrapper is an automatic method for information extraction from a semi-structured source, and “wrapper induction” is the field of research whose goal is to generate a wrapper by generalizing over a set of examples [18]. Although semi-structured sources possess syntactic regularities, it is not trivial to process and extract information from those sources. The main issues that wrapper generation systems deal with are scalability and flexibility. Scalability concerns the multiplicity variations of layout and format between many sites, and flexibility concerns the robustness of the wrappers to frequent layout and format change.

1.1 Motivation

To better illustrate the problem, let’s look at an extract of the Web page from the Internet Movie Database¹ (IMDb) for the movie *Garden State* (Figure 1-1). The page contains a top panel, a left panel, and a main section. The top panel has the IMDb logo and some links to the site services. The left panel consists of search boxes and a list of links to related and non-related Web pages. The main section

¹<http://www.imdb.com>

contains the movie’s properties such as the director, the genre, and the actors. A wrapper generation system needs to identify the regions of relevant information on the page, associate those regions with semantic values, and create a wrapper for the data resource that represents a general form of the page. In this example, the page is in HTML format and is populated from a database of movies and their properties. It is generated by filling a movie template with the movie properties. We can notice that properties are in different format such as a single value for year-of-release, a list of values for genre, and a table for the cast. The wrapper works as a smart index of such pages and is able to identify the properties and extract their values for use by information extraction systems. Automating this task is not trivial, even though the page is a simple structured Web page. For example, how can a wrapper generation system identify that the picture under the movie’s title is the movie poster and not the picture of the director when there is no textual indicator. The same difficulty applies to the movie title and year of release properties. In other cases where there are textual indicators such as the director property, it is easier to identify the information and its meaning. All wrapper generation systems’ goal is to be able to extract this information even if the textual clue “Directed by” and the property location change. After the wrapper has been created, it can identify the movie properties for any movie page from the IMDb site and, for example, should identify *Zack Snyder* as the director of the movie *300 (2006)* even though the layout has changed (Figure 1-2).

Existing wrapper generation systems for structured sources and for semi-structured sources do not make use of the global document structure and are dependent on a large training set. Newer systems have focused on building end-user wrapper induction gadgets that reduced the amount of training data needed, leveraged the hierarchical structure of HTML, and learned patterns by aligning multiple example trees using a *tree edit distance* algorithm. Wrapster has built on these ideas; it represents the full semantics of the page, adds automatic testing and repair facilities, and partly automates the labeling process.

This thesis extends previous work in several significant ways for automated wrapping of semi-structured resources, and presents an underlying language for the setup

Earth's Biggest Movie Database™

NOW PLAYING

MOVIE / TV NEWS

MY MOVIES

DVD / VIDEO

IMDb TV

MESSAGE BOARDS

SHOWTIMES & TICKETS

[Home](#) | [Top Movies](#) | [Photos](#) | [Independent Film](#) | [GameBase](#) | [Browse](#) | [Help](#)

[more tips](#)

[IMDb](#) > [300](#) (2006)

[photo gallery](#)
[message boards](#)
[view trailer](#)
[add to My Movies](#)
[IMDb Pro details](#)

Directed by [Zack Snyder](#)

Writing credits ([WGA](#))
[Zack Snyder](#) (screenplay) & [Kurt Johnstad](#) (screenplay) ...
[more](#)

[Register](#) or [login](#) to rate this title

★★★★★☆☆☆☆☆
 User Rating: 8.1/10 ([61,923 votes](#))
[more](#)

[Quicklinks](#)

[Top Links](#)
[- trailers](#)
[- full cast and crew](#)
[- trivia](#)
[- official sites](#)
[- memorable quotes](#)

[Overview](#)
[- main details](#)
[- combined details](#)
[- full cast and crew](#)
[- company credits](#)
[- tv schedule](#)

[Promotional](#)
[- taglines](#)
[- trailers](#)
[- posters](#)
[- photo gallery](#)

[Awards & Reviews](#)
[- user comments](#)
[- external reviews](#)
[- newsgroup reviews](#)
[- awards](#)
[- user ratings](#)
[- recommendations](#)
[- message board](#)

[Plot & Quotes](#)

300 (2006)

[more](#)

Showtimes & Tickets

[\(Register to personalize\)](#)

Release Date: 9 March 2007 (USA) [more](#) [view trailer](#)

Genre: [Action](#) / [History](#) / [War](#) [more](#)

Tagline: Prepare for glory! [more](#)

Plot Outline: Based on Frank Miller's graphic novel about the Battle of Thermopylae in 480 B.C. [more](#)

Plot Keywords: [Based On Comic Book](#) / [Bare Butt](#) / [Severed Leg](#) / [Gore](#) / [Betrayal](#) [more](#)

User Comments: Stupendous movie making!! A masterpiece!!! [more](#)

Cast

(Cast overview, first billed only)

	Gerard Butler	... King Leonidas
	Lena Headey	... Queen Gorgo

Figure 1-2: *300* movie Web page from IMDB site

and extension of this system. Given a set of Web pages that describe related objects, and the names of those objects, Wrapster incrementally forms a common template that captures relevant slots for extraction from the pages by comparing the related Web pages and using a metric of *tree edit distance*. Next, Wrapster labels relevant regions using a trained classifier with region content and its context as features. Then, the system induces the value type for the discovered properties (e.g. number with possible units of measure, text, named entity, list of values, or table of values). The system uses this type specification and constructs robust scripts to extract type of values for the specified properties. Because wrapper scripts are generated on the basis of the conceptual organization of the semi-structured site, and not tied to specific formatting choices. Therefore, the scripts accommodate small and even medium-sized formatting differences. In addition, the system can be configured to generate scripts that identify named entities within value strings, or that standardize accessed values in a suitable manner for use in answering complex questions within a question answering system.

1.2 Applications

Price engines, question answering systems, news trackers and web manipulating tools are several of many applications which require robust wrapper generation systems.

1.2.1 Price Engines

Price engines, such as Froogle² and PriceGrabber³, have become very popular in the past few years. A price engine is a search engine that finds the best prices for specific items such as books and computer hardware. It searches, indexes all items from retail sites on the Web, and then identifies similar items for comparative purposes. For example, if a user searches for the price of Larry A. Wasserman's book, *All of Statistics*, the above price engines should provide a list of online retailers that offer

²<http://www.google.com>

³<http://www.google.com/products>

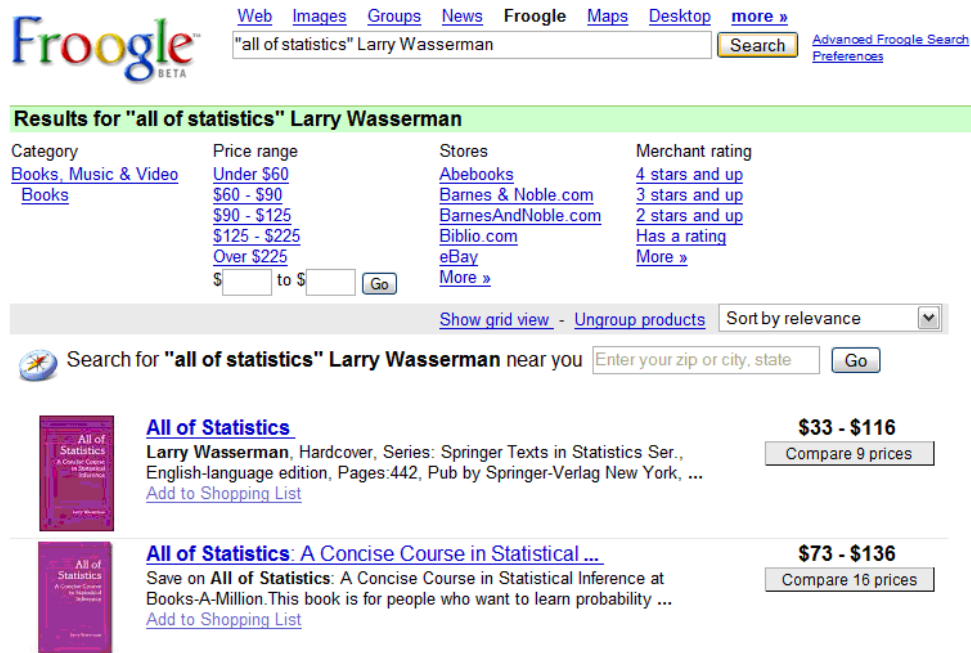


Figure 1-3: Froogle's result page for Larry Wasserman's *All of Statistics* book.

the book for sale (Figure 1-3). They also provide advanced search features, such as sorting the items by price or by merchant rating. Price engines use manually created or semi-automatic wrappers that require a lot of maintenance and technical skills. They are not flexible enough to support format changes and lack automatic repair facilities since they usually look for specific keywords in the searched page. Wrapster is robust to such changes, identifies the item's properties for indexing by price engines, and provides automatic wrapper repairing.

