

Extracting Data From Web Sources

Giansalvatore Mecca

Università della Basilicata

mecca@unibas.it

<http://www.difa.unibas.it/users/gmecca/>

EDBT Summer School 2002

August, 26 2002

Outline

○ Subject of this talk

⇒ *Information extraction from the Web*

○ Scenario

⇒ the Web is probably the largest “knowledge base” ever developed

⇒ information conceived to be consumed (“browsed”) by humans

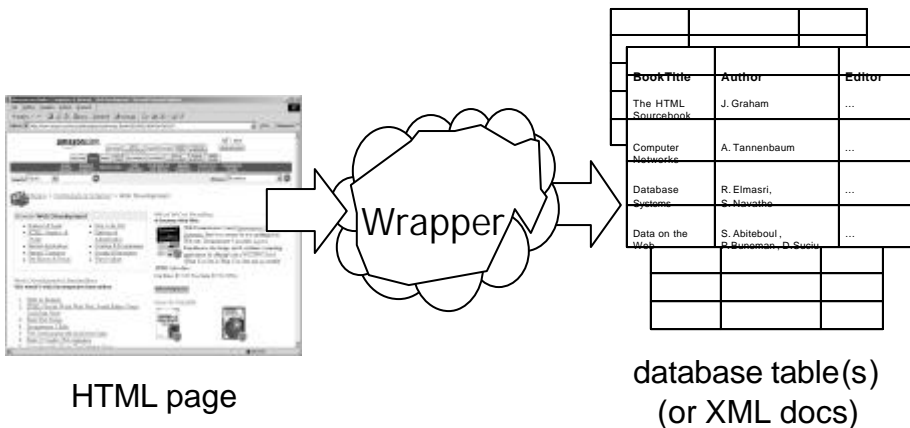
⇒ the goal is to make this information available to computer applications (“agents”)

Outline

- A general description of a software agent's tasks
 1. receive a goal
 2. locate relevant sources on the Web
 3. gather information from these sources
 4. process this content
 5. deliver results to the user
- In this talk
 - ⇒ we concentrate exclusively on task 3
- Task carried on by “wrappers”

3

Outline Notion of Wrapper



4

Outline

○ Part I: Information Extraction from HTML

⇒ wrapper inference techniques

⇒ limitations

○ Part II: Grammar Inference

⇒ Gold's theorem and identification in the limit

⇒ algorithms from grammar inference

⇒ limitations

○ Part III: Bridging the Gap

⇒ the RoadRunner Project

5

Outline

Part I: Information Extraction from HTML

○ What's in a wrapper

○ Recent research on semi-automatic wrapper generation techniques

⇒ machine learning approaches

⇒ wrapper induction

○ Limitations of these approaches

⇒ related to supervised learning

⇒ wrappers tend to be brittle

⇒ the maintenance problem

6

Outline

Part II: Grammar Inference

- Wrappers are essentially grammar parsers
 - ⇒ wrapper inference is strongly related to grammar inference
- A jump backward: 30 years of grammar inference
 - ⇒ accomplishments of the grammar inference community
 - ⇒ why grammar inference techniques are not applicable to information extraction

7

Outline

Part III: Bridging the Gap

- The Schema Finding Problem
 - ⇒ a variant of the traditional grammar inference problem for information extraction
- Markup Grammars
 - ⇒ a class of grammars that can be practically inferred with unsupervised algorithms
- Limitations and opportunities
 - ⇒ future research directions

8

Part I

Information Extraction from HTML

9

Information Extraction Task

○ Information extraction task

- ⇒ source format: plain text with HTML markup (no semantics)
- ⇒ target format: database table or XML file (adding structure, i.e., semantics)
- ⇒ extraction step: parse the HTML and return data items in the target format

○ Wrapper

- ⇒ piece of software for the extraction step
- ⇒ intuition: use extraction rules based on HTML markup

