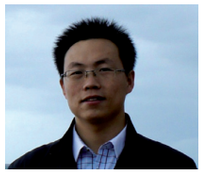
In this book, we develop two frameworks to tackle the task of semi-structured Web data record extraction. We first present a record segmentation search tree framework in which a new search structure, named Record Segmentation Tree (RST), is designed and several efficient search pruning strategies on the RST structure are proposed to identify the records in a given Web page. We also present another DOM Structure Knowledge Oriented Global Analysis (Skoga) framework which can perform robust detection of different kinds of data records and record regions. Skoga can conduct a global analysis on the DOM structure to achieve effective detection. Finally, we present a framework that can make use of the detected data records to automatically populate existing Wikipedia categories. This framework takes a few existing entities that are automatically collected from a particular Wikipedia category as seed input and explores their attribute infoboxes to obtain clues for the discovery of more entities for this category and the attribute content of the newly discovered entities.



Lidong Bing

He is currently a postdoc fellow in The Chinese University of Hong Kong, where he received his PhD in 2012. Before that, he obtained his MPhil and BSc degrees from Peking University and Northeast Normal University respectively. He has research interests in Information Extraction, Information Retrieval, Web Mining, Natural Language Processing, etc.

978-3-659-20611-5

Information Discovery from Web Records

Lidong Bing
Wai Lam

Information Discovery from Semi-structured Record Sets on the Web

From Web Pages to Knowledge

LAP LAMBERT Academic Publishing

Bing, Lam

Lidong Bing
Wai Lam

Information Discovery from Semi-structured Record Sets on the Web

Lidong Bing
Wai Lam

Information Discovery from Semi-structured Record Sets on the Web

From Web Pages to Knowledge

LAP LAMBERT Academic Publishing

Impressum / Imprint
Bibliografische Information der Deutschen Nationalbibliothek: Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Bibliographic information published by the Deutsche Nationalbibliothek: The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Coverbild / Cover image: www.ingimage.com

Verlag / Publisher:
LAP LAMBERT Academic Publishing
ist ein Imprint der / is a trademark of
OmniScriptum GmbH & Co. KG
Heinrich-Böcking-Str. 6-8, 66121 Saarbrücken, Deutschland / Germany
Email: info@lap-publishing.com

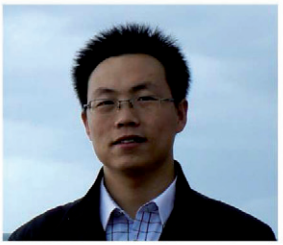
Herstellung: siehe letzte Seite /
Printed at: see last page
ISBN: 978-3-659-20611-5

Zugl. / Approved by: Hong Kong, The Chinese University of Hong Kong, Diss., 2012

Information Discovery from Semi-structured Record Sets on the Web

Lidong Bing Lam Wai

About the Authors



Lidong Bing received his Ph.D. degree in the Department of Systems Engineering and Engineering Management at The Chinese University of Hong Kong in 2012, Before that, he received his M.Phil. degree in Computer Application Technology from Peking University in 2009, and BSc. degree in Computer Science and Technology from Northeast Normal University in 2006. He is currently a postdoctoral fellow in The Chinese University of Hong Kong. His research interests include Information Extraction, Information Retrieval, Web Mining, Natural Language Processing, etc. His research results have been published in ACM Transactions on the Web, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Conference on Information and Knowledge Management, ACM International Conference on Web Search and Data Mining, etc. He has good connections and collaborations with large technology companies, such as Alibaba, Baidu, Nosh's Ark Lab of Huawei, etc.

Lam Wai received a Ph.D. in Computer Science from the University of Waterloo. He obtained his BSc. and M.Phil. degrees from The Chinese University of Hong Kong. After completing his Ph.D. degree, he conducted research at Indiana University Purdue University Indianapolis (IUPUI) and the University of Iowa. He joined The Chinese University of Hong Kong, where he is currently a professor. His research interests include intelligent information retrieval, text mining, digital library, machine learning, and knowledge-based systems. He has published articles in IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Knowledge and Data Engineering, ACM Transactions on Information Systems, etc. His research projects have been funded by the Hong Kong SAR Government General Research Fund (GRF) and DARPA (USA). He also managed industrial projects funded by Innovation and Technology Fund (industrial grant) and IT companies.

Abstract

The World Wide Web has been extensively developed since its first appearance two decades ago. Various applications on the Web have unprecedentedly changed humans' life. Although the explosive growth and spread of the Web have resulted in a huge information repository, yet it is still under-utilized due to the difficulty in automated information extraction (IE) caused by the heterogeneity of Web content. Thus, Web IE is an essential task in the utilization of Web information. Typically, a Web page may describe either a single object or a group of similar objects. For example, the description page of a digital camera describes different aspects of the camera. On the contrary, the faculty list page of a department presents the information of a group of professors. Corresponding to the above two types, Web IE methods can be broadly categorized into two classes, namely, description details oriented extraction and object records oriented extraction. In this book, we focus on the later task, namely semi-structured data record extraction from a single Web page.

In this book, we develop two frameworks to tackle the task of data record extraction. We first present a record segmentation search tree framework in which a new search structure, named Record Segmentation Tree (RST), is designed and several efficient search pruning strategies on the RST structure are proposed to identify the records in a given Web page. The subtree groups corresponding to possible data records are dynamically generated in the RST structure during the search process. Therefore, this framework is more flexible compared with existing methods such as MDR and DEPTA that have a static manner of generating subtree groups. Furthermore, instead of using string edit distance or tree edit distance, we propose a token-based edit distance which takes each DOM node as a basic unit in the cost calculation.

Many existing methods, including the RST framework, for data record extraction from Web pages contain pre-coded hard criteria and adopt an exhaustive search strategy for

traversing the DOM tree. They fail to handle many challenging pages containing complicated data records and record regions. In this book, we also present another DOM Structure Knowledge Oriented Global Analysis (Skoga) framework which can perform robust detection of different kinds of data records and record regions. Skoga can conduct a global analysis on the DOM structure to achieve effective detection. The DOM structure knowledge consists of background knowledge as well as statistical knowledge capturing different characteristics of data records and record regions as exhibited in the DOM structure. Specifically, the background knowledge encodes some logical relations governing certain structural constraints in the DOM structure. The statistical knowledge is represented by some carefully designed features that capture different characteristics of a single node or a node group in the DOM. The feature weights are determined using a development data set via a parameter estimation algorithm based on structured output Support Vector Machine model which can tackle the inter-dependency among the labels on the nodes of the DOM structure. An optimization method based on divide and conquer principle is developed making use of the DOM structure knowledge to quantitatively infer the best record and region recognition.

Finally, we present a framework that can make use of the detected data records to automatically populate existing Wikipedia categories. This framework takes a few existing entities that are automatically collected from a particular Wikipedia category as seed input and explores their attribute infoboxes to obtain clues for the discovery of more entities for this category and the attribute content of the newly discovered entities. One characteristic of this framework is to conduct discovery and extraction from desirable semi-structured data record sets which are automatically collected from the Web. A semi-supervised learning model with Conditional Random Fields is developed to deal with the issues of extraction learning and limited number of labeled examples derived from the seed entities. We make use of a proximate record graph to guide the semi-supervised leaning process. The graph captures alignment similarity among data records. Then the semi-supervised learning process can leverage the benefit of the unlabeled data in the record set by controlling the label regularization under the guidance of the proximate record graph.

I dedicate this book to my family that I love dearly.
I am deeply grateful to every person in my life!

– Lidong Bing

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Web Era and Web IE

The World Wide Web has been extensively developed since its first appearance two decades ago. Various applications on the Web have unprecedentedly changed humans' life. Web search provides us a fast and accurate access to the useful information on the Web. Online encyclopedia contains human knowledge in different areas and makes it accessible to every Web user. Online shopping saves us the time for searching items in the malls. Corresponding to the various applications on the Web, some important and challenging research topics have attracted a lot of attention from the research community. To facilitate a better Web search experience for the users, several directions are well studied such as search result ranking [57, 37], query log analysis [34, 62] and query reformulation [82, 35, 5]; Online encyclopedias such as Wikipedia are employed to upgrade the performance of document classification and clustering methods [31, 53, 78] as well as to generate a hybrid ontology by merging them with the expert-edit ontology [60, 63, 32, 8]; Different methods in Web information extraction (IE) are proposed to tackle the problem of extracting useful information from different types of Web pages, such as product description details extraction [84, 85, 92] and Web data record extraction [4, 90, 49].

Although the explosive growth and spread of the Web have resulted in a huge information repository, it is still under-utilized due to the difficulty in automated information extraction caused by data's heterogeneity. Therefore, Web IE is an essential task in the utilization of Web information. Traditional information extraction dates back to the late 1970s [15]. It aimed at extracting a particular kind of information from basically unstruc-

tured free text in the early days by natural language processing (NLP) method [47]. In contrast, Web IE deals with Web documents (or Web pages) which are semi-structured and coded with HTML [13]. Except the NLP based methods, some new features of Web documents are leveraged in the task of Web IE such as text's appearance, HTML tag template structured, etc. The output of Web IE may be structured data records or free text fragments having specific meaning.

Typically, a Web page may describe either a single object or a group of similar objects. For example, the description page of a digital camera describes different aspects of the camera. On the contrary, the faculty list page of a department presents the information of a group of professors. Corresponding to the above two types, Web IE methods can be broadly categorized into two classes, namely, description details oriented extraction [85, 92] and object records oriented extraction [42, 49]. The former aims at extracting the description details of a single object from its description page, while the later aims at extracting a set of similar object records. In this book, we focus on the later task, namely semi-structured data record extraction from a single Web page. Furthermore, we also present a framework that can make use of the detected data records to automatically populate existing Wikipedia categories.

1.2 Semi-structured Record and Region Detection

1.2.1 Problem Setting

Many Web sites make use of pre-designed templates to format and present information units, known as *data records*, which have similar attributes or fields. Some sites prefer to display the record information in a semi-structured way in static Web pages to facilitate easy browsing, such as a list of faculty, a list of breaking events, etc. Some sites run a server-side program to fill products' information, retrieved from back-end databases, in a pre-defined template to generate Web pages, which are referred to as deep or dynamic Web pages. Therefore, semi-structured information on the Web is tremendously popular. If such information can be exploited, it is very useful for developing various applications such as online market intelligence, knowledge base population, etc.

Two types of data record regions on the Web are given in Figure 1.1, namely, a flat record region in Figure 1.1(a) and a nested record region in Figure 1.1(b). Web page is

normally represented in a structure known as DOM [48] which is a programming interface specification being developed by the World Wide Web Consortium (W3C) and it lets a programmer create and modify HTML pages and XML documents as full-fledged program objects. The DOM structures of the examples in Figures 1.1(a) and 1.1(b) are given in Figures 1.1(c) and 1.1(d). In the DOM tree corresponding to the flat region, each row of the table, excluding the header row $\mathcal{S}_1$, is a data record, such as record R_1 corresponding to $\mathcal{S}_2$ and record R_2 corresponding to $\mathcal{S}_3$. While in the nested region, each row of the table contains three data records and it can be regarded as a subregion. Figure 1.2 depicts two more complicated record regions, namely, a complicated flat record in Figure 1.2(a) with its DOM given in Figures 1.2(c) and an intertwined record region in Figure 1.2(b) with its DOM given in Figure 1.2(d). In the complicated flat region, each data record is composed of several table rows. For example, record R_1 is composed of three rows, namely, $\mathcal{S}_2$, $\mathcal{S}_3$, and $\mathcal{S}_4$. In the intertwined region, different fields of R_1 and R_2 are intertwined in the first three rows.

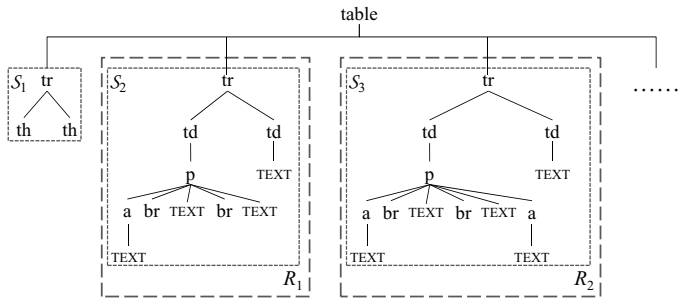
Consider a Web page and its corresponding DOM tree structure. The extraction of data records from the page is equivalent to identifying the record region subtree and the data record subtrees in the DOM tree structure. Returning to the example in Figures 1.1(a) and 1.1(c), a record detection algorithm should identify that the subtree corresponding to the <table> tag is a *record region*. Also the subtrees $\mathcal{S}_2$ and $\mathcal{S}_3$ should be identified as *data records*. For the nested region in Figures 1.1(b) and 1.1(d), the subtree corresponding to the <table> tag should be identified as a *record region*. Furthermore, the subtrees corresponding to the <tr> tags should be identified as *subregions* of records, and the subtrees corresponding to the <td> tags should be identified as *data records*.

1.2.2 Observations and Challenges

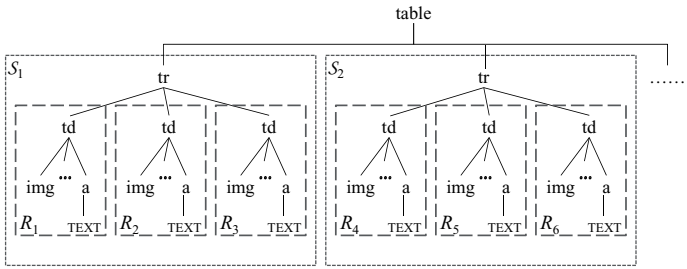
From the examples given in Figures 1.1 and 1.2, two basic observations on data record and record region can be perceived: 1) A group of data records describing a set of similar objects are typically presented in a contiguous region, known as record region, of a page and are formatted using similar HTML tags; 2) A set of similar data records are formed by some child subtrees of the same parent node. Based on these observations, one way to detect record region and record boundary is to examine the similarity (or distance) between subtrees. This strategy is adopted by MDR [42], ViPER [67], and DEPTA [91].

(a) A page fragment of flat region.

(b) A page fragment of nested region.



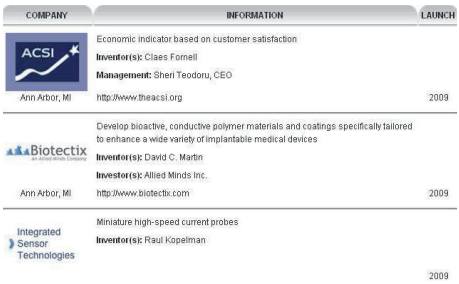
(c) DOM tree of (a).



(d) DOM tree of (b).

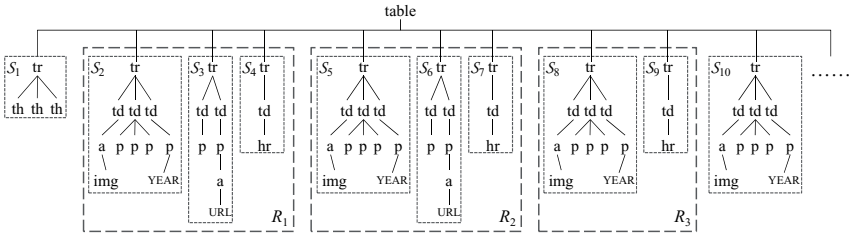
Figure 1.1: Flat and nested record regions.

MDR utilizes string edit distance to assess whether two adjacent subtree groups, named *generalized nodes*, are a repetition of the same data type. ViPER is another work which also calculates string edit distance of the subtree pairs to detect record region. Then it involves some visual information to segment a detected region into records. DEPTA calculates tree edit distance between generalized nodes. In summary, MDR and DEPTA

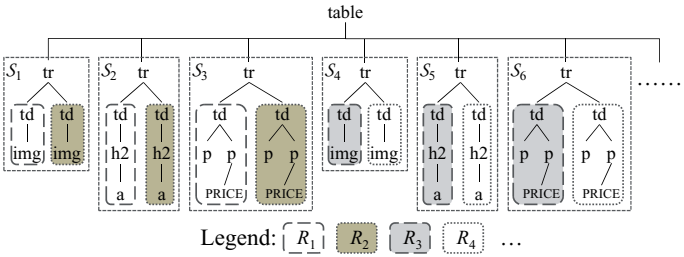


(a) A page fragment of complicated flat region.

(b) A page fragment of intertwined region.



(c) DOM tree of (a).



(d) DOM tree of (b).

Figure 1.2: Complicated flat and intertwined record regions.

calculate similarity for the pairs of neighboring generalized nodes, while ViPER calculates similarity for every pair of subtrees.

The advantage of the above similarity-based approaches is that they can tolerate the optional fields or tags in a record. They can also tackle the problem of approximate matching to identify repeating objects. However, one disadvantage is that their method of determining the granularity involved in the similarity calculation is not flexible. In a record region, records may contain different number of subtrees, as shown in Figure 1.2(c).

Thus, the number of possible subtree combinations to compose records is exponential with respect to the number of subtrees. To tackle this problem, MDR and DEPTA introduce a concept known as generalized node, which contains several subtrees. The limitation of the MDR family methods [42, 90, 91] is the greedy search for the best generalized node combination, which is time consuming as noticed by other works [49, 67]. To speed up the greedy search, one inflexible constraint is introduced in their definition of record region. This constraint specifies that all generalized nodes must have the same length, i.e. the number of subtrees, in the same record region. This constraint will fail to handle some cases in which the records contain different number of subtrees. For the record region in Figure 1.2(c), MDR and DEPTA identify that the subtree sequence of $\mathcal{S}_2 \cdots \mathcal{S}_7$ is one record region and contains two data records namely R_1 and R_2, but the record R_3 is missed. Such difficulties are often observed from the regions like the ones in Figure 1.2. Noticing this limitation, ViPER only calculates the similarity for single subtree pairs and constructs a similarity matrix. Then with this matrix, some heuristic rules are employed to detect the record region. Another disadvantage of these existing methods is that they decompose record extraction task into two separate steps, namely, record region detection, and record segmentation. Thus, given a subtree sequence such as $\mathcal{S}_1\mathcal{S}_2 \cdots \mathcal{S}_n$ in Figure 1.2(c), these methods first detect whether it contains some record regions. Then, they will further segment the detected regions into data records. The reason they adopted this two-step approach is that the calculated similarity or distance in region detection, namely, the edit distance of generalized nodes in MDR, and the similarity of pairwise subtrees in ViPER, cannot be utilized to decide record boundary directly, since the calculated information is not necessarily boundary related.

Another major limitation of most existing works such as MDR [42], DEPTA [91], ViPER [67], and TPC [49] is that they contain pre-coded hard criteria related to characteristics embedded in the HTML source code or related to visual perception. When processing a new page, the pre-coded hard criteria, such as the similarity between the subtree groups in MDR, DEPTA, and tag path similarity in TPC, are applied to determine whether a fragment of Web page should be recognized as a record region. After that, the next step is to detect the data records inside the region. Despite the fact that these methods can be applied to certain kinds of Web pages, various types of data record formation make it infeasible to use some pre-coded hard criteria to accurately capture different

types of formatting manners. For example, in the DOM structure shown in Figure 1.1(d), the hard criteria would mistakenly treat the three data records under the subtree $\mathcal{S}_1$ as three different aspects of a single record. Moreover, if noise subtrees exist in the subtree sequences given in Figures 1.1 and 1.2, the performance of existing methods will be inevitably affected. It is because their greedy search manner in region detection and record segmentation has the limitation of myopic analysis and is lacking of a global analysis to quantitatively differentiate a variety of possible style of data records and record regions. Therefore, a quantitative measure is needed to accurately detect records and regions in a global manner.

1.2.3 Our Proposed First Framework - Record Segmentation Tree

As discussed above, the similarity based methods have limitations in the manner of determining the subtree group granularity. Because of these limitations, they decompose record extraction task into two separate steps, namely, record region detection, and record segmentation. To tackle these limitations, we propose a new model known as Record Segmentation Tree (RST) method. The main contributions of the RST method are as follows:

- The RST method examines the similarity between the dynamically generated subtree groups taking into account the characteristics of the current record region. Therefore, these subtree groups may have different length. Furthermore, the two steps in the existing methods are combined in a unified model with the advantage that the dynamically generated subtree groups are record boundary related.

- Based on the new search structure provided by RST, several key issues in record extraction are handled in a uniform manner. For example, whether the given region contains a record region, if so, from which subtree it starts, where it ends, and how to segment it into records.

- To obtain solutions for these issues, we develop several efficient search pruning strategies on the RST structure of a given region to identify the correct record segmentation. When a correct segmentation is successfully detected, it implies that the record region is also successfully obtained. Based on such approach, our method is able to combine the two steps in the existing works in a unified model.

- The research effort inspiring or producing this framework has been published in the ACM international Conference on Information and Knowledge Management 2011 [4].

1.2.4 Our Proposed Second Framework - DOM Structure Knowledge Oriented Global Analysis

To overcome the limitations that existing methods need some pre-coded hard criteria and merely conduct myopic analysis to better tackle various types of data records and record regions, we present a more powerful DOM Structure Knowledge Oriented Global Analysis (Skoga) framework which can perform robust detection of different kinds of data records and record regions as exemplified in Figures 1.1 and 1.2. The main contributions of our Skoga framework are as follows:

- This framework has a DOM structure knowledge driven detection model as well as a record segmentation search tree model. The DOM structure knowledge consists of background knowledge as well as statistical knowledge capturing different characteristics of data records and record regions as exhibited in the DOM structure. Specifically, the background knowledge encodes some logical relations governing certain structural constraints in the DOM structure. The statistical knowledge is represented by some carefully designed features that capture different characteristics of a single node or a node group in the DOM such as the similarity of the neighboring subtrees, the similarity of one subtree with its siblings, etc.

- Different from most existing works such as MDR, DEPTA, ViPER and RST which adopt a exhaustive search manner by traversing the DOM tree to detect data record regions, this framework can conduct a global analysis to achieve effective recognition. For the cases shown in Figures 1.1 and 1.2, the data records and record regions can be identified directly.

- An optimization method based on divide and conquer principle is developed making use of the DOM structure knowledge to quantitatively infer the best record and region recognition for a page. Taking the flat region in Figure 1.1(c) as an example, this framework can quantitatively assign the highest score to a recognition that correctly identifies $\mathcal{S}_2$ and $\mathcal{S}_3$ as data records and the subtree corresponding to the

<table> node as a record region. For the nested example in Figure 1.1(d), this framework can accurately identify the subtrees corresponding to the <td> nodes as data records. At the same time, it identifies the subtrees corresponding to the <tr> nodes as subregions of data records even though these subtrees have similar structure. Note that some existing methods such as MDR [42], DEPTA [91], and RST [4] would wrongly identify them as data records.

- The DOM structure knowledge in this framework can also evaluate the regularity of the subtree sequences as exemplified in Figure 1.2. Specifically, a particular subtree structure occurs regularly across the subtree sequences, such as the structure shared by $\mathcal{S}_2$, $\mathcal{S}_5$, $\mathcal{S}_8$ and $\mathcal{S}_{10}$ in Figure 1.2(c) and the structure shared by $\mathcal{S}_1$ and $\mathcal{S}_4$ in Figure 1.2(d). This framework is also able to overcome the limitation of requiring the generalized nodes to be of the same length in the same region as in MDR and DEPTA.

- To allow different impacts for different features, there is a weight associated with each feature. The feature weights in the DOM structure knowledge are determined using a development data set via a parameter estimation algorithm based on structured output Support Vector Machine model [73] which can handle inter-dependency among the labels on the nodes of the DOM structure. A record region oriented loss function is designed so that the acquired statistical knowledge can deal with multiple regions in one page. The development data set was arbitrarily collected from different Web sites covering different kinds of record regions such as the examples given in Figures 1.1 and 1.2. Once the DOM structure knowledge including the feature weights is determined, this framework can be directly applied to detect common kinds of record regions and data records as illustrated above from any Web sites and domains without the need of labeled data or training.

- The research effort inspiring or producing this framework has been published in ACM Transactions on the Web [6].

One Droopy Knight	
Droopy series	
Directed by	Michael Lah
Produced by	William Hanna Joseph Barbera
Story by	Homer Brightman
Narrated by	Bill Thompson
Voices by	Bill Thompson Daws Butler
Music by	Scott Bradley
Animation by	Bill Schipek Ken Southworth Irvin Spence Herman Cohen
Layouts by	Ed Benedict
Backgrounds by	F. Montealegre

Figure 1.3: The infobox of a cartoon entity.

1.3 Entity Expansion and Attribute Acquisition with Semi-structured Data Records

1.3.1 Problem Setting

Semi-structured Web data records are very useful for developing various applications such as Wikipedia category population. As a remarkable and rich online encyclopedia, Wikipedia provides a wealth of general knowledge about various aspects. Some Wikipedia articles have a structured information block known as infobox, as exemplified in Figure 1.3, typically on the upper right of the Wikipedia page. An infobox is composed of a set of attribute name and value pairs that summarize the key information of the entity. Infoboxes are extensively explored in some existing projects such as DBpedia [3], Freebase [9], and YAGO [70, 71]. Although Wikipedia already covers a large number of popular entities, many entities have not been included.

Figure 1.4 presents a fragment of a Web page containing a semi-structured list that describes some cartoon films. After manual checking, we find that quite a number of cartoons in this list have not been covered by Wikipedia. For example, no existing Wikipedia entities describe the cartoons numbered 2, 3, 100 and 101. On the other hand, the ones numbered 319 to 321 in the same list are well contained by Wikipedia and each of them

2: CLEANING HOUSE
Rel 2/19/38
Supervised by Robert Allen

3: BLUE MONDAY
Rel 4/2/38
Supervised by William Hanna

... ...
... ...

100: WHAT'S BUZZIN' BUZZARD?
Rel 11/27/43
Directed by Tex Avery; Animation: Ed Love, Ray Abrams, Preston Blai

101: THE STORK'S HOLIDAY
Rel 10/23/43
Directed by George Gordon; Animation: Michael Lah, Rudy Zamora, C
Al Grandmain; Story: Otto Englander, Webb Smith

... ...
... ...

319: SCAT CATS
Rel 7/26/57
Directed by Joseph Barbera and William Hanna;
Animation: Kenneth Muse, Carlo Vinci, Lewis
Marshall; Story: Homer Brightman; Layout: Dick
Bickenbach; Backgrounds: Robert Gentle

320: ONE DROOPY KNIGHT
Rel 12/6/57
Directed by Michael Lah; Animation: Bill Schipek, Ken Southworth, Ir
Story: Homer Brightman; Layout: Ed Benedict; Backgrounds: F. Monte

321: FEEDIN' THE KIDDIE
Rel 6/7/57
Directed by Joseph Barbera and William Hanna; Animation: Irvin Spen
Ray Patterson; Layout: Dick Bickenbach; Backgrounds: Don Driscoll; F
ORPHAN.

Figure 1.4: A Web page fragment of MGM cartoon list.

has a description article and an infobox. Therefore, with these records that are already described by some Wikipedia entities as clues, we may infer that the remaining records in the list shown in Figure 1.4 are talking about the same type of entities. For example, one can infer that, from the record number 100, there exists a cartoon with title "`What's Buzzin' Buzzard`" which does not exist in Wikipedia. Furthermore, considering the infobox of cartoon entity "`One Droopy Knight`" as shown in Figure 1.3 and its corresponding data record, i.e. record number 320 in Figure 1.4, one can infer that the text fragment "`Directed by Michael Lah`" in the record 320 is related to the director attribute and its value. As a result, one can further infer that the director attribute of the newly discovered entity "`What's Buzzin' Buzzard`" is associated with the value

"`Tex Avery`".