1.2.2 Question Answering

Given a collection of documents question answering systems retrieve answers to questions posed in natural language. They return precise answers, as opposed to search engines that return lists of whole documents. Complex question answering require the integration of nuggets of information from multiple data resources. An answer to the complex question "What is the population of the capital of India?" involves two queries. First, the system acquires the value for the capital of India and then queries for the population of the acquired value (Figure 1-4). Therefore, question answering

systems can benefit from unified access to information residing on the Web. One such system is START⁴, a high-precision question answering system, which uses the Web as part of its knowledge base [31, 28, 29]. START uses a system called Omnibase [30] as its uniform access interface to the Web. The main portion of Omnibase is a database of scripts that extract information from various websites. Omnibase uses an object property value relational model, and the execution of a script generates the value of a predefined property from a predefined site. A wrapper for a site constitutes all the scripts that belong to it, with their corresponding property names. These scripts are manually written and include low-level regular expressions that extract information. Moreover, these scripts are tailored to the specific sources, and are not robust to format and layout changes. An effort to tackle these issues led to the creation of Hap-Shu [44], a system for locating information in an HTML document, which abstracts the scripts from low-level regular expressions and HTML-specific vocabulary to text landmarks. Hap-Shu introduced a new set of commands that selects information based on context and type. This tool has facilitated the manual creation of wrappers but it fails with format changes, such as paraphrasing, and such layout changes as a change in content representation. In addition, this tool still requires programming skills and the use of regular expressions inside scripts for user-presentation purposes.

Wrapster tackles these issues and eliminates the need for programming skills and, to a large extent, the process of scripts creation. It has been successfully deployed on START; however it is a generic tool to be used by any information access system that uses the Web as its knowledge base.

1.2.3 News Tracker

News trackers such Google News, and Event Monitor⁵ need a uniform way to access news stories from varied online news Web sites.

Google News uses wrappers to extract the title name, abstract (if one exists) and

⁴<http://start.csail.mit.edu>

⁵<http://www.eventmonitor.com>

START's reply

==> What is the population of the capital of India?

I know that India's capital is New Delhi, India (source: START KB).

Using this information, I determined what is the population of New Delhi, India:

The population of New Delhi, India is 7,174,755.

Source: START KB

- ♦ [Go back to the START dialog window.](#)

Figure 1-4: START's answer for the question "What is the capital of India?"

text of news articles from many online newspapers. It then clusters the news articles by similar topic and presents the clustered articles on its Web pages ordered by topic. Each cluster is presented with the title of the highest ranked article, a brief extract of the text and the links to all news sources that have published the same story.

Many startup companies, such as Event Monitor, have emerged recently to provide investment management firms and trading operations with an analysis and recommendations about the market and their investments. Their ultimate goal is to provide smart automatic agents that correctly invest the company's resources and create more value for the company. However this goal is not yet achieved. The state-of-the-art systems are recommendation systems that track business news articles and generate warnings or risk factors about each of the company's investments. For example if a company investment is being sued for patent infringement, the system might recommend examining a particular investment again and send a message to the management for immediate attention. Such systems are built on top of wrappers that extract relevant information from business newspapers and online resources. The robustness of wrappers is crucial for such companies to be able to receive accurate analysis.

With Wrapster, such systems can easily add more data sources without the need of programming skills, and generate wrappers that are robust to layout and format changes. Wrapster’s repairing module constantly checks if any wrapper’s script is failing and immediately fixes the failing scripts using the stored extracted Web page instances for that script.

1.2.4 Wrapper Generation for End-Users

End-user systems, such as Sifter [25] and Threster [22], allow users to enhance their browsing experience with expert tools and provide services such as sorting a list of items on a Web page in an order that is not provided by the site. Usually they are built as browser extensions, and are intuitive to the non-expert user. They require no training but are not robust to format changes and do not make use of global document structure.

1.3 Basic System Outline and Assumptions

The input for Wrapster is the set of objects and their Web page links on a site (Table 1.1). The output is a wrapper that consists of a finite set of scripts and their corresponding semantic labels that represent the properties of the site.

1.3.1 Components

Wrapster consists of the following five modules:

1. Template Creation: This module creates a general template that identifies and marks the relevant information from the pages of the data source. Wrapster incrementally constructs the template the pages share. Wrapster computes the alignment between a small subset of pages using the *tree edit distance* technique [12, 13, 16, 20, 41, 27]. Using the alignment, Wrapster identified and discards the elements in common, and leaves us with the relevant slots for extraction. Then, Wrapster clusters the slots from all the aligned pages using

IMDb (The Internet Movie Database, www.imdb.com)	
<i>La Dolce Vita</i>	http://www.imdb.com/title/tt0053779/
<i>Finding Neverland</i>	http://www.imdb.com/title/tt0308644/
<i>Memoirs of a Geisha</i>	http://www.imdb.com/title/tt0397535/
...	

Weather Channel (www.weather.com)	
<i>Lima, Peru</i>	http://www.weather.com/outlook/travel/businesstraveler/local/PEXX0011?from=search_city
<i>Paris, France</i>	http://www.weather.com/outlook/travel/businesstraveler/local/FRXX0076?from=search_city
<i>Boston, MA</i>	http://www.weather.com/outlook/travel/businesstraveler/local/FRXX0076?from=search_city
...	

Table 1.1: Wrapster input examples for IMDb and Weather Channel.

a trained classifier [45], creates regions of neighboring slots, and identifies data structures such as lists and tables comparing the HTML structural and content similarity. Wrapster only needs a few examples of training data because it uses a variation of active learning technique [19]. Given a new set of unlabeled data it classifies the data and then retrains the classifier expanding the training data with new classified instances that have high confidence classification score. See Chapter 3 for an in-depth discussion of template creation.

2. Wrapper Induction: After identifying the content regions of the the source pages, Wrapster creates the scripts that extract the information from the template created in the previous step. In addition, it creates scripts with different focuses, i.e., a response for question fusion needs to be precise for value comparison while other question types prefer HTML fragments as answers for human-friendly display purposes. The generated scripts comprise the wrapper for the data source. They are written in a high-level description language that abstracts away from HTML-specific vocabulary and low-level regular expressions. As a result, such scripts are robust to format and layout changes, in contrast to regular expressions and high-level tag paths that break easily with such changes and

need constant maintenance. See Chapter 4 for more information on wrapper induction.

3. **Semantic Annotation:** The annotation module attaches semantic labels to the wrappers' scripts. Wrapster labels the regions using a trained classifier with features such as context and type of value. It is important to give meaningful labels to the scripts instead of using randomly generated ones for reasons such as facilitating the annotation process that maps properties to English sentences and paraphrases. See Chapter 5 for more information on semantic annotation.
4. **User Interface:** The graphical user interface displays the wrapper generated in the previous steps and asks the user to approve that the wrapper includes the relevant properties and correct labels. The user can add, delete and modify the regions, change the labels, and then approve the deployment of the wrapper to the online system. This verification step is optional but crucial to high-precision question answering systems such as START. See Chapter 6 for full description of the developed Web user interface.
5. **Repair Module:** The repair module is a background service that constantly verifies the wrapper's correctness. When scripts fail, the system tries to repair the failing scripts if possible or else regenerates the wrapper. Wrapster repairs failing scripts by finding the most similar region on the object's page that matches the stored extracted instances of the script. Wrapster uses the same trained classifier from the template creation module. The verification component deals successfully with frequently updated resources such as The Weather Channel and Yahoo Stocks. Although the scripts were built to be robust to format changes, this component is a safety net in cases of total format changes or word paraphrase changes. See Chapter 7 for more information on the repair module.

1.4 Thesis Overview

Chapter 2 describes related work. Chapters 3 through 8 explain in detail Wrapster's five components. Chapter 9 summarizes the results and contributions of Wrapster, and describes possible future work.

Chapter 2

Related Work

2.1 Wrapper Generation

Wrapper generation is the creation of wrappers which contains scripts that extract and integrate data from data sources, mostly from Web data sources due to the large amount of data available on the World Wide Web. A Wrapper can be seen as a set of specialized programs that extracts data from a data source such as a Web site and manipulates the information into a suitably structured format to enable further processing. Usually after creating a wrapper for a Web site, the wrapper is post-processed and annotated with semantic properties. This process enables the extracted data to be further manipulated by other specialized programs. For example, an answer to “What are the 10 richest countries?” might require the question answering system to request the GDP field of all countries from a country site wrapper, sort the answers from richest to poorest, and present the 10 highest answers.

In this section I will describe the various approaches developed to automate the process of wrapper generation.

2.1.1 Wrapper Induction

Wrapper induction is a method that generates wrappers automatically using machine learning techniques. In wrapper induction, wrappers are generated from a set of ex-

amples by computing a generalization that explains the observations. This process is usually supervised, because it requires expert intervention to label training examples and manually correct exceptions that cannot be generalized by the learning algorithm.

Nikola Kushmerick introduced the term “wrapper induction”. In their WIEN system, Kushmerick et al. [32] used a four-feature wrapper induction: HLRT, which stands for Head, Left, Right, and Tail regular expression patterns of the property to be extracted. Muslea et al. [39] have introduced STALKER, a hierarchical wrapper induction system. STALKER uses greedy-covering, an inductive learning algorithm, which incrementally generates extraction rules from training examples. The rules are represented as linear landmark automata. A linear landmark automaton is a non-deterministic finite automaton where transition between states is allowed if the input string matches the rule for this transition. In each step, STALKER tries to generate new automata for the remaining training examples, until all the positive examples are covered. The system’s output is a simple landmark grammar where each branch corresponds to a learned landmark automaton. Hsu et al. [24] used finite state transducers as their model in their SoftMealy system. In contrast to WIEN, the SoftMealy and STALKER systems’ representations support missing values and variable permutations, and require less training examples.