10

Intuition Behind Extraction



<i>teamName</i>	<i>town</i>
Atalanta	Bergamo
Inter	Milano
Juventus	Torino
Milan	Milano
...	...

```

<html><body>
<h1>Italian Football Teams</h1>
<ul>
<li><b>Atalanta</b> - <i>Bergamo</i><br>
<li><b>Inter</b> - <i>Milano</i><br>
<li><b>Juventus</b> - <i>Torino</i><br>
<li><b>Milan</b> - <i>Milano</i><br>
</ul>
</body></html>
    
```

Wrapper Procedure:
 Scan document for
 While there are more occurrences
 scan until
 extract teamName between
 and
 scan until <i>
 extract town between <i> and </i>
 output [teamName, town]

A Complex Extraction Task



20-30KB of HTML

>10 attributes
with nesting

image	brand	model	color	price
	Asics	GEL-1070	White/Midlevel/Black	\$89.95
	Asics	Men's Gel-100 10™	White/White/Neon Navy	\$59.95
	Asics	GEL-MC RUSH 2	White/White/Pink	\$99.95
	Asics	GEL-1070	Light Silver/Black/Pink	\$74.95
	Asics	GEL-1070	White/Light Silver/White/Gold	\$74.95
	Asics	Men's GEL-Foundation II	White/Black/Black	\$79.95

Why Information Extraction is Relevant ?

- Wrappers are largely used today
 - ⇒ shopping agents (es: Jango, BargainFinder, ShopBot ...)
 - ⇒ integration agents (es: molecular biology)
 - ⇒ personal information brokers
- In these applications, wrappers are usually written by hand
 - ⇒ es: Excite's shopping agent, Jango, relies on several hundred wrappers [Kushmerick, 2000]

13

Wrapper Development

- There are several tools that can be used to support the wrapper development process
 - ⇒ both research and commercial tools
 - ⇒ for a list see:
www.wifo.uni-mannheim.de/~kuhlins/wrappertools/
- Using these tools it is usually possible to rapid prototype a wrapper very quickly
 - ⇒ example: Lixto [Baumgartner et al, VLDB 2001], a GUI-based wrapper generation toolkit
- Problem: wrappers tend to be brittle

14

Problem: Brittleness

- Web sites change quite frequently
- Any change in the HTML layout may disrupt the extraction rules
 - ⇒ es: Jango's wrapper mean time to failure was approximately one month [Kushmerick, 2000]
- Maintaining a wrapper is very costly
 - ⇒ this has motivated the study of (semi)-automatic wrapper generation techniques

15

A Note on the Semantic Web

- It is a common opinion that neither XML nor the Semantic Web initiative will represent for now a solution to the information extraction problem
- Reason I
 - ⇒ the Web is a giant "legacy system"
 - ⇒ many sites will never move to the new technology
- Reason II
 - ⇒ many organizations will not be willing to expose their data in XML

16

Semi-Automatic Wrapper Generation

- Many proposals, from different inspirations
- A rough classification
 - ⇒ grammar-based (“finite-state”) machine learning approaches
 - ⇒ relational machine learning approaches
 - ⇒ ontology based approaches [Embley et al, 98]
- We discuss only finite-state approaches
 - ⇒ see the references for a complete list
 - ⇒ we also don’t cover information extraction from free text (a related but quite different problem)

17

Finite-State Approaches

- Typical learning system
 - ⇒ a supervisor provides a set of training samples (i.e., labeled HTML pages)
 - ⇒ a learning algorithm is used to learn the extraction rules
 - ⇒ a wrapper is generated based on the extraction rules
 - ⇒ the wrapper is used on the target pages
- This process is called “wrapper induction”

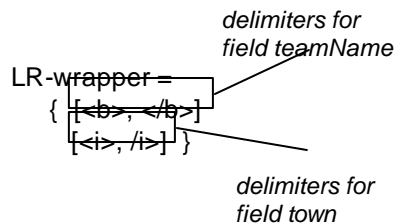
18

Wrapper Induction

- Seminal work by Kushmerick et al
 - ⇒ [Kushmerick et al, 1997] [Kushmerick, 2000]
- Six wrapper classes
 - ⇒ increasing expressibility
 - ⇒ decreasing efficiency in the learning
- The simplest class: LR (left-right wrappers)
 - ⇒ targeted at pages containing multiple records
 - ⇒ one extraction rule per field
 - ⇒ each extraction rule is a pair of delimiting strings (left and right)