Inspired by the above observation, we develop a framework to achieve the goal of new entity discovery and attribute extraction for Wikipedia categories by mining the rich and valuable semi-structured data records on the Web as exemplified in Figure 1.4.

1.3.2 Our Proposed Framework - Semi-supervised CRF Regularized by Proximate Graph

Although semi-structured data is abundant on the Web and its inherent advantages are appealing for the task of entity discovery and attribute extraction, one big challenge of this task is that the number of available seed entities is typically very limited. As shown by the work [25], when the seed number is small, the trained semi-Markov CRF based extractor cannot perform well. However, it is time consuming and labor force intensive to prepare more seed entities' as well as their attributes. Semi-supervised approaches are proposed to ease the difficulty of lacking enough training data by taking the unlabeled data into account [24, 69]. In this book, we propose a semi-supervised learning framework to extract the entities and their attribute content from the semi-structured data record sets. The main contributions of this framework are as follows:

- Our framework makes use of a few existing seed Wikipedia entities and their infoboxes automatically extracted from a particular category as clues to discover more entities of this category. It can leverage the existing infoboxes of the seed entities to automatically harvest attribute content of the newly discovered entities. Entity attribute extraction is essential for the usability of the entities in downstream applications such as knowledge base construction.

- This framework automatically derives training examples by using the seed Wikipedia entities and their infoboxes. We employ semi-Markov CRF as the basic sequence classification learning model. Due to the limited number of seed entities, the derived training examples are usually not sufficient to train a reliable model for extracting the entities and their attributes. A semi-supervised learning model with CRF is developed to solve this problem by exploiting the unlabeled data in the semi-structured data record set.

- To connect the derived training examples and the unlabeled records in the semi-supervised learning, a proximate record graph with each node representing one data record is constructed. This proximate graph is employed to guide the regularization of the posterior label distribution of the text segments in the unlabeled records with that of the derived training examples. Armed with the pairwise sequence alignment based similarity measure between the record nodes, the labels of derived training examples can regularize the labels of directly or indirectly aligned segments in the unlabeled records effectively.

- The research effort inspiring or producing this framework has been published in the ACM International Conference on Web Search and Data Mining [7].

1.4 Outline of the Book

After the high level introduction of the major problems focused on in this book, the rest of the chapters in the book are organized as follows.

In Chapter 2, we review some closely related works for data record extraction and entity expansion. We compare our frameworks with these works, pointing out some shortcomings of these works and the superiority of our proposed frameworks.

In Chapter 3, we first present a record segmentation search tree framework in which a new search structure, named Record Segmentation Tree (RST) framework, is designed and several efficient search pruning strategies on the RST structure are proposed to identify the records in a given Web page.

In Chapter 4, we present our second framework, named DOM Structure Knowledge Oriented Global Analysis (Skoga) framework, for Web data record extraction. Skoga, composed of a DOM structure knowledge driven detection model and a record segmentation search tree model, can conduct a global analysis on the DOM structure to achieve effective record detection. The feature weights of statistical structure knowledge are determined using a development data set via a parameter estimation algorithm based on structured output Support Vector Machine model which can tackle the inter-dependency among the labels on the nodes of the DOM structure.

An optimization method based on divide and conquer principle is developed making use of the DOM structure knowledge to quantitatively infer the best record and region recognition.

In Chapter 5, extensive experiments have been conducted on three data sets including flat, nested, and intertwined records to evaluate the performance of our proposed frameworks in Chapters 3 and 4. The experimental results demonstrate that our frameworks achieve higher accuracy compared with state-of-the-art methods.

In Chapter 6, we present a framework for entity expansion and attribute acquisition. This framework takes a few existing entities that are automatically collected from a particular Wikipedia category as seed input and explores their attribute infoboxes to obtain clues for the discovery of more entities for this category and the attribute content of the newly discovered entities.

In Chapter 7, we review the main contributions of the book and summarize the significance and applicability of the proposed frameworks. We also discuss some possible extensions and future research directions of the research topics in this book.

Chapter 2

Literature Survey

2.1 Semi-structured Record Extraction

Semi-structured Web information extraction (IE) from the Web pages has been studied extensively. The task of record-level extraction from an arbitrary single input page is one active direction in Web IE [42, 67, 77, 91]. The data record extraction frameworks in this book fall into this category. Techniques that address record extraction from a single page can be categorized into five classes: early methods based on heuristics [10, 19], repetitive pattern based methods [14, 77], similarity-based extraction methods [42, 67, 91], tag path based methods [49], and visual feature based methods [23, 45, 94]. Methods based on heuristic rules cannot be generalized well. Repetitive pattern based methods such as IEPAD [14] and DeLa [77] show some potential in solving this problem because similar templates are used in formatting the records, which make it feasible to mine some repetitive patterns as clues for locating records in the page. One limitation of such pattern mining methods is that it is not robust against optional data and tags inserted into the records.

The similarity-based method tackles this limitation with approximate matching to identify repeating objects. MDR [42] and DEPTA [91] are such techniques, which utilize string or tree edit distance to assess whether two adjacent subtree groups, known as generalized nodes, are a repetition of the same data type. ViPER [67] is another work which computes the similarity of each pair of single subtrees to detect record region, then involves some visual perception to segment the detected regions into records. Because these methods highly depend on the pairwise similarity computation of subtrees or subtree

groups, they separate the task into two steps, namely record region detection and record segmentation. They search the possible record regions in the entire DOM tree with a traversal manner. In contrast, our DOM structure knowledge driven framework can efficiently analyze the DOM tree structure with a global view and find the regions of high potential of containing data records. Furthermore, our RST framework is free from their limitations on the subtree grouping manner, namely fixed length of generalized node or single subtree pairs.

Miao et al. investigated the tag paths in a Web page to perform record extraction [49]. They transform a DOM tree into pieces of tag paths, and cluster the paths according to the defined similarity measure to detect record region. The limitation of this method is that it cannot take into account the record boundary information during region detection, and needs a separate step to segment records after region detection. On the contrary, our DOM structure knowledge driven framework can directly report the unity data records with the statistical analysis. Furthermore, this method clusters the tag paths across the entire page, and does not consider the proximate relations of the paths. Thus, the same tag path may be used in different blocks of the page, even these blocks are far away from each other.

Although ViPER [67] and the work by Miao et al. [49] utilize some visual information from rendered Web page to assist record segmentation, they depend on tag structure to detect record region. In contrast, ViNTs [94] utilizes the visual information first to identify content regularities, and then combines them with tag structure regularities to generate wrappers. ViNTs cannot separate horizontally arranged records, e.g., nested records in a table, and identify multiple regions. Pure visual feature based methods include VENTex [23] and ViDE [45], and they are effective to extract records from pages with well organized visual features. With the help of visual information of the rendered pages, these methods are able to select some major blocks that may have high potential of containing data records. However, they suffer two limitations, namely, the inefficiency of Web page rendering, and the difficulty of correctly rendering. For a single Web page in a repository, its related cascading style sheet (CSS) and JavaScript files are normally not cached by the repository. Therefore, it is very likely that this page cannot be correctly rendered. Furthermore, the rendering operation is time consuming. In this book, we do not use these expensive features although our framework is open to incorporate them. Some

other researchers employ pre-defined domain ontology [18] or automatically generated domain ontology [68] to assist the record extraction task.

Another branch of semi-structured Web data extraction mainly depends on manually-constructed wrappers [2, 44]. These methods are difficult to maintain and be applied to different Web sites, because they are very labor intensive. Semi-automatic methods [29, 30, 38, 39, 50, 92, 95, 96], known as wrapper induction, were proposed to tackle this problem. These methods need some labeled pages in the target domain as input to perform the induction. First, the target data or record in a set of training pages are labeled manually. The system then learns the extraction rules from the labeled data automatically, and uses them to extract records from new pages originated from the target domain. To enhance the adaptation capability of the inducted wrappers, a vertical, i.e. domains such as book and auto, oriented method was proposed in [27]. This method takes a labeled example site of a particular vertical as input and learns attribute related knowledge of the vertical. After that, this knowledge is adapted to a new Web site of the same vertical to learn new wrappers for it. Some attempts were made by Zhao et al. [93] in conducting domain-independent Web IE aiming at extracting open-domain attribute name and value pairs from Web pages. They formulated the task as a structured classification problem, which shares some resemblance to our framework, on the structured representation of Web pages. However, our framework targets at extracting different information, namely, data records. Furthermore, we propose a record region oriented loss function as well as a refined training method taking into account this loss function.

It should be noted that the problem setting in our data record extraction frameworks is significantly different from the unsupervised instance-based learning data extraction methods, such as RoadRunner [16] and EXALG [1]. Their methods tackle the task of site-oriented data extraction by taking several pages coming from the same Web site as input, and extract the underlying template or schema automatically. However, our problem setting is to detect data records and regions not limited to particular sites and does not require that several pages from the same site are available.

2.2 Entity Expansion and Attribute Acquisition

Entity set expansion takes a few user given seed entities of a particular class as input, and aims at collecting more entities of the same class. SEAL/iSEAL [79, 80] exploit "list" style of data, which can be considered as a simplified kind of semi-structured data records in this book, to discover more entities for expanding a given entity set. They extract named entities with wrappers, each of which is composed of a pair of character-level prefix and suffix [81]. However, they do not perform record region detection, consequently the wrappers may extract noise from the non-record regions of the page. The work [25] by Gupta and Sarawagi also processes the "list" style of data records to extract entity names and their attributes. It takes several user input data records, including entity names and attribute values, as seeds to discover more entities as well as similar specified attributes by a trained semi-Markov Conditional Random Field (semi-Markov CRF) [65] based extractor. This method requires considerable manual effort if we wish to apply it in large scale of domains.

Some other works focus on a more general problem setting for the task of entity set expansion [54, 55, 58]. They first obtain a set of candidate entities by some linguistics techniques such as pattern-based method. Then the similarity of a candidate with the seeds is calculated using their context distributions on the Web or Web query. Because of the general setting, the targeted classes have more coarse granularity such as city, country, etc. Moreover, these methods are not able to conduct attribute extraction of the new entities. Different from the above methods that explore positive seed instances only, Li et al [40] proposed a learning method that takes both positive and unlabeled learning data as input and generates a set of reliable negative examples from the candidate entity set. Then the remaining candidates are evaluated with the seeds as well as the negative examples. Ensemble semantics methods [12, 59] assemble the existing set expansion techniques as well as their information resources to boost the performance of a single approach on a single resource. The output of named entity recognition [51] can serve as one source to perform set expansion [55, 58, 59]. Entity set acquisition systems [11, 21] do not need input seeds. They leverage domain independent patterns, such as "is a" and "such as", to harvest the instances of a given class. Open information extraction [20, 22] and table semantifying [41, 74, 76] focus more on extracting or annotating large amount of facts and relations.

With respect to the goal of entity attribute extraction, some existing works [28, 72, 86, 88] train extractors on the free text of Wikipedia articles that are automatically annotated with the corresponding articles' infoboxes. Different from their direction, we explore the semi-structured data records which exist in large quantity on the Web. For example, we can extract the entity attributes of release date, director, etc. from the semi-structured list of the page shown in Figure 1.4. Since each data record describes one entity, the extracted attribute information can be connected to the correct subjects (i.e. entities) automatically. Consequently, our entity discovery and attribute extraction framework eliminates some error-prone operations such as coreference resolution. The methods of weakly-supervised attribute acquisition [54, 56] can also be applied in identifying important attributes for the categories. Thus, the existing infoboxes can be polished, and the non-infobox categories can obtain proper attributes to establish their own infobox schemata.

Recently, researches performed some exploration on automatic wiki article generation. The work by Sauper and Barzilay [66] takes articles of the existing entities from a particular Wikipedia category to derive a category specific article model. Then article generation for a new entity of the same category is performed under the guidance of this model taking some entity related documents as input. BioSnowball [46] is another work to automatically generate description article for a person. Our framework can cooperate with these works to make a discovered entity have not only infobox but also description article.

A number of projects have investigated Wikipedia as information source to construct knowledge bases or taxonomy, including Freebase [9], DBpedia [3], YAGO [70, 71], KOG [87], and work by Ponzetto and Strube [61]. These works harvest the structured infobox or category hierarchy of Wikipedia to deduce facts and relations such as isA, bornIn, spouseOf, etc. The entity expansion framework proposed in this book aims at harvesting more entities which are absent in Wikipedia from the public Web, as well as the attributes of these new entities. Therefore, it can be regarded as an preparation step for the above works. Nevertheless, some techniques in these projects are also useful for our work. Some existing works employ the facts and relations provided by Wikipedia or other knowledge base originated from it to perform new relation and fact extraction from natural language documents in free text. KYLIN [86], LUCHS [28], and WOE [88] utilize the attribute information in the infobox together with the articles of corresponding

entities to generate training examples automatically, then train a CRF based extractor in lexicalized [86, 28] or unlexicalized [88] manner. After that, the constructed extractor can be utilized to process other entity articles or the documents outside Wikipedia. Different from the above works, our framework extracts attribute values from semi-structured data records that describe particular entities. Thus, the extracted attribute information can be connected to the correct subjects automatically. Consequently, our method is free from the curse of ambiguity and coreference resolution.

Among the semi-supervised CRF approaches, one class of methods consider data sequence granularity [24, 33, 83]. Precisely, these methods incorporate one more term in the objective function of CRF. This term captures the conditional entropy of the CRF model or the minimum mutual information on the unlabeled data. The extra term can be interpreted by information theory such as rate distortion theory. However, the objective function does not possess the convexity property any more. Subramanya et al. also constructed a graph to guide the semi-supervised CRF learning in part-of-speech tagging problem [69]. This method regularizes the posterior probability distribution on each single token in a 3-gram graph. Its n-gram based graph construction is not applicable to the problem tackled in our entity expansion framework. The reason is that the length of our desirable text segments cannot be fixed in advance. In contrast, our proximate record graph can capture both record level and segment level similarities at the same time. Furthermore, the proximate record graph is also able to capture the position of a text segment in the alignment so that the aligned segments with higher chance describing the same functionality components of different records.

Chapter 3

Record Segmentation Tree (RST) Framework for Data Record Detection

3.1 Overview

In the Web page fragment given in Figure 1.2(a), the company information is organized in a "table", and each company corresponds to one data record. The table's DOM tree is given in Figure 1.2(c). We can see that the records share some common fields such as company description, investor, and established year. One can also notice that some records do not contain certain fields. For example, the third record does not have URL information. Furthermore, the records are formatted with similar HTML templates, and each record is composed of several rows in the table. We use $\mathcal{T}$ to denote the DOM tree given in Figure 1.2(c), and $\mathcal{T}$'s subtree sequence is denoted by $\mathbf{S}$, and an element in $\mathbf{S}$ is referred to as $\mathcal{S}_i$. In the DOM tree in Figure 1.2(c), $\mathcal{T}$ is "table", and $\mathbf{S}$ includes $\mathcal{S}_1$, $\mathcal{S}_2$, etc. $\mathcal{T}$ and $\mathcal{S}_i$ are also used to refer to the root nodes of the corresponding DOM trees. A fragment of the sequence $\mathbf{S}$ is denoted by $\mathcal{S}_{i..j}$ or $\mathcal{S}_i \cdots \mathcal{S}_j$, where $1 \leq i \leq j \leq |\mathbf{S}|$.

Considering the characteristics of data record and record region, previous works rely on two basic observations which are reviewed as follows:

Observation 1. *A group of data records describing a set of similar objects are typically presented in a particular region of a page and are formatted using similar HTML tags.*

Observation 2. *A group of similar data records being placed in a specific region is reflected in the tag tree by the fact that they are under one parent node, although we do not know which parent. It is very unlikely that a data record starts from an inner node of a child subtree and ends at an inner node of another child subtree of the parent node.*

Based on these two observations, existing works separate the record extraction task into two sub-tasks, namely, record region detection and record segmentation. In this framework, we propose a unified method which can tackle these two sub-tasks simultaneously. Returning to the example in Figures 1.2(a) and 1.2(c), given the subtree sequence $\mathbf{S}$ of the table $\mathcal{T}$, data record extraction aims at identifying the sequence $\mathcal{S}_{i..j}$ $(i < j)$ and a set of separating indexes $\mathbf{b}$ in which each $b_k \in \mathbf{b}$ *s.t.* $i < b_k \leq j$. The identified sequence $\mathcal{S}_{i..j}$ is a *record region*, and it is separated into *data records* by the indexes in $\mathbf{b}$. In the above example, the record region is $\mathcal{S}_{2..|\mathbf{S}|}$, and separating indexes set $\{4, 7, 9 \cdots\}$ indicates the boundary of records in this region. Thus, we know that the records are $\mathcal{S}_{2..4}$, $\mathcal{S}_{5..7}$, $\mathcal{S}_{8..9}$, etc. It is important to notice that a record region may not start from the first subtree of $\mathcal{T}$, and the length (number of subtrees) of different records may be different. In addition, some DOM trees may contain 0 or more than one regions. If there is no record region, no subtree sequence should be identified. If there are several regions, several subsequences of $\mathbf{S}$ should be identified.

3.2 Record Segmentation Tree

Given a subtree sequence $\mathbf{S}$, we design a new search structure, named Record Segmentation Tree (RST), to detect possible records in this sequence. Based on this search structure, the proposed algorithm can search and identify data records in the sequence. If some data records are identified, they naturally compose record regions. Therefore, region detection and record segmentation are performed simultaneously. For the convenience of interpretation, we use the subsequence starting from $\mathcal{S}_1$ to illustrate RST construction, as well as the subsequent algorithms. Our framework can detect the region starting from any subtrees in $\mathbf{S}$. We assume that one record at most contains K subtrees in $\mathbf{S}$.

3.2.1 Basic Record Segmentation Tree

Definition 1. ***Record segmentation tree*** *is a search tree with the following properties:*

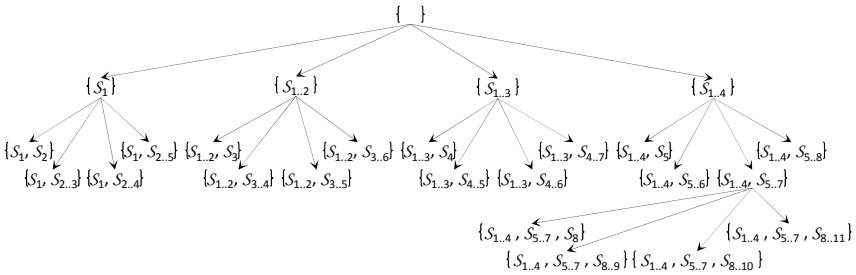


Figure 3.1: An example of basic record segmentation tree with $K = 4$.

- *Each node* $\mathbf{R}$ *represents a possible record region. The root node represents an empty region.*

- *Each* $\mathbf{R}$ *covers a prefix subsequence* $\mathcal{S}_{1..n}$ *of* $\mathbf{S}$*, referred to as* $\mathbf{S_R}$*, where* $0 \leq n \leq |\mathbf{S}|$*, and has a separating indexes set* $\mathbf{b}$ *which segments* $\mathcal{S}_{1..n}$ *into records. Each record of* $\mathbf{R}$ *is denoted by* R_i*. The root has an empty prefix subsequence, i.e.* $n = 0$*, and an empty separating indexes set.*

- *Each* $\mathbf{R}$ *with* $\mathcal{S}_{1..n}$ *and* $\mathbf{b}$ *has at most* K *children. Each child of* $\mathbf{R}$ *covers* $\mathcal{S}_{1..m}$ *where* $n + 1 \leq m \leq \min\{n + K, |\mathbf{S}|\}$*, and has a separating indexes set* $\mathbf{b} \cup \{m\}$.

From the above definition, it can been seen that each node is recursively defined with its parent node, and contains one more record. The record extraction task in $\mathbf{S}$ is transformed into a problem that searches a node in the RST structure which best matches with the true records in $\mathbf{S}$.

The segmentation tree with $K = 4$ is given in Figure 3.1. In this example, each node is labeled by its record set. For instance, the node $\mathbf{R} = \{\mathcal{S}_{1..4}, \mathcal{S}_{5..7}\}$ indicates that there are two records, namely, $R_1 = \mathcal{S}_{1..4}$, and $R_2 = \mathcal{S}_{5..7}$. The covered prefix is $\mathcal{S}_{1..7}$, and the separating indexes set is $\{4, 7\}$. $|R_i|$ denotes the length of R_i, and is equal to the number of subtrees it contains. For instance, $|R_1| = 4$, $|R_2| = 3$. We use $|\mathbf{R}|$ to denote the number of records in $\mathbf{R}$. $L_{\mathbf{R}}$ denotes the average length of the records in $\mathbf{R}$, calculated as $(\sum_{R_i \in \mathbf{R}} |R_i|)/|\mathbf{R}|$ or $|\mathbf{S_R}|/|\mathbf{R}|$.

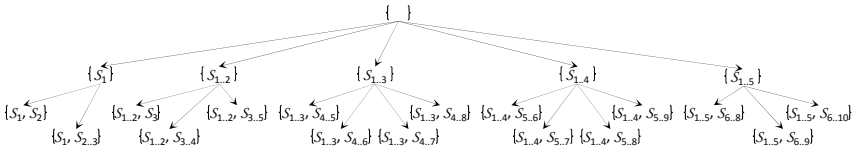


Figure 3.2: An example of slimmed segmentation tree with $K = 5$.

3.2.2 Slimmed Segmentation Tree

In the same record region, it is almost impossible that the lengths of different records are significantly different, say some records contain only 1 subtree, while some others contain 5 subtrees. Based on this observation, we design a slimmed segmentation tree, in which the length of previous records is used to predict that of the next record. Suppose for a node $\mathbf{R}$ with $L_{\mathbf{R}} = 2$, we may think that it is impossible that a child of $\mathbf{R}$ has a new record with length 4 or more. Therefore, we only consider the children of $\mathbf{R}$ in which the new records' length is smaller than 4. Another observation is that when $L_{\mathbf{R}}$ is large, it is more probable that different records have a larger absolute length difference. For example, in a page in our experiment data set, the average length of the records is 7. Some records contain 4 subtrees, while some others contain 9 subtrees. Taking the above two observations into consideration, we introduce a slimmed version of segmentation tree.

Definition 2. ***Slimmed segmentation tree*** *is a sub-graph structure of the basic record segmentation tree. It keeps the first two layers of basic RST. Each child of a non-root node* $\mathbf{R}$ *with prefix sequence* $\mathcal{S}_{1..n}$ *has prefix sequence* $\mathcal{S}_{1..m}$*, where* $n + (L_{\mathbf{R}} - \lfloor L_{\mathbf{R}}/2 \rfloor) \leq m \leq \min\{n + \min\{L_{\mathbf{R}} + \lceil L_{\mathbf{R}}/2 \rceil, K\}, |\mathbf{S}|\}$.

An example is given in Figure 3.2, in which K is 5. It can be seen that in the third layer, we do not construct all possible children of a node $\mathbf{R}$ in the second layer. If $L_{\mathbf{R}}$ is smaller, we construct fewer number of children for it. Otherwise, we construct more.

3.2.3 Utilize RST in Record Extraction

Each node in RST is one possible segmentation of the record region starting from $\mathcal{S}_1$ in the sequence $\mathbf{S}$. According to the observation that the records in the same region are formatted using similar tags, the correct segmentation should be the node that achieves higher average pairwise record similarity. If we cannot find a node with pairwise record

similarity greater than a pre-defined threshold, we may conclude that no record region exists starting from $\mathcal{S}_1$.

Precisely, given a DOM tree $\mathcal{T}$ and its subtree sequence $\mathbf{S}$, record extraction with the RST structure of $\mathbf{S}$ aims at finding a node $\mathbf{R}^*$ such that:

$$\mathbf{R}^* = \underset{\mathbf{R}\in\{\mathbf{R}|Q(\mathbf{R})\geq\theta\}}{\operatorname{argmax}} |\mathbf{R}|, \tag{3.1}$$

where θ is a pre-defined threshold. $Q(\cdot)$ is the quality function of an RST node $\mathbf{R}$, which is defined as the average pairwise record similarity of records in $\mathbf{R}$:

$$Q(\mathbf{R}) = \frac{\sum_{R_i,R_j\in\mathbf{R}\ s.t.\ i<j} sim(R_i, R_j)}{|\mathbf{R}|(|\mathbf{R}|-1)/2}, \tag{3.2}$$

where sim is a similarity function between two records. Let $\mathbb{R}^*$ denote $\{\mathbf{R}|Q(\mathbf{R}) \geq \theta\}$, each element in $\mathbb{R}^*$ is a node whose quality is not less than θ. Among these nodes in $\mathbb{R}^*$, the one with the maximum number of records is determined as the correct record region $\mathbf{R}^*$ starting from $\mathcal{S}_1$ in $\mathbf{S}$. If more than one nodes have the maximum number of records, we select the one with the best quality as $\mathbf{R}^*$. If $\mathbb{R}^* = \emptyset$, it means that no record region exists starting from $\mathcal{S}_1$.

For un-slimmed RST, when we expand it to layer $|\mathbf{S}|/K$, each inner node has K children. Thus, the total number of nodes in layer $|\mathbf{S}|/K$ is $K^{|\mathbf{S}|/K}$. In the slimmed version, suppose each inner node has $\hat{K}$ children on average, this number is $\hat{K}^{|\mathbf{S}|/K}$. Usually, in a possible record region, $|\mathbf{S}|$ is much larger than K. Constructing and searching such a segmentation tree is extremely time and space consuming because of its exponential size. Therefore, some search pruning strategies have been developed to allow efficient utilization of the RST structure.

3.3 Search Pruning Strategies

In this subsection, we introduce a threshold-based top k search that can prune the RST significantly, and reduce the complexity to $O(|\mathbf{S}|^2)$ without considering pairwise similarity computation of subtrees. Furthermore, instead of calculating the similarity of all record pairs in Equation 3.2, we may only check the similarity of a record and its nearest previous neighbors. Accordingly, the time complexity is further reduced to $O(|\mathbf{S}|)$.

3.3.1 Threshold-Based Top k Search

Initialization

In the beginning of RST construction, we have no idea about the record length in the given subtree sequence. The only information on hand is that each record has at most K subtrees. To attain a better starting, we fully expand the RST in the first 2 layers as shown in Figure 3.1. Each non-leaf node has K children in these layers. Then we have K^2 initial candidate nodes, denoted by $\tilde{\mathbb{R}}$, for future expansion and search. Note that K is very small, and the full expansion in the beginning will not cause much increasing of time complexity.

Before proceeding to the next layer of RST, we prune the candidates in $\tilde{\mathbb{R}}$ according to their quality, which is defined in Equation 3.2. For a particular node $\mathbf{R}$ in $\tilde{\mathbb{R}}$, it will be pruned if $Q(\mathbf{R}) < \theta'$. The meaning of θ' will be clear later. After the nodes with lower quality are pruned, if the number of retained nodes is more than a pre-defined threshold k, only the top k nodes with the best quality are retained, and denoted by $\mathbb{R}$. For example in Figure 3.1, the node $\{\mathcal{S}_1, \mathcal{S}_{2..5}\}$ is very likely to be pruned. If $\mathbb{R} = \emptyset$, then no region starting from $\mathcal{S}_1$ exists in $\mathbf{S}$.