There are also more general wrapper generation systems, such as RAPIER [7, 8] and WHISK [42] that extract information from unstructured text data. Given a hand-tagged template, these systems learn pattern-matching rules to extract the tagged information in the template.

The bottleneck in all the above systems is the labeling of training examples. Each system uses different wrapper representations and requires technical users to annotate input examples. In addition, it is hard to compare the above systems because of the different underlying representations.

Wrapster addresses representational issues and introduces a general language that describes the wrapper, independently of the system’s model. In addition, it reduces the amount of training and labeled examples and gets us closer to the goal of unsupervised wrapper induction.

2.1.2 Automatic Wrapper Generation

Different approaches have been developed for detail pages and list pages. A detail page is a page that focuses on a single item (see Figure 1-1). A list page is a page that lists several items (see Figure 1-3). Previous work developed separate systems to deal with each data type. Wrapster currently handles detail pages; however, it detects and extract lists from details pages such as a list of actors in a movie detail page as described in detail in Chapter 3.

Wrapper Generation for Detail Pages

Crescenzi et al. [14] developed RoadRunner. RoadRunner proposes a matching algorithm to infer union-free regular expressions from multiple pages with the same template. The resulting regular expressions are then used to extract data from other similar pages. This approach has been improved by Arasu et al. [3], who presented a polynomial time algorithm by using several heuristics. The input for such systems is similar to Wrapster’s input and they also require several pages from the same template. Systems developed for this task need to deal with the following:

- Pages contain irrelevant information such as advertisements and user comments which wrapper generation systems need to discard.
- Not all pages contains the same set of fields; therefore, wrapper generation systems need to handle missing and extra fields.

Wrapper Generation for List Pages

This subfield is also known data record extraction. It has been shown by Liu et al. [35] that one input page is sufficient for for this task. Wrapper generation systems for list pages first identify data record boundaries and then generate patterns to extract the data records their properties.

Embley et al. [17] conducted a study to automatically identify data record boundaries. Their method is based on set of heuristic rules such as tag repetition and ontology matching. Buttler et al. [6] proposed additional heuristics such as partial

path heuristic and eliminated the need for a domain ontology which is costly to build. However, those methods do not extract data records. Chang et al. [9] proposed to find patterns from HTML tag strings based on Patricia trees and sequence alignment to extract the data items. Liu et al. [35] proposed MDR which only identify data records. Zhai et al. [47] improved the MDR using visual rendering information to deal with erroneous HTML tags and then applied partial tree matching to extract the data record contents. Reis et al. [41] used tree matching to find article content from Web newspapers.

2.1.3 Interactive Wrapper Generation

Interactive wrapper generation is also called visual wrapper generation. Such an approach provides specialized pattern specification languages to help the user construct data extraction programs. Interactive wrapper generation systems hide their complexities under a graphical user interface.

Systems that follow this approach include WICCAP [34], Wargo [40], Lixto [4], DEBye [33], Dapper¹, etc.

In contrast to these systems, Wrapster provides an infrastructure that manages a large number of sites and provides an interactive web interface to edit wrappers.

2.1.4 Wrapper Generation for End-Users

Wrapper generation for end-user systems augments the user browsing experience. Such systems usually require no training and they provides simple services as such adding a person contact to the personal address book and sorting a list of items in a new order.

Bolin et al. [5] have developed Chickenfoot, a Firefox extension that the user can use to write scripts to manipulate web pages and automate web browsing. It is based on LAPIS's rich library of patterns and parsers. LAPIS [37] enables Chickenfoot to recognize text structure automatically. While Wrapster currently uses regular

¹<http://www.dapper.net>

expressions to recognize text structure, rich libraries like LAPIS’s might someday be able to provide it with more robust pattern generation.

Hogue et al. have introduced Thresher [22] that lets non-technical users extract semantic content from the Web. Thresher is an end-user wrapper and part of the Haystack Project. It reduces the amount of training data needed, leverages the hierarchical structure of HTML, and learns patterns by aligning multiple example trees using a *tree edit distance* algorithm. However, Thresher fails to create wrappers that make use of global document structures. It can wrap only a small subset of a Web page, referred to as records, and if the time to generate a wrapper passes the threshold usually expected of an end-user program, the system will be unusable.

Huynh et al. [25] recently tried to augment the sorting and filtering ability of retail sites on the Web. Since the resulting items from a query on a retail site have the same structure, it is easy to recognize the underlying structure by comparing the tree structure of the items. Even with just the simple ability to sort and filter items interactively while visiting an online retail site, great value has been added to the browser, as reported by their user study.

2.2 Tree Data Structures and Algorithms

Typically, information resources on the Web are in HTML. HTML pages can be manipulated via DOM [1], an application programming interface for HTML and XML documents. DOM documents have a tree-logical structure. Leveraging the tree structure of Web pages is a recent trend in wrapper generation from the Web. Reis et al. [41], Hogue et al. [22], and Huynh et al. [25] used *tree edit distance* to find and compare information within Web pages.

Since Wrapster models wrappers from DOM trees, algorithms such as locating tree nodes, tree-pattern matching, tree isomorphism, and *tree edit distance* are needed to extract and compare information from DOM documents:

- Locating nodes in DOM documents is addressed by the XPath standard [2].

XPath allows users to specify path-structured queries which return sets of

matching nodes from the DOM document tree. However, XPath requires technical skills.

- *Tree pattern matching* has been well studied in fields such as computer science theory and compiler optimization [12, 16, 20]. Most approaches require the system to have great knowledge of the underlying data, and have focused on tree matching of unordered trees.
- *Tree isomorphism* [23, 36] tries to locate similar structures in trees. It generally finds structural similarities in trees without including node labels or taking order into consideration. Tree isomorphism is a specialization of graph isomorphism [46], whose complexity is NP (not solvable in polynomial time).
- *Tree edit distance* [27], upon which our approach is based, is an extension of string edit distance [13]. The edit distance between strings or trees is computed in polynomial time using dynamic programming. The edit distance algorithm finds the minimal cost of delete, insert, and change operations to turn one object into another.

Tree pattern matching and tree isomorphism are out of the scope of this thesis since they are used for automatic record detection and extraction research which tries to find similar tree patterns inside a single data page. In the next section I will describe in detail the definition and my implementation of *tree edit distance*.

2.2.1 Tree Edit Distance

The *tree edit distance* between two trees T_1 and T_2 is defined as the cost of edit operations to transform T_1 into T_2 . The three edit operations are changing a node's label, deleting a node from the tree by making its children the children of the parent node, and inserting a new node into the tree which is the complement of delete operation. Each edit operation has an associated cost γ_{modify} , γ_{insert} , and γ_{delete} . A mapping M is sequence of edit operations that transform T_1 into T_2 . The *best mapping* M is the lowest-cost set of set of edit operations to transform T_1 into T_2 .

Formally the triple (M, T_1, T_2) or simply M is defined to be a mapping from T_1 to T_2 (Zhang et al. [27]):

- M is any set of pairs of integers (i, j) satisfying:
 1. $1 \leq i \leq |T_1|, 1 \leq j \leq |T_2|$
 2. For any pair (i_1, j_1) and (i_2, j_2) in M ,
 - (a) $i_1 = i_2$ if-f $j_1 = j_2$ (one-to-one),
 - (b) $T_1[i_1]$ is to the left of $T_1[i_2]$ if-f $T_2[j_1]$ is to the left of $T_2[j_2]$ (sibling order is preserved),
 - (c) $T_1[i_1]$ is the ancestor of $T_1[i_2]$ if-f $T_2[j_1]$ is the ancestor of $T_2[j_2]$ (ancestor order relationships preserved)
- The cost $\gamma(M)$ of a mapping M from T_1 to T_2 , is defined to be

$$\gamma(M) = \sum_{i,j \in M} \gamma(T_1[i] \rightarrow T_2[j]) + \sum_{i \notin M} \gamma(T_1[i] \rightarrow \Lambda) + \sum_{j \notin M} \gamma(\Lambda \rightarrow T_2[j])$$

where Λ is the empty node.

In Wrapster’s implementation, we defined the cost function for each operation to be

$$\gamma(v \rightarrow w) = \begin{cases} 1 & \text{if } v \neq w \\ 0 & \text{otherwise.} \end{cases}$$

To calculate the least-cost *tree edit distance* between v and w we implemented the dynamic programming algorithm defined in Andrew Hogue’s master thesis [21].

2.3 Classification

Wrapster uses Support Vector Machines (SVM) for its classification task. SVM has been in the spotlight in the machine learning community because of its theoretical soundness and practical performance. In particular, we used $\text{SVM}^{\text{light}}$, a scalable and

efficient implementation of SVM [26]. In brief, SVM is defined as the optimal hyperplane that separates the vector sets belonging to different classes with the maximum distance to its closest samples.

Chapter 3

Template Creation

In this chapter we describe the first component of Wrapster, how we create a template for a Web site. A site is a Web data source that contains information about a specific domain. For example, the IMDb is an online database which contains information about movies, actors, television shows, production crew personnel, and video games. As a database has objects with records containing fields of data about the objects, “semi-structured” Web sites have “objects” with Web pages containing information about the objects. While a semi-structured Web site doesn’t contain formal fields like a true database, nevertheless the objects’ pages have similar structure and “fields” can be detected programatically with fairly high precision and recall.