19

LR-wrappers: Example



```
<html><body>  
<h1>Italian Football Teams</h1>  
<ul>  
<li><b>Atalanta</b> - <i>Bergamo</i> <br>  
<li><b>Inter</b> - <i>Milano</i> <br>  
<li><b>Juventus</b> - <i>Torino</i> <br>  
<li><b>Milan</b> - <i>Milano</i> <br>  
</ul>  
</body></html>
```

Wrapper Procedure:

While there are more occurrences
scan until
extract teamName between
and
scan until <i>
extract town between <i> and </i>
output [teamName, town]

20

Wrapper Induction

○ Note the connection with grammars

- ⇒ the extraction rules define a regular grammar with labeled nonterminals
- ⇒ wrapper: parser for the grammar + knowledge about the target structure (set of records)

LR-wrapper =
{ [****, ****]
[*<i>*, *</i>*] }

Σ^*

(**** *teamName* **** Σ^* *<i>* *town* *</i>*)^{*} Σ^* : any string

Σ^*

output: records of the form (*teamName*, *town*)

21

Wrapper Induction

- In order to induce the wrapper, the first step is to label the training data
- A human inspects some sample pages and manually labels the fields

```
<html><body>
<h1>Italian Football Teams</h1>
<ul>
<li><b>teamName</b> - <i>town</i> <br>
<li><b>teamName</b> - <i>town</i> <br>
<li><b>teamName</b> - <i>town</i> <br>
<li><b>teamName</b> - <i>town</i> <br>
</ul>
</body></html>
```

22

Wrapper Induction

- Then a learning algorithm is run on the sample pages to find the delimiters
- The algorithm is quite simple
 - ⇒ it enumerates over potential values for each delimiter selecting the first such that the wrapper works correctly on the training data
 - ⇒ the learning is efficient because delimiters can be learned independently
 - ⇒ quadratic time

23

Wrapper Induction

- Obviously, not all pages can be wrapped using LR-wrappers
 - ⇒ in [Kushmerick, 2000] it is reported that LR-wrappers were able to handle 53% of the sites in a survey conducted by the author

```
<html><body>
<b>Italian Football Teams</b>
<ul>
<li><b>Atalanta</b> - <i>Bergamo</i> <br>
<li><b>Inter</b> - <i>Milano</i> <br>
<li><b>Juventus</b> - <i>Torino</i> <br>
<li><b>Milan</b> - <i>Milano</i> <br>
</ul>
</body></html>
```

24

Wrapper Induction

- To handle these cases, Kushmerick introduces more complex classes of wrappers
- Example: HLRT (head-left-right-tail wrappers)
 - ⇒ two additional delimiters, h and t
 - ⇒ h is used to skip potentially confusing text in the head page
 - ⇒ t is used to skip potentially confusing text in the tail of the page

25

HLRT-wrappers: Example

```
<html><body>
<b>Italian Football Teams</b>
<ul>
<li><b>Atalanta</b> - <i>Bergamo</i> <br>
<li><b>Inter</b> - <i>Milano</i> <br>
<li><b>Juventus</b> - <i>Torino</i> <br>
<li><b>Milan</b> - <i>Milano</i> <br>
</ul>
</body></html>
```

```
 $\Sigma^* \langle ul \rangle \Sigma^*$ 
 $(\langle b \rangle \text{teamName} \langle /b \rangle \Sigma^* \langle i \rangle \text{town} \langle /i \rangle)^*$ 
 $\Sigma^* \langle /html \rangle \Sigma^*$ 
```

HLRT-wrapper = *head and tail delimiters*

```
{ [ <ul>, </html> ]
  [ <b>, </b> ]
  [ <i>, /i> ] }
```

Wrapper Procedure:

Scan document for

While (there are more occurrences of before </html>)

scan until

extract teamName between and

scan until <i>

extract town between <i> and </i>

output [teamName, town]