Pruning Search

Suppose $\mathbf{R}$ is a node in $\mathbb{R}$, the last record in $\mathbf{R}$ is $\mathcal{S}_{i..n}$. We construct some children of $\mathbf{R}$, namely, $\mathbf{R} \cup \{\mathcal{S}_{n+1}\}, \mathbf{R} \cup \{\mathcal{S}_{n+1..n+2}\}, \cdots, \mathbf{R} \cup \{\mathcal{S}_{n+1..n+K}\}$. Then the similarity between newly generated records and the existing ones in $\mathbf{R}$ are calculated. Precisely, for each new record $\mathcal{S}_{n+1..m}$ $(n+1 \leq m \leq n+K)$ and each $R \in \mathbf{R}$, $sim(\mathcal{S}_{n+1..m}, R)$ is calculated. Then, the quality of $\mathcal{S}_{n+1..m}$ is defined as:

$$Q(\mathcal{S}_{n+1..m}) = \frac{\sum_{R \in \mathbf{R}} sim(\mathcal{S}_{n+1..m}, R)}{|\mathbf{R}|}. \tag{3.3}$$

In the same way, each $\mathbf{R}$ in $\mathbb{R}$ is expanded, and the quality of each new record is calculated.

After the RST structure is expanded one more layer, a new pruning strategy different from that for $\tilde{\mathbb{R}}$ is adopted. For a new RST node $\mathbf{R} \cup \{\mathcal{S}_{n+1..m}\}$, if $Q(\mathcal{S}_{n+1..m}) \geq \theta'$ and $Q(\mathbf{R} \cup \{\mathcal{S}_{n+1..m}\}) \geq \theta$, $\mathbf{R} \cup \{\mathcal{S}_{n+1..m}\}$ is retained, otherwise it is pruned, where θ' is smaller than θ. The rationale behind this strategy is that some record, say $\mathcal{S}_{n+1..m}$, may have a larger difference compared with the previous records. But we assume that the difference should not be large. Precisely, $Q(\mathcal{S}_{n+1..m})$ should not be less than a looser threshold θ'. If

this condition is met, we continue to check the quality of $\mathbf{R} \cup \{\mathcal{S}_{n+1..m}\}$. In this way, the search procedure can overcome the problems caused by some outlier records, meanwhile, the quality of the outliers is also bounded by θ'. Recall that we use θ' instead of θ in the pruning of the initial candidate nodes $\tilde{\mathbb{R}}$, it is because each element of $\tilde{\mathbb{R}}$ contains two records. For a particular $\mathbf{R}$, if no $\mathbf{R} \cup \{\mathcal{S}_{n+1..m}\}$ is retained after pruning, $\mathbf{R}$ is put into $\mathbb{R}^*$. Again, if the number of retained nodes is more than k, we keep the top k nodes with the best quality, and construct a new $\mathbb{R}$.

The above procedure is repeated on the new $\mathbb{R}$ until it is empty or the end of $\mathbf{S}$ is reached. Then we finish building an RST structure for $\mathbf{S}$ starting from $\mathcal{S}_1$, at the same time, $\mathbb{R}^*$ is also built. Note that the RST structure needs not reach $\mathcal{S}_{|\mathbf{S}|}$, which indicates that some subtrees in the end of $\mathbf{S}$ should not be included in the record region.

Retrospect with Short-term Memory

In the above expanding and pruning process, when examining the quality of a new record, all previous records are considered, as shown in Equation 3.3. This thorough retrospect is not only time consuming, but also unnecessary. Especially in the region with many records, the cost of such exhaustive comparison outweighs the benefit gained. We adopt a short-term memory retrospect strategy to reduce the computation workload. When a node is expanded one more layer, we only calculate the similarity between a new record and the nearest r previous records. Thus, in Equation 3.3, the number of pairwise similarity calculations is reduced from $|\mathbf{R}|$ to r. And in Equation 3.2, the cost of evaluating one node is reduced from $|\mathbf{R}|(|\mathbf{R}|-1)/2$ to $r|\mathbf{R}|$.

3.3.2 Complexity Analysis

The computation workload of pairwise record similarity calculation is related to the number of subtrees in the involved records. Hence, we use a finer unit to analyze the complexity of utilizing RST in record extraction.

In Section 3.3.1, when evaluating the quality of new records with Equation 3.3, we need to calculate the similarity between each existing record in $\mathbf{R}$ and each new record. The computation workload is:

$$|\mathbf{R}| \sum_{1 \le l \le K} (L_{\mathbf{R}} l) = \frac{K(K+1)|\mathbf{S}_{\mathbf{R}}|}{2}, \tag{3.4}$$

where $L_{\mathbf{R}}$ is the average length of records in $\mathbf{R}$, $\mathbf{S}_{\mathbf{R}}$ is the subtree sequence covered by $\mathbf{R}$, and $L_{\mathbf{R}}l$ indicates the computation needed for calculating the similarity between the records having $L_{\mathbf{R}}$ and l subtrees respectively. Thus, the computation workload of segmenting entire $\mathbf{S}$ is:

$$\sum_{1\leq x\leq(\frac{|\mathbf{S}|}{L_{\mathbf{R}}}-1)}(x\sum_{1\leq l\leq K}(L_{\mathbf{R}}l))=\frac{K(K+1)(\frac{|\mathbf{S}|}{L_{\mathbf{R}}}-1)|\mathbf{S}|}{4}, \tag{3.5}$$

In the top k strategy, the overall computation needed is $kK(K+1)(|\mathbf{S}|/L_{\mathbf{R}}-1)|\mathbf{S}|/4$. In our framework, the method of calculating pairwise record similarity is a variant of edit distance. Due to the dynamic programming employed in edit distance, when calculating the distance matrix between two strings, the existing matrix between their prefix strings can be reused. In Equation 3.3, when calculating the similarity between $\mathcal{S}_{n+1..m}$ and an existing record R, the distance matrix between $\mathcal{S}_{n+1..m-1}$ and R can be reused. Consequently, the above workload can be further reduced to $k(K+1)(|\mathbf{S}|/L_{\mathbf{R}}-1)|\mathbf{S}|/4$. The value of k is usually smaller than K, and both k and K are small and can be treated as constants. Thus, the overall time complexity is $O(|\mathbf{S}|^2)$.

Under the strategy of short-term memory retrospect, the workload in Equation 3.5 becomes:

$$(\frac{|\mathbf{S}|}{L_{\mathbf{R}}}-1)r\sum_{1\leq l\leq K}(L_{\mathbf{R}}l)=\frac{rK(K+1)|\mathbf{S}|}{2}, \tag{3.6}$$

where r is the number of records retrospected from current layer. Similarly, the overall complexity can be further reduced to $r(K+1)|\mathbf{S}|/2$, which can be regarded as $O(|\mathbf{S}|)$.

For each sequence $\mathbf{S}$, MDR [42] has time complexity of $O(|\mathbf{S}|)$ to calculate the similarity among the same length generalized node. In MDR, only the adjacent generalized node pairs are considered in similarity calculation, which is similar to our retrospect strategy with $r=1$. ViPER [67] has time complexity of $O(|\mathbf{S}|^2)$ to construct the upper triangular similarity matrix of all subtree pairs. Note that the above time complexity is only the part needed by MDR or ViPER for record region detection, and they need another step to segment the detected region into records.

3.3.3 Composite Node Pruning

Suppose each subtree in $\mathbf{S}$ is one record, all RST nodes with the form $\{\mathcal{S}_{1..i},\mathcal{S}_{i+1..2i},\mathcal{S}_{2i+1..3i},\cdots\}$ $(1\leq i\leq K)$ have quite high quality. The reason is that the combination of i records

is similar to that of the other i records. In some cases, this kind of combination may even achieve higher quality than the correct record segmentation. For example, there are four records in a region $\mathbf{R} = \{\mathcal{S}'_1\mathcal{S}_2, \mathcal{S}'_3, \mathcal{S}'_4\mathcal{S}_5, \mathcal{S}'_6\}$, where the primed subtrees are matched ones in different records. The node $\mathbf{R}^c = \{\mathcal{S}'_1\mathcal{S}_2\mathcal{S}'_3, \mathcal{S}'_4\mathcal{S}_5\mathcal{S}'_6\}$ has a higher quality than $\mathbf{R}$. We call $\mathbf{R}^c$ a composite node of $\mathbf{R}$. Composite nodes will increase the computation workload and involve noise during RST search. To tackle this problem, after a period of search, say when all nodes in $\mathbb{R}$ exceed $\mathcal{S}_n$ in $\mathbf{S}$, we check the relation between the nodes in $\mathbb{R}$. If one node is a composite node of any other node, it is pruned. Note that for the node involving subtrees after $\mathcal{S}_n$, we only consider its records before $\mathcal{S}_n$ in the detection of composite node.

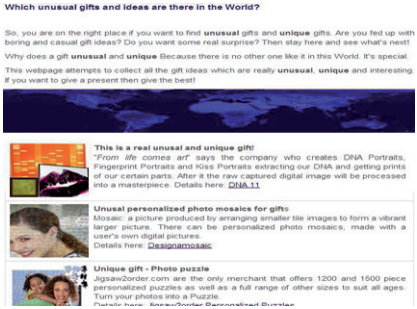
3.3.4 More Challenging Record Region Discussion

In this subsection, we discuss how several kinds of more challenging record regions can be handled with the proposed RST structure and search strategy.

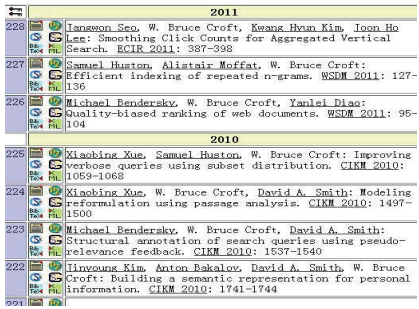
Embedded Region and Non-continuous Region

Record region in $\mathbf{S}$ may not start from $\mathcal{S}_1$ because of the existence of header information, one example and its DOM tree are given in Figures 3.3(a) and 3.3(c). In the same way, the region also may not end at $\mathcal{S}_{|\mathbf{S}|}$. We name this kind of region as *embedded region*. The way to detect embedded regions in our framework is straightforward. In the beginning, we start from $\mathcal{S}_1$, and generate an initial set $\tilde{\mathbb{R}}_{\mathcal{S}_1}$. If all elements in $\tilde{\mathbb{R}}_{\mathcal{S}_1}$ are pruned, we move to $\mathcal{S}_2$, and repeat the above procedure. In this way, the header subtrees in the beginning are excluded. If $\mathbb{R} = \emptyset$ before coming to $\mathcal{S}_{|\mathbf{S}|}$, it means that there is some footer information which should not be treated as part of any record.

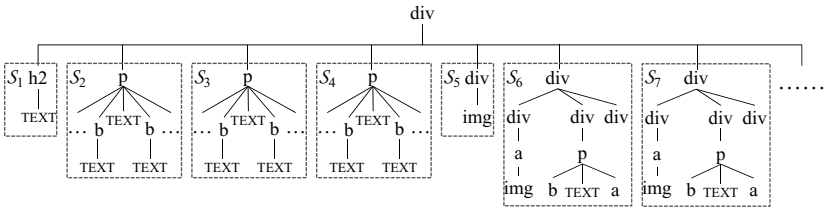
In some other cases, there may be more than one regions in $\mathbf{S}$. For example, in an on-line shopping Web page, $\mathbf{S}$ may contain two subsequences introducing new arrival products and featured products respectively. And they are separated by some other information. We name this kind of region as *non-continuous region*. The above procedure for dealing with embedded regions can also be applied to non-continuous region easily. After finishing one region detection, the detection algorithm will move on to the sequence of the remaining subtrees if there are some, and perform the search procedure again to detect the second region.



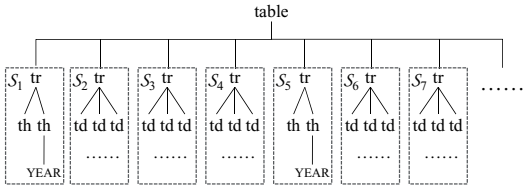
(a) A page fragment of embedded region.



(b) A page fragment of noisy subtree region.



(c) DOM tree of (a).



(d) DOM tree of (b).

Figure 3.3: Several kinds of more challenging record regions.

Nested Region

In some Web pages, the records are formatted in a nested manner. For example, the region formatted with <table>, each record is packed with one <td>, each <tr> has several <td>'s, one nested region and its DOM tree are given in Figures 1.1(b) and 1.1(d). The region with <table> as root node is know as *nested region*. In our framework, with a top-down manner to scan the DOM tree, <table> is detected as a record region with <tr>'s as records. Then, each <tr> is detected as a region with <td>'s as records. After the <tr>'s are detected as regions, along with the fact that these <tr>'s have been detected

as records of the same root node <table> in the preceding detection, we may postulate that the <table> is a nested region. To inspect our postulation, we compare the records, i.e., <td>'s, from different <tr>'s, and check whether they are similar. If so, we conclude the node <table> is a nested region. An orphan record may exist in the last "row" and needs to be detected. Note that the above "table" is used to demonstrate the detecting process, our detection method does not rely on any particular tag.

Intertwined Records

Intertwined record, also known as *non-continuous record* in DEPTA [91], refers to the record whose attributes intertwine together with other records' attributes. One example and its DOM tree are given in Figures 1.2(b) and 1.2(d). Each record has 3 attributes, namely, image, title, and price, and these attributes scattered in 3 different subtrees (<tr>'s). In our framework, the subtree sequence of <tr>'s in the table is first passed to the record detection algorithm, and each successive non-overlapping 3-subtree segment is detected as one record. After that, each <tr> is passed to the detection algorithm, and detected as a region with each <td> as one record. Up to now, the above procedure resembles that of nested region detection. We continue to check the similarity between records in neighboring <tr>'s, and they are found dissimilar to each other. Thus, we conclude that a intertwined region is detected, and the correct records can be generated by reassembling the detected regions. In this example, three regions (image 1, image 2), (title 1, title 2) and (price 1, price 2) are reassembled to generate records (image 1, title 1, price 1) and (image 2, title 2, price 2).

Noise Subtree Exclusion

In Figures 3.3(b) and 3.3(d), a publication list is separated into different sections according to the published year. It can be regarded as a non-continuous region. Theoretically, with proper setting of thresholds, we can expect that publications in 2011 will be detected as the first region, and publications in 2010 will be detected as the second region. However, it is also possible that the year row 2011 fuses into record 228, and the year row 2010 fuses into record 225. If a record contains such noise subtrees, the similarity between this record and its context records becomes lower. Meanwhile, the number of subtrees in this record is more than that in others. Using these observations as clues, it is not difficult

to exclude noise subtrees from records. Basically, if some subtrees in the first record are detected as noise and excluded, we need to reexamine this record segmentation. The reason is that the polluted first record may lead to wrong segmentation of the region.

3.4 Similarity Measure

In this section, we discuss the method used for measuring the similarity between two records (sequences of subtrees), in sequence **S**. There are mainly two existing approaches. One is string edit distance based [42, 67], the other is tree edit distance based [91]. In string edit distance based method, each subtree is encoded with a string which is obtained by traversing the subtree in pre-order, and appending the name of visited tag node to the string. Due to the fact that names of different tags may have different length, using the string of tag name directly is not suitable. In spite of this limitation, string edit distance can tackle repetitive fields properly by tandem repeats detection [67], as well as optional fields. In [91], top-down distance, which is a restricted version of tree edit distance, is employed to measure the distance between two subtrees. Its problem is that any crossing layers' operation is not permitted, making it unable to handle optional tags. On the other hand, the tree edit distance can overcome the limitation in string edit distance since it considers each DOM node as an inseparable unit. To adopt their good points and avoid the shortcomings, we propose a token-based edit distance method.

3.4.1 Encoding Subtree with Tokens

Aiming at fixing the ill-formatted Web pages, some existing works [90, 91] introduce visual information from rendered Web page into DOM tree building. To avoid this time consuming rendering operation, we employ an HTML cleansing package, namely, HtmlCleaner[1], to clean the Web pages. Then the DOM structure of the cleaned page is built.

In the DOM tree, we have two kinds of nodes, namely, tag node and text node. Each tag node has a name such as "tr", "div", etc. A piece of visible text between a pair of ">" and "<" is regarded as a basic text unit, and normalized to a text node. Table 3.1 shows the types of text nodes defined in our framework. The priority indicates the order in which different types are attempted when normalizing a piece of text. Each subtree

[1] `http://htmlcleaner.sourceforge.net/`

Table 3.1: Types of text node.

Type Name	Meaning	Priority
EMAIL	An email address	1
URL	A URL string	2
PRICE	Digital number with a currency symbol	3
TIME	Time in predefined format, such as hh:mm:ss	4
DATE	Date in predefined format, such as yyyy/mm/dd	5
YEAR	Four digits, arrange from 1900 to 2050	6
TEXT	Other kinds of text	7

in the DOM is encoded with the sequence of node names in it, which is obtained by traversing the tree in pre-order. Each node name in the sequence is called one *token*, and it is an inseparable unit in similarity calculation.

3.4.2 Tandem Repeat Detection and Distance-based Measure

A typical characteristic of data records is that they vary in optional or repetitive fields. For example, a book record may have discounted price or not, and one author or several. An obvious disadvantage of edit distance computation is that repetitive and optional fields increase the edit cost. As noticed by ViPER [67], similarity threshold is suitable to solve the problem caused by optional fields, but not suitable for repetitive fields. ViPER identifies *tandem repeats* in two strings before distance calculation, and allows zero cost for deletion and insertion inside additional repetitions.

In our token-based edit distance scenario, the above idea is also applicable. We implement the algorithm proposed by Gusfield et al. [26] to detect the tandem repeats. This algorithm utilizes suffix tree structure to identify the tandem repeats which are not longer than z in a sequence of length n in $O(n + z)$ time. The difference is that the basic unit in our sequences is token (node name), not single character. Thus we need to perform token comparison instead of character comparison in the detection of tandem repeats. Furthermore, if the token sequence is originated from several subtrees, it is constrained that tandem repeats are only detected in the token sequence of each single subtree. We

refine the tandem repeats based edit distance proposed in ViPER [67] and make it suitable for the token scenario in our framework. For the details of tandem repeat detection and its application in edit distance, we would like to direct the readers to [26] and [67]. Then the calculated distance is normalized into the interval [0, 1], and the negative value of the normalized distance is added by 1. Thus, we obtain the record similarity measure used in our framework.

Chapter 4

DOM Structure Knowledge Oriented Global Analysis (Skoga) Framework for Data Record Detection

4.1 Overview

In our proposed Structure Knowledge Oriented Global Analysis (Skoga) framework, the goal of record region detection and data record extraction is tackled by identifying appropriate portions in the DOM structure as record regions as well as data records in a region. It can be formulated as a problem of assigning suitable labels to the nodes of the DOM tree. Taking the flat region in Figure 1.1(a) with its DOM tree given in Figure 1.1(c) as an example, the label "`REC-S`" (namely, *rec*ord composed of a *s*ingle subtree) should be assigned to the <tr> tags which are the roots of the subtrees $\mathcal{S}_2$, $\mathcal{S}_3$, etc. Furthermore, the label "`REGION`" should be assigned to the root <table> tag of the table, and the label "`REGNOT`" (namely, *reg*ion *not*e node) should be assigned to the <tr> tag of $\mathcal{S}_1$. For the nested region in Figure 1.1(b) with its DOM tree given in Figure 1.1(d), the label "`REC-S`" should be assigned to the <td> tags. Also the label "`SUBREG`" (namely, *subreg*ion of records) should be assigned to the <tr> tags which are the roots of the subtrees $\mathcal{S}_1$, $\mathcal{S}_2$, etc., and the label "`REGION`" should be assigned to the root <table> tag.

Formally, let $\mathbf{x}$ denote the DOM tree of a particular Web page, and a single node in $\mathbf{x}$ is denoted by x. Let $\mathbf{y}$ denote a label assignment for $\mathbf{x}$, and a single label is denoted by $y \in \mathcal{Y}$ where $\mathcal{Y}$ is the set of all possible labels. To achieve the goal of record region

detection and data record extraction, we can formulate it as an optimization problem via a global objective function to obtain $\mathbf{y}^*$ such that:

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}} F(\mathbf{x}, \mathbf{y}; \mathbf{w}), \tag{4.1}$$

where F is an objective function that evaluates the fitness of $\mathbf{y}$ for $\mathbf{x}$ with the guidance of the DOM structure knowledge $\mathbf{w}$. Such design facilitates a global analysis on the DOM structure to achieve accurate detection of records and regions. The DOM structure knowledge is composed of background knowledge and statistical knowledge. The background knowledge encodes the semantics of labels indicating general constituents of data records and regions. In addition, it captures some logical relations governing certain structural constraints among the labels to be assigned to the nodes of the DOM structure. The statistical knowledge consists of the design of features capturing different characteristics of a single node or a node group in the DOM. Furthermore, these features are able to distill the difference among different types of record regions so as to identify them accurately. Some examples of the features are the structure feature of a single node, the similarity of the neighboring subtrees, the similarity of one subtree with its siblings, etc. To allow different impacts for different features, each feature is associated with a weight.

Fundamentally, some existing methods such as DEPTA [91], ViPER [67], FiVaTech [36], and RST [4] can also be represented in the form of $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$. For example, MDR and DEPTA calculate the similarity between two neighboring generalized nodes in a particular region derived from the DOM tree $\mathbf{x}$. According to the degree of satisfaction on some predefined heuristic criteria such as similarity threshold, a certain label assignment $\mathbf{y}$ is returned. The criteria they employ can also be regarded as a simple kind of DOM structure knowledge. Different from these methods, our model conducts a global analysis on the fitness of $\mathbf{y}$ for $\mathbf{x}$ driven by the DOM structure knowledge and it takes the inter-dependency among the labels on the nodes into consideration. To infer the best label assignment $\mathbf{y}^*$ for $\mathbf{x}$, an efficient optimization method is developed using divide-and-conquer principle in polynomial time.

Let us return to the complicated flat region given in Figure 1.2(c). Our model is able to detect the data records accurately by assigning the label "`REC-B`" (*b*eginning segment of a *rec*ord) to the subtrees $\mathcal{S}_2$, $\mathcal{S}_5$, $\mathcal{S}_8$, etc., and the label "`REC-I`" (*i*nside segment of a *rec*ord) to the subtrees $\mathcal{S}_3$, $\mathcal{S}_4$, $\mathcal{S}_6$, $\mathcal{S}_7$, $\mathcal{S}_9$, etc. The challenges raised by the variable number of subtrees in these records are tackled with the global analysis on the characteristics of

the subtree sequence in this record region. Specifically, all possible label sequences are evaluated during the inference of the best label assignment for the sequence $\mathcal{S}_1, \ldots, \mathcal{S}_n$. Finally, the label sequence " 'REGNOT', 'REC-B', 'REC-I', 'REC-I', 'REC-B', 'REC-I', 'REC-I', 'REC-B', 'REC-I', 'REC-B', ..." achieves the highest value for the global objective function. With respect to the intertwined example given in Figure 1.2(d), our model assigns the label "REC-B" to the subtrees $\mathcal{S}_1$, $\mathcal{S}_4$, etc., and the label "REC-I" to the subtrees $\mathcal{S}_2$, $\mathcal{S}_3$, $\mathcal{S}_5$, $\mathcal{S}_6$, etc. Thus, the composite data records such as $\mathcal{S}_{1:3}$ and $\mathcal{S}_{4:6}$ are detected. Each segment <tr> in the composite records contains the constituents of two intertwined data records. And then, our Skoga framework invokes a separate assembling stage to assemble the intertwined records R_1, R_2, R_3, and R_4 from the detected composite records in Figure 1.2(d).

The feature weights in the DOM structure knowledge are determined using a development data set via a parameter estimation algorithm. To allow better generalization capability of the estimated feature weights when tackling the heterogenous Web pages, the maximum margin principle with soft margin is employed. Specifically, the parameter estimation algorithm is developed based on structured output Support Vector Machine (SVM) model [73]. The label output of a node in the DOM structure exhibits a tight interaction with the labels of its connected nodes such as its parent node and siblings. The structured output SVM model is able to tackle the inter-dependency among the labels on the nodes of the DOM structure so as to provide more accurate labeling solutions. Furthermore, a record region oriented loss function is designed to penalize the missing of record regions. Therefore, the acquired statistical knowledge can effectively identify multiple regions, if exist, in a single page. The development data set was arbitrarily collected from different Web sites covering different kinds of record regions such as the examples given in Figures 1.1 and 1.2. An optimization method, namely hierarchical Viterbi algorithm, is developed based on divide-and-conquer principle, which makes use of the DOM structure knowledge to quantitatively infer the optimal label assignment of record and region recognition from a page in polynomial time.

Once the DOM structure knowledge including the feature weights is determined, Skoga framework can be directly applied to detect common kinds of record regions and data records as illustrated above from any Web sites and domains without the need of labeled data or training. As a result, our framework is more robust when processing different types

of record regions so that it can achieve better effectiveness and higher efficiency. Note that we do not generate any wrapper and Skoga is site-independent. Another characteristic of Skoga is that when there is a need to detect application oriented record regions and data records which are different from the common kinds of regions or records, Skoga can conduct a training process with the application-specific labeled data to generate a tailor-made detection model for the intended application.

4.2 Design of DOM Structure Knowledge

4.2.1 Background Knowledge

As mentioned above, each node in the DOM tree is assigned a label. We design eight types of labels denoted as $\mathcal{Y}$ to capture general constituents of record regions and data records. The details of the labels are given as follows.

`REGION`: The DOM node with this label is the root node of a subtree corresponding to a record region. The region should contain either a set of data records or a set of subregions. A data record under this region may be composed of several subtrees of the current region, such as the examples given in Figure 1.2.

`SUBREG`: The DOM node with this label is the root node of a subtree corresponding to a subregion. A subregion should contain a set of data records such as the examples given in Figure 1.1(b). Each data record under this subregion may be composed of several subtrees of the current subregion.

`REC-S`: The DOM node with this label is the root node of a subtree corresponding to a complete data record and it may be composed of a group of components. For example, each data record in Figure 1.1(d) contains an image <img> and a link <a>.

`REC-B`: The DOM node with this label is the root node of a subtree corresponding to the beginning segment of a data record such as $\mathcal{S}_2$, $\mathcal{S}_5$ in Figure 1.2(c) and $\mathcal{S}_1$, $\mathcal{S}_4$ in Figure 1.2(d).

`REC-I`: The DOM node with this label is the root node of a subtree corresponding to an inside segment of a data record such as $\mathcal{S}_3$, $\mathcal{S}_4$ in Figure 1.2(c) and $\mathcal{S}_2$, $\mathcal{S}_3$

in Figure 1.2(d). Note that a segment node (labeled with `REC-B` or `REC-I`) may be composite and contain constituents of several data records such as $\mathcal{S}_1$, $\mathcal{S}_2$, etc. in Figure 1.2(d).

`REGNOT`: The DOM node with this label is the root node of a subtree containing some explanation information on the data records in a record region or a subregion, such as $\mathcal{S}_1$'s in Figures 1.1(c) and 1.2(c). Note that such node is not a part of any data record.

`RECCMP`: The DOM node with this label is the root node of a subtree corresponding to a component of a data record. Each constituent with any granularity in a data record or record segment can be regarded as a component. Thus, each descendant node under a data record or record segment has this label, such as the nodes <img> and <a> in Figure 1.1(d).