Given a list of objects and their corresponding Web URLs (see Table 1.1 for an example), Wrapster can be configured to either select randomly several items from the object–URL pair list or the user can select which items to generate the template from. The generated template for a site is a list of all identified regions of relevant information that a wrapper for that site should be able to extract.

3.1 Tree Mapping

First, Wrapster computes the tree mapping between all selected item pairs. It parses the selected items’ Web pages into DOM parse trees using the Cyberneko [10] HTML parser library, a simple HTML parser that enables the information inside the HTML

to be accessed using XML interfaces. Then, Wrapster computes *tree edit distance* on all DOM tree pairs and gets the optimal mapping between each pair of DOM trees. The algorithm to compute *tree edit distance* was described in detail in Section 2.2.1. To get more intuition about how we identify regions in an HTML page using *tree edit distance*, see Figure 3-1, which shows a rendered fragment of the *Fahrenheit 9/11* movie Web page where some text content is marked with yellow background. This page is achieved by computing the tree alignment with the *Garden State* movie Web page (Figure 3-2) and marking the aligned HTML elements that are different in both Web pages. We can notice that the year next to the movie title is not marked since both movies have the same value “(2004)”. We can also notice that the “Writer” region has the value WGA for both movies so that this region is also not identified by the tree alignment

Although previous work in wrapper generation for list pages has found it to be sufficient to examine a single page, this is clearly not the case for details pages. To guarantee that all possible regions were covered, it would be necessary to perform every alignment. Currently, Wrapster requires the user to select the specific sample pages to align, and/or the number of sample pages for the system to randomly select. In the future, we will investigate automating the process of alignment using heuristics to choose when to stop the process.

Having the mapping between each pair we continue to the next section to filter out the irrelevant information.

3.2 Region Identification

This subcomponent receives as input the tree mapping between each DOM tree pair computed in the previous step. Its purpose is to filter out the irrelevant information and to identify the regions to extract. First we generate a list of DOM node candidates that are possible subregions. We discard any mapping under the “head” node since the data under the “head” tag is not visible in the browser. We add to the list of node candidates all the text nodes under the “body” node whose node value is different

IMDb > [Fahrenheit 9/11 \(2004\)](#)



add to My Movies

Quicklinks

[main details](#)

Top Links

- [trailers](#)
- [full cast and crew](#)
- [trivia](#)
- [official sites](#)
- [memorable quotes](#)

Overview

[main details](#)

- [combined details](#)
- [full cast and crew](#)
- [company credits](#)
- [tv schedule](#)

Promotional

- [taglines](#)
- [trailers](#)
- [posters](#)
- [photo gallery](#)

Awards & Reviews

- [user comments](#)
- [external reviews](#)
- [newsgroup reviews](#)
- [awards](#)
- [user ratings](#)
- [recommendations](#)
- [message board](#)

Plot & Quotes

- [plot summary](#)
- [nint kawwnrrls](#)

Fahrenheit 9/11 (2004)

[photos](#)
[board](#)
[trailer](#)
[IMDb Pro details](#)


[Register](#) or [login](#) to rate this title

User Rating: 7.7/10 (80,455 votes)

[more](#)

Photo Gallery (see all 55 photos)





[more](#)

[SHOP FAHRENHEIT...](#)
[amazon.com](#)
[Download](#)
[All](#)

[DVD](#)
[VHS](#)
[CD](#)

advertisement

Overview

Director: [Michael Moore](#)

Writer (WGA): [Michael Moore](#) (written by)

Release Date: [25 June 2004 \(USA\)](#) [more](#) [view trailer](#)

Genre: [Documentary](#) / [War](#) [more](#)

Plot Outline: Michael Moore's view on what happened to the United States after September 11; and how the Bush Administration allegedly used the tragic event to push forward its agenda for unjust wars in Afghanistan and Iraq. [more](#)

Plot Keywords: [Political Protest](#) / [Propaganda](#) / [Good Versus Evil](#) / [Fat Man](#) / [Expose](#) [more](#)

Awards: [25 wins & 12 nominations](#) [more](#)

User Comments: [A whirlwind tour of corruption and diplomatic deceit](#) [more](#)

Cast (Cast overview, first billed only)

	Ben Affleck	...	Himself (archive footage)
	Stevie Wonder	...	Himself (archive footage)
	George W. Bush	...	Himself (archive footage)
	James Baker III	...	Himself - Former Secretary of State (archive footage)

Figure 3-1: Tree mapping of *Fahrenheit 9/11* movie Web page aligned with *Garden State* movie Web page from IMDB site

IMDb > Garden State (2004)



+ add to My Movies

Quicklinks

main details

Top Links

- trailers
- full cast and crew
- trivia
- official sites
- memorable quotes

Overview

- main details
- combined details
- full cast and crew
- company credits
- tv schedule

Promotional

- taglines
- trailers
- posters
- photo gallery

Awards & Reviews

- user comments
- external reviews
- newsgroup reviews
- awards
- user ratings
- recommendations
- message board

Garden State (2004)

photos board trailer **IMDb PRO** details

Register or login to rate this title



User Rating: 7.9/10 (56,988 votes)

more

Photo Gallery (see all 57 photos)



more

Overview

Director: [Zach Braff](#)

Writer (WGA): [Zach Braff](#) (written by)

Release Date: 22 September 2004 (Trinidad And Tobago) [more](#) [view trailer](#)

Genre: [Comedy](#) / [Drama](#) / [Music](#) / [Romance](#) [more](#)

Plot Outline: A quietly troubled young man returns home for his mother's funeral after being estranged from his family for a decade. [more](#)

Plot Keywords: [Boyfriend Girlfriend Relationship](#) / [Dog](#) / [Overhead Camera Shot](#) / [Soundtrack](#) / [Animal In Cast Credits](#) [more](#)

Awards: 10 wins & 24 nominations [more](#)

User Comments: A blooming, Wonderful Garden State! [more](#)

Cast (Cast overview, first billed only)

	Zach Braff	...	Andrew Largeman
	Kenneth Graymez	...	Busboy



Figure 3-2: Screen shot of *Garden State* movie Web page from IMDB site

in the tree mapping (edit distance is not equal to zero). The text nodes are the leaf nodes of the HTML DOM tree and they contain all the content of an HTML document.

Each node is a candidate region. We build a node-path tree of the region candidate content list. Each node has a unique node path. Using the node-path we identify nodes that belong to the same region: We traverse the tree and merge recursively all child nodes if there is no break between them. In Wrapster, a break is one of the following closing tags DIV, P, BR, TABLE, TH, TR, TD, UL, OL, DL, LI, DT, and DD. That is, in this step we will merge nodes separated only by constant text strings; for example, we will merge “Documentary” and “War” into a single region “Documentary / War” (see the “Genre” line in Figure 3-1). It is acceptable to treat table and list elements as separate regions here because we will merge lists and tables later in the process. This step yields region instances.

At the end this merging step, each region contains the list of all candidate content nodes. At this point, we expand each region with all the close context (context within the break node), if the context exists, which we will later use in the script generation process. All nodes in a region are ordered and each region stores its tree mapping properties such as the edit distance cost for each node. In addition, each region stores the previous and next general context (context not within the break node). Given the list of regions, we infer structure such as lists and tables using the regions’ node-paths.

To summarize, in this step Wrapster identifies the region instances within each object’s Web page. In the next section, we will describe how we cluster matching regions from all objects’ Web pages in order to form our general template for the site.

3.3 Region Clustering

Having a basic template containing region *instances* for each alignment pair, we need to compute the general template of *regions* for all objects, taking into account all layout and format differences. This step detects slot insertion or deletion since usually some objects have different properties from others.

In designing an algorithm to perform this step, our first intuition was to create a manual rule-based program that incrementally built a general template by matching regions from basic templates. However, this approach didn't perform as well as we expected because the rules did not generalize well across the data sources. The maximum performance we achieved with this approach is 90.58% F-Measure. We dropped this approach and adopted a supervised machine learning technique using Support Vector Machine classification, as described below.

3.4 Classification

To cluster the identified region instances from all the selected items' Web pages, Wrapster uses SVM^{light} , a Support Vector Machine classifier implementation, that takes as input all pairs regions instances and decide for each pair if the regions match or not, as described in detail in the next paragraph. Then, Wrapster uses the classifier decision for each pair to create clusters for matching regions that form the basic template for the wrapped site.

Given two region instances from different objects' Web pages, the classifier task is to determine if those region instances belong to the same region. To build the training data seed, we annotated 2,517 region instances from five Web pages. The training data samples came from IMDb movie and The World Factbook¹ Web pages. They consist of 156 matching region instances and 2,361 non-matching instances. (The features used for training the SVM are described in Section 3.4.1.) However, such a small dataset does not provide enough training data to reliably generalize feature settings. Therefore, we used a variation of active learning technique to automatically annotate data and retrain the classifier (see Chapter 8).

3.4.1 Features

The training data are in HTML format; therefore, the features should leverage the hierarchical structure of HTML. The features are mostly similarity measures between

¹<https://www.cia.gov/library/publications/the-world-factbook/index.html>

the two region instances. The features we used are the following:

- Region size ratio: The ratio of the number of nodes in the two regions.
- Content similarities:
 - Exact match disregarding HTML tags.
 - String edit distance disregarding HTML tags.
- Region node match ratio: The ratio of the number nodes in the two regions that have zero tree edit cost (close context).
- Node-paths similarities: How far apart are regions in the HTML document.