26

Wrapper Induction

○ Six classes of wrappers

- ⇒ LR (left-right)
- ⇒ HLRT (head-left-right-tail)
- ⇒ OCLR (open-close-left-right): open and close delimiters identify each record in the document
- ⇒ HOCLRT (head-close-left-right-tail)
- ⇒ NLR (nested LR): to handle nested tabular data
- ⇒ NHLRT (nested HLRT)

27

Wrapper Induction

○ Tradeoff between expressibility and complexity of the learning

- ⇒ the first classes can be learned in polynomial time wrt to the length of the training documents
- ⇒ in last two classes the learning is unfeasible (exponential in the size of the training documents)

○ Overall

- ⇒ the six classes were able to handle 70% of the sites selected by the author in his survey

28

Extended Wrapper Induction Techniques

- Subsequent works have extended Kushmerick's work in several respects
 - ⇒ missing attributes
 - ⇒ multiple orderings
 - ⇒ disjunctive delimiters
 - ⇒ arbitrary nesting
- Stalker [Muslea et al, 1999]
- SoftMealy [Hsu, Dung, 98]
- NoDoSE [Adelberg, 1998]
- Can wrap virtually any form of HTML layout

29

Limitations of Supervised Techniques

- If the site changes, it is necessary to maintain the wrapper
- Maintenance Problem I: verification
 - ⇒ it is necessary to detect when the wrapper stops to work properly
 - ⇒ delimiter-based wrappers do not always allow to detect changes in the site
- Maintenance Problem II: re-induction
 - ⇒ in general, a new user intervention is necessary in order to learn the new wrapper

30

Wrapper Verification Example



```
<html><body>
<h3>Italian Football Teams</h3>
<table>
<tr><td>Atalanta</td> <td>Bergamo</td>
<tr><td>Inter</td> <td>Milano</td>
<tr><td>Juventus</td> <td>Torino</td>
<tr><td>Milan</td> <td>Milano</td>
</table>
<b>Last modified</b>: <i>Aug. 2002</i>
</body></html>
```

LR-wrapper =
{ [****, ****]
[*<i>*, *</i>*] }

teamName	town
Last modified	Aug. 2002

the reason for this is that the grammar specified by the delimiters is not very strict, and allows a wide degree of variations

31

Wrapper Maintenance

- There has been some work on wrapper maintenance
 - ⇒ [Kushmerick, WWWJ 2000]: regression testing to detect wrapper disruption
 - ⇒ [Lerman, Minton, 2000]: data extracted when the wrapper was working properly can help to automatically relabel the new samples
- However
 - ⇒ these results are quite preliminary

32

Wrapper Maintenance

Ideal Goals

- Ideally, we would like to achieve two goals
- First, the wrapper should give a tighter description of the grammatical structure of the page
 - ⇒ this would simplify wrapper verification
- Second, the wrapper should be learnable in an unsupervised way (fully automatically, without the need for training examples)
 - ⇒ this would simplify wrapper reinduction

33

Wrapper Maintenance

- In this way, wrapper induction becomes a typical grammar inference problem
 - ⇒ given a set of samples, automatically infer the **minimal** grammar to which they belong
- Grammar inference is a well established research field, with a rich literature
 - ⇒ nice theoretical framework
 - ⇒ unsupervised algorithms
- Why not grammar inference for information extraction ?

34



Part II Grammar Inference

35



Grammar Inference

- Grammar Inference is a 30-years old subject (early studies in the sixties)
 - ⇒ focus on computational theory, and not on information extraction
- Preliminaries:
 - ⇒ let's fix a finite alphabet of symbols Σ
 - ⇒ a language over Σ is any collection of strings over Σ
 - ⇒ a class of languages over Σ is any collection of languages over Σ

36

Grammar Inference

○ The problem

⇒ given a collection of sample strings, find the language they belong to

○ The computational model: identification in the limit [Gold, 1967]