`OTHNOD`: The DOM node with this label is the root node of a subtree corresponding to any other portion located outside record regions. Therefore, such node is not a part of any record region.

Figure 4.1 shows an example of label assignment for the record region in Figure 1.2(a) with DOM structure given in Figure 1.2(c). Our label design allows broad and general types of data records and record regions. We also design some logical relations among the labels based on the common understanding of data records and record regions. Let x_p denote a particular node and x_c denote one child node of x_p. Let $\mathbf{y}$ denote the label assignment of the DOM $\mathbf{x}$ from where x_p and x_c originate; $\mathbf{y}(x_p)$ and $\mathbf{y}(x_c)$ denote the labels of x_p and x_c given by $\mathbf{y}$. The logical relations, namely BK1 to BK8, are presented as follows.

- BK1: $\mathbf{y}(x_c) = \texttt{REGION} \rightarrow \mathbf{y}(x_p) = \texttt{OTHNOD}$
- BK2: $\mathbf{y}(x_c) = \texttt{SUBREG} \rightarrow \mathbf{y}(x_p) \in \{\texttt{REGION}, \texttt{SUBREG}\}$
- BK3: $\mathbf{y}(x_c) = \texttt{REC-S} \rightarrow \mathbf{y}(x_p) \in \{\texttt{REGION}, \texttt{SUBREG}\}$
- BK4: $\mathbf{y}(x_c) = \texttt{REC-B} \rightarrow \mathbf{y}(x_p) \in \{\texttt{REGION}, \texttt{SUBREG}\}$
- BK5: $\mathbf{y}(x_c) = \texttt{REC-I} \rightarrow \mathbf{y}(x_p) \in \{\texttt{REGION}, \texttt{SUBREG}\}$

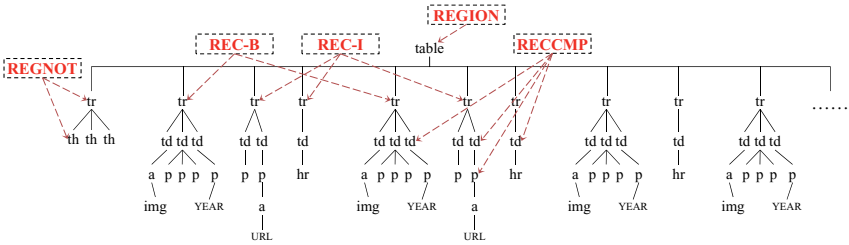


Figure 4.1: Label assignment for Figure 1.2(c).

- BK6: $\mathbf{y}(x_c) = \texttt{REGNOT} \rightarrow \mathbf{y}(x_p) \in \{\texttt{REGION}, \texttt{SUBREG}, \texttt{REGNOT}\}$
- BK7: $\mathbf{y}(x_c) = \texttt{RECCMP} \rightarrow \mathbf{y}(x_p) \in \{\texttt{REC-S}, \texttt{REC-B}, \texttt{REC-I}, \texttt{RECCMP}\}$
- BK8: $\mathbf{y}(x_c) = \texttt{OTHNOD} \rightarrow \mathbf{y}(x_p) = \texttt{OTHNOD}$

Taking BK3 as an example, if a particular node is labeled as `REC-S`, we only need to consider two candidate labels for its parent node, namely, `REGION` and `SUBREG`. These logic formulae are employed to provide guidance and constraints for the inference algorithm (presented in Section 4.3) when labeling a new DOM tree so that the inference results are more robust against noise and the inference algorithm is more efficient. These logic formulae are also used in the feature weight estimation algorithm (presented in Section 4.4) in which the inference for the optimal and the second optimal label assignments is needed during the optimization of the cost function.

4.2.2 Statistical Knowledge

As mentioned above, our Skoga framework captures the statistical knowledge represented as features and their corresponding weights. This knowledge is utilized to quantitatively evaluate the fitness of a label assignment $\mathbf{y}$ for the DOM tree $\mathbf{x}$ of a particular Web page via an objective function F. Let $\mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ denote the combined feature representation of $\mathbf{x}$ and its label assignment $\mathbf{y}$. Thus, the objective function F in Equation 4.1 is formulated as:

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) \equiv \langle \mathbf{\Psi}(\mathbf{x}, \mathbf{y}), \mathbf{w} \rangle, \tag{4.2}$$

which is the linear combination of the features in $\mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ with their corresponding weights given in $\mathbf{w}$. To exploit the tree structure and capture the inter-dependency among labels,

we design three types of features in $\mathbf{\Psi}(\mathbf{x}, \mathbf{y})$, namely, single node features, sibling features, and parent-children features.

Single node features are summarized from a single DOM node. Let $\mathbf{\Phi}(x)$ denote the feature vector related to a particular node x. The combined feature map of x and its label $y \in \mathcal{Y}$ is defined as:

$$\mathbf{\Psi}(x, y) \equiv \mathbf{\Phi}(x) \otimes \mathbf{\Lambda}^c(y), \tag{4.3}$$

where $\otimes$ is the operator of tensor multiplication. $\mathbf{\Lambda}^c(y)$ is the canonical representation of the label y:

$$\mathbf{\Lambda}^c(y) \equiv (\delta(y_1, y), \delta(y_2, y), \cdots, \delta(y_{|\mathcal{Y}|}, y)), \tag{4.4}$$

where δ is an indicator function which has the value 1 if $y_i = y$ and the value 0 otherwise. From Equation 4.3, it can be seen that each single feature is mapped to a dimension according to the label y of x. We design four types of single node features to depict different characteristics of a single node and they are described in the first section of Table 4.1. Note that Skoga is tag-independent and the tag related features only add some contribution in the overall evaluation.

Sibling features capture the relations between two neighboring nodes. Let x_{c_t} and $x_{c_{t+1}}$ be the root nodes of a pair of neighboring sibling subtrees, and their labels are y_{c_t} and $y_{c_{t+1}}$ respectively. The combined feature map of the pair $\langle x_{c_t}, x_{c_{t+1}} \rangle$ and the corresponding labels is defined as:

$$\mathbf{\Psi}(\langle x_{c_t}, x_{c_{t+1}} \rangle, y_{c_t}, y_{c_{t+1}}) \equiv \mathbf{\Phi}(\langle x_{c_t}, x_{c_{t+1}} \rangle) \otimes \mathbf{\Lambda}^c(y_{c_t}) \otimes \mathbf{\Lambda}^c(y_{c_{t+1}}), \tag{4.5}$$

where $\mathbf{\Phi}(\langle x_{c_t}, x_{c_{t+1}} \rangle)$ represents the features summarized from this sibling pair, $\otimes$ and $\mathbf{\Lambda}^c(\cdot)$ are defined as above. We design three types of sibling features and they are described in the second section of Table 4.1.

Parent-children features are designed to capture the relations between a particular node and its entire sibling sequence. Let $\langle x_p, x_c \rangle$ denote a parent-child pair, and their labels are y_p and y_c respectively. The combined feature map of the pair $\langle x_p, x_c \rangle$ and the corresponding labels is defined as:

$$\mathbf{\Psi}(\langle x_p, x_c \rangle, y_p, y_c) \equiv \mathbf{\Phi}(\langle x_p, x_c \rangle) \otimes \mathbf{\Lambda}^c(y_p) \otimes \mathbf{\Lambda}^c(y_c), \tag{4.6}$$

where $\mathbf{\Phi}(\langle x_p, x_c \rangle)$ represents the features summarized from this parent-child pair. Similarly, the feature map for the triple $\langle x_p, x_{c_t}, x_{c_{t+1}} \rangle$ and the corresponding labels can be

Table 4.1: The features used in the statistical knowledge.

Single Node Features
Tag features: These features indicate whether the current node is a special tag type, such as <table>,   , <dl>, etc. These tags have higher chance to be used in formatting data records. Although the examples in Figures 1.1 and 1.2 are rooted at <table>, it should be noted that our framework is not tag-dependent and the tag features only add some contribution in the overall evaluation.
Text appearance features: These features summarize the general characteristics of text content contained by the current node, such as the fraction of anchor text, the number of different fonts, etc.
Structure features: These features capture the aggregated characteristics of the subtree rooted at the current node, such as the number of child nodes, the number of different tags among its children, standard deviation of child subtrees' height, etc.
Special functionality features: These features capture some special characteristics that are often observed in data records, such as URL string, cash symbol, image, etc.
Sibling Features
Structure similarity: This feature is defined as the similarity of the subtrees rooted at x_{c_t} and $x_{c_{t+1}}$. Normally, the neighboring records as well as subregions as exemplified in Figure 1.1 share higher similarity, while the neighboring record segments as exemplified in Figure 1.2 share less similarity.
Skeleton similarity: Different from the structure similarity, these features capture the similarity of the skeletons of the subtrees rooted at x_{c_t} and $x_{c_{t+1}}$. Skeleton refers to the top level structure of a subtree, for instance, 2-layer or 3-layer skeletons. Skeleton similarity can overcome the dissimilarity caused by optional fields, such as the <a> node in the bottom right portion of R_2 in Figure 1.1(c).
Text similarity features: These features summarize the text similarity between the contents of x_{c_t} and $x_{c_{t+1}}$. For example, the text overlapping feature indicates the fraction of the overlapping words between the text contents. The text length difference feature indicates the length difference of the text contents.
Parent-children Features
Occurrence based on structure similarity: The number of occurrences of x_c (or $\langle x_{c_t}, x_{c_{t+1}} \rangle$) among its sibling sequence based on structure similarity. For example, the subtree structure corresponding to the data records in Figure 1.1(c) occurs many times in the child sequence of the table.
Occurrence difference between siblings: The difference of the occurrence number between x_{c_t} and $x_{c_{t+1}}$. This feature can help us capture the beginning of the record sequence more precisely, such as the ones given in Figures 1.1(c) and 1.2(c).
Occurrence interval: These features capture the characteristics of the intervals between two successive occurrences of x_{c_t} (or $\langle x_{c_t}, x_{c_{t+1}} \rangle$). For example, the interval length feature is defined as the average length of the intervals. The standard deviation feature indicates whether the intervals are regular, such as the subtree structure corresponding to the beginning segment of the records (i.e., $\mathcal{S}_2$, $\mathcal{S}_5$, etc.) in Figure 1.2(c) which occurs regularly.
Occurrence span: The number of siblings spanned by the first occurrence and the last occurrence of x_c (or $\langle x_{c_t}, x_{c_{t+1}} \rangle$). Normally, records occupy a large fraction of the subtree sequence in a record region.
Occurrence-based pivot score: This feature is defined to capture the beginning segment of a record and it is calculated as: $ps(x_{c_t}) = \begin{cases} ps(x_{c_k}) & \textit{if } x_{c_t} \textit{ is a reoccurence of } x_{c_k} \ (k < t) \\ f(x_{c_t}) \frac{1}{I\_var(x_{c_t})+\sigma} - \max\{ps(x_{c_1}), \ldots, ps(x_{c_{t-1}})\} & \textit{otherwise} \end{cases}$ where $f(x_{c_t})$ is the number of occurrence of x_{c_t}; $I\_var(x_{c_t})$ is the variance of the occurrence intervals; $\sigma = 0.01$ is used to avoid zero denominator. Considering the example in Figure 1.2(c), $ps(\mathcal{S}_2)$ is large, $ps(\mathcal{S}_3)$ and $ps(\mathcal{S}_4)$ are small, $ps(\mathcal{S}_5)$ is also large since $\mathcal{S}_5$ is a reoccurrence of $\mathcal{S}_2$.

defined as:

$$\mathbf{\Psi}(\langle x_p, x_{c_t}, x_{c_{t+1}} \rangle, y_p, y_{c_t}, y_{c_{t+1}}) \equiv \mathbf{\Phi}(\langle x_p, x_{c_t}, x_{c_{t+1}} \rangle) \otimes \mathbf{\Lambda}^c(y_p) \otimes \mathbf{\Lambda}^c(y_{c_t}) \otimes \mathbf{\Lambda}^c(y_{c_{t+1}}), \quad (4.7)$$

where $\mathbf{\Phi}(\langle x_p, x_{c_t}, x_{c_{t+1}} \rangle)$ represents the features summarized from the triple. We design five types of parent-children features and they are described in the third section of Table 4.1.

The combined feature representation of the DOM tree $\mathbf{x}$ and its label assignment $\mathbf{y}$ is the combination of the above types of feature maps:

$$\mathbf{\Psi}(\mathbf{x},\mathbf{y}) \equiv \begin{pmatrix} \sum_x \mathbf{\Psi}(x,y)^T \\ \sum_{\langle x_{c_t}, x_{c_{t+1}}\rangle} \mathbf{\Psi}(\langle x_{c_t}, x_{c_{t+1}}\rangle, y_{c_t}, y_{c_{t+1}})^T \\ \sum_{\langle x_p, x_c\rangle} \mathbf{\Psi}(\langle x_p, x_c\rangle, y_p, y_c)^T \\ \sum_{\langle x_p, x_{c_t}, x_{c_{t+1}}\rangle} \mathbf{\Psi}(\langle x_p, x_{c_t}, x_{c_{t+1}}\rangle, y_p, y_{c_t}, y_{c_{t+1}})^T \end{pmatrix}^T . \tag{4.8}$$

As shown above, different features are combined and the difference of their impacts will be captured by the corresponding weights in $\mathbf{w}$ determined via a parameter estimation algorithm with the guidance of a development data set.

4.3 Finding Optimal Label Assignment

One major task in Skoga framework is to find the optimal label assignment of a DOM tree by maximizing $F(\mathbf{x},\mathbf{y};\mathbf{w})$ as in Equation 4.1 with F defined in Equation 4.2. As a result, the aim is to solve the following optimization problem:

$$\begin{aligned} \mathbf{y}^* &= \operatorname*{argmax}_{\mathbf{y}} F(\mathbf{x},\mathbf{y};\mathbf{w}) \\ &= \operatorname*{argmax}_{\mathbf{y}} \langle \mathbf{\Psi}(\mathbf{x},\mathbf{y}), \mathbf{w}\rangle. \end{aligned} \tag{4.9}$$

It is referred to as the inference task. The inference algorithm of the entire DOM tree can be tackled with divide-and-conquer principle. Specifically, a hierarchical Viterbi algorithm is designed to obtain the optimal label assignment in polynomial time. It conducts the inference in a bottom-up manner and starts from the subtree whose height is 1. After that, the intermediate results are utilized in the inference of their parent trees.

4.3.1 Inference for Bottom Subtrees

We first describe the inference algorithm for the subtrees whose height is 1. Let x_p denote the parent node and x_{c_t} denote a child node under x_p. Let $\mathbf{y}_p = \{y_{p,1}, y_{p,2}, \cdots\}$ denote the candidate labels of x_p and $\mathbf{y}_{c_t} = \{y_{c_t,1}, y_{c_t,2}, \cdots\}$ denote the candidate labels of x_{c_t}. Take the tree $\mathbf{x} = \{x_p, x_{c_1}, x_{c_2}, \cdots, x_{c_T}\}$ in Figure 4.2 as an example, where the filled nodes represent the root node and the leaf nodes respectively. The unfilled nodes represent the label candidates of the corresponding DOM nodes. T is the total number of the children.

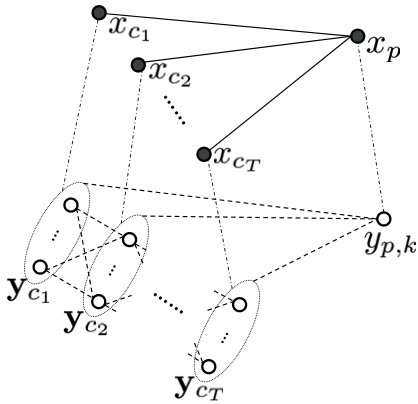


Figure 4.2: Inference lattice structure of a subtree whose height is 1.

We first fix the label of the root x_p to be $y_{p,k}$, and infer the optimal labels for the child sequence from their candidate label sets. The inference mechanism considering different combinations of labels can be represented as a lattice structure. After that, we enumerate all possible candidate labels of x_p in $\mathbf{y}_p$ to obtain the global optimal labeling.

Let $\hat{F}^{p,k}_{c_t,i}$ denote the intermediate maximum objective value on $\mathbf{x} = \{x_p, x_{c_1}, \cdots, x_{c_t}\}$ achieved by an assignment that assigns the label $y_{p,k}$ to the root x_p and $y_{c_t,i}$ to the child x_{c_t}. Then, $\hat{F}^{p,k}_{c_{t+1},j}$ is calculated as follows:

$$\hat{F}^{p,k}_{c_{t+1},j} = \max_{y_{c_t,i} \in \mathbf{y}_{c_t}} \{\hat{F}^{p,k}_{c_t,i} + \Delta F^{p,k}_{(c_t,i),(c_{t+1},j)}\}, \tag{4.10}$$

where $\Delta F^{p,k}_{(c_t,i),(c_{t+1},j)}$ denotes the increment of the objective value when taking the node $x_{c_{t+1}}$ into account with label $y_{c_{t+1},j}$, and it is calculated as:

$$\Delta F^{p,k}_{(c_t,i),(c_{t+1},j)} = \langle \Delta\mathbf{\Psi}(x_p, x_{c_t}, x_{c_{t+1}}, y_{p,k}, y_{c_t,i}, y_{c_{t+1},j}), \mathbf{w} \rangle, \tag{4.11}$$

where $\Delta\mathbf{\Psi}$ is the change of the feature vector including the single node features of $x_{c_{t+1}}$, the sibling features of $\langle x_{c_t}, x_{c_{t+1}} \rangle$, and the parent-children features of $\langle x_p, x_{c_{t+1}} \rangle$ and $\langle x_p, x_{c_t}, x_{c_{t+1}} \rangle$. To be precise, $\Delta\mathbf{\Psi}$ is represented as:

$$\Delta\mathbf{\Psi}(x_p, x_{c_t}, x_{c_{t+1}}, y_{p,k}, y_{c_t,i}, y_{c_{t+1},j}) = \begin{pmatrix} \mathbf{\Psi}(x_{c_{t+1}}, y_{c_{t+1},j})^T \\ \mathbf{\Psi}(\langle x_{c_t}, x_{c_{t+1}} \rangle, y_{c_t,i}, y_{c_{t+1},j})^T \\ \mathbf{\Psi}(\langle x_p, x_{c_{t+1}} \rangle, y_{p,k}, y_{c_{t+1},j})^T \\ \mathbf{\Psi}(\langle x_p, x_{c_t}, x_{c_{t+1}} \rangle, y_{p,k}, y_{c_t,i}, y_{c_{t+1},j})^T \end{pmatrix}^T. \tag{4.12}$$

The contributed objective value by x_p and x_{c_1} is included in $\hat{F}^{p,k}_{c_1,i}$, which is calculated as:

$$\hat{F}^{p,k}_{c_1,i} = \left\langle \begin{pmatrix} \mathbf{\Psi}(x_{c_1}, y_{c_1,i})^T + \mathbf{\Psi}(x_p, y_{p,k})^T \\ \mathbf{0} \\ \mathbf{\Psi}(\langle x_p, x_{c_1} \rangle, y_{p,k}, y_{c_1,i})^T \\ \mathbf{0} \end{pmatrix}^T, \mathbf{w} \right\rangle. \tag{4.13}$$

Recall that the labels of x_p and its child x_{c_t} should satisfy the logic formulae in Section 4.2.1. Taking these formulae into consideration in the calculation of Equation 4.10, when a particular $y_{c_{t+1},j}$ together with $y_{p,k}$ violates any formula, it can be automatically pruned. Similarly, Equation 4.13 is also constrained by these logic formulae.

For a pre-assigned label $y_{p,k}$ of x_p, the maximum objective value that can be achieved by labeling its child nodes is denoted by $\hat{F}^{p,k}$, which is calculated as:

$$\hat{F}^{p,k} = \max_{y_{c_T,i} \in \mathbf{y}_{c_T}} \hat{F}^{p,k}_{c_T,i}, \tag{4.14}$$

Finally, the maximum objective value achieved by the optimal label assignment can be obtained by enumerating all possible $y_{p,k}$ for x_p:

$$\hat{F} = \max_{y_{p,k} \in \mathbf{y}_p} \hat{F}^{p,k}. \tag{4.15}$$

The time complexity of the above inference for a bottom single-depth tree can be derived from that of the standard Viterbi algorithm [64]. For each candidate label of the root x_p, the standard Viterbi algorithm is invoked to infer the optimal labels for the children. Therefore, the time complexity is $\mathbf{O}(|\mathcal{Y}|^3 * T)$, where $\mathcal{Y}$ is the set of all possible labels. Note that the above time complexity analysis has not taken the logical relations (see Section 4.2.1) into consideration. Incorporating the logical relations makes the inference algorithm even more efficient.

4.3.2 Recursive Inference for Higher Subtrees

In the previous subsection, each child node is assumed to be a leaf. When processing the higher level subtrees, the intermediate results from the lower level subtrees rooted at each child x_{c_t} are taken into consideration. Let $\hat{F}^{c_t,i}$ denote the optimal value achieved in labeling the subtree rooted at x_{c_t} which is labeled with $y_{c_t,i}$. The objective value increment $\Delta F^{p,k}_{(c_t,i),(c_{t+1},j)}$ is calculated as:

$$\Delta F^{p,k}_{(c_t,i),(c_{t+1},j)} = \hat{F}^{c_{t+1},j} + \langle \Delta \mathbf{\Psi}'(x_p, x_{c_t}, x_{c_{t+1}}, y_{p,k}, y_{c_t,i}, y_{c_{t+1},j}), \mathbf{w} \rangle, \tag{4.16}$$

where $\Delta\mathbf{\Psi}'$ is calculated as:

$$\Delta\mathbf{\Psi}'(x_p, x_{c_t}, x_{c_{t+1}}, y_{p,k}, y_{c_t,i}, y_{c_{t+1},j}) = \begin{pmatrix} \mathbf{0} \\ \mathbf{\Psi}(\langle x_{c_t}, x_{c_{t+1}}\rangle, y_{c_t,i}, y_{c_{t+1},j})^T \\ \mathbf{\Psi}(\langle x_p, x_{c_{t+1}}\rangle, y_{p,k}, y_{c_{t+1},j})^T \\ \mathbf{\Psi}(\langle x_p, x_{c_t}, x_{c_{t+1}}\rangle, y_{p,k}, y_{c_t,i}, y_{c_{t+1},j})^T \end{pmatrix}^T . \quad (4.17)$$

Similarly, $\hat{F}^{p,k}_{c_1,i}$ is calculated as:

$$\hat{F}^{p,k}_{c_1,i} = \hat{F}^{c_1,i} + \left\langle \begin{pmatrix} \mathbf{\Psi}(x_p, y_{p,k})^T \\ \mathbf{0} \\ \mathbf{\Psi}(\langle x_p, x_{c_1}\rangle, y_{p,k}, y_{c_1,i})^T \\ \mathbf{0} \end{pmatrix}^T , \mathbf{w} \right\rangle . \quad (4.18)$$

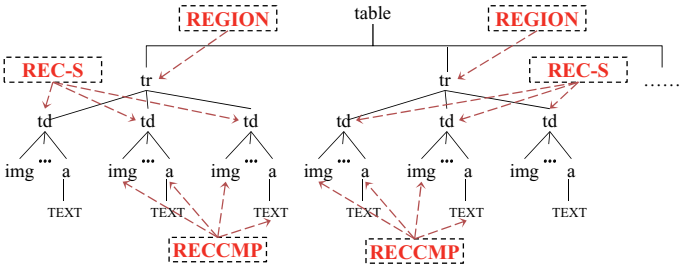
With the above recursive property, the inference is conducted layer by layer starting from the bottom of the DOM tree. After the recursion is finished at the root of the DOM tree, the global optimal value of F is obtained.

Figure 4.3 shows the recursive inference procedure for the nested region in Figure 1.1(b). When inferring the labels of the subtrees rooted at <tr>'s, referring to Figure 4.3(a), the optimal label for the <tr>'s is "`REGION`" and the optimal label for the <td>'s is "`REC-S`". Currently, the label "`SUBREG`" for <tr> is not the optimal choice. When coming to the higher level <table>, referring to Figure 4.3(b), the label of <tr>'s favors towards "`SUBREG`" and <table> tends to be labeled as "`REGION`".

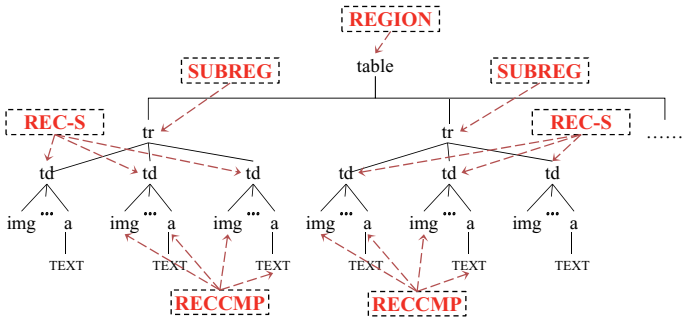
Given that the inference of all lower level subtrees has been done, the time complexity of the higher level subtree inference is $\mathbf{O}(|\mathcal{Y}|^3 * T')$ where T' is the child number of this subtree. Therefore, the time complexity of the entire DOM tree inference can be decomposed into those of all child sequences in any layer of the DOM tree. Let T_i denote the length of a particular child sequence, the overall time complexity for the entire DOM tree is computed as $\mathbf{O}(\sum_i |\mathcal{Y}|^3 * T_i) = \mathbf{O}(|\mathcal{Y}|^3 * |\mathbf{x}|)$ where $|\mathbf{x}|$ denotes the total number of nodes in the DOM $\mathbf{x}$.

4.3.3 Backtracking for the Optimal Label Assignment

The backtracking of the optimal label for the entire DOM tree is carried out with a top-down manner starting from the root of the tree. The best label of the root node is



(a) Inference results for the subtrees rooted at <tr>'s.



(b) Inference result for the nested region <table>.

Figure 4.3: Recursive inference for the nested region in Figure 1.1(b).

obtained by:

$$y_{p,*} = \underset{y_{p,k} \in \mathbf{y}_p}{\operatorname{argmax}} \hat{F}^{p,k}. \tag{4.19}$$

Let $y_{c_t,*}$ denote the label of x_{c_t} in the optimal label assignment for the child sequence with the root node x_p labeled with $y_{p,*}$. Thus, the optimal label of the last child x_{c_T} is:

$$y_{c_T,*} = \underset{y_{c_T,i} \in \mathbf{y}_{c_T}}{\operatorname{argmax}} \hat{F}^{p,*}_{c_T,i}. \tag{4.20}$$

By backtracking, $y_{c_t,*}$ is obtained as:

$$y_{c_t,*} = \underset{y_{c_t,i} \in \mathbf{y}_{c_t}}{\operatorname{argmax}} \{\hat{F}^{p,*}_{c_t,i}. + \Delta F^{p,*}_{(c_t,i),(c_{t+1},*)}\} \tag{4.21}$$

After the optimal label $y_{c_t,*}$ of each x_{c_t} is obtained, we can repeat the same procedure as given in Equations 4.20 and 4.21 to obtain the optimal labels for the children of x_{c_t}.

The above backtracking can be implemented by keeping backward pointers during the recursive calculation of the global optimal objective value. Thus, given the optimal label

$y_{p,*}$ of the DOM root, we can obtain the optimal label assignment for the entire DOM tree by backtracking through backward pointers in a top-down manner.