We add the same features for preceding and following contexts.

3.5 Annotator Tool

We developed a graphical user interface that lets the user add and edit training instances. The user has two ways of annotating with this tool.

Firstly, the initial page of the annotator (Figures 3-3 and 3-4) lists all the uploaded training data. The user can select training instances to view using the following options:

- Classification score: This constraint lets the user view all the training data which are in the range of specified classification scores interval. This option allows the user to, for example, constrain the scores to between -1 and 1 to view only outlying instances.
- Match constraint: This constraint has the following values: list all training data, list data whose regions match, list data whose regions don't match, and list data that the annotator is not sure about.
- Automatic match: This constraint lets the user restrict the displayed training data to just automatically annotated data or manually annotated data.

The user can combine any of the above constraints to customize how to inspect and edit all training data.

To upload more training data the user has the option to type or paste XML in the input text area or upload a file (Figure 3-5); for example, if training needs to be readjusted later to take into account an example that was missed in earlier training. The input is in XML format for which the DTD is shown in Figure 3-6.

The “annotate” page shown in Figure 3-7 lists the next training data item to annotate. In addition, the page contains the classification score computed by the trained classifier. The input radio button is preselected according to the classifier score. If the classification is above or below certain thresholds, in this case 1 and -1, the program marks that the data item as matching or not matching correspondingly. If the classification is in the threshold interval, the program chooses the “not sure” option. The user can edit the program suggestion and move to the next item to annotate. The user may have to annotate a very large number of examples, which can be time-consuming and tedious. The automatic classifier can often decide with fairly high accuracy if the regions match. Therefore, to ease the classification process for the user and save them the trouble of clicking again and again, the user is given a countdown timer of seven seconds to edit the program suggestion. By default, the form is submitted and the next item to annotate is shown to the user. The user can stop the countdown timer with a mouse click on the page and resume the timer with another mouse click.

3.5.1 Automatic Annotation

The user’s second method for annotating is via the automatic annotation page (Figure 3-8), which provides quick batch annotation. The user can choose to create a batch annotation for a certain amount of training data. After the batch run has finished, the user can inspect the automatic annotation result and correct the classifier if needed by viewing it in the List data page.

 Home
  List
  Upload
  Annotate
  Serialize To File
  Logout

Listing ranges

Constraints:

Match: Select list constraints

Range id: from to

Classification score: from to

Automatic match: Select list constraints

	1st region	2nd region
Data	Peter Jackson	David Fincher
Previous Content	Directed by	Directed by
Next Content	Writing credits WGA	Writing credits WGA

[\(show details...\)](#)

Do the regions match?

Yes ☒ No ☐ Not Sure! ☐

[Show](#) [Edit](#) [Destroy](#)

Range id: 7169 Classification score : 1.1709073 Automatically matched : 1

	1st region	2nd region
Data	LORD OF THE...	FIGHT CLUB
Previous Content	(more) Awards: (more) Quotes: (more) Goofs: (more) Trivia: (canton of Vaud) (canton of Geneva) Japan:PG-12 Certification: Sound Mix: (DeLuxe) Color Color: English Language: USA Country: Runtime: MPAA : (more) Cast overview, first billed only: User Rating: (more) User Comments: (view trailer) (more) Plot Outline: (more) Tagline: (more) Genre: more Photo Gallery (140 photos) IMDbPro Details Photo Gallery Add to MyMovies / WGA Writing credits Directed by	(more) Awards: (more) first lines Quotes: (more) Goofs: (more) Trivia: (canton of Vaud) (canton of Geneva) Hong Kong: IIB Certification: Dolby Digital EX DTS-ES Sound Mix: Color Color: English Language: USA Country: Runtime: MPAA : Also Known As: (more) Cast overview, first billed only: User Rating: (more) User Comments: (view trailer) (more) Plot Outline: (more) Tagline: (more) Genre: more Photo Gallery (166 photos) IMDbPro Details Photo Gallery Add to MyMovies (novel) / WGA Writing credits Directed by
Next Content	Memorabilia Books All Products	Memorabilia Books All Products

[\(show details...\)](#)

Do the regions match?

Yes ☒ No ☐ Not Sure! ☐

[Show](#) [Edit](#) [Destroy](#)

Range id: 7207 Classification score : 1.0295992 Automatically matched : 1

Figure 3-3: Annotator tool screen shot of the List data page. Selected options appear at the top and all selected training data items appear below; the first two items can be seen here.

	1st region	2nd region
Data	Peter Jackson	David Fincher
Previous Content	Directed by	Directed by
Next Content	Writing credits (WGA)	Writing credits (WGA)

([show details...](#))

1st region

order	xpath	content	cost
0	DIV[@id="root"] LAYER[1] TABLE[3] TR[1] TD[3] TABLE[1] TR[1] TD[1] TABLE[1] TR[1] TD[2] #text[2]		0
1	DIV[@id="root"] LAYER[1] TABLE[3] TR[1] TD[3] TABLE[1] TR[1] TD[1] TABLE[1] TR[1] TD[2] A[1]	Peter Jackson	2

2nd region

order	xpath	content	cost
0	DIV[@id="root"] LAYER[1] TABLE[3] TR[1] TD[3] TABLE[1] TR[1] TD[1] TABLE[1] TR[1] TD[2] #text[2]		0
1	DIV[@id="root"] LAYER[1] TABLE[3] TR[1] TD[3] TABLE[1] TR[1] TD[1] TABLE[1] TR[1] TD[2] A[1]	David Fincher	2

Do the regions match?

Yes ☒ No ☐ Not Sure! ☐ [Submit](#)

[Show](#) [Edit](#) [Destroy](#)

Range id: 7169 Classification score : 1.1709073 Automatically matched : 1

Figure 3-4: Annotator tool screen shot of the List data page, zooming on one training data item.

 Home
  List
  Upload
  Annotate
  Serialize To File
  Logout

Annotation load data page

Select input format: Text ☒ File ☐

Input text:

[Load Data](#)

[Create a new range form](#)

Figure 3-5: Annotator tool print-screen of the Upload page.

```

<!ELEMENT CONTENT ( DATA, DATA_WITH_ATTR, MAP ) >

<!ELEMENT DATA ( #PCDATA ) >

<!ELEMENT DATA_WITH_ATTR ( #PCDATA ) >

<!ELEMENT DOCUMENT ( RANGE+ ) >
<!ATTLIST DOCUMENT count NMTOKEN #REQUIRED >
<!ATTLIST DOCUMENT first_document_url CDATA #REQUIRED >
<!ATTLIST DOCUMENT second_document_url CDATA #REQUIRED >

<!ELEMENT MAP ( NODE_PROPERTIES+ ) >

<!ELEMENT NEXT_CONTENT ( DATA, DATA_WITH_ATTR, MAP ) >

<!ELEMENT NODE_PROPERTIES EMPTY >
<!ATTLIST NODE_PROPERTIES content CDATA #REQUIRED >
<!ATTLIST NODE_PROPERTIES content_with_attr CDATA #REQUIRED >
<!ATTLIST NODE_PROPERTIES cost NMTOKEN #REQUIRED >
<!ATTLIST NODE_PROPERTIES order NMTOKEN #REQUIRED >
<!ATTLIST NODE_PROPERTIES xpath CDATA #REQUIRED >

<!ELEMENT PREV_CONTENT ( DATA, DATA_WITH_ATTR, MAP ) >

<!ELEMENT RANGE ( REGION+ ) >
<!ATTLIST RANGE range_id NMTOKEN #REQUIRED >
<!ATTLIST RANGE xpath_from_first_document CDATA #REQUIRED >
<!ATTLIST RANGE xpath_from_second_document CDATA #REQUIRED >

<!ELEMENT REGION ( CONTENT, PREV_CONTENT, NEXT_CONTENT ) >

<!ELEMENT ROOT ( DOCUMENT+ ) >

```

Figure 3-6: Annotator tool DTD of training data XML input format.

Home
 List
 Upload
 Annotate
 Serialize To File
 Logout

Annotate data page

Constraints:

Range id: *from*

Do the regions match? (Classification Result = -1.8684667) Automatic Submission in :

Yes ☐ No ☒ Not Sure!

	1st region	2nd region
Data	Trivia: Sean Astin reckons that there were 50 separate days of the 15 month shoot were the Hobbits had their prosthetic feet applied and they were never seen on camera. (more)	Awards: 1 win & 2 nominations (more)
Previous Content	(canton of Vaud) (canton of Geneva)	(more) Quotes:
Next Content	Goofs: (more)	Memorabilia Books All Products

[\(show details...\)](#)

[Show](#) [Edit](#) [Destroy](#)

Range id: 7376
[Edit](#)

Figure 3-7: Annotator tool screen shot of the Manual annotate data page. The above training sample is a comparison between a “Trivia” and an “Awards” region. The Annotator Tool decided that those regions do not match since the classification score is -1.86 outside of the threshold range.

Home
 List
 Upload
 Annotate
 Serialize To File
 Logout

Batch Automatic Annotation

Constraints:

Range id: *from*

How many to annotate:

Figure 3-8: Annotator tool screen shot of the Batch automatic annotate request page

Chapter 4

Wrapper Induction

After creating the general template for the site being wrapped, our next step is to generate the wrapper. The template generated in the previous chapter contains all the clustered identified regions from all the selected Web pages. The wrapper is comprised of clustered region instances and a list of extraction rules for each cluster. The extraction rules are the specialized scripts that extract specific properties for any given object from that site. The region instances will serve us later in the repairing module to test and fix the wrapper in case a script fails to execute (see Chapter 7).