⇒ the inference system is presented with a larger and larger corpus of examples

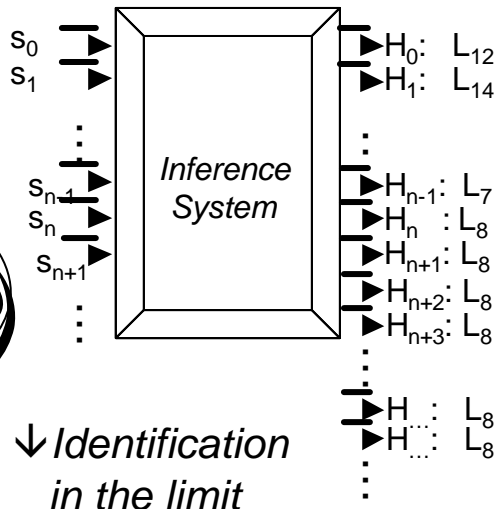
⇒ it simultaneously makes a sequence of guesses about the language to infer

⇒ if the sequence eventually converges, the inference is correct

37

Identification in the Limit

Class of Languages



38

Gold's Theorem

- *Any class of languages over S that contains all finite languages over S and at least one infinite language cannot be identified in the limit from positive samples only [Gold, 1967]*
- This means, for example, that regular grammars cannot be identified in the limit from positive samples only

39

Gold's Theorem: Intuition

- Consider a set of samples of the form $a, ab, abb, abbb, abbbb, \dots$
 - ⇒ the learner will never be able to definitely tell whether the target language is $a \cup ab \cup abb \cup abbb \cup \dots$ or ab^*
- Hint: the problem is related to disjunction
 - ⇒ but r.e. without \cup are not identifiable either
- Research directions
 - ⇒ sub-classes that are identifiable in the limit
 - ⇒ different computational model (es: positive and negative examples or additional information)

40

Restricted Classes of Regular Expressions

- Dana Angluin has characterized the classes of languages identifiable in the limit
- A class of languages L is identifiable in the limit iff each language of the class admits a characteristic sample [Angluin, 1980]
- Characteristic sample
 - ⇒ A finite set $T \subset L \in L$ is said a *characteristic sample* of L in L if L is the smallest language from L containing T

41

Restricted Classes of Regular Expressions

- Several subclasses of regular expressions have been proven to identifiable in the limit
 - ⇒ reversible languages [Angluin, 1982]
 - ⇒ terminal-distinguishable languages [Radakrishnan and Nagaraja, 1987]
 - ⇒ testable languages [Garcia and Vidal, 1990]
 - ⇒ ...
- The definition of these classes is quite involved

42

f-Distinguishable Languages

- Fernau has recently generalized many previously known classes of identifiable languages under the notion of f-distinguishable languages
- Class of f-Distinguishable Languages
 - ⇒ class of regular languages identified by a distinguishing function [Fernau, 2000]
- Distinguishing function
 - $f: \Sigma^* \rightarrow F$ (with F finite) such that
 - $f(w) = f(z) \iff f(wu) = f(zu)$, for all $w, z, u \in \Sigma^*$
- All these classes of languages are identifiable in the limit (polynomial algorithm)

43

k-Reversible Languages

- 0-reversible regular language L
 - ⇒ consider the automaton A associated with L
 - ⇒ consider the reversed automaton A' of A obtained by exchanging initial and final states and reversing each transition edge
 - ⇒ L is 0-reversible if both A and A' are deterministic
- k -reversible regular language L
 - ⇒ A is deterministic
 - ⇒ A' is deterministic with “lookahead” k

44

k-Reversible Languages

○ Characteristic sample for a k-reversible language (intuition)

⇒ a set of strings that touch each state of A, and exercise each transition of A

⇒ the strings need to have minimum length

○ Formally, given $A=(Q, \Sigma, \delta, q_0, Q_F)$

$$\chi = \{ u(q)v(q) \mid q \in Q \} \cup \{ u(q)av(\delta(q,a)) \mid q \in Q, a \in \Sigma \}$$

where $u(q)$ and $v(q)$ are words of minimal length such that: $\delta^*(q_0, u(q)) = q$ and $\delta^*(q, v(q)) \in Q_F$