4.3.4 Second Optimal Label Assignment

Owning to the fact that the feature weight estimation is conducted with a maximum margin principle based algorithm, the second optimal label assignment is needed when maximizing the margin.

Second Optimal Inference for Bottom Subtrees

Returning to the example in Figure 4.2, recall that its optimal label assignment is denoted as $(y_{p,*}, y_{c_1,*}, y_{c_2,*}, \cdots, y_{c_T,*})$. The second optimal objective value is achieved by one of the following cases: (1) The label of x_p is not $y_{p,*}$ and the labels of the children also change accordingly. Let $\ddot{F}^{p,\bar{*}}$ denote the achieved value in this case; (2) The label of x_p is still $y_{p,*}$, but the label sequence of the children is different from $(y_{c_1,*}, y_{c_2,*}, \cdots, y_{c_T,*})$. Let $\ddot{F}^{p,*}$ denote the achieved value in this case. Then the second optimal objective value is obtained by:

$$\ddot{F} = \max\{\ddot{F}^{p,\bar{*}}, \ddot{F}^{p,*}\}. \tag{4.22}$$

$\ddot{F}^{p,\bar{*}}$ and $\ddot{F}^{p,*}$ are calculated by Equations 4.23 and 4.24:

$$\ddot{F}^{p,\bar{*}} = \max_{y_{p,k} \in \mathbf{y}_p \setminus y_{p,*}} \hat{F}^{p,k}, \tag{4.23}$$

$$\ddot{F}^{p,*} = \max\{\max_{y_{c_T,i} \in \mathbf{y}_{c_T} \setminus y_{c_T,*}} \hat{F}^{p,*}_{c_T,i}\ ,\ \ddot{F}^{p,*}_{c_T,*}\}. \tag{4.24}$$

The first term of Equation 4.24 is the best value achieved by labeling the last child with another label other than $y_{c_T,*}$. The second term of Equation 4.24 denotes the second optimal value achieved by still labeling the last child with $y_{c_T,*}$, which can be recursively calculated. Let $\ddot{F}^{p,*}_{c_{t:T},*}$ denote the second optimal value achieved by labeling the child sequence from t to T with the optimal labels $(y_{c_t,*}, \cdots, y_{c_T,*})$. $\ddot{F}^{p,*}_{c_{t:T},*}$ is recursively calculated as:

$$\ddot{F}^{p,*}_{c_{t:T},*} = \max\{\ddot{F}^{p,*}_{c_{t-1:T},*}\ ,\ \max_{y_{c_{t-1},i} \in \mathbf{y}_{c_{t-1}} \setminus y_{c_{t-1},*}} \{\hat{F}^{p,*}_{c_{t-1},i} + \Delta F^{p,*}_{(c_{t-1},i),(c_t,*)}\} + \Delta F^{p,*}_{c_{t:T},*}\}, \tag{4.25}$$

where the second term is composed of two parts, namely, the partial second optimal value achieved up to the child x_{c_t} which is labeled with $y_{c_t,*}$, and the objective value increment

$\Delta F^{p,*}_{c_t:T,*}$ achieved by the optimal labels $(y_{c_t,*}, \cdots, y_{c_T,*})$. $\Delta F^{p,*}_{c_t:T,*}$ is calculated as:

$$\Delta F^{p,*}_{c_t:T,*} = \sum_{t'=t}^{T-1} \Delta F^{p,*}_{(c_{t'},*),(c_{t'+1},*)}. \tag{4.26}$$

Since we assume that each child node is a leaf, the termination condition of the recursion in Equation 4.25 is $t = 2$, i.e., the label of the first child changes and the remaining children's labels do not change:

$$\ddot{F}^{p,*}_{c_2:T,*} = \max_{y_{c_1,i} \in \mathbf{y}_{c_1} \setminus y_{c_1,*}} \{\hat{F}^{p,*}_{c_1,i} + \Delta F^{p,*}_{(c_1,i),(c_2,*)}\} + \Delta F^{p,*}_{c_2:T,*}. \tag{4.27}$$

Second Optimal Inference for Higher Subtrees

When processing the higher level subtrees, we can take the lower level subtrees rooted at each child x_{c_t} into consideration. Thus, the global second optimal value may be achieved in the case that all labels of x_p and its children x_{c_t}'s are still the same as those for the global optimal value, and the labels of some descendants of a certain x_{c_t} change. Following the above notation, this value is denoted as $\ddot{F}^{p,*}_{c_1:T,*}$, which is calculated as:

$$\ddot{F}^{p,*}_{c_1:T,*} = \max_{x_{c_t}} \{\hat{F} - \hat{F}^{c_t,*} + \ddot{F}^{c_t,*}\}, \tag{4.28}$$

where $\ddot{F}^{c_t,*}$ denotes the second optimal value achieved in labeling the subtree rooted at x_{c_t} which is still labeled with $y_{c_t,*}$. $\ddot{F}^{c_t,*}$ can be calculated by Equation 4.24. Finally, the global second optimal objective value $\ddot{F}$ is calculated as:

$$\ddot{F} = \max\{\ddot{F}^{p,\bar{*}}, \ddot{F}^{p,*}, \ddot{F}^{p,*}_{c_1:T,*}\}. \tag{4.29}$$

Thus, $\ddot{F}$ can be obtained with a recursive manner starting from the bottom of the DOM tree. Similarly, backward pointers are kept for backtracking the second optimal label assignment.

4.4 Statistical Knowledge Acquisition

In this section, we discuss the determination of the feature weights of the statistical knowledge using a development data set via a parameter estimation algorithm based on structured output Support Vector Machine (SVM) model [73]. This model can tackle the inter-dependency among the labels on the nodes of the DOM structure. Meanwhile, the maximum margin principle of SVM allows the determined feature weights a better generalization capability to harness the heterogeneity of Web content.

4.4.1 Finding Feature Weights via Structured Output SVM Learning

Let $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ denote a set of development data instances, and $\Delta(\mathbf{y}_i, \mathbf{y})$ denote the loss caused by assigning the label assignment $\mathbf{y}$ to $\mathbf{x}_i$. The quadratic program form of the SVM model with slack re-scaled by the loss is:

$$\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$
$$s.t. \quad \forall i, \forall \mathbf{y} \in \mathcal{Y}^{|\mathbf{x}_i|} \setminus \mathbf{y}_i : \langle \delta\mathbf{\Psi}_i(\mathbf{y}), \mathbf{w}\rangle \geq 1 - \frac{\xi_i}{\Delta(\mathbf{y}_i, \mathbf{y})}, \tag{4.30}$$

where ξ_i is the slack variable of $\mathbf{x}_i$, $C > 0$ is a tradeoff constant of the two parts, and $\langle \delta\mathbf{\Psi}_i(\mathbf{y}), \mathbf{w}\rangle = F(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}) - F(\mathbf{x}_i, \mathbf{y}; \mathbf{w})$ is the margin between the objective values of $\mathbf{y}_i$ and $\mathbf{y}$. Tsochantaridis et al. proposed a cutting plane based algorithm to solve this optimization problem in its dual formulation [73]. It selects a subset of constraints from the exponential full set $\mathcal{Y}^{|\mathbf{x}_i|}$ to ensure a sufficiently accurate solution. The procedure of finding the feature weights is briefly summarized in Algorithm 1. S_i is the working set of selected constraints for the instance $\mathbf{x}_i$, α's are the Lagrange multipliers, and ε is the precision parameter. The algorithm proceeds by finding the most violated constraint for $\mathbf{x}_i$ involving $\mathbf{y}^*$ (refer to Line 5). If the margin violation of this constraint exceeds the current ξ_i by more than ε (refer to Line 7), the working set S_i of $\mathbf{x}_i$ is updated. α's and $\mathbf{w}$ are also updated with the updated working set accordingly. We refer the reader to [73] for more details of the algorithm.

4.4.2 Region-oriented Loss

In record region detection and data record extraction, we wish to avoid the missing of data record regions since it will result in false negative predictions on all data records in this region. As observed in the existing works [67], one major difficulty of record region detection is to find out all the regions in a given Web page. Some existing methods only reported the largest region and missed the others [94]. To tackle this issue, we define a loss function that penalizes the missing of record regions.

Let $\mathbf{x}_{reg}$ denote the set of root nodes of the subtrees corresponding to record regions in a Web page, the loss function is defined as:

$$\Delta(\mathbf{y}_i, \mathbf{y}) \equiv \exp\{\frac{\sum_{x\in\mathbf{x}_{reg}} \bar{\delta}(\mathbf{y}_i(x), \mathbf{y}(x))}{|\mathbf{x}_{reg}|}\}, \tag{4.31}$$

Algorithm 1: Finding feature weights via structured output SVM learning.

1: **initialization**: $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, C, ε, $\forall i : S_i \leftarrow \emptyset$
2: **repeat**
3: **for** $i = 1, \cdots, n$ **do**
4: $H(\mathbf{y}) \equiv (1 - \langle \delta\mathbf{\Psi}_i(\mathbf{y}), \mathbf{w} \rangle)\Delta(\mathbf{y}_i, \mathbf{y})$ *//cost function*
5: $\mathbf{y}^* = \text{argmax}_{\mathbf{y} \in \mathcal{Y}^{|\mathbf{x}_i|}} H(\mathbf{y})$ *//cutting plane*
6: $\xi_i = \max\{0, \max_{\mathbf{y} \in S_i} H(\mathbf{y})\}$
7: **if** $H(\mathbf{y}^*) > \xi_i + \varepsilon$ **then**
8: $S_i \leftarrow S_i \cup \{\mathbf{y}^*\}$
9: update α's and $\mathbf{w}$ with $\cup_i S_i$
10: **end if**
11: **end for**
12: **until** no S_i has changed during iteration

where $\bar{\delta}$ is an indicator function which has the value 0 if $\mathbf{y}_i(x) = \mathbf{y}(x)$ and the value 1 otherwise. The loss function Δ is monotonically increasing with respect to the number of wrongly labeled regions. Note that we do not adopt zero diagonal loss function. The reason is that our loss function does not consider all the nodes of a DOM tree and it concentrates on the essential parts, namely, record regions.

4.4.3 Cost Function Optimization

In the learning procedure as depicted in Algorithm 1, it is required to optimize the cost function in Line 4 for finding the most violated constraint corresponding to $\mathbf{y}^*$:

$$\mathbf{y}^* = \underset{\mathbf{y} \in \mathcal{Y}^{|\mathbf{x}_i|}}{\text{argmax}} H(\mathbf{y}). \tag{4.32}$$

When $H(\mathbf{y}^*) \leq 0$, $\mathbf{y}^*$ will not be added into the working set S_i since the margin $\langle \delta\mathbf{\Psi}_i(\mathbf{y}^*), \mathbf{w} \rangle$ is larger than or equal to 1 (refer to Line 7). Thus, we only need to consider the cases when $H(\mathbf{y}^*) > 0$, i.e., $\langle \delta\mathbf{\Psi}_i(\mathbf{y}^*), \mathbf{w} \rangle < 1$.

We first conduct inference for $\mathbf{x}_i$ based on the current $\mathbf{w}$. Let $\hat{\mathbf{y}}$ denote the label assignment that achieves the optimal objective value $F(\mathbf{x}_i, \hat{\mathbf{y}}; \mathbf{w})$ denoted as $F_i(\hat{\mathbf{y}})$ for short. If $\langle \delta\mathbf{\Psi}_i(\hat{\mathbf{y}}), \mathbf{w} \rangle \geq 1$, we directly move to $\mathbf{x}_{i+1}$. The reason is that given $F_i(\hat{\mathbf{y}}) \geq F_i(\mathbf{y}^*)$, $\langle \delta\mathbf{\Psi}_i(\hat{\mathbf{y}}), \mathbf{w} \rangle = F_i(\mathbf{y}_i) - F_i(\hat{\mathbf{y}})$ and $\langle \delta\mathbf{\Psi}_i(\mathbf{y}^*), \mathbf{w} \rangle = F_i(\mathbf{y}_i) - F_i(\mathbf{y}^*)$, we have $\langle \delta\mathbf{\Psi}_i(\mathbf{y}^*), \mathbf{w} \rangle \geq \langle \delta\mathbf{\Psi}_i(\hat{\mathbf{y}}), \mathbf{w} \rangle \geq 1$. If $\langle \delta\mathbf{\Psi}_i(\hat{\mathbf{y}}), \mathbf{w} \rangle < 1$, we utilize $\hat{\mathbf{y}}$ to derive $\mathbf{y}^*$.

Proposition 1. *If all regions in* $\mathbf{x}_{reg}$ *are wrongly labeled in* $\hat{\mathbf{y}}$ *and given* $\langle \delta\mathbf{\Psi}_i(\hat{\mathbf{y}}), \mathbf{w}\rangle < 1$*, then we have* $\mathbf{y}^* = \hat{\mathbf{y}}$*.*

Proof. Note that the loss function Δ is monotonically increasing with respect to the number of wrongly labeled regions. Then $\Delta(\mathbf{y}_i, \hat{\mathbf{y}})$ is maximized when all regions in $\mathbf{x}_{reg}$ are wrongly labeled. In addition, $\hat{\mathbf{y}}$ maximizes F so that $1 - \langle \delta\mathbf{\Psi}_i(\hat{\mathbf{y}}), \mathbf{w}\rangle$ is also maximized. Taking $\Delta(\mathbf{y}_i, \hat{\mathbf{y}}) > 0$ and $\langle \delta\mathbf{\Psi}_i(\hat{\mathbf{y}}), \mathbf{w}\rangle < 1$ into consideration, $H(\hat{\mathbf{y}})$ is the maximum. □ □

Proposition 2. *Let* $\mathbf{y}'$ *be a label assignment that has less than or equal number of wrongly labeled regions than* $\hat{\mathbf{y}}$ *and given* $\langle \delta\mathbf{\Psi}_i(\hat{\mathbf{y}}), \mathbf{w}\rangle < 1$*, then we have* $H(\mathbf{y}') \leq H(\hat{\mathbf{y}})$*.*

Proof. Because $\hat{\mathbf{y}}$ maximizes F, $F_i(\mathbf{y}') \leq F_i(\hat{\mathbf{y}})$. Thus, we have $(1 - \langle \delta\mathbf{\Psi}_i(\mathbf{y}'), \mathbf{w}\rangle) \leq (1 - \langle \delta\mathbf{\Psi}_i(\hat{\mathbf{y}}), \mathbf{w}\rangle)$. In addition, $0 < \Delta(\mathbf{y}_i, \mathbf{y}') \leq \Delta(\mathbf{y}_i, \hat{\mathbf{y}})$ and $(1 - \langle \delta\mathbf{\Psi}_i(\hat{\mathbf{y}}), \mathbf{w}\rangle) > 0$, therefore $H(\mathbf{y}') \leq H(\hat{\mathbf{y}})$. □ □

Based on Proposition 2, we only need to optimize $H(\mathbf{y})$ among the label assignments that have more wrongly labeled regions than $\hat{\mathbf{y}}$, at the same time satisfying $\langle \delta\mathbf{\Psi}_i(\mathbf{y}), \mathbf{w}\rangle < 1$. Let $\mathbf{x}^{\times}_{reg}$ denote the regions labeled wrongly in $\hat{\mathbf{y}}$. The procedure of deriving $\mathbf{y}^*$ is summarized as Proposition 3.

Proposition 3. *Let* $\{\mathbf{x}'_{reg}\}$ *be all subsets of* $\mathbf{x}_{reg}$ *having more elements than* $\mathbf{x}^{\times}_{reg}$*, and let* $\mathbf{y}'$ *be the label assignment that achieves the largest* F *value when all regions in* $\mathbf{x}'_{reg}$ *are wrongly labeled and all regions out of* $\mathbf{x}'_{reg}$ *are correctly labeled. The label* $\mathbf{y}^*$ *maximizing* $H(\mathbf{y})$ *is from* $\cup_{\mathbf{x}'_{reg}}\{\mathbf{y}'\} \cup \{\hat{\mathbf{y}}\}$*.*

We can first fix the loss value to a constant by selecting an $\mathbf{x}'_{reg}$ and removing the label `REGION` from the candidate label set of the regions in $\mathbf{x}'_{reg}$, meanwhile the label `REGION` is pre-assigned to the other regions out of $\mathbf{x}'_{reg}$. Then, the term $(1 - \langle \delta\mathbf{\Psi}_i(\mathbf{y}), \mathbf{w}\rangle)$ in H is maximized with the inference algorithm and $\mathbf{y}'$ is obtained. If $\langle \delta\mathbf{\Psi}_i(\mathbf{y}'), \mathbf{w}\rangle \geq 1$, $\mathbf{y}'$ is pruned. Otherwise, $\mathbf{y}'$ is one candidate for finding $\mathbf{y}^*$. The same procedure is repeated for all the other $\mathbf{x}'_{reg}$. Typically, the number of regions in a page is limited, normally, no more than 5. So the above enumeration method can work in affordable time in practice. To save the computational time, some existing intermediate results in the computation of $\hat{\mathbf{y}}$ can be reused.

Algorithm 2: Assembling intertwined data records.

input: data record $\mathbf{R} = \{\mathcal{S}_{i..j}\}$,
  where $i < j$ for each $\mathcal{S}_{i..j}$
output: intertwined data records $\mathbf{R}'$
$c \leftarrow 0$, $\mathbf{R}' \leftarrow \emptyset$
for $\mathcal{S}_{i..j} \in \mathbf{R}$ **do**
  if $is\_composite\_record(\mathcal{S}_{i..j})$ **then**
    $c \leftarrow c + 1$
  end if
end for
if $c/|\mathbf{R}| \geq \theta'$ **then**
  for $\mathcal{S}_{i..j} \in \mathbf{R}$ **do**
    $\mathbf{S}_k \leftarrow$ subtrees of $\mathcal{S}_k$, where $i \leq k \leq j$
    $\mathcal{S}_l^k$ denotes the l-th subtree of $\mathcal{S}_k$
    $size \leftarrow \max_{k=i}^{j} |\mathbf{S}_k|$
    for $l = 1, \cdots, size$ **do**
      $\mathbf{R}' \leftarrow \mathbf{R}' \cup \{\mathcal{S}_l^i \ldots \mathcal{S}_l^j\}$
    end for
  end for
end if

proc $is\_composite\_record(\mathcal{S}_{i..j})$
$s_1 \leftarrow 0$, $s_2 \leftarrow 0$, $c_1 \leftarrow 0$, $c_2 \leftarrow 0$
$\mathbf{S}_k \leftarrow$ subtrees of $\mathcal{S}_k$, where $i \leq k \leq j$
for $k = i, \cdots, j$ **do**
  for $\mathcal{S}_l^k, \mathcal{S}_r^k \in \mathbf{S}_k$ **and** $l \neq r$ **do**
    $s_1 \leftarrow s_1 + sim(\mathcal{S}_l^k, \mathcal{S}_r^k)$
    $c_1 \leftarrow c_1 + 1$
  end for
  for $t = i + 1, \cdots, j$ **do**
    for $\mathcal{S}_l^k \in \mathbf{S}_k$ **and** $\mathcal{S}_r^t \in \mathbf{S}_t$ **do**
      $s_2 \leftarrow s_2 + sim(\mathcal{S}_l^k, \mathcal{S}_r^t)$
      $c_2 \leftarrow c_2 + 1$
    end for
  end for
end for
if $s_1/c_1 \geq \theta$ $s_2/c_2 < \theta$ **then**
  return true
else
  return false
end if

Note that it is possible that $\hat{\mathbf{y}}$ is the same as $\mathbf{y}_i$. To tackle this issue, we infer the second best label assignment and use it instead of $\hat{\mathbf{y}}$ to go through the same procedure as above.

4.5 Assembling Intertwined Records

Intertwined records, also known as *non-continuous records* in DEPTA [91] and *cross records* in [95], refer to records whose attributes intertwine together with other records' attributes. One example and its DOM tree are given in Figures 1.2(b) and 1.2(d) respectively. Each record has 3 attributes, namely, image, title, and price, and these attributes are scattered in 3 successive <tr>'s, such as $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$. Such subtree group, e.g., $\mathcal{S}_{1..3}$, is named *composite record*.

Our Skoga framework described above is able to detect the composite records together with the complicated flat records using a global analysis. To determine and assemble possible intertwined records, Skoga invokes the method depicted in Algorithm 2. Let $\mathbf{R} = \{\mathcal{S}_{i..j}\}$ denote such a set of data records from the same record region. We first judge whether $\mathbf{R}$ is a set of composite data records. In Line 6 of the main procedure on the left-hand side, each $\mathcal{S}_{i..j}$ is passed to the sub-procedure *is_composite_record* depicted on the right-hand side to determine whether it is a composite record. Referring to Line 16 of *is_composite_record*, if the average similarity s_1/c_1, calculated as in Lines 5 to 8, of the subtree pairs from a single $\mathcal{S}_k$ $(i \leq k \leq j)$ is greater than or equal to a threshold θ and the average similarity s_2/c_2, calculated as in Lines 9 to 14, of the subtree pairs with two subtrees from $\mathcal{S}_k$ and $\mathcal{S}_t$ $(k \neq t)$ is less than θ, we are confident to determine that $\mathcal{S}_{i..j}$ is a composite record. If the ratio of composite records in $|\mathbf{R}|$ is no less than a ratio threshold θ', as shown in Line 10 of the main procedure, we conclude that $\mathbf{R}$ is a composite record set. Then, the intertwined records can be easily assembled from the composite records, as shown in Lines 11 to 18 of the main procedure. Returning to the example given in Figures 1.2(b) and 1.2(d), two records (image 1, title 1, price 1) and (image 2, title 2, price 2) are assembled from the first three <tr>'s.

Chapter 5

Experimental Results of Data Record Extraction

5.1 Evaluation Data Sets

In our experiments, four evaluation data sets are used to evaluate the performance of the proposed Skoga framework. The first data set is the testbed collected by Yamada et al. [89]. It can be considered as a benchmark data set [1] which has been used for evaluation in some works such as ViPER [67] and TPC [49]. It is referred to as TB1 in this paper. TB1 has 253 Web pages from 51 Web sites randomly drawn from 114,540 Web pages with search forms of various search engines, such as picture search, product search and document search. One sample data record of each record region in the pages was given by the collectors of this data set. Almost all pages contain flat regions or complicated flat regions. In our experiment, two sites were excluded because of garbled code or ambiguous record annotation. Thus, we used the remaining 49 sites including 243 pages and 4,326 data records in total.

Different from TB1, in which the records are search results in dynamic Web pages generated by server-side programs with predefined templates, the second data set is composed of static Web pages in which the data records are presented in formats exhibiting some regularities. This data set is referred to as TB2 [2]. The pages were collected from different online shopping and university Web sites. The targeted university pages are the

[1] It is publicly available at `http://daisen.cc.kyushu-u.ac.jp/TBDW/`.

[2] It is publicly available at `http://www.se.cuhk.edu.hk/~textmine/`.

ones that contain the faculty list of a particular department. To obtain such kind of pages, we issued a synthetic query in the form of "faculty list site:xxx.edu" to Google and then browsed the result pages to find the ones with record regions. To collect the shopping Web pages, we investigated the online shopping Web sites one by one in an online shopping yellow page `http://www.usaonlineshoppingguide.com/`. There are 37 categories such as "Art Collectibles" and "Baby Stores" in this yellow page, and each category has 10 recommended shopping Web sites on average. We clicked the navigation links in the randomly selected sites to obtain record pages and at most 2 pages were collected from a single site. This data set contains 200 pages and 5,713 data records in total.

We prepared the third data set to examine the performance of Skoga on complicated flat regions and intertwined regions. The pages with flat and complicated flat regions were collected from different online shopping sites and university sites. The pages with intertwined regions were only collected from online shopping sites since very few university Web sites adopt the intertwined manner in presenting the faculty information. This data set was collected in the same manner as above and it is referred to as TB3 [3]. TB3 contains 100 pages and 2,158 data records in total. There are 50 pages containing some product lists and at most 5 pages were collected from a single shopping site. The other pages were collected from department sites of different universities.

The last data set contains data records of user-generated content such as reader comments, customer reviews, and forum posts. The pages were collected from news channels, such as BBC, CNN, ABC, etc., online shopping sites, such as Amazon, eBay, AliExpress, etc., and online forums, such as forums.asp.net, forums.d2jsp.org, forums.phpfreaks.com, etc. The posts or reviews are written by the users and embedded in the predefined templates of the corresponding Web sites. We regard each of them as one data record and test whether Skoga framework can accurately extract them. In total, this data set contains 100 pages and 2,318 data records coming from about 30 different Web sites and at most 4 pages were collected from a single site. This data is referred to as TB4.

[3] It is publicly available at `http://www.se.cuhk.edu.hk/~textmine/`.

5.2 Experimental Setup

One of the comparison methods is MDR [42] [4] which is able to deal with flat, nested, and intertwined records. DEPTA [91] is another comparison method in our experiment. DEPTA employs some rendering information to construct the DOM tree and uses tree edit distance instead of string edit distance in the calculation of generalized node similarity. Since no implementation of this method is available, we implemented this method by following the pseudo-code presented in [91]. To enhance it so that it can take the advantage of the development data set, we develop a parameter estimation process for automatically determining a major model parameter on top of the basic DEPTA algorithm using the development data set. Specifically, the basic DEPTA algorithm has one major parameter that affects the performance of record detection, namely, the threshold τ of the normalized tree distance between generalized nodes. Basically, we varied this parameter and determined the parameter value that achieves the highest performance in the development data set. The found value for τ was 0.36. Considering the heterogeneity of the record regions, the maximum generalized node length used was 12 in our experiments and it is lager than the value 10 in [91] to obtain better performance although it increases the running time of the program. This enhanced DEPTA is referred to as DEPTA+ in this paper. The authors of FiVaTech [36] kindly provided us the demo system. Therefore, we can also conduct comparison with FiVaTech on all the evaluation data sets except TB3 since this method is not designed to tackle intertwined record regions. We also compared with another existing method, namely, TPC [49] by simply retrieving the experimental results of TPC available on TB1 data set we used.

We employ the commonly used precision, recall, and F-measure as the evaluation metrics. They are calculated as follows:

$$precision = \frac{|\{\text{true data records}\} \cap \{\text{identified data records}\}|}{|\{\text{identified data records}\}|}, \tag{5.1}$$

$$recall = \frac{|\{\text{true data records}\} \cap \{\text{identified data records}\}|}{|\{\text{true data records}\}|}, \tag{5.2}$$

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \tag{5.3}$$

[4] MDR is publicly available at `http://www.cs.uic.edu/~liub/WebDataExtraction/`.

Both micro-averaged and macro-averaged values are reported to provide a more comprehensive perspective on the performance. A micro-averaged value is computed by first aggregating the ground truth data records and the identified data records from all the pages. Then micro-averaged precision, recall, and F-measure are calculated based on these two global sets. In contrast, macro-averaged values are calculated by first calculating precision, recall, and F-measure for each Web page and finally taking the average values of all the pages.

As mentioned before, a separate development data set was collected to conduct the estimation of feature weights of statistical structure knowledge. It was also used to enhance the existing method DEPTA for comparison. This data set contains Web pages originated from different types of sources. Some pages come from the data set 3 in ViNTs [94] [5]. This data set is originated from Omini [10] testbed collected by Buttler et al. which consists of more than 2,000 Web pages collected from 50 Web sites. ViNTs took one random page per Web site to construct its data set 3. The pages in this set are mainly search result pages from different search engines including vertical search engines and general search engines. The regions in these pages are of flat type or complicated flat type. We also collected some other pages from the online shopping Web sites listed in another online shopping yellow page `http://www.toponlineshopping.com/`. There are 22 categories such as "Art & Collectibles" and "Beauty & Fragrances" in this yellow page, and each category has about 6 sub-categories on average. Under each sub-category, we arbitrarily selected 3 recommended Web sites. We directly clicked the navigation links in the home page to obtain data record pages. In this way, we collected 100 data record pages presenting some product lists. About half of these pages contain nested record regions, and the other pages contain other types of data record regions.