4.1 Script Generation

The extraction rules for the wrapped site regions are lists of patterns derived from each cluster of region instances. Regular expressions have been used extensively to extract regions. Recently, commercial systems leveraged the HTML hierarchical structure and used XPath to extract information from Web pages; however, this technique requires parsing of Web pages, which is slower than just matching string patterns. Furthermore, using XPath is limiting since the XPath language cannot extract text fields that are not wrapped by an HTML tag. Other techniques, such as Hap-Shu [44] and LAPIS [37], use rich pattern matching libraries which can extract regions from Web pages if the region can be identified via text landmarks. Since most Web pages are updated frequently, those rich libraries aim to make the script more

robust to layout changes. Wrapster currently generates standard regular expressions matching surrounding context, but rich libraries will yield more-robust scripts and will be explored in future work.

4.1.1 Pattern Induction

The clustered regions contain the content of all regions and their corresponding context along with their order in the document. The extraction rules are a list of prefixes and suffixes that are matched on the data page to extract a specific region. To compute the prefix and suffix patterns for each region, we first align the text nodes for all region instances. Then, we search for starting and ending text landmarks that occurs in all regions and select the ones that surround the region content in all region instances. If we cannot find starting and ending patterns in the close context of the region we select them from the preceding and the following context.

4.1.2 Region Focuses

By manipulating the extraction rules we return exact values in addition to HTML fragments. Since each region has mapping information from the tree alignment, we know which parts of the region are constant and which parts have the actual content. Depending on the type of third-party system which will be using the wrapper, we may want the wrapper to create scripts with different focuses; that is, we may want to be able to retrieve more than one version of each region value, such as the full HTML value for human-friendly display purposes or an “exact” value for question fusion, value comparison, etc. For example, we might want to retrieve “21 wins & 12 nominations” as a movie awards value for human display (see Figure 3-1), but retrieve “21” as part of answering “What movies have received more than twenty awards?”. Creating scripts with alternate focuses is adds post-processing to each script with context filtering. One common heuristic is removing links and context text from the script output. In the near future, we plan to explore more sophisticated heuristics to create scripts with alternate focuses.

Chapter 5

Semantic Annotation

When we start the semantic annotation step of wrapper creation, the wrapper generated by the previous steps contains the scripts and the stored region instances. However, to use the wrapper in third-party systems such as question answering systems, the wrapper needs to store semantic information so that the required information can be extracted from the object's page. For example, if a QA system needs to answer “How many people live in the United States?”, it will seek for the population property information from the queried country's Web page. (It is the responsibility of the QA system to first identify the “United States” as a country and determine that the question refers to the population of that country.)

Giving the scripts meaningful names, such as “director”, instead of machine generated names, such as “script-1242”, makes the connection of the wrapper with a third-party application more intuitive. Automating this process is a hard task because there are not always text clues for all regions on the Web pages. Furthermore, if a text clue exists there is often more than one to choose from, and its format varies greatly for each site. To get a sense of where semantic information is located in the source document, Figure 5-1 shows the “POPULATION” region for the United States from The World Factbook¹ Web site and its HTML code. To extract semantic annotation for this region is not trivial: we need to select the “Population” from the previous table column and filter out other text clues such as “People” and the image

¹<https://www.cia.gov/library/publications/the-world-factbook/index.html>

People	United States	Top of Page
Population:   		
298,444,215 (July 2006 est.)		

(a)

```

...
<td class="FieldLabel" valign="top" width="20%">
  <div align="right">Population:</div>
</td>
<td bgcolor="#ffffff" valign="top" width="80%">
  <a href="../docs/notesanddefs.html#2119">
    </a>
    <a href="../fields/2119.html">
      </a>
      <a href="../rankorder/2119rank.html">
        </a>
      <br>
      298,444,215 (July 2006 est.)
    </td>
...

```

(b)

Figure 5-1: United States population region extract from The World Factbook Web site (a) and the corresponding HTML source (b).

Director:	Robert De Niro
------------------	--------------------------------

(a)

```

...
<div class="info">
<h5>Director:</h5>
<a href="/name/nm0000134/">Robert De Niro</a><br>
</div>
...

```

(b)

Figure 5-2: *The Good Shepherd* movie director region extract from the IMDb Web site (a) and the corresponding HTML source (b).

links. In this case, the default script name is the content of a “div” element. In a second example (Figure 5-2) the semantic information for the director region is in the close context of the region and is the content of an “h5” element. Wrapster computes a basic semantic annotation and then lets the user annotate the wrapper using the Wrapster Web User Interface (see Chapter 6).

Wrapster’s heuristic selects the text occurring in all region instances for the given attribute and we select the text which is closest to the region of interest in the preceding context. Non-alphanumeric characters are trimmed from each end of the selected text. For example, in the cases in the above paragraph, we would select “Population” and “Director”, respectively. This basic suggestion is presented to the user for confirmation. Earlier implementation of semantic annotation [15, 43] systems relied mainly on ontologies such as WordNet. In the future, we intend to improve semantic annotation and implement a component that takes into account the region content and leverages HTML structure.

Chapter 6

Wrapster User Interface

To eliminate the need for programming skills usually required to generate a wrapper for the site, we developed a Web user interface for Wrapster that manages all tasks needed for wrapper generation. The main purpose of the user interface is to give a novice user the ability monitor the automatic wrapper generation, edit the final version of the generated wrappers and add the semantic annotation for each property discovered.

Wrapster's home page (Figure 6-1) provides the user with two options. The first is to create a new wrapper for a site and the second is to edit an existing wrapper. To create a wrapper the user chooses the Web pages from which Wrapster will generate the wrapper. The user chooses the pages interactively using an HTML proxy server by adding the browsed pages to a *basket* using a control bar inserted on the top of the Web pages (Figure 6-2). After selecting the Web pages, Wrapster generates the wrapper and redirects the user to the wrapper editing Web page. The wrapper editing page lists the wrapper's scripts for the site. The user can add semantic information for each script, edit the extraction patterns, test, and serialize the scripts into START's Omnibase script format or XML format. The Web wrapper editor, as seen in Figure 6-3, has three panes. The "Scripts" pane lists all the wrapper's scripts, the "Wrapper properties" pane lists the attributes that Wrapster used to generate the wrapper with, and the "Work" pane enables the user to test the scripts interactively for any object's URL.



Figure 6-1: Wrapster web user interface main page



Figure 6-2: IMDb *Sean Connery* Web page opened using Wrapster HTML proxy

Scripts

Goofs - (id 0)

Quotes - (id 1)

Awards - (id 2)

Tagline - (id 3)

Plot - (id 4)

Pattern

Pattern 0 - (HTML)

- Prefix: <h5>Plot Outline:</h5>
- Suffix: plotsummary">more

Examples

Example 0 - http://www.imdb.com/title/tt0120586/

- Plot Outline: A former neo-nazi skinhead (Norton) tries to prevent his younger brother (Furlong) from going down the same wrong path that he did. **(more) (view trailer)**

Example 1 - http://www.imdb.com/title/tt0120737/

Example 2 - http://www.imdb.com/title/tt0137523/

Example 3 - http://www.imdb.com/title/tt0381717/

Example 4 - http://www.imdb.com/title/tt0408306/

Comments - (id 5)

Rating - (id 6)

MPAA - (id 7)

Runtime - (id 8)

Certification - (id 9)

Trivia - (id 10)

Sound Mix - (id 11)

Genre - (id 12)

Photos - (id 13)

Add Comments - (id 14)

Create new script

Wrapper Properties

Name: imdb_train1

Apply url: http://www.imdb.com/title/tt030

Urs: http://www.imdb.com/title/tt030

Save

Work pane

Script: Plot

URL : http://www.imdb.com/title/tt0303706/

Execute script

Plot Outline:

A quietly troubled young man returns home for his mother's funeral after being estranged from his family for a decade. **more**

Figure 6-3: Wrapster Web user interface editing the IMDb wrapper

Chapter 7

Repair Module

Data sources such as Web sites are subject to changes in format and wording, sometimes frequent changes, which cause wrapper scripts to fail. For example, in the IMDb site, if a “director” script is looking for a text landmark such “Directed by” (Figure 7-1) to extract the director property for the movie *Garden State* it will fail if the site has been updated and the keywords “Directed by” have changed to “Director:” (Figure 7-2). In addition, we can notice that the layout has changed (Figure 7-3). In order to be useful, wrapper scripts must be robust to at least moderate changes in format in wording. Wrapster’s repair module monitors and automatically repairs Wrapster scripts, and so is central to Wrapster’s robustness. It handles cases where scripts fail to extract the previously annotated information from a site, when the layout or format or both have changed, by looking for regions identified in a new template whose values match values stored using the old template.

The repairing module runs in the background and tests all the wrappers for failing scripts where stored values are available. The region instances calculated during template creation serve as stored values for repair. If a failing script is detected the module repeats the process of template creation and wrapper induction, described in Chapter 3 and 4, for that site and update only the failing script or all the site scripts according to the saved program configuration.