45

k-Reversible Languages: Example

○ Professors and their courses

⇒ name, email (optional), one or more courses

```
<html><body>
<h1>John Doe</h1> email: <b>doe@dot.edu</b>
<p>Courses:<br>
<i>Distributed Systems</i><br>
</body></html>
```



```
<html><body>
<h1>John Doe</h1>
<p>Courses:<br>
<i>Databases</i><br>
<i>Operating Systems</i><br>
</body></html>
```

class of pages
with homogeneous
content and layout

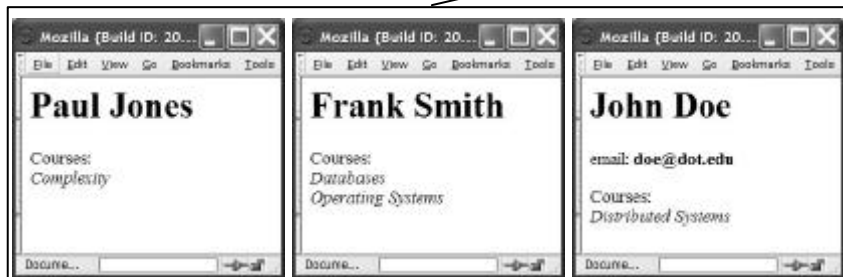
46

k-Reversible Languages: Example

○ This is a 1-reversible language

```
<html><body>
<h1>#PCDATA</h1>
(email: <b>#PCDATA</b>)?
<p>Courses:<br>
(<i>#PCDATA</i><br>)+
</body></html>
```

the sources of these
three HTML pages
form a characteristic
sample for the grammar



47

k-Reversible Languages

- For each fixed k , there is a polynomial time algorithm that correctly infers a k -reversible language from positive samples
- The algorithm is completely unsupervised
 - ⇒ automata theoretic, state-merging techniques
- Please note:
 - ⇒ the algorithm requires the knowledge of k
 - ⇒ to identify the language, the input must contain a characteristic sample (otherwise, the grammar is not correctly inferred)

48

k-Reversible Languages

○ For example, the samples below cannot be used to identify the language

⇒ although they touch each state of A and exercise all transitions



this does not contain a characteristic sample the learning fails to infer the grammar

49

Why Not Grammar Inference for IE ?

○ Grammar inference techniques might in principle solve the wrapper maintenance problem

⇒ tight grammars

⇒ unsupervised algorithms

○ However, virtually none of the recent approaches to information extraction from HTML is based on grammar inference techniques

○ There are several reasons for this

50

Why Not Grammar Inference for IE ?

- Reason I: Assumptions on the input to the inference algorithm
 - ⇒ availability of a characteristic sample
- The minimal length requirement makes highly unlikely its availability
 - ⇒ a simple probabilistic argument shows that in practice there is a very low probability to find a characteristic sample in a bunch of randomly selected HTML pages

51

Why Not Grammar Inference for IE ?

- Reason II: The grammar by itself is not a wrapper
- Recall that a wrapper is
 - ⇒ a grammar
 - ⇒ plus labeled terminals (attribute names)
 - ⇒ plus a target structure for the output
- In essence, the wrapper needs to know
 - ⇒ how to output strings that have been parsed

52

Why Not Grammar Inference for IE ?

- Ideally, grammar inference algorithms would be useful if we could find a class of grammars that
 - ⇒ are natural with respect to HTML pages
 - ⇒ have a more practical notion of characteristic sample
 - ⇒ are such that the target structure can be identified easily by looking at the grammar itself

53

Part III Bridging the Gap

54

The RoadRunner Approach

○RoadRunner

⇒[Crescenzi, Mecca, Merialdo 2001] a research project on information extraction from Web sites

○The goal

⇒developing fully automatic techniques for information extraction from Web sites

○Contributions

⇒it bridges the gap between wrapper induction and traditional grammar inference techniques

55

The RoadRunner Approach

○The target

⇒large Web sites with HTML pages generated by programs (scripts) based on the content of an underlying data store

⇒the data store is usually a relational database

○The typical page-generation process

⇒data are extracted from the relational tables and possibly joined and nested

⇒the resulting dataset is exported in HTML format by attaching tags to values

56

Example: A Fictional Bookstore

Name	Books				
	Title	Description	Editions		
			Details	Year	Price
John Smith	Database Primer	This book ...	First Edition, Paperback	1998	20\$
	Computer Systems		Second Edition, Hard Cover	2000	30\$
Paul Jones	XML at Work				
	HTML and Scripts				
	JavaScripts				
...			