A Firefox plug-in program was developed to assist the annotators in preparing the development data instances from the pages in the development data set. The annotators appended the labels as attributes to the DOM nodes inside record regions with the plug-in program. Take the nested region in Figure 1.1(d) as an example, after the annotation, each <td> node becomes the form of "<td label='`REC-S`'>", each <tr> node becomes the form of "<tr label='`SUBREG`'>", etc.

We used the development data set to tune the parameter values in the RST framework.

[5] It is publicly available at `http://www.data.binghamton.edu:8080/vints/`.

θ is set to 0.75, θ' is set to 0.65, K is set to 10, both k and r are set to 5. In the Skoga framework, the parameters C and ε in the feature weight estimation algorithm depicted in Algorithm 1 were set to be 5 and 0.5 respectively. The similarity threshold for counting the occurrence in the parent-children features in Section 4.2.2 was set to be 0.6 and the similarity measure used is the same as the one proposed in the RST framework as presented in Section 3.4. In Algorithm 2, the average similarity threshold θ is set to be 0.65 and the ratio threshold θ' is set to be 80%. We ran MDR on the evaluation data sets with the default similarity threshold 60% and extracted the records reported.

5.3 Experimental Results on TB1

TPC [49] also conducted experiment on this data set. The authors kindly provided us the Web site IDs in the subset they used, which contains 43 sites out of 49 sites we use. Therefore, without implementing their method, we can still conduct a fair comparison. MDR could not produce output for two Web sites because the MDR program terminated abnormally. We report the results without these two sites.

The experimental results on TB1 are given in Table 5.1. "MDR handled" refers to the subset that MDR program can handle, "TPC reported" refers to the subset used by the TPC method reported in [49], and "ALL" refers to the entire set of pages. "Ground" denotes the number of ground truth records. "TP" denotes the number of true positives detected by a method, and "FP" denotes the number of false positives. "P-mi", "R-mi", "F-mi", "P-ma", "R-ma" and "F-ma" are the micro-averaged and macro-averaged precision, recall, and F-measure values respectively. "NA" means that the corresponding result was not reported in [49]. In P-ma calculation, if both TP and FP are 0 for a particular page, its precision is set to be 0.

Our frameworks outperform MDR, DEPTA+, FiVaTech, and TPC. The recall of our frameworks is significantly better than that of MDR. Compared with DEPTA+, our frameworks achieves 14% to 20% improvement in both precision and recall. Compared with FiVaTech, the improvements achieved are 7% to 12%. In addition, the precision and recall of our frameworks outperform TPC about 8% and 4% respectively. For MDR, the macro-precision is much smaller than the micro-precision. It is because for some pages, although the MDR program terminated normally, it could not give any output. Thus,

Table 5.1: Experimental results on TB1.

		Ground	TP	FP	P-mi	R-mi	F-mi	P-ma	R-ma	F-ma
MDR handled	MDR	4261	2692	230	0.920	0.640	0.754	0.620	0.636	0.622
	Skoga	4261	4226	27	0.994	0.992	0.993	0.991	0.979	0.983
TPC reported	TPC	3897	NA	NA	NA	NA	NA	0.904	0.931	NA
	Skoga	3897	3873	27	0.993	0.994	0.993	0.989	0.985	0.983
ALL	DEPTA+	4326	3550	704	0.835	0.823	0.829	0.788	0.807	0.802
	FiVaTech	4326	3782	340	0.918	0.874	0.895	0.883	0.889	0.871
	RST	4326	4239	105	0.976	0.980	0.978	0.977	0.967	0.970
	Skoga	4326	4289	49	0.989	0.991	0.990	0.989	0.976	0.975

both true positive and false positive are 0.

The details of the extracted records on TB1 are given in Figure 5.1. Skoga produces some false positives for the sites 14 and 21. After checking the pages manually, we found that each page in the site 14 contains several record regions presenting the search results from different sources. The given ground truth by the collectors of TB1 only contains two regions, and regards the others as non-record regions. Some pages in the site 21 contain several recommended books on the right side bar. These books were not regarded as data records in the given ground truth since they are more likely to be advertisement items. Only the books formatted with <table> in the center of the page were included in the ground truth. For this site, MDR only outputs the books annotated. Skoga missed a few results in the sites 14, 18, 23, and 25. The main reason is that some regions adopt a different formatting manner in the first few data records compared with the remaining records. For example, the heading information is included in the first data record. Consequently, this record is identified as `REGNOT`. For the site 36, we find that each field of a record is packed in a single subtree, such as id, title, URL, each of digest sentences, etc. Since different records have different number of subtrees, both DEPTA+ and FiVaTech output some false positives. DEPTA+ wrongly outputs many records in some sites such as 19, 23, etc. It is mainly because DEPTA+ has limitations in region detection and record identification brought in by the constraints on the length of generalized node.

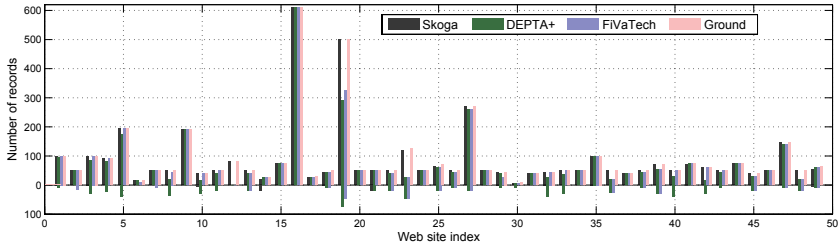


Figure 5.1: Extracted records for each site in TB1. TP number and FP number are shown by the bars above and below the axis respectively.

5.4 Experimental Results on TB2

On data set TB2, we conducted comparison with MDR, DEPTA+, and FiVaTech. MDR could not produce output for 13 pages because the MDR program terminated abnormally. The experimental results on TB2 are given in Table 5.2. The headers of the rows and columns have the same meaning as those in Table 5.1. The recall of our frameworks is significantly better than that of MDR. The reason is that for quite a few pages the MDR program could not give any output, although it terminated normally. Compared with DEPTA+, our frameworks achieve 10% to 21% improvements in both precision and recall, and about 18% improvements in both micro and macro F-measure values. Compared with FiVaTech, improvements achieved by Skoga are about 6% to 9%. Skoga outperforms the RST framework by 3% and 2.5% in micro and macro F-measure values respectively.

When tackling nested record regions, DEPTA+ first identifies the record region by regarding each subregion, such as <tr>'s in Figure 1.1(d), as a generalized node. In the record identifying step of DEPTA+, it attempts to identify the data records by finding lower level generalized nodes from each generalized node in the detected region. The performance of DEPTA+ in this record identifying step is affected by three difficulties. First, if the similarity among the records in the subregion is lower than the required similarity threshold, the entire subregion is regarded as one data record. The second difficulty is caused by the lack of a global analysis. Precisely, the record identification from one subregion does not consider the identification results from other subregions. Therefore, the identification results from different subregions can be significantly different.

Table 5.2: Experimental results on TB2.

		Ground	TP	FP	P-mi	R-mi	F-mi	P-ma	R-ma	F-ma
MDR handled	MDR	5364	3451	832	0.806	0.643	0.715	0.629	0.637	0.634
	Skoga	5364	4893	441	0.917	0.912	0.915	0.902	0.896	0.897
ALL	DEPTA+	5713	4503	1854	0.708	0.788	0.746	0.697	0.767	0.731
	FiVaTech	5713	4712	804	0.854	0.825	0.840	0.843	0.832	0.834
	RST	5713	5006	607	0.892	0.876	0.884	0.881	0.861	0.868
	Skoga	5713	5325	472	0.919	0.932	0.925	0.906	0.914	0.908

The third difficulty for DEPTA+ is that, if the neighboring records in one subregion are separated by separator subtrees, DEPTA+ cannot identify the records accurately. Regarding FiVaTech, after checking the pages that cannot be well tackled, we found that FiVaTech suffers from the difficulty brought in by the optional tags in data record templates. FiVaTech does not perform tree matching across multiple layers in peer node recognition so that it may not be able to induce effective wrappers when the templates have more variants.

In general, Skoga can better handle these difficulties with the global analysis that evaluates the entire nested region as a whole. However, the heterogeneity of Web data records also causes some failure cases for Skoga. In some pages from the university sites, the professors are grouped according to their titles and formatted with different formats in the same region. This causes some difficulty for Skoga which adopts a global analysis to favor the regions whose records follow similar formats. In contrast, DEPTA+ and RST can handle such cases better since they adopt a greedy search manner in region detection from one subtree sequence and can report several regions each of which has its own record format.

5.5 Experimental Results on TB3

On data set TB3, we conducted comparison with MDR and DEPTA+. FiVaTech was not used for comparison since it is not designed to tackle intertwined record regions. MDR could not produce output for 5 pages because the MDR program terminated abnormally. The experimental results are given in Table 5.3. The headers of the rows and columns have the same meaning as those in Table 5.1. Compared with DEPTA+, Skoga achieves

Table 5.3: Experimental results on TB3.

		Ground	TP	FP	P-mi	R-mi	F-mi	P-ma	R-ma	F-ma
MDR handled	MDR	2049	1468	179	0.891	0.716	0.794	0.715	0.694	0.702
	Skoga	2049	1993	42	0.979	0.972	0.976	0.973	0.969	0.970
ALL	DEPTA+	2158	1853	457	0.802	0.859	0.827	0.796	0.833	0.811
	RST	2158	1979	143	0.933	0.917	0.925	0.929	0.917	0.919
	Skoga	2158	2099	54	0.975	0.973	0.974	0.981	0.969	0.972

Table 5.4: Experimental results on TB4.

	Ground	TP	FP	P-mi	R-mi	F-mi	P-ma	R-ma	F-ma
MDR	2318	1809	266	0.872	0.780	0.824	0.857	0.768	0.816
DEPTA+	2318	1979	287	0.873	0.854	0.863	0.858	0.837	0.850
FiVaTech	2318	2056	335	0.860	0.887	0.873	0.859	0.842	0.853
Skoga	2318	2185	96	0.958	0.943	0.950	0.947	0.936	0.938

7% to 18% improvements in both precision and recall, and more than 10% improvements in both micro and macro F-measure values.

In the detection of intertwined data records, DEPTA+ first detects each intertwined field as one pseudo data record region. In reality, each record in the pseudo-region is a record field. Then it assembles the true data records from the adjacent pseudo-regions. However, the record region detection step in DEPTA+ would face a difficulty in detecting the desirable pseudo-regions in certain kinds of intertwined regions. Taking the region shown in Figure 1.2(b) with its DOM given in Figure 1.2(d) as an example, with the top-down searching detection manner DEPTA+ first identifies that the table is one record region and it has the generalized nodes $\mathcal{S}_{1..3}$, $\mathcal{S}_{4..6}$, etc. In the next step of identifying data records, DEPTA+ regards each generalized node, e.g., $\mathcal{S}_{1..3}$, as a data record but not a pseudo-region since the subtrees, i.e., $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$, are dissimilar to one another. Skoga can tackle this type of intertwined regions well. It first identifies the composite records with the labels "`REC-B`" and "`REC-I`", such as $\mathcal{S}_{1..3}$ and $\mathcal{S}_{4..6}$. Then, the assembling algorithm assembles the true data records from each composite record based on the regularity that each single subtree, e.g., $\mathcal{S}_1$, has several repetitive fields and the neighboring subtrees, e.g., $\mathcal{S}_1$ and $\mathcal{S}_2$, have different repetitive fields. Finally, the correct records can be reassembled.

5.6 Experimental Results on TB4

On data set TB4, we conducted comparison with MDR, DEPTA+, and FiVaTech. The experimental results on TB4 are given in Table 5.4. The headers of the rows and columns have the same meaning as those in Table 5.1. In this data set, most of the data records are composed of a single subtree and the data items are embedded in predefined templates. In general, all methods can achieve good performance except MDR which cannot generate output for a few pages. Skoga outperforms DEPTA+ and FiVaTech by around 8% in micro and macro F-measure values. One major type of difficulty in this data set is caused by the quotations. The users usually quote the record of previous users when giving their own comments, reviews or posts. This behavior results in embedding format of some data records so that the dissimilarity among records is enlarged and the detection accuracy is affected. Another type of difficulty is the content length diversity of this type of user-generated content resulting in different number of <p>'s or 's for paragraphs. Skoga and FiVaTech can overcome the difficulty of repeating content by tandem repeat detection and set detection respectively. MDR and DEPTA+ are not able to tackle this difficulty properly.

5.7 Running Time

The running time of Skoga framework for each site in TB1 is reported in Figure 5.2. The reported time for each site is the average time needed for processing its pages. The implementation of the feature extraction part is in Java and the running time on different steps of this part is depicted by T1, T2 and T3. T1 indicates the time for loading a Web page and building its DOM tree structure. T2 indicates the time for calculating feature values. T3 indicates the time for generating feature vectors and saving the testing instance of a page into the disk. The inference part is implemented by extending the structured output SVM framework [73] in C and the inference time is depicted by T4 and T5. T4 indicates the time for loading a testing instance from the disk and rebuilding tree structure. T5 indicates the time for inferring the optimal label assignment for a testing instance. The experiment is run on a PC with Quad core CPU @2.66 GHz and 3GB RAM. In fact, the program has one thread and consume relatively low memory, so it does not require powerful computing resource. From Figure 5.2, it can be observed that Skoga

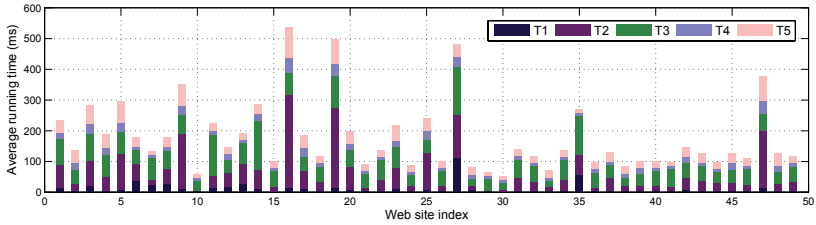


Figure 5.2: The running time of Skoga for each site in TB1. The reported time for each site is the average time needed for processing its pages.

is very efficient and one page takes around 200ms on average.

5.8 Empirical Case Studies

Some empirical case studies are presented to offer a close look at how Skoga framework can effectively tackle some difficult cases. These studies can also provide explanations on the failure of existing methods. The intermediate computational results in Skoga is also presented to show the procedure how the correct labels are assigned for the studied pages. The record region detection and record identification results of DEPTA+ are also illustrated.

5.8.1 Case Study One

In the page given in Figure 3.3(b) with its corresponding DOM tree in 3.3(d), the heading rows of the published year, namely, $\mathcal{S}_1$, $\mathcal{S}_5$, $\mathcal{S}_{20}$, etc., create challenges for existing methods to conduct accurate record extraction. Recall that DEPTA+ requires all the generalized nodes in the same region have the same length and they are all adjacent. Thus it recognizes that the first region is composed of several generalized nodes and each of which contains four table rows, such as $\mathcal{S}_{1..4}$, $\mathcal{S}_{5..8}$ and $\mathcal{S}_{9..12}$. In the record identifying step, DEPTA+ identifies data records from each single generalized node in the regions. The main idea is that if a generalized node contains two or more data records, one more iteration of finding finer generalized nodes in the current generalized node can find the data records. It assumes that the lower level finer generalized nodes, i.e., data records, need to satisfy the condition of covering all the data items in the original generalized node. This assumption

Table 5.5: Portion of intermediate computational results of Skoga for record detection from the Web page in Figure 3.3(b) with the root node <table> labeled with `REGION`. L is the label ID, specifically, `REC-S`=3 and `REGNOT`=6. N is the subtree ID in Figure 3.3(d).

L \ N	1	2	3	4	5	6	7	8	9	10	11
3	^{0}7.679 (5.194)	^{6}36.868 (29.156)	^{3}66.059 (29.153)	^{3}95.250 (29.156)	^{3}100.452 (5.194)	^{6}129.630 (29.156)	^{3}158.833 (29.158)	^{3}188.040 (29.173)	^{3}222.156 (34.082)	^{3}246.417 (24.227)	^{3}270.681 (24.229)
6	^{0}7.702 (5.218)	^{6}36.801 (29.180)	^{6}66.055 (29.172)	^{3}95.219 (29.167)	^{3}100.472 (5.218)	^{6}129.559 (29.178)	^{3}158.820 (29.170)	^{3}188.011 (29.183)	^{3}222.175 (34.141)	^{6}246.388 (24.217)	^{3}270.627 (24.216)

L \ N	12	13	14	15	16	17	18	19	20	21	22
3	^{3}294.947 (24.230)	^{3}319.214 (24.231)	^{3}348.401 (29.151)	^{3}372.673 (24.236)	^{3}400.330 (27.622)	^{3}429.528 (29.163)	^{3}453.779 (24.218)	^{3}489.429 (35.617)	^{6}494.646 (5.194)	^{6}518.949 (24.250)	^{3}543.215 (24.231)
6	^{3}294.896 (24.220)	^{3}319.162 (24.221)	^{3}348.378 (29.170)	^{3}372.621 (24.225)	^{3}400.294 (27.627)	^{3}429.508 (29.184)	^{3}453.735 (24.211)	^{3}489.409 (35.693)	^{3}494.699 (5.218)	^{6}518.934 (24.223)	^{3}543.165 (24.221)

L \ N	23	24	25	26	27	28	29	30	31	32	33
3	^{3}572.408 (29.157)	^{3}596.670 (24.227)	^{3}620.935 (24.229)	^{3}642.504 (21.536)	^{3}671.693 (29.156)	^{3}700.898 (29.171)	^{3}730.095 (29.162)	^{3}754.355 (24.226)	^{3}783.556 (29.166)	^{3}807.815 (24.225)	^{3}832.101 (24.253)

makes DEPTA+ fail to detect the correct records from the first two generalized nodes. Precisely, the existence of $\mathcal{S}_1$ and $\mathcal{S}_5$ hinders the accurate detection of data records in the corresponding generalized nodes, namely, $\mathcal{S}_{1..4}$ and $\mathcal{S}_{5..8}$.

This case also hinders the accurate record detection of other existing methods such as FiVaTech and RST. FiVaTech identifies each group of publications in a particular year as one data record. The wrapper induction manner in FiVaTech favors a wrapper that can achieve a record recognition covering longer repeats. Thus, it induces a wrapper that treats each publication as one repetitive field in a detected "record", i.e. a group of publications in one year. Note that the output given by FiVaTech can also be regarded to be correct from a more general perspective on record definition. RST outputs the data records such as $\mathcal{S}_{1..2}$, $\mathcal{S}_3$, $\mathcal{S}_4$, $\mathcal{S}_{5..6}$, $\mathcal{S}_7$, etc. We can see that the heading rows of the published year are fused into the neighboring records and inaccurate data records are reported such as $\mathcal{S}_{1..2}$ and $\mathcal{S}_{5..6}$.

Skoga framework can consider the entire sequence of subtrees in Figure 3.3(d) to conduct more accurate record detection with a global analysis. It identifies all data records correctly and also labels all heading rows of the published year with `REGNOT`. A portion of intermediate computational results of Skoga for record detection is given in Table 5.5. L is

Table 5.6: Portion of intermediate computational results of Skoga for record detection from the Web page in Figure 3.3(a) with the root node labeled with `REGION`. L is the label ID, specifically, `RECORD`=3 and `REGNOT`=6. N is the subtree ID in Figure 3.3(c).

L \ N	1	2	3	4	5	6	7	8	9
3	^{0}3.845 (3.384)	^{6}15.876 (11.507)	^{6}27.670 (11.463)	^{6}39.513 (11.485)	^{6}41.248 (1.479)	^{6}61.514 (19.592)	^{3}79.265 (17.739)	^{3}95.494 (16.214)	^{3}110.904 (15.396)
6	^{0}4.108 (3.389)	^{6}15.940 (11.490)	^{6}27.755 (11.479)	^{6}39.570 (11.486)	^{6}41.675 (1.565)	^{6}61.515 (19.480)	^{6}79.156 (17.638)	^{3}95.333 (16.070)	^{3}110.718 (15.226)

L \ N	10	11	12	13	14	15	16	17	18
3	^{3}126.318 (15.400)	^{3}142.520 (16.189)	^{3}158.721 (16.187)	^{3}174.905 (16.171)	^{3}191.099 (16.181)	^{3}208.839 (17.727)	^{3}226.571 (17.718)	^{3}242.785 (16.201)	^{3}258.155 (15.356)

the label ID and N is the subtree ID in Figure 3.3(d). The optimal label of the root node <table> is `REGION`. Following the notations in Section 4.3, we have $y_{p,*} =$ `REGION`. In each cell of the table, the superscript is the backtracking pointer, i.e. $y_{c_t,*}$, and the superscript 0 indicates that the backtracking pointer is ineffective. The value after the superscript is the intermediate maximum objective value, i.e. $\hat{F}^{p,*}_{c_t,i}$. The value in the brackets is the optimal value for the corresponding subtree with the root labeled with the corresponding label, i.e. $\hat{F}^{c_t,i}$. Take the cell at $N = 3$ and $L = 3$ as an example, we have $y_{c_2,*} = 3$, $\hat{F}^{p,*}_{c_3,3} = 66.059$, and $\hat{F}^{c_3,3} = 29.153$. From the intermediate maximum objective values in the last column in the third section of Table 5.5, we can obtain that $y_{c_{33},*} = 3$. Thus, the optimal label sequence of the subtrees can be obtained by backtracking the pointers starting from $y_{c_{33},*}$. Finally, the subtrees $\mathcal{S}_1$, $\mathcal{S}_5$ and $\mathcal{S}_{20}$ corresponding to the published year rows are correctly labeled with `REGNOT`. The publication records are correctly labeled with `REC-S`.

5.8.2 Case Study Two

Figure 3.3(a) depicts another challenging record region and its DOM tree is given in Figure 3.3(c). We can see that there are five nodes in the beginning of this region, namely $\mathcal{S}_1 \cdots \mathcal{S}_5$, which should not be identified as data records. The first data record is $\mathcal{S}_6$ and each subtree afterward is one data record. The difficulty of record detection in this region is how to correctly avoid the subtrees $\mathcal{S}_2$, $\mathcal{S}_3$, and $\mathcal{S}_4$ being identified as data records, which share high similarity.

DEPTA+ detects record regions from a subtree sequence with a greedy search manner. Therefore, it first identifies the sequence $\mathcal{S}_2 \cdots \mathcal{S}_4$ as the first record region and then continues to identify the sequence $\mathcal{S}_6 \cdots \mathcal{S}_{18}$ as the second record region. For identifying the records, DEPTA+ identifies each single subtree in the above regions as one data record. Thus, false positive records are output by DEPTA+ from $\mathcal{S}_2 \cdots \mathcal{S}_4$. The challenges created by this case also degrade the performance of other existing methods such as ViPER [67] and TPC [49]. Their heuristic manner for determining the true data records and excluding the description nodes would hinder the accurate detection and output false positives.

With a brief scan through the page, humans can immediately identify the correct records and regard $\mathcal{S}_1 \cdots \mathcal{S}_5$ as some description information about these records. Our Skoga framework can tackle this issue via a global analysis to obtain the correct records. A portion of the intermediate computational results for record detection are given in Table 5.6. It can been seen that all data records are correctly identified and all of $\mathcal{S}_1 \cdots \mathcal{S}_5$ are successfully excluded from being identified as data records and labeled as `REGNOT`.

Chapter 6

Entity Expansion and Attribute Acquisition with Semi-structured Data Records

6.1 Overview

A Wikipedia category contains a set of existing entities. Taking the "`MGM animated short films`" category as an example, its existing entities include "`Scat Cats`", "`One Droopy Knight`", etc. Each single entity may have an infobox, as exemplified in Figure 1.3, which contains a set of attributes. Each attribute is a name-value pair (n, V), where n is the attribute name and V is the set of values. In Figure 1.3, an example of attribute is ("`Produced by`", {"`William Hanna`", "`Joseph Barbera`"}). Given a Wikipedia category C with several existing entities and infoboxes automatically extracted from C as clues, this framework aims at discovering more entities for the category C as well as extracting the attributes for the newly discovered entities. For discovering new entities, one example is the entity "`What's Buzzin' Buzzard`" found in the semi-structured list shown in Figure 1.4 and the director attribute of this new entity is "`Tex Avery`".

The architecture of this framework is depicted in Figure 6.1. First, the component of semi-structured data record set collection aims at automatically obtaining from the Web a collection of semi-structured data record sets which provide the sources for new entity discovery and attribute extraction. Given a Wikipedia category C, it takes several seed entities $\mathcal{S}$ automatically extracted from C as clues for constructing a synthetic query to

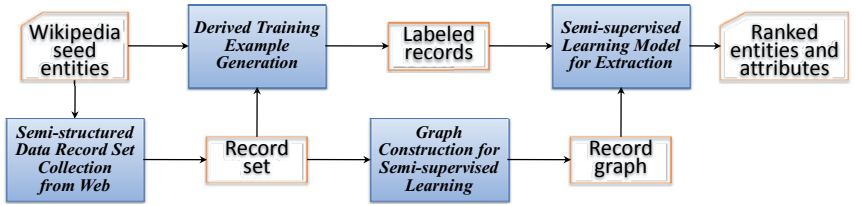


Figure 6.1: Architecture of our framework.

retrieve Web pages. Then semi-structured data record sets that likely contain new entities and attributes are automatically detected from the retrieved pages. One example of such record set is given in Figure 1.4. Let $\mathcal{D}$ denote a record set discovered from a Web page. Some records in $\mathcal{D}$, corresponding to the seed entities in $\mathcal{S}$, are called seed records. The remaining records likely describe some new entities as well as their attribute values. In each record, the key information of the entity described is presented with text segments, such as cartoon name (i.e. entity name of this cartoon), release date, director, etc. The component of semi-supervised learning model for extraction in our framework aims at detecting these desirable text segments. Considering the characteristics of the task, we formulate it as a sequence classification problem. Our classification problem is different from the standard learning paradigm since we do not have manually labeled training examples. Instead, our framework automatically derives the training examples from the seed records. Such labels are known as derived labels and the labeled records are known as derived training examples. For instance, consider the record number 320 corresponding to the seed entity "`One Droopy Knight`" in Figure 1.3, our framework automatically derives labels for the text fragments corresponding to the entity name and attribute values in this record to generate a derived training example.