After generating the basic wrapper for the site, the module matches the newly generated scripts and the existing ones using the stored regions instances. It classifies

Garden State (2004)



Directed by
[Zach Braff](#)

[▶ watch the trailer](#)

Writing credits ([WGA](#))
[Zach Braff](#) (written by)



[Add to MyMovies](#)



[Photo Gallery](#)



[IMDbPro Details](#)

Figure 7-1: Screen shot of *Garden State* movie Web page from 2006 IMDB site

Garden State (2004)



[photos](#)



[board](#)



[trailer](#)



[IMDb Pro details](#)

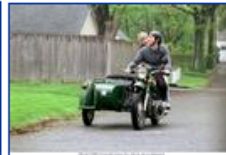


[Register](#) or [login](#) to rate this title

User Rating: 8.0/10 ([55,020 votes](#))

[more ▶](#)

Photo Gallery ([see all 57 photos](#))



[more ▶](#)

Overview

Director: [Zach Braff](#)

Writer ([WGA](#)): [Zach Braff](#) (written by)

Figure 7-2: Screen shot of *Garden State* movie Web page from 2007 IMDB site

(a)

```
<b class="blackcatheader">Directed By</b><br>
<a href="http://imdb.com/name/nm0103785/">Zach Braff</a><br>
<br>
```

(b)

```
<div class="info">
<h5>Director:</h5>
<a href="/name/nm0103785/">Zach Braff</a><br/>
</div>
```

Figure 7-3: *Garden State* movie HTML code from the IMDb site (a) 2006 layout (b) 2007 layout.

all pairs of regions instances using the trained classifier and chooses the script with the highest match ratio. Then, the user-edited information such as semantic information is transferred to the newly generated scripts from the failing scripts. Finally, the module can notify the administrator to verify the correction of the new scripts and apply the changes.

In conclusion, we achieve this module almost without effort using the previous steps of wrapper generation due to the simple design and modularity of Wrapster (evaluated in Chapter 8).

Chapter 8

Experiments and Results

To evaluate Wrapster we need to evaluate some components separately and the performance of the system as a whole. We conducted experiments based on data from seven sites that have details pages (Table 8.1). Four of the datasets, IMDb person, POTUS, MSN Money, and The Weather Channel, are available at the RISE repository [38], and have been evaluated by previous wrapper generation systems such STALKER [39], WIEN [32], and WL² [11]. The other three sites were chosen because they are in use by the START Question Answering system.

<i>Web sites</i>	<i>location</i>	<i>Number of content regions</i>
IMDb Movie	http://imdb.com/title	30
IMDb Person	http://imdb.com/name	14
The World Factbook	https://www.cia.gov/cia/publications/factbook	229
POTUS	http://www.ipl.org/div/potus/	25
MSN Money	http://moneycentral.msn.com/	28
The Weather Channel	http://www.weather.com/	16
50 States	http://www.50states.com/	77

Table 8.1: Evaluation dataset.

<i>source</i>	<i>total training samples</i>	<i>positive</i>	<i>negative</i>
IMDb Movie	9566	466	9100
The World Factbook	566	463	103

Table 8.2: SVM training data

8.1 Classification

We evaluated the SVM classification on 10,132 training data samples using 10-fold cross validation (Table 8.2). The training data were annotated using our annotator tool described in Chapter 3. Tables 8.4 and Table 8.3 show the average F-Measure, precision, and recall values for the SVM classifier across all feature groups for the full training data samples and for IMDb Movie training data. We noticed that the FULL feature group on IMDb data had the best performance. The NEXT CONTEXT feature group appears to be more informative for classification than the PREVIOUS CONTENT feature group since the former group outperformed the latter group in all datasets.

In addition we evaluated our system on 2,992 annotated instances of 62,010 data samples gathered from the rest of the sites in our datasets where their classifier confidence score was inside the $[-1,1]$ range (outliers). The classifier accuracy was 53.41 (37.36% precision and 60.3% recall). Even though the results are not high, they show satisfactory performance of the classifier on the outlier points which are much harder to classify. For example, by examining the annotated data samples the classifier didn’t classifier correctly the “Actress Filmography” and “Actor Filmography” regions of IMDb. In the future we need incorporate more complex features such as using morphology as a similarity measure.

8.2 Comprehensive Evaluation

Table 8.5 and Table 8.6 present the performance of Wrapster on the dataset.¹ Wrapster identified and generated extraction rules correctly for most of the content regions

¹The IMDb movie and The World Factbook sites were used both in the evaluation and in training the classifier (see Chapter 3). Objects were randomly selected, separately, for both these purposes, so that any overlap between selected objects is coincidental.

<i>Feature set</i>	<i>F-Measure</i>	<i>Precision</i>	<i>Recall</i>
RULE BASED CONTENT	63.23	58.0	69.5
RULE BASED CONTENT + CONTEXTS	90.58	98.7	83.7
CONTENT	91.04	98.7	84.5
PREVIOUS CONTEXT	87.71	86	89.5
CONTENT + PREVIOUS CONTEXT	94.57	97.3	92
CONTENT + NEXT CONTEXT	95.72	97.3	94.2
NEXT CONTEXT	87.28	84.2	90.6
FULL	95.46	97.3	93.7

Table 8.3: SVM feature sets and performance on all training data

<i>Feature set</i>	<i>F-Measure</i>	<i>Precision</i>	<i>Recall</i>
RULE BASED CONTENT SCORE	49.07	44.5	54.7
RULE BASED CONTENT + CONTEXTS SCORE	92.21	97.1	87.8
CONTENT	93.6	97.3	90.2
CONTENT + PREVIOUS CONTEXT	94.797	95.6	94.0
PREVIOUS CONTEXT	86.71	92.9	81.3
CONTENT + NEXT CONTEXT	97.68	99.0	96.4
NEXT CONTEXT	87.59	83.6	92
FULL	97.89	98.2	97.6

Table 8.4: SVM feature sets and performance on IMDB movie training data

<i>Web sites</i>	<i>Regions discovered</i>	<i>Scripts generated</i>	<i>Regions discovered Recall</i>
IMDb Movie	39	38	100%
IMDb Person	31	5	100%
The World Factbook	198	164	82.9%
POTUS	26	20	88%
MSN Money	18	5	46.4%
The Weather Channel	26	16	62.5%
50 States	64	47	81.8%

Table 8.5: Wrapper generation content discovery

<i>Web sites</i>	<i>Regions discovered</i>	<i>Number of scripts generated</i>
IMDb Movie	43	38
IMDb Person	46	29
The World Factbook	203	161
POTUS	23	20
MSN Money	43	41
The Weather Channel	20	19
50 States	63	48

Table 8.6: Wrapper generation whole evaluation on the updated Web sites on April 23rd

from the test sites. Wrapster identified on average 88.22% of the content regions for those sites. However, there are some cases where multiple scripts are generated for the same region because our classifier, as mentioned earlier, fails to classify correctly cases such a “Actress Filmography” and “Actor Filmography”. This mis-classification can be exploited in the future to discover underlying properties on the Web pages such as gender. We can notice that Wrapster was able to generate a significant proportion of scripts for updated versions for most of the sites. The sites that benefited the most are IMDb person and MSN Money. The wrapper scripts generated by Wrapster increased from 5 scripts each to 29 and 41 respectively, based on which we may conjecture that these sites changed their layouts in order to increase consistency.

An interesting experiment is to evaluate Wrapster’s repair module on the IMDb movie site because it experience a single recent change and demonstrates how much a site may change in a single update. It identified 100% of regions for the site and was

able to attach with 36.36% accuracy the semantic information such as the names of the scripts from the old wrapper. Even though the accuracy is low for migrating the semantic information, we are working to improve it in the near future work and it doesn't damage Wrapster performance significantly since the user interface provides us the tools to add the semantic information easily.

Chapter 9

Conclusion and Future Work

Many available information sources on the Web are semi-structured, which creates an opportunity for automatic tools to process and extract their information for easy access through a uniform interface language using wrappers. A wrapper is an automatic tool for information extraction from a semi-structured source. Although semi-structured sources possess syntactic regularities, it is not trivial to process and extract information from those sources because they are not uniform in format and are frequently updated, making it hard to create flexible wrappers that can adapt to changing sources, and to scale wrapper-based systems.

In this thesis we presented Wrapster, a wrapper generation system for detail pages which leverages the HTML structure and reduces the amount of training data needed, using an active learning technique. The interactive Web user interface enables people with little or no programming skills to create and edit wrappers. In addition, once a wrapper for a site is created, Wrapster checks the correctness of all stored wrappers and automatically fixes their failing scripts.

9.1 Contributions

We have made the following contributions in this research area:

1. We developed a platform-independent, end-to-end wrapper generation system.

2. We applied *tree edit distance* to construct the template of a detail page.
3. We introduced a general wrapper representation that includes not only text landmarks, but also region instances, extraction rules, and semantic annotations which aid in repair and integration.
4. We reduced the amount of training examples needed to create a wrapper using an active learning technique.
5. We implemented an interactive Web user interface so that a user can edit the generated wrapper without the need for programming skills.
6. We developed a wrapper repair module that checks the correctness of the wrappers and fixes any failing scripts.