Source Dataset



HTML Pages

57

The Schema Finding Problem

○ Page class

- ⇒ collection of pages generated by the same script from a common dataset
- ⇒ these pages typically share a common structure

○ Schema finding problem

- ⇒ *given a set of sample HTML pages belonging to the same class, automatically recover the source dataset*

○ Solution: Wrapper

- ⇒ structure of the underlying dataset
- ⇒ extraction rules

58

The Schema Finding Problem

Wrapper Output	A	C	D	B		
				F	G	H
<pre><HTML><BODY><TABLE> <TR> <TD>books.com</TD> <TD><A> John Smith</TD></TR> <TR> <TD>Database Primer</TD> <TD><A> This book ...</TD> <TD> First Edition, ...</TD> <TD> 1998</TD> <TD> 20\$</TD> </TR> <TR> <TD>Computer Systems</TD> <TD><A> An undergraduate ...</TD> <TD> First Edition, ...</TD> <TD> 1995</TD> <TD> 40\$</TD> </TR> <TR> <TD>KML at Work</TD> <TD><A> A comprehensive ...</TD> <TD> First Edition, ...</TD> <TD> 1999</TD> <TD> 30\$</TD> </TR> <TR> <TD>HTML and Scripts</TD> <TD><A> An useful HTML ...</TD> <TD> null</TD> <TD> 1993</TD> <TD> 30\$</TD> </TR> <TR> <TD>JavaScripts</TD> <TD><A> A must in ...</TD> <TD> null</TD> <TD> 2000</TD> <TD> 50\$</TD> </TR> </tbody></table></pre>	John Smith	Database Primer	This book ...	First Edition, ...	1998	20\$
	Paul Jones	Computer Systems	An undergraduate ...	First Edition, ...	1995	40\$
...

Wrapper = Grammar

```
<HTML><BODY><TABLE>
<TR>
<TD><FONT>books.com</FONT></TD>
<TD><A> #PCDATA</A></TD></TR>
<TR>
<TD><FONT> #PCDATA </FONT></TD>
<TD><A> #PCDATA </A></TD>
<TD><B> #PCDATA </B> </TD>
<TD><B> #PCDATA </B> </TD>
<TD><B> #PCDATA </B> </TD>
</TR>
</tbody></table>
```

Target Schema

```
SET (
  TUPLE (A : #PCDATA;
    B : SET (
      TUPLE (C : #PCDATA;
        D : #PCDATA;
        E : SET (
          TUPLE (F : #PCDATA;
            G : #PCDATA;
            H : #PCDATA)))
    )
  )
)
```

The Schema Finding Problem

○ In essence

- ⇒ we are not making any assumptions on the target structure (which can be arbitrarily nested)
- ⇒ we are not relying on user inputs (the algorithm should be unsupervised)

○ This is a grammar inference problem

- ⇒ but we want the data source schema, not only the grammar
- ⇒ we want a more natural notion of characteristic sample