Let $\mathcal{D}_L$ denote the set of derived training examples in $\mathcal{D}$. Our goal is to predict the labels of text fragments for each remaining record in $\mathcal{D}$. We employ semi-Markov Conditional Random Field (semi-Markov CRF) [65] as the basic sequence classification learning model. Typically, the amount of $\mathcal{D}_L$ is quite limited, so the trained classifier from pure supervised learning would not perform well. A semi-supervised learning model is developed to solve this problem by exploiting the data records in $\mathcal{D}$ that are not in $\mathcal{D}_L$. These data records are called unlabeled data and denoted as $\mathcal{D}_U$. For a better utilization of the unlabeled data, a proximate record graph, with each node of the graph representing

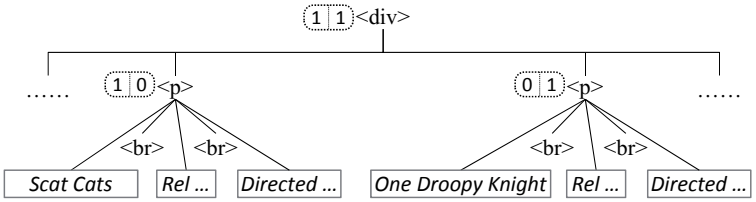


Figure 6.2: The DOM tree of the record set in Figure 1.4 with the flag array added.

one data record in $\mathcal{D}$, is proposed based on the proximate relation of records. During the semi-supervised learning process, the posterior label distribution of the text segments in the unlabeled data records is regularized under the guidance of the proximate graph. Then the labels of records in $\mathcal{D}_U$ are inferred based on the regularized posterior. The data records in $\mathcal{D}_U$ with inferred labels are taken into account in the subsequent iteration of the training process.

Finally, the new entities and their attributes extracted from different data record sets are integrated together. Then a ranking criterion is applied to return a ranked list of the entities together with their attributes.

6.2 Semi-structured Data Record Set Collection

As mentioned in the overview, given a Wikipedia category C, a seed entity set $\mathcal{S}$ is automatically selected from C. These seed entities are used for constructing a synthetic query to retrieve Web pages which are expected to contain semi-structured data record sets as exemplified in Figure 1.4. The synthetic query is composed of the seed entity names and the nouns in the category title. In addition, the query term "list" is added because this term is often used in the page titles and article titles of the desirable Web pages. One sample synthetic query for "MGM animated short films" category is ⟨scat cats, one droopy knight, [mgm, film, list]⟩, where the square brackets denote optional query terms. The synthetic query is then submitted to Google API to retrieve relevant documents, and the pages from Wikipedia are excluded in order to discover entities not in Wikipedia.

In each Web page obtained above, we need to detect the semi-structured data record set describing a group of targeted entities. To do so, we design a method with two steps,

namely, candidate record region identification, and record extraction. To identify the possible record region, we make use of the seed entities as clues. We first build the DOM tree of the page and assign a flag array to each inner node of it. The number of flags in the array is equal to the number of the seed entities in $\mathcal{S}$. Each flag indicates whether the corresponding entity appears in the subtree rooted at the current node. After that, the flag array of each node is set following the post-order traversal. Finally, we identify the DOM node which has all 1's in its flag array, and none of its descendants meets this condition. For example, the DOM tree of the list in Figure 1.4 with the flag array added is shown in Figure 6.2, and is the DOM node that satisfies the condition. The subtree identified above is the candidate region in the page that may contain the desirable record set. Then, we employ the framework presented in Chapter 4 to conduct data record detection from the sub DOM tree with the identified node (e.g.) as the root node.

6.3 Semi-supervised Learning Model for Extraction

In our framework, the extraction of new entities and their attributes from a particular data record set is formulated as a sequence classification problem. Recall that $\mathcal{D}$ is a semi-structured data record set identified from a Web page. Each data record $\mathbf{x}_i$ in $\mathcal{D}$ is composed of a sequence of text tokens and HTML tags. Hence, it can be represented by a token sequence $\mathbf{x}_i = x_i^1 \cdots x_i^{|\mathbf{x}_i|}$. Since some records in $\mathcal{D}$ correspond to the seed entities known as seed records, we attempt to automatically identify the text fragments corresponding to entity names and attribute values based on the infobox information. For example, consider the sample data record 320, i.e., the cartoon "`One Droopy Knight`" in Figure 1.4. By using the infobox of the seed Wikipedia entity as given in Figure 1.3, the text fragment "`ONE DROOPY KNIGHT`" is identified as the entity name and labeled with "ENTITY_NAME" label. The text fragment "`Michael Lah`" is identified as the director attribute value of the cartoon and labeled with "DIRECTED_BY" label. Note that these attribute labels come from the infobox. The label "OTHER" is used to label the tokens that do not belong to the entity names and attribute values.

Unlike traditional labels that are provided by human, our labels are automatically derived from the available data. Therefore, we call them derived labels. After the seed records are automatically labeled with the derived labels, we obtain a set of derived

training examples denoted as $\mathcal{D}_L = \{(\mathbf{x}_i, \mathbf{s}_i)\}$, where $\mathbf{s}_i = s_i^1 \cdots s_i^{|\mathbf{s}_i|}$ and $s_i^q = \langle t_i^q, u_i^q, y_i^q \rangle$ is a token segment with the beginning index t_i^q, the ending index u_i^q, and the derived label y_i^q. Our goal is to predict the labels of the text fragments for each record in the remaining part of $\mathcal{D}$ (i.e., $\mathcal{D} - \mathcal{D}_L$) so as to extract new entity names and their attribute values described by the record. The details of the component for generating the derived training examples will be presented in Section 6.4.

We adopt the semi-Markov CRF [65] as the basic sequence classification learning model. Typically, the amount of $\mathcal{D}_L$ is quite limited since there are just a few seed entities available. If we apply ordinary supervised learning, the performance of the trained classifier is limited. One way to deal with such issue is to exploit the data records in $\mathcal{D}$ that are not in $\mathcal{D}_L$. In particular, we treat them as unlabeled data $\mathcal{D}_U$, where $\mathcal{D}_U = \mathcal{D} - \mathcal{D}_L$. To tackle this problem, we develop a semi-supervised learning model by exploiting the data records in $\mathcal{D}_U$. To better utilize $\mathcal{D}_U$, we design a graph-based component to guide the semi-supervised learning process. Specifically, we propose a graph called proximate record graph where each node of the graph represents a record in $\mathcal{D}$. Each edge represents the connected data records with high degree of similarity in both of text content and HTML format. The high-level pseudo-code of our semi-supervised learning algorithm is given in Algorithm 3. At the beginning, we train the initial parameters of the semi-Markov CRF on $\mathcal{D}_L$ in Line 3. Before performing the semi-supervised learning, the construction of the proximate record graph $\mathcal{G}$ is conducted using the records in $\mathcal{D}$ as shown in Line 4. The details of the construction will be discussed in Section 6.3.1. In Lines 7 and 8, the proximate record graph $\mathcal{G}$ guides the regularization of the posterior label distribution $\mathbf{P}$ of the records. The details are presented in Section 6.3.3. In Lines 9 and 10, the regularized distribution $\mathbf{P}^*$ is interpolated with the original posterior distribution $\mathbf{P}$ to produce an updated label distribution $\tilde{\mathbf{P}}$ which is used in the inference to get the predicted labels for the records in $\mathcal{D}_U$. The details are presented in Section 6.3.4. Then, if the stopping conditions in Line 11 are not met, the algorithm proceeds to the next iteration, and the records in $\mathcal{D}_U$ with the inferred labels in the current iteration are involved in the semi-Markov CRF training in Line 14 as discussed in Section 6.3.5. Finally the labeling results of $\mathcal{D}_U$ in the last iteration are returned as the output.

After all record sets are processed, the extracted entities and attribute values are integrated together. The details are described in Section 6.3.6.

Algorithm 3: The algorithm of our Semi-supervised Learning Model.

1: **input**: record set $\mathcal{D}$ ($\mathcal{D}_L \cup \mathcal{D}_U$)
2: $\mathcal{D}_U^{(0)} \leftarrow \mathcal{D}_U$, $n \leftarrow 0$
3: $\Lambda^{(0)} \leftarrow$ train semi-Markov CRF on $\mathcal{D}_L$
4: $\mathcal{G} \leftarrow$ construct proximate record graph on $\mathcal{D}$
5: $\hat{\mathbf{P}} \leftarrow$ calculate empirical distribution on $\mathcal{D}_L$
6: **while** *true* **do**
7: $\quad \mathbf{P} \leftarrow$ estimate label distribution for each record $\mathbf{x}_i$ in $\mathcal{D}$ with $\Lambda^{(n)}$
8: $\quad \mathbf{P}^* \leftarrow$ regularize $\mathbf{P}$ with $\hat{\mathbf{P}}$ according to graph $\mathcal{G}$
9: $\quad \tilde{\mathbf{P}} \leftarrow$ interpolate distributions of $\mathbf{P}^*$ and $\mathbf{P}$
10: $\quad \mathcal{D}_U^{(n+1)} \leftarrow$ inference label of $\mathcal{D}_U$ with $\tilde{\mathbf{P}}$
11: $\quad$ **if** $\mathcal{D}_U^{(n+1)}$ same as $\mathcal{D}_U^{(n)}$ **or** $n = maxIter$ **then**
12: $\quad\quad$ **goto** Line 17
13: $\quad$ **end if**
14: $\quad \Lambda^{(n+1)} \leftarrow$ train semi-Markov CRF on $\mathcal{D}_L \cup \mathcal{D}_U^{(n+1)}$
15: $\quad n \leftarrow n + 1$
16: **end while**
17: **return** $\mathcal{D}_U^{(n+1)}$

6.3.1 Proximate Record Graph Construction

A proximate record graph $\mathcal{G} = (\mathcal{D}, \mathcal{E})$ is an undirected weighted graph with each record in $\mathcal{D}$ as a vertex and $\mathcal{E}$ is the edge set. Recall that each record $\mathbf{x}_i$ in $\mathcal{D}$ is represented by a token sequence $\mathbf{x}_i = x_i^1 \cdots x_i^{|\mathbf{x}_i|}$. Each edge $e_{ij} = (\mathbf{x}_i, \mathbf{x}_j)$ connecting the vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ is associated with a weight w_{ij} calculated as:

$$w_{ij} = \begin{cases} \mathbb{A}(\mathbf{x}_i, \mathbf{x}_j) & if\ \mathbf{x}_i \in \mathbb{K}(\mathbf{x}_j)\ or\ \mathbf{x}_j \in \mathbb{K}(\mathbf{x}_i) \\ 0 & otherwise \end{cases}, \tag{6.1}$$

where $\mathbb{A}$ is a pairwise sequence alignment function returning a score in $[0, 1]$ which indicates the proximate relation between the two record sequences of $\mathbf{x}_i$ and $\mathbf{x}_j$. $\mathbb{K}(\cdot)$ is a function that can output the top k nearest neighbors. In the proximate record graph, only the vertices (i.e., records) sharing similar format and content will be connected by the edges. Taking the record set given in Figure 1.4 as an example, let $\mathbf{x}_i$ denote the token sequence obtained from the data record number i. There are edges such as $(\mathbf{x}_{319}, \mathbf{x}_{320})$

and $(\mathbf{x}_{319}, \mathbf{x}_{321})$. However, it is unlikely that there is an edge between $\mathbf{x}_2$ and $\mathbf{x}_{319}$.

To design the algorithm for pairwise sequence alignment, we employ a modified Needleman-Wunsch algorithm [52] which performs a global alignment on two sequences using the dynamic programming technique. In particular, we use unit penalty per gap, zero penalty per matching, and infinite penalty per mismatching. It is required that each pair of matching tokens have exactly the same string. For example, consider data records $\mathbf{x}_2$ and $\mathbf{x}_3$ in Figure 1.4, their token sequences are "*<p> DIGIT : cleaning house
 rel DATE* $\cdots$", and "*<p> DIGIT : blue monday
 rel DATE* $\cdots$" respectively. The alignment result is illustrated as below:

$$
\begin{array}{cccccccccc}
< p > & DIGIT & : & cleaning & house & - & - & < br > & rel & \cdots \\
| & | & | & & & & & | & | & \\
< p > & DIGIT & : & - & - & blue & monday & < br > & rel & \cdots
\end{array}
$$

where "-" represents a gap token. After sequence alignment, we obtain a set of aligned token pairs $\mathcal{A}^t_{ij} = \{(x^m_i, x^n_j)\}$, where (x^m_i, x^n_j) indicates that x^m_i from $\mathbf{x}_i$ and x^n_j from $\mathbf{x}_j$ are aligned. Furthermore, we can also define the set of aligned segment pairs of $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$
\begin{aligned}
\mathcal{A}^s_{ij} \quad &= \quad \{(s^q_i, s^r_j)\} \\
&s.t. \quad (a).\neg\exists(x^m_i, x^n_j) \in \mathcal{A}^t_{ij} \wedge x^m_i \in s^q_i \wedge x^n_j \in s^r_j, and \\
&\qquad (b).(x_i^{t^q_i - 1}, x_j^{t^r_j - 1}) \in \mathcal{A}^t_{ij} \wedge (x_i^{u^q_i + 1}, x_j^{u^r_j + 1}) \in \mathcal{A}^t_{ij}, and \\
&\qquad (c).u^q_i - t^q_i \leq maxL \wedge u^r_j - t^r_j \leq maxL.
\end{aligned}
$$

The first condition constrains that an aligned segment pair does not contain any aligned token pairs. The second condition constrains that the left (right) neighboring tokens of the segment pair are aligned. The last condition constrains that the length of both segments should be less than or equal to the maximum length $maxL$. In the above example, one aligned segment pair is ("*cleaning house*", "*blue monday*"). Note that the alignment relation between segments is transitive. Therefore, although $\mathbf{x}_2$ and $\mathbf{x}_{319}$ are not directly connected by an edge, the segments "*cleaning house*" and "*scat cats*" can still be aligned indirectly with the paths from $\mathbf{x}_2$ to $\mathbf{x}_{319}$ in $\mathcal{G}$.

As shown by the above example, it is beneficial to make the labels of aligned segments favor towards each other. The reason is that the text fragments having the same label in different records often share the same or similar context tokens. Meanwhile, they are often presented in the similar relative locations in their own token sequences. The

aligned segments generated from our pairwise sequence alignment algorithm can capture the above two characteristics at the same time. The context tokens of some "OTHER" segments may also be similar to that of the desirable segments. The "OTHER" segments in the unlabeled records are often directly or indirectly aligned to the "OTHER" segments in the labeled records. The label regularization in the semi-supervised learning can help predict the correct labels for such situations.

6.3.2 Semi-Markov CRF and Features

As mentioned in the overview of our framework in Section 6.1, our semi-supervised extraction model employs semi-Markov CRFs [65] as the basic classification learning model. In particular, a linear-chain CRF model is used. Let $\mathbf{s}_i = s_i^1 \cdots s_i^{|\mathbf{s}_i|}$ denote a possible segmentation of $\mathbf{x}_i$, where $s_i^q = \langle t_i^q, u_i^q, y_i^m \rangle$ as defined above. Then the likelihood of $\mathbf{s}_i$ can be expressed as:

$$P(\mathbf{s}_i|\mathbf{x}_i;\Lambda) = \frac{1}{Z(\mathbf{x}_i)} \exp\{\Lambda^T \cdot \sum_q \mathbf{f}(y_i^{q-1}, s_i^q, \mathbf{x}_i)\}, \tag{6.2}$$

where $\mathbf{f}$ is a feature vector of the segment s_i^q and the state of the previous segment. Λ is a weight vector that establishes the relative importance of each feature in $\mathbf{f}$. $Z(\mathbf{x}_i)$ is a normalizing factor.

To avoid the risk of overfitting with only a few derived training examples in our problem setting, the only feature template used in our framework is the separator feature:

$$f_{j,j',t,t',d}(y_i^{q-1}, s_i^q, \mathbf{x}_i) = \mathbf{1}_{\{y_i^q=d\}} \mathbf{1}_{\{x_i^{t_i^q-j}=t\}} \mathbf{1}_{\{x_i^{u_i^q+j'}=t'\}}, \tag{6.3}$$

where $j, j' \in \{1, 2, 3\}$. t and t' vary over the separators such as delimiters and tag tokens in the sequence $\mathbf{x}_i$. d varies over the derived labels of the record set from where $\mathbf{x}_i$ originates. This class of features are unrelated to the previous state, so the feature vector $\mathbf{f}$ can be simplified to $\mathbf{f}(s_i^q, \mathbf{x}_i)$ in our framework.

6.3.3 Posterior Regularization

In Lines 7 and 8 of the semi-supervised learning model shown in Algorithm 3, the posterior label distribution is regularized with the guidance of the proximate graph $\mathcal{G}$. Once the parameters of the CRF model are trained, the posterior probability of a particular segment

s_i^q in the sequence $\mathbf{x}_i$, denoted by $P(s_i^q|\mathbf{x}_i;\Lambda)$, can be calculated as:

$$P(s_i^q|\mathbf{x}_i;\Lambda) = \frac{1}{Z'}\exp\{\Lambda^T \cdot \mathbf{f}(s_i^q, \mathbf{x}_i)\}, \tag{6.4}$$

where $Z' = \Sigma_y \exp\{\Lambda^T \cdot \mathbf{f}(\langle t_i^q, u_i^q, y\rangle, \mathbf{x}_i)\}$. Let the vector $\mathbf{P}_{s_i^q} = (P(\langle t_i^q, u_i^q, y\rangle|\mathbf{x}_i;\Lambda))^T$ denote the posterior label distribution of s_i^q. To regularize this distribution with the proximate graph $\mathcal{G}$, we minimize the following function:

$$O(\mathbf{P}) = O_1(\mathbf{P}) + \mu O_2(\mathbf{P}), \tag{6.5}$$

where μ is a parameter controlling the relative weights of the two terms. $O_1(\mathbf{P})$ and $O_2(\mathbf{P})$ are calculated as in Equations 6.6 and 6.7:

$$O_1(\mathbf{P}) = \sum_{\mathbf{x}_i \in \mathcal{D}_L} \sum_{b=1}^{|\mathbf{x}_i|} \sum_{\substack{s:\langle b, b+l-1, y\rangle \\ s.t. 1 \le l \le maxL \\ b+l-1 \le |\mathbf{x}_i|}} \|\hat{\mathbf{P}}_s - \mathbf{P}_s\|, \tag{6.6}$$

$$\begin{aligned} O_2(\mathbf{P}) \;=\; & \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{E}} w_{ij} \Big(\sum_{(x_i^m, x_j^n) \in \mathcal{A}_{ij}^t} \|\mathbf{P}_{x_i^m} - \mathbf{P}_{x_j^n}\| \\ & + \sum_{(s_i^q, s_j^r) \in \mathcal{A}_{ij}^s} \|\mathbf{P}_{s_i^q} - \mathbf{P}_{s_j^r}\| \Big), \end{aligned} \tag{6.7}$$

where $\|\cdot\|$ is the Euclidean norm. $\hat{\mathbf{P}}_s$ is the empirical label distribution of the segment s in $\mathbf{x}_i$. When $l > 1$, $\hat{P}(\langle b, b+l-1, y\rangle|\mathbf{x}_i)$ is calculated as:

$$\hat{P}(\langle b, b+l-1, y\rangle|\mathbf{x}_i) = \sum_{b \le m \le b+l-1} \hat{P}(\langle m, m, y\rangle|\mathbf{x}_i)/l, \tag{6.8}$$

and $\hat{P}(\langle m, m, y\rangle|\mathbf{x}_i)$ is obtained from the derived label sequence of $\mathbf{x}_i$. In Equation 6.5, the term $O_1(\mathbf{P})$ regularizes the estimated posterior distribution of the derived training examples in $\mathcal{D}_L$ with the original empirical labeling. The term $O_2(\mathbf{P})$ regularizes the estimated posterior distribution of the aligned token and segment pairs.

To achieve efficient computation, we employ the iterative updating method to obtain the suboptimal solution $\mathbf{P}^*$ of Equation 6.5. The updating formulae are:

$$'\mathbf{P}_{s_i^q} = \hat{\mathbf{P}}_{s_i^q} \mathbf{1}_{\{\mathbf{x}_i \in \mathcal{D}_L\}} + \mu \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{E}} w_{ij} \sum_{(s_i^q, s_j^r) \in \mathcal{A}_{ij}^s} \mathbf{P}_{s_j^r}, \tag{6.9}$$

and

$$'\mathbf{P}_{x_i^m} = \hat{\mathbf{P}}_{x_i^m} \mathbf{1}_{\{\mathbf{x}_i \in \mathcal{D}_L\}} + \mu \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{E}} w_{ij} \sum_{(x_i^m, x_j^n) \in \mathcal{A}_{ij}^t} \mathbf{P}_{x_j^n}, \tag{6.10}$$

where $\mathbf{P}$ is the old estimation from the last iteration. Note that after each iteration, $'\mathbf{P}$ should be normalized so as to meet the condition $\|'\mathbf{P}\|_1 = 1$, where $\|\cdot\|_1$ is the ℓ_1 norm. Then $'\mathbf{P}$ will be regarded as the old estimation for the next iteration.

6.3.4 Inference with Regularized Posterior

In Lines 9 and 10 of the algorithm shown in Algorithm 3, the obtained $P^*(s_i^q|\mathbf{x}_i)$ in the regularization is employed to adjust the original $P(s_i^q|\mathbf{x}_i;\Lambda)$ which is calculated with the parameters obtained from the training in the last iteration. The interpolation function is given in Equation 6.11:

$$\tilde{P}(s_i^q|\mathbf{x}_i;\Lambda) = (1-\upsilon)P^*(s_i^q|\mathbf{x}_i) + \upsilon P(s_i^q|\mathbf{x}_i;\Lambda), \tag{6.11}$$

where υ is a parameter controlling the relative weights of P^* and P. Then we calculate the new feature value related to s_i^q on $\mathbf{x}_i$ as $\tilde{P}(s_i^q|\mathbf{x}_i;\Lambda) * Z'$. This value is utilized in the Viterbi decoding algorithm [75] to infer new label sequences for the records in $\mathcal{D}_U$.

Recall that the proximate graph $\mathcal{G}$ captures the alignment relation between similar segments and this relation is transitive. Therefore, the decoded labels with the interpolated posterior distribution favor towards the labels of their aligned neighboring segments.

6.3.5 Semi-supervised Training

As shown in Line 11 of the algorithm in Algorithm 3, if the inferred labels of the records in $\mathcal{D}_U$ from the previous step are the same as the ones obtained in the last iteration, the semi-supervised learning process terminates. Otherwise, if n is still smaller than the maximum iteration number, the records in $\mathcal{D}_U$ with new labels will be taken into consideration in the next iteration of the training process. In this training, we maximize the penalized log likelihood function as shown below:

$$\begin{aligned} \ell(\Lambda^{(n)}) &= \eta \sum_{(\mathbf{x}_i,\mathbf{s}_i)\in\mathcal{D}_L} \log P(\mathbf{s}_i|\mathbf{x}_i;\Lambda^{(n)}) \\ &+ (1-\eta) \sum_{(\mathbf{x}_i,\mathbf{s}_i^*)\in\mathcal{D}_U^{(n)}} \log P(\mathbf{s}_i^*|\mathbf{x}_i;\Lambda^{(n)}) + \gamma\|\Lambda^{(n)}\|, \end{aligned} \tag{6.12}$$

where η is a parameter controlling the relative weights of the contribution from the records in $\mathcal{D}_L$ and $\mathcal{D}_U$. The term $\gamma\|\Lambda^{(n)}\|$ is the Euclidean norm penalty factor weighted by γ. $\mathbf{s}_i^*$ is the segmentation of $\mathbf{x}_i$ inferred from the previous step. Obviously, the above objective function is still concave and it can thus be optimized with the efficient gradient descent methods such as L-BFGS [43].

6.3.6 Result Ranking

In our semi-supervised leaning model for extraction as shown in Algorithm 3, each data record set is processed separately. Thus, the same entity may be extracted from different record sets where it appears with its different variant names. Before ranking the result entities, we first conduct entity deduplication. The number of occurrence of the same entity is counted during the deduplication process. Then the entities are ranked according to their number of occurrence. After entity deduplication, the attributes of the same entity collected from different record sets are integrated together, and they are also deduplicated and ranked similarly. The details of variant collecting and approximate string matching utilized in the deduplication will be presented in Section 6.4 since they are also used in the generation of derived training examples.

6.4 Derived Training Example Generation

As mentioned before, in a semi-structured data record set $\mathcal{D}$, the seed records refer to the data records that correspond to the seed entities in $\mathcal{S}$. The goal of derived training example generation is to automatically identify seed records in $\mathcal{D}$ and determine the sequence classification labels for these records using the information of the seed infoboxes. Since such labels are not directly provided by human as in the standard machine learning paradigm, we call them derived labels. Moreover, the records, after determining the derived labels, are called derived training examples. The generation task can be decomposed into two steps, namely, seed record finding and attribute labeling.

To find the seed record in $\mathcal{D}$ for a seed entity E, we first find the records that contain E's name or its variants as a sub-sequence. The name variants are obtained from synonym in WordNet and the redirection relation in Wikipedia. If the entity name is a person name detected by YagoTool[1], we also collect its variants by following the name conventions, such as middle name removal, given name acronym, etc. In addition, we allow an approximate string matching as supplement in case that the collected variants are not sufficient. The found records are regarded as candidate seed records of E, and the matching sub-sequences are regarded as the candidate record name segments. If multiple record name segments are found in one candidate seed record, the one that has the smallest index in the record

[1] http://www.mpi-inf.mpg.de/yago-naga/javatools/

is retained and others are discarded. We adopt this strategy because the subject of a record, i.e. the record name segment, is usually given in the very beginning of the record [17]. When multiple candidate seed records are found for E, the one whose record name segment has the smallest index in its own token sequence is returned as the seed record. This treatment can handle the case that the entity name of E appears as an attribute value or plain text in other non-seed records. The found record name segment of the seed record is labeled with the derived label "ENTITY_NAME".

The seed records found above compose the derived training set $\mathcal{D}_L$. The procedure of labeling the attribute values in these seed records is similar to the labeling of the entity name. The attribute values for labeling are collected from the seed entities' infoboxes and their variants are obtained in a similar manner as above. In addition, we use YagoTool to normalize the different formats of date and number types. The variant set of each attribute value goes through the same procedure above for entity name except that we only search the value variant in its own seed record. The derived labels of attribute values are obtained from the seed entity's infobox, such as "DIRECTED_BY", "STORY_BY", etc. After the labeling of attribute values, the remaining unlabeled parts in the seed records are labeled with "OTHER".

6.5 Experiments

6.5.1 Experiment Setting

We collect 16 Wikipedia categories as depicted in Table 6.1 to evaluate the performance of our framework. These categories are well-known so that the annotators can collect the full set of the ground truth without ambiguity. Therefore, the experiment results are free from the annotation bias. Moreover, some of these categories are also used in the experimental evaluation of existing works, such as SEAL [79] which will also be compared in our experiment.

To obtain the ground truth of entity elements, the annotators first check the official Web site of a particular category if available. Normally, the full entity list can be found from the Web site. For example, the teams of NBA can be collected from a combox in the home page. For the categories that do not have their own official Web sites such as African country, our annotators try to obtain the entity elements from other related

Table 6.1: The details of the Wikipedia categories collected for the experiments.

Category ID	Category Name	# of entities
1	African countries	55
2	Best Actor Academy Award winners	78
3	Best Actress Academy Award winners	72
4	Best Picture Academy Award winners	83
5	Counties of Scotland	33
6	Fields Medalists	52
7	First Ladies of the United States	44
8	Leone d'Oro winners	57
9	Member states of the European Union	27
10	National Basketball Association teams	30
11	Nobel laureates in Chemistry	160
12	Nobel laureates in Physics	192
13	Presidents of the United States	44
14	Prime Ministers of Japan	66
15	States of the United States	50
16	Wimbledon champions	179

organization Web sites, e.g. the Web site of The World Bank. In addition, the entity lists of some categories are also available in Wikipedia.