9.2 Future Work

Our work with Wrapster suggests several interesting topics for future exploration:

1. Test different approaches for template creation such as sequence alignment and visual rendering alignment. In addition, create nested templates for sites where an object spans over multiple Web pages.
2. Experiment with smart features such as using morphology and value type as similarity measures to improve the performance of the classifier.
3. Try various metrics for region clustering.
4. Develop and use rich pattern libraries such as Hap-Shu and LAPIS that leverage the HTML structure and are more robust to format changes than simple regular expressions. In addition, build an interactive Web tool to refine scripts using a collaborative learning framework such as PLOW [2].
5. Augment user interface functionalities to combine simple scripts into new complex scripts.

6. Create user interface for repair; allow the user to align new and old templates in order to transfer semantic information from old scripts to new; store user alignments as training data to improve the repair module's automatic alignment of new and old templates.
7. Automatically annotate the generated scripts with possible questions for each script.
8. Eliminate the manual input for Wrapster and build a tool that crawls a Web site and discovers its objects.
9. Automate the selection of objects within a site that we use to generate the wrapper, using heuristics, to improve the odds of finding all regions for that site.
10. Allow for repair on a configured subset of objects in a site by storing values for them; use heuristics to repair scripts for all objects by examining value type and other features rather than by using stored values.

Bibliography

- [1] Document Object Model (DOM). <http://www.w3.org/DOM/>.
- [2] James Allen, Nathanael Chambers, George Ferguson, Lucian Galescu, Hyuckchul Jung, Mary Swift, and William Taysom. Demonstration of PLOW: A dialogue system for one-shot task learning. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 1–2, Rochester, New York, USA, April 2007. Association for Computational Linguistics.
- [3] Arvind Arasu and Hector Garcia-Molina. Extracting structured data from Web pages. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on management of data*, pages 337–348, New York, NY, USA, 2003. ACM Press.
- [4] Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Visual Web Information Extraction with Lixto. In *The VLDB Journal*, pages 119–128, 2001.
- [5] Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. Automation and customization of rendered web pages. In *UIST '05: Proceedings of the 18th annual ACM symposium on user interface software and technology*, pages 163–172, New York, NY, USA, 2005. ACM Press.
- [6] David Buttler, Ling Liu, and Calton Pu. A Fully Automated Object Extraction System for the World Wide Web. In *Proceedings of the 2001 International Conference on Distributed Computing Systems (ICDCS'01)*, pages 361–370, Phoenix, Arizona, May 2001.

- [7] Mary Elaine Califf. *Relational Learning Techniques for Natural Language Information Extraction*. PhD thesis, Department of Computer Sciences, University of Texas, Austin, TX, August 1998. Also appears as Artificial Intelligence Laboratory Technical Report AI 98-276 (see <http://www.cs.utexas.edu/users/ai-lab>).
- [8] Mary Elaine Califf and Raymond J. Mooney. Bottom-up relational learning of pattern matching rules for information extraction. *Journal of Machine Learning Research*, 4:177–210, 2003.
- [9] Chia-Hui Chang and Shao-Chen Lui. IEPAD: information extraction based on pattern discovery. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 681–688, New York, NY, USA, 2001. ACM Press.
- [10] Andrew Clark. Cyberneko HTML Parser. <http://people.apache.org/~andyc/neko/doc/index.html>.
- [11] William W. Cohen, Matthew Hurst, and Lee S. Jensen. A flexible learning system for wrapping tables and lists in HTML documents. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 232–241, New York, NY, USA, 2002. ACM Press.
- [12] Richard Cole, Ramesh Hariharan, and Piotr Indyk. Tree pattern matching and subset matching in deterministic $O(n \log^3 n)$ -time. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1999.
- [13] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 2nd edition, 2001.
- [14] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 109–118, 2001.
- [15] Yihong Ding and David W. Embley. Using Data-Extraction Ontologies to Foster Automating Semantic Annotation. In *ICDEW '06: Proceedings of the 22nd*

International Conference on Data Engineering Workshops (ICDEW'06), page 138, Washington, DC, USA, 2006. IEEE Computer Society.

- [16] M. Dubiner, Z. Galil, and E. Magen. Faster Tree Pattern Matching. In *Proc. of the Symposium on Foundations of Computer Science (FOCS'90)*, pages 145–150, 1990.
- [17] David W. Embley, Yue Jiang, and Yiu-Kai Ng. Record-boundary discovery in Web documents. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 467–478, 1999.
- [18] Dayne Freitag. Information Extraction from HTML: Application of a General Machine Learning Approach. *AAAI/IAAI*, 1998.
- [19] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective Sampling Using the Query by Committee Algorithm. *Machine Learning*, 28(2-3):133–168, 1997.
- [20] Christoph M. Hoffmann and Michael J. O'Donnell. Pattern Matching in Trees. *Journal of the ACM*, 29(1):68–95, January 1982.
- [21] Andrew Hogue. Tree Pattern Inference and Matching for Wrapper Induction on the World Wide Web. Master's thesis, MIT, 2004.
- [22] Andrew Hogue and David Karger. Thresher: automating the unwrapping of semantic content from the World Wide Web. *WWW*, 2005.
- [23] John E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs Preliminary Report. In *STOC '74: Proceedings of the sixth annual ACM symposium on theory of computing*, pages 172–184, New York, NY, USA, 1974. ACM Press.
- [24] Chun-Nan Hsu and Ming-Tzung Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems*, 23(9):521–538, 1998.

- [25] David F. Huynh, Robert C. Miller, and David R. Karger. Enabling web browsers to augment web sites' filtering and sorting functionalities. In *UIST '06: Proceedings of the 19th annual ACM symposium on user interface software and technology*, pages 125–134, New York, NY, USA, 2006. ACM Press.
- [26] Thorsten Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.
- [27] D. Shasha K. Zhang. Simple fast algorithms for the editing distance between trees and related problems. In *SIAM J. Computing*, number 18 (6), pages 1245–1262, December 1989.
- [28] Boris Katz. Using English for Indexing and Retrieving. Technical Report AIM-1096, 1988.
- [29] Boris Katz. Annotating the World Wide Web Using Natural Language. In *Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet (RIAO '97)*, Montreal, Canada, 1997.
- [30] Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran. Omnibase: Uniform Access to Heterogeneous Data for Question Answering. In *Proc. of the 7th Int. Workshop on Applications of Natural Language to Information Systems (NLDB '02)*, Stockholm, Sweden, June 2002.
- [31] Boris Katz and Jimmy J. Lin. Start and Beyond. <http://citeseer.ist.psu.edu/556306.html>, 2002.
- [32] Nicholas Kushmerick, Daniel S. Weld, and Robert B. Doorenbos. Wrapper Induction for Information Extraction. In *Intl. Joint Conference on Artificial Intelligence (IJCAI)*, pages 729–737, 1997.
- [33] Alberto H. F. Laender, Berthier Ribeiro-Neto, and Altigran S. da Silva. DEByE - Data extraction by example. *Data Knowledge Engineering*, 40(2):121–154, 2002.

- [34] Zhao Li and Wee Keong Ng. WICCAP: From Semi-structured Data to Structured Data. *Engineering of Computer-Based Systems*, 00:86, 2004.
- [35] Bing Liu, Robert Grossman, and Yanhong Zhai. Mining data records in Web pages. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining*, pages 601–606, New York, NY, USA, 2003. ACM Press.
- [36] David W. Matula. An Algorithm for Subtree Identification. *SIAM Review*, 10:273–274, 1968. Abstract.
- [37] Robert C. Miller and Brad A. Myers. LAPIS: smart editing with text structure. In *CHI '02: CHI '02 extended abstracts on human factors in computing systems*, pages 496–497, New York, NY, USA, 2002. ACM Press.
- [38] Ion Muslea. RISE: Repository of online information sources used in information extraction tasks. <http://www.isi.edu/info-agents/RISE/>, 1998.
- [39] Ion Muslea, Steve Minton, and Craig Knoblock. A hierarchical approach to wrapper induction. In *AGENTS '99: Proceedings of the third annual conference on Autonomous Agents*, pages 190–197, New York, NY, USA, 1999. ACM Press.
- [40] Juan Raposo, Alberto Pan, Manuel Álvarez, Justo Hidalgo, and Ángel Viña. The Wargo System: Semi-Automatic Wrapper Generation in Presence of Complex Data Access Modes. In *DEXA '02: Proceedings of the 13th International Workshop on Database and Expert Systems Applications*, pages 313–320, Washington, DC, USA, 2002. IEEE Computer Society.
- [41] Davi De Castro Reis, Paulo B. Golgher, Alberto H.F. Laender, and Altigran S. da Silva. Automatic web news extraction using tree edit distance. *WWW*, 2004.
- [42] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Journal of Machine Learning*, 34(1-3):233–272, 1999.

- [43] Hui Song, Suraj Giri, and Fanyuan Ma. Data Extraction and Annotation for Dynamic Web Pages. In *EEE '04: Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04)*, pages 499–502, Washington, DC, USA, 2004. IEEE Computer Society.
- [44] Baris Temelkuran. Hap-Shu: A Language for Locating Information in HTML Documents. Master's thesis, Massachusetts Institute of Technology, May 2003.
- [45] Vladimir Vapnik. *Statistical Learning Theory*. Wiley-Interscience, New York, 1998.
- [46] Eric W. Weisstein. Isomorphic Graphs. <http://mathworld.wolfram.com/IsomorphicGraphs.html>. Wolfram MathWorld—the web most extensive mathematics resource, accessed 21 May 2007.
- [47] Yanhong Zhai and Bing Liu. Web data extraction based on partial tree alignment. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 76–85, New York, NY, USA, 2005. ACM Press.