Wikipedia already contains articles for some entities in the above categories, and quite many of these articles have well maintained infoboxes. This is helpful for us to collect the ground truth attribute values with high quality for conducting the evaluation of the attribute extraction results. For each entity, the annotators collect the ground truth attribute values for each attribute that also appeared in the seed entities' infoboxes of the same category. Since these infoboxes are used to generate the derived training examples, the extracted attribute values have the same label set as the derived label set from these infoboxes. Hence, the collected ground truth attribute values can be used to evaluate the results. During the collection of attribute values, if the entity exists in Wikipedia and has a good quality infobox, our annotators use the infobox first. After that, they search the

Web to collect the values for the remaining attributes.

For each category, the semi-structured data collection component randomly selects two seed entities from the existing ones in Wikipedia to generate a synthetic query. This query is issued to Google API to download the top 200 hit Web pages. The discovery of entities and the extraction of their attribute are carried out with the semi-structured record sets detected from the downloaded pages. This procedure is executed 3 times per category so as to avoid the possible bias introduced by the seed entity selection. The average of these 3 runs is reported as the performance on this category. It is worthwhile to notice that when three or more seeds are available, we can enhance the performance by generating several synthetic queries with different combinations of the seeds and aggregate the results of these synthetic queries. For example, three synthetic queries each of which involves two seeds can be generated from three seed entities.

In the proximate record graph construction, each vertex is connected to its 3 nearest neighbors. The maximum segment length $maxL$ is set to 6. The iteration numbers are 20 and 10 for regularization updating and semi-supervised learning respectively. Both supervised and semi-supervised models are regularized with a squared ℓ_2-norm regularizer with weight γ set to 0.01. The remaining parameter settings are $\mu = 0.1$, $\upsilon = 0.2$, and $\eta = 0.01$.

6.5.2 Entity Expansion

For entity expansion, we conduct comparison with two methods. The first one is a baseline that employs supervised semi-Markov CRF as the extraction method (called CRF-based baseline). It also makes use of the collected semi-structured record sets by our framework as the information resource and takes the derived training examples in our framework as the training data to perform supervised learning. The second comparison method is an existing work, namely SEAL [79]. SEAL also utilizes the semi-structured data on the Web to perform entity set expansion. It generates a character-level wrapper for each Web page with the seed entities as clues. We collect 200 results from the system[2] Web site of SEAL per seed set per category.

[2]http://www.boowa.com/

Table 6.2: The precision performance of entity discovery of different methods.

	Our framework					CRF-based baseline					SEAL				
#	@5	@20	@50	@100	@200	@5	@20	@50	@100	@200	@5	@20	@50	@100	@200
1	1.00	1.00	0.94	0.48	0.25	1.00	0.90	0.80	0.47	0.25	1.00	1.00	0.92	0.47	0.25
2	1.00	1.00	0.94	0.76	0.38	1.00	0.90	0.84	0.68	0.38	1.00	1.00	0.92	0.71	0.38
3	1.00	0.95	0.92	0.67	0.35	1.00	0.90	0.82	0.58	0.31	1.00	0.85	0.74	0.48	0.29
4	1.00	1.00	0.98	0.81	0.41	1.00	0.93	0.90	0.77	0.41	1.00	1.00	0.98	0.79	0.40
5	0.53	0.43	0.28	0.16	0.11	0.40	0.35	0.28	0.16	0.11	0.40	0.40	0.22	0.12	0.07
6	1.00	1.00	0.94	0.50	0.25	1.00	0.95	0.84	0.43	0.25	1.00	1.00	0.86	0.46	0.24
7	1.00	0.93	0.72	0.42	0.21	0.93	0.85	0.60	0.41	0.21	1.00	0.80	0.62	0.40	0.21
8	0.87	0.85	0.74	0.44	0.24	0.73	0.68	0.48	0.34	0.24	0.80	0.72	0.51	0.26	0.14
9	0.80	0.72	0.46	0.23	0.12	0.60	0.55	0.44	0.23	0.12	0.80	0.55	0.42	0.22	0.12
10	1.00	1.00	0.54	0.28	0.14	1.00	0.93	0.48	0.25	0.14	1.00	1.00	0.52	0.27	0.14
11	1.00	0.95	0.90	0.88	0.66	1.00	0.87	0.84	0.68	0.52	1.00	0.95	0.86	0.71	0.48
12	1.00	0.97	0.86	0.84	0.73	1.00	0.83	0.78	0.62	0.58	1.00	0.95	0.92	0.57	0.52
13	1.00	1.00	0.80	0.41	0.21	1.00	0.93	0.72	0.37	0.19	1.00	0.98	0.76	0.39	0.20
14	1.00	1.00	0.84	0.52	0.32	1.00	0.95	0.76	0.44	0.30	1.00	0.98	0.82	0.42	0.30
15	1.00	1.00	0.95	0.48	0.24	1.00	0.93	0.88	0.48	0.24	1.00	1.00	0.96	0.48	0.24
16	0.80	0.68	0.52	0.33	0.19	0.67	0.62	0.48	0.33	0.18	0.47	0.35	0.34	0.25	0.14
avg.	0.94	0.91	0.77	0.52	0.31	0.90	0.82	0.68	0.46	0.28	0.90	0.85	0.71	0.44	0.26
P-value in pairwise t-test						0.014	9.90E-08	1.25E-05	0.002	0.025	0.074	0.011	0.004	0.001	0.010

Both precision and recall are used to evaluate the ranking results of entity discovery. We first report the result of precision at K (P@K) where K is the number of entities considered from the top of the ranked output list by a method. For each method, the value of K varies over 5, 20, 50, 100, and 200. The results of entity discovery of different methods are given in Table 6.2. It can be observed that all three methods achieve encouraging performance. Their average P@5 values are 0.94, 0.90 and 0.90 respectively. This demonstrates that semi-structured data on the Web is very useful in this task. Therefore, making use of the semi-structured Web data to enrich some categories of Wikipedia is a feasible and practical direction. On average, the performance of our framework with semi-supervised learning extraction is better than SEAL. One reason for the superior performance is that our framework detects the semi-structured data record region before the extraction. This can eliminate the noise blocks in the Web pages. The character-level wrapper of SEAL may be affected by these noise blocks. In addition, with only a few seeds, the character-level wrapper of SEAL cannot overcome the difficulty caused by the cases that the seed entity names have inconsistent contexts. For example, one seed name is embedded in a tag <font> while the other seed name is not. The wrapper induction procedure will be misled by this inconsistency. Moreover, when the text segments of the names have similar format contexts as that of other segments of the records, the obtained wrapper will also extract more false positives. The CRF-based baseline also suffers from the above context related difficulties. Our semi-supervised learning method can cope with this problem by taking the unlabeled data records into account. Specifically, the sequence alignment based proximate record graph regularizes the labels according to the alignment relation so as to overcome the difficulties brought in by the ambiguous context. We conduct pairwise t-test using a significance level of 0.05 and find that the improvements of our framework are significant in most cases. The P-values in the significance tests are given in the last row of Table 6.2.

We manually check some categories with low performance. One major type of error for Category 9 is due to the non-EU European countries, such as “Ukraine” and “Switzerland”. When we retrieve the semi-structured data records from the Web, the seeds of EU member countries also serve as the seeds for non-EU European countries in a counterproductive manner. For the category of Scotland county, many noisy place names are extracted. The main reason is that the entity names in this category are also widely used

Table 6.3: The recall performance of entity discovery of our framework and SEAL.

	Our framework			CRF-based baseline			SEAL		
K / #	50	100	200	50	100	200	50	100	200
1	–	0.91	0.94	–	0.89	0.94	–	0.89	0.94
2	–	0.79	0.87	–	0.78	0.84	–	0.79	0.82
3	–	0.74	0.83	–	0.68	0.83	–	0.62	0.79
4	–	1.00	1.00	–	0.95	1.00	–	0.97	0.99
5	0.45	0.52	0.71	0.44	0.52	0.71	0.35	0.39	0.45
6	–	1.00	1.00	–	0.92	1.00	–	0.92	1.00
7	0.81	1.00	1.00	0.72	0.97	1.00	0.74	0.90	0.95
8	–	0.71	0.84	–	0.62	0.80	–	0.41	0.51
9	0.84	0.92	0.96	0.81	0.92	0.96	0.81	0.88	0.96
10	0.96	1.00	1.00	0.86	0.89	1.00	0.93	0.96	1.00
11	–	–	0.81	–	–	0.66	–	–	0.61
12	–	–	0.74	–	–	0.59	–	–	0.52
13	0.93	0.97	1.00	0.86	0.88	0.90	0.89	0.90	0.93
14	–	0.83	1.00	–	0.75	1.00	–	0.76	1.00
15	0.98	1.00	1.00	0.92	1.00	1.00	1.00	1.00	1.00
16	–	–	0.19	–	–	0.18	–	–	0.16
avg.	0.83	0.88	0.87	0.77	0.85	0.84	0.79	0.80	0.79

to name other places all over the nations of British Commonwealth. Consequently, the collected large amount of noisy semi-structured data records affect the performance.

We also report the recall of our framework and SEAL. Since the number of the ground truth entities in different categories varies from dozens to near 200, the recall values are calculated at different K values of the result list, namely 50, 100, and 200. If the ground truth number is no more than a particular K value, the recall for this K value is calculated. The recall results of our framework and SEAL are given in Table 6.3. On average, our framework outperforms SEAL by about 7%. One reason is that the noise output of SEAL affects the ranking of the true positives, and some of them are ranked lower. Another possible reason is that the search engine cannot return sufficient number

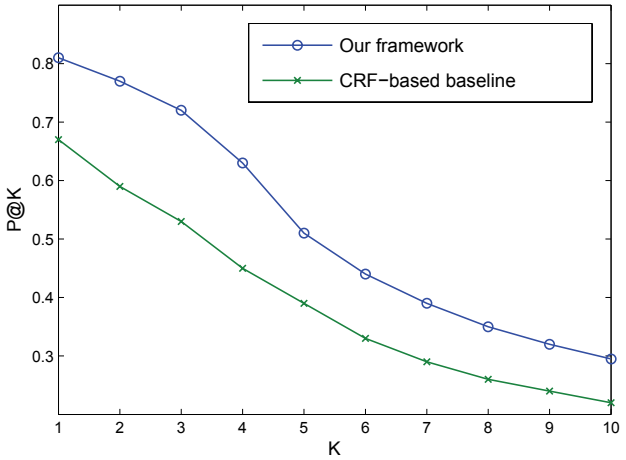


Figure 6.3: The attribute extraction performance of our framework and the CRF-based baseline.

of semi-structured record sets. This also affects the performance of our framework.

6.5.3 Attribute Extraction

Another major goal of our framework is to extract attribute values for the detected entities. SEAL does not perform entity attribute extraction, therefore, we only report the performance of our framework and the CRF-based baseline. For each domain, we only examine the extracted attributes of the correct entities in its answer entity list and report the average precision at different K values from 1 to 10 in all domains. Note that about one fifth correctly extracted entities do not have detected attributes and they are excluded from this evaluation.

The result of attribute extraction performance is given in Figure 6.3. It can been seen that our framework can perform significantly better than the CRF-based baseline with a difference about 11% on average at different K levels. It is worth emphasizing that our framework can achieve 16% improvement on average when K is no more than 5. It indicates that our framework is rather robust. We manually check the extracted attributes of CRF-based baseline and find that besides noise segments, it sometimes wrongly identifies the name of an entity as an attribute value. The reason is that the CRF-based baseline

only depends on the separator features to identity segments. Thus they cannot distinguish those segments with similar context tokens. Our framework can handle these cases by regularizing the label distribution with the proximate record graph as guidance. As a result, noise segments can be regularized to favor towards the segments labeled as "OTHER" in the derived training examples. Similarly, the segments of the entity name and the attribute value with similar context tokens can also be properly identified. In addition, it can be seen that the attributes of higher rank have better precision. Specifically, our framework can output 2 true positive attribute values among the top 3 extracted attribute results. It is because the important attributes of an entity are repeatedly mentioned in different data record sets, such as the capital and GDP of an African country, the party and spouse of a US president, etc. Therefore, they are ranked higher according to the counted occurrence time. We also find that some correct attributes are not ranked very high. One possible reason is due to the simple design of the final ranking method which only counts the number of occurrence of the extracted attributes.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this book, we first present a framework to extract data records in a Web page. The proposed RST structure can be utilized to address several key issues in the record extraction task. The two essential sub-tasks, namely, record region detection, and record segmentation, are handled in a unified manner with the proposed search pruning techniques on the RST structure. Different from the existing similarity-based methods, our method examines the similarity between the dynamically generated subtree groups taking into account the characteristics of the current record region. Owing to the pruning strategies, our method has a comparative complexity compared with the existing methods. Furthermore, we propose a new similarity measure in which each DOM node is regarded as an inseparable unit. Together with the detected tandem repeats, our similarity measure can tackle optional and repetitive fields in records properly.

We also present the second framework for performing robust detection of different kinds of Web data records. This framework can effectively eliminate the limitations of the existing works brought in by their pre-coded hard criteria. It conducts a quantitative analysis of the DOM structure in a global manner in the detection of record regions and data records with a DOM structure knowledge driven model. To allow different impacts for different features in the structure knowledge, there is a weight associated with each feature which is determined using a development data set via a parameter estimation algorithm based on structured output Support Vector Machine model. An optimization method based on divide and conquer principle was developed making use of the DOM

structure knowledge to quantitatively infer the best record and region recognition for a page.

The third presented framework can conduct Wikipedia entity expansion and attribute extraction from the semi-structured data record sets. This framework takes a few seed entities automatically collected from a particular category as well as their infoboxes as clues to harvest more entities as well as attribute content by exploiting the semi-structured data records on the Web. To tackle the problem of lacking sufficient training examples, a semi-supervised learning model is proposed. A proximate record graph is designed, based on pairwise sequence alignment, to guide the semi-supervised learning. Extensive experimental results can demonstrate that our framework can outperform a state-of-the-art existing system. The semi-supervised learning model achieves significant improvement compared with pure supervised learning.

7.2 Future Work

In the research of data record extraction, several directions are worth exploring in the future work. One direction is to enhance our frameworks by incorporating some visual perception from the rendered Web pages. Another direction is to extend the DOM structure knowledge driven framework to tackle the extraction of description details from the pages describing a single object. In the research of entity expansion and attribute extraction, one potential direction is to investigate how to perform derived training example generation with the article of the seed entity when its infobox does not exist. Another direction is to detect new attributes in the semi-structured data record sets which are not mentioned by the existing infoboxes. Such new attributes are valuable to populate the relations in the knowledge base.

For better utilization of the semi-structured data records on the Web, we will further investigate appropriate models for storing, indexing and retrieving this kind of data. The search in the relational database has very strict constraint on the format and structure of the data. On the other hand, Web document search does not have much constraint on the format and structure of the data. Both of these two kinds of search aim at serving the information need of users. However, they work on different kinds of information, and have different underlying methodologies of organizing and searching the data. Both of

them cannot tackle the semi-structured record data effectively due to the gap between their methodologies and the new characteristics of the semi-structured record data. Considering the richness and usefulness of data records on the Web and the lacking of suitable techniques for indexing and retrieving it, the necessity of deep investigation on storing, indexing and retrieving this kind of data becomes obvious. In our future work, we intend to propose a set of models and algorithms to leveraging this kind of data.

Bibliography

[1] Arvind Arasu and Hector Garcia-Molina. Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 337–348, 2003.

[2] Gustavo O. Arocena and Alberto O. Mendelzon. Weboql: restructuring documents, databases, and webs. *Theory and Practice of Object Systems*, 5:127–141, August 1999.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data The Semantic Web. In *Proceedings of 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference (ISWC+ASWC 2007)*, volume 4825, pages 722–735, November 2007.

[4] Lidong Bing, Wai Lam, and Yuan Gu. Towards a unified solution: data record region detection and segmentation. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1265–1274, 2011.

[5] Lidong Bing, Wai Lam, and Tak-Lam Wong. Using query log and social tagging to refine queries based on latent topics. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 583–592, 2011.

[6] Lidong Bing, Wai Lam, and Tak-Lam Wong. Robust detection of semi-structured web records using a dom structure-knowledge-driven model. *ACM Trans. Web*, 7(4):21:1–21:32, November 2013.

[7] Lidong Bing, Wai Lam, and Tak-Lam Wong. Wikipedia entity expansion and attribute extraction from the web using semi-supervised learning. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 567–576, New York, NY, USA, 2013. ACM.

[8] Lidong Bing, Bai Sun, Shan Jiang, Yan Zhang, and Wai Lam. Learning ontology resolution for document representation and its applications in text mining. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1713–1716, 2010.

[9] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[10] David Buttler, Ling Liu, and Calton Pu. A fully automated object extraction system for the world wide web. In *Proceedings of the The 21st International Conference on Distributed Computing Systems*, pages 361–370, Washington, DC, USA, 2001.

[11] Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. Knowitnow: fast, scalable information extraction from the web. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 563–570, 2005.

[12] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, Jr., and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110, 2010.

[13] Chia-Hui Chang, Mohammed Kayed, Moheb Ramzy Girgis, and Khaled F. Shaalan. A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18:1411–1428, October 2006.

[14] Chia-Hui Chang and Shao-Chen Lui. Iepad: information extraction based on pattern discovery. In *Proceedings of the 10th international conference on World Wide Web*, pages 681–688, 2001.

[15] Jim Cowie and Wendy Lehnert. Information extraction. *Communications of the ACM*, 39(1):80–91, January 1996.

[16] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 109–118, 2001.

[17] Eric Crestan and Patrick Pantel. Web-scale table census and classification. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 545–554, 2011.

[18] 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 pages. *Data & Knowledge Engineering*, 31:227–251, November 1999.

[19] D. W. Embley, Y. Jiang, and Y.-K. Ng. Record-boundary discovery in web documents. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 467–478, 1999.

[20] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, December 2008.

[21] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th international conference on World Wide Web*, pages 100–110, 2004.

[22] Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam Mausam. Open information extraction: the second generation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume One*, pages 3–10, 2011.

[23] Wolfgang Gatterbauer, Paul Bohunsky, Marcus Herzog, Bernhard Krüpl, and Bernhard Pollak. Towards domain-independent information extraction from web tables. In *Proceedings of the 16th international conference on World Wide Web*, pages 71–80, 2007.

[24] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. pages 529–236. Cambridge, MA, 2005.

[25] Rahul Gupta and Sunita Sarawagi. Answering table augmentation queries from unstructured lists on the web. *Proceedings of the VLDB Endowment*, 2:289–300, 2009.

[26] Dan Gusfield and Jens Stoye. Linear time algorithms for finding and representing all the tandem repeats in a string. *Journal of Computer and System Sciences*, 69:525–546, December 2004.

[27] Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 775–784, 2011.

[28] Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 286–295, 2010.

[29] Andrew Hogue and David Karger. Thresher: automating the unwrapping of semantic content from the world wide web. In *Proceedings of the 14th international conference on World Wide Web*, pages 86–95, 2005.

[30] Chun-Nan Hsu and Ming-Tzung Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems*, 23:521–538, December 1998.

[31] Jian Hu, Lujun Fang, Yang Cao, Hua-Jun Zeng, Hua Li, Qiang Yang, and Zheng Chen. Enhancing text clustering by leveraging wikipedia semantics. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 179–186, 2008.

[32] Shan Jiang, Lidong Bing, Bai Sun, Yan Zhang, and Wai Lam. Ontology enhancement and concept granularity learning: keeping yourself current and adaptive. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1244–1252, 2011.

[33] Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 209–216, 2006.

[34] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.

[35] Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, pages 387–396, 2006.

[36] Mohammed Kayed and Chia-Hui Chang. Fivatech: Page-level web data extraction from template pages. *IEEE Transactions on Knowledge and Data Engineering*, 22:249–263, February 2010.

[37] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, September 1999.

[38] Nicholas Kushmerick. Wrapper induction: efficiency and expressiveness. *Artificial Intelligence*, 118:15–68, April 2000.

[39] Alberto H. F. Laender, Berthier Ribeiro-Neto, and Altigran S. da Silva. Debye - date extraction by example. *Data & Knowledge Engineering*, 40:121–154, February 2002.

[40] Xiao-Li Li, Lei Zhang, Bing Liu, and See-Kiong Ng. Distributional similarity vs. pu learning for entity set expansion. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 359–364, 2010.

[41] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proceedings of the VLDB Endowment*, 3:1338–1347, 2010.

[42] Bing Liu, Robert Grossman, and Yanhong Zhai. Mining data records in web pages. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–606, 2003.

[43] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[44] Ling Liu, Calton Pu, and Wei Han. Xwrap: An xml-enabled wrapper construction system for web information sources. In *International Conference on Data Engineering*, pages 611–621, 2000.

[45] Wei Liu, Xiaofeng Meng, and Weiyi Meng. Vide: A vision-based approach for deep web data extraction. *IEEE Transactions on Knowledge and Data Engineering*, 22:447–460, March 2010.

[46] Xiaojiang Liu, Zaiqing Nie, Nenghai Yu, and Ji-Rong Wen. Biosnowball: automated population of wikis. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 969–978, 2010.

[47] Christopher D. Manning and Hinrich Schuetze. *Foundations of Statistical Natural Language Processing*. 1 edition, June 1999.

[48] Joe Marini. *Document Object Model*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 2002.

[49] Gengxin Miao, Junichi Tatemura, Wang-Pin Hsiung, Arsany Sawires, and Louise E. Moser. Extracting data records from the web using tag path clustering. In *Proceedings of the 18th international conference on World wide web*, pages 981–990, 2009.

[50] Ion Muslea, Steven Minton, and Craig A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4:93–114, March 2001.

[51] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30:3–26, 2007.

[52] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[53] Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. Cross lingual text classification by mining multilingual topics from wikipedia. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 375–384, 2011.

[54] Marius Paşca. Organizing and searching the world wide web of facts – step two: harnessing the wisdom of the crowds. In *Proceedings of the 16th international conference on World Wide Web*, pages 101–110, 2007.

[55] Marius Paşca. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 683–690, 2007.

[56] Marius Paşca and Benjamin Van Durme. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 19–27, 2008.

[57] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, 1998.

[58] Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, pages 938–947, 2009.

[59] Marco Pennacchiotti and Patrick Pantel. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, pages 238–247, 2009.

[60] Simone Paolo Ponzetto and Roberto Navigli. Large-scale taxonomy mapping for restructuring and integrating wikipedia. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 2083–2088, 2009.

[61] Simone Paolo Ponzetto and Michael Strube. Deriving a large scale taxonomy from wikipedia. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, pages 1440–1445, 2007.

[62] Filip Radlinski and Thorsten Joachims. Query chains: learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248, 2005.

[63] Maria Ruiz-Casado, Enrique Alfonseca, and Pablo Castells. Automatic assignment of wikipedia encyclopedic entries to wordnet synsets. In *Proceedings of the Third international conference on Advances in Web Intelligence*, pages 380–386, 2005.

[64] Matthew S. Ryan and Graham R. Nudd. The viterbi algorithm. Technical report, Coventry, UK, UK, 1993.

[65] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In *Neural Information Processing Systems*, pages 1185–1192, 2004.

[66] Christina Sauper and Regina Barzilay. Automatically generating wikipedia articles: a structure-aware approach. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, pages 208–216, 2009.

[67] Kai Simon and Georg Lausen. Viper: augmenting automatic information extraction with visual perceptions. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 381–388, 2005.

[68] Weifeng Su, Jiying Wang, and Frederick H. Lochovsky. Ode: Ontology-assisted data extraction. *ACM Transactions on Database Systems*, 34:12:1–12:35, July 2009.

[69] Amarnag Subramanya, Slav Petrov, and Fernando Pereira. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 167–176, 2010.

[70] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, 2007.

[71] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A large ontology from wikipedia and wordnet. *Web Semantics*, 6:203–217, 2008.

[72] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. Sofie: a self-organizing framework for information extraction. In *Proceedings of the 18th international conference on World wide web*, pages 631–640, 2009.

[73] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, December 2005.

[74] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. Recovering semantics of tables on the web. *Proceedings of the VLDB Endowment*, 4:528–538, 2011.

[75] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, September 2006.

[76] Jingjing Wang, Bin Shao, Haixun Wang, and Kenny Q. Zhu. Understanding tables on the web. Technical report, 2010.

[77] Jiying Wang and Fred H. Lochovsky. Data extraction and label assignment for web databases. In *Proceedings of the 12th international conference on World Wide Web*, pages 187–196, 2003.

[78] Pu Wang and Carlotta Domeniconi. Building semantic kernels for text classification using wikipedia. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–721, 2008.

[79] Richard C. Wang and William W. Cohen. Language-independent set expansion of named entities using the web. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 342–350, 2007.

[80] Richard C. Wang and William W. Cohen. Iterative set expansion of named entities using the web. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 1091–1096, 2008.

[81] Richard C. Wang and William W. Cohen. Character-level analysis of semi-structured documents for set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, pages 1503–1512, 2009.

[82] Xuanhui Wang and ChengXiang Zhai. Mining term association patterns from search logs for effective query reformulation. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 479–488, 2008.

[83] Yang Wang, Gholamreza Haffari, Shaojun Wang, and Greg Mori. A rate distortion approach for semi-supervised conditional random fields. In *Neural Information Processing Systems*, pages 2008–2016. 2009.

[84] Tak-Lam Wong, Lidong Bing, and Wai Lam. Normalizing web product attributes and discovering domain ontology with minimal effort. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 805–814, 2011.

[85] Tak-Lam Wong, Wai Lam, and Shing-Kit Chan. Collaborative information extraction and mining from multiple web documents. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 440–450, 2006.

[86] Fei Wu and Daniel S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 41–50, 2007.

[87] Fei Wu and Daniel S. Weld. Automatically refining the wikipedia infobox ontology. In *Proceeding of the 17th international conference on World Wide Web*, pages 635–644, 2008.

[88] Fei Wu and Daniel S. Weld. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127, 2010.

[89] Yasuhiro Yamada, Nick Craswell, Tetsuya Nakatoh, and Sachio Hirokawa. Testbed for information extraction from deep web. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 346–347, 2004.

[90] Yanhong Zhai and Bing Liu. Web data extraction based on partial tree alignment. In *Proceedings of the 14th international conference on World Wide Web*, pages 76–85, 2005.

[91] Yanhong Zhai and Bing Liu. Structured data extraction from the web based on partial tree alignment. *IEEE Transactions on Knowledge and Data Engineering*, 18:1614–1628, December 2006.

[92] Yanhong Zhai and Bing Liu. Extracting web data using instance-based learning. *World Wide Web*, 10:113–132, June 2007.

[93] Bin Zhao, Xiaoxin Yin, and Eric P. Xing. Max margin learning on domain-independent web information extraction. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1305–1310, 2011.

[94] Hongkun Zhao, Weiyi Meng, Zonghuan Wu, Vijay Raghavan, and Clement Yu. Fully automatic wrapper generation for search engines. In *Proceedings of the 14th international conference on World Wide Web*, pages 66–75, 2005.

[95] Shuyi Zheng, Ruihua Song, Ji-Rong Wen, and C. Lee Giles. Efficient record-level wrapper induction. In *Proceeding of the 18th ACM conference on Information and knowledge management*, pages 47–56, 2009.

[96] Shuyi Zheng, Ruihua Song, Ji-Rong Wen, and Di Wu. Joint optimization of wrapper generation and template detection. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 894–902, 2007.