The RoadRunner Approach

○Nested Types

⇒ nested relation types with optional attributes

○Source Dataset

⇒ instance of a nested type

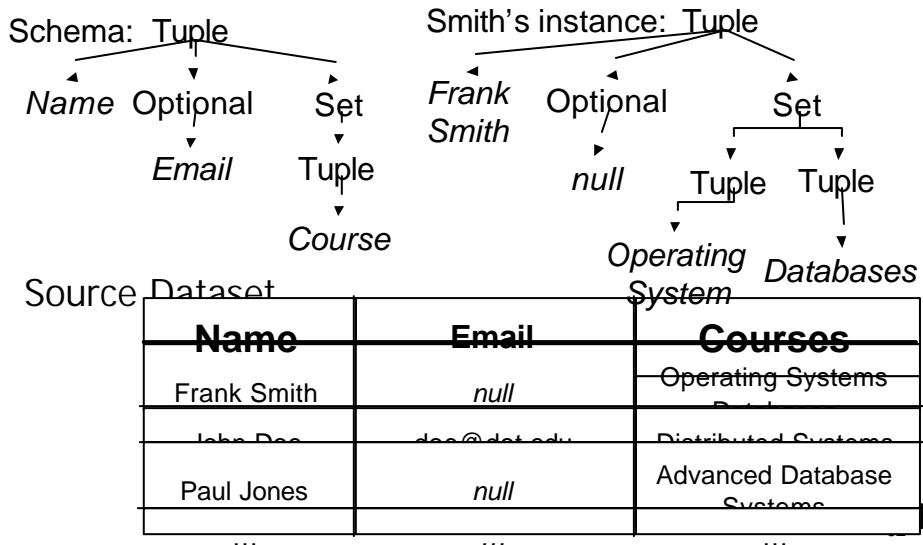
○Target class of grammars

⇒ markup-grammars

⇒ intuition: languages obtained by serializing a dataset with HTML markup

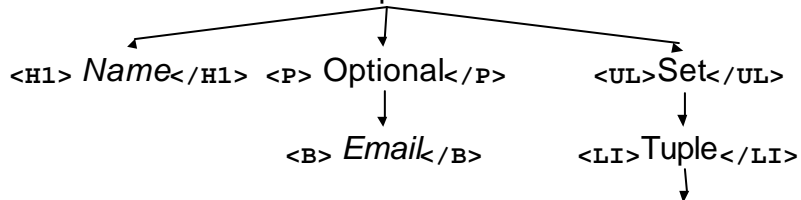
61

Nested Relations with Optionals



Serializing Instance Mark-Up Encodings

Tagged Schema: `<HTML><BODY>Tuple</BODY></HTML>`



Smith's Instance Encoding:

```
<HTML><BODY>
  <H1>Frank Smith</H1>
  <P></P>
  <UL>
    <LI>Operating System</LI>
    <LI>Databases</LI>
  </UL>
</BODY></HTML>
```

Mark-up Language: *Course*

```
<HTML><BODY>
<H1> #PCDATA </H1>
<P> ( <B>#PCDATA </B> )? </P>
<UL>
  ( <LI> #PCDATA </LI> )+
</UL>
</BODY></HTML>
```

63

Prefix Markup-Encodings

○ Mark-up grammar

⇒ any regular language generated by a markup-encoding of a nested type

○ Prefix markup grammar (intuition)

⇒ markup grammar such that the resulting language is LL(1) in both directions

○ In essence

⇒ we have identified a subclass of union-free regular expressions

⇒ this class has a number of nice properties

64

Prefix Markup-Encodings

- Property I: There is a straightforward mapping from grammars to database types
 - ⇒ concatenation in the grammar correspond to database tuples
 - ⇒ each + in the grammar corresponds to a database set
 - ⇒ each ? in the grammar corresponds to an optional
- In fact
 - ⇒ given a mark-up language, it is possible to recover the underlying type in linear time

65

Prefix Markup-Encodings

- Property II: It is identifiable in the limit
 - ⇒ note that regular expressions with . and * only are not identifiable in the limit
- In fact
 - ⇒ it is possible to prove that is a subclass of a (slight generalization of) a class of f-distinguishable languages [Fernau, 2000]
 - ⇒ therefore each language has a characteristic sample

66



Prefix Markup-Encodings

- Property III: It has a very natural notion of characteristic sample
 - ⇒ we can drop the minimality requirement
- More specifically
 - ⇒ all we need is a *rich set of instances*
- Rich set of instances
 - ⇒ intuitively: a set of instances that makes “full use” of the underlying type

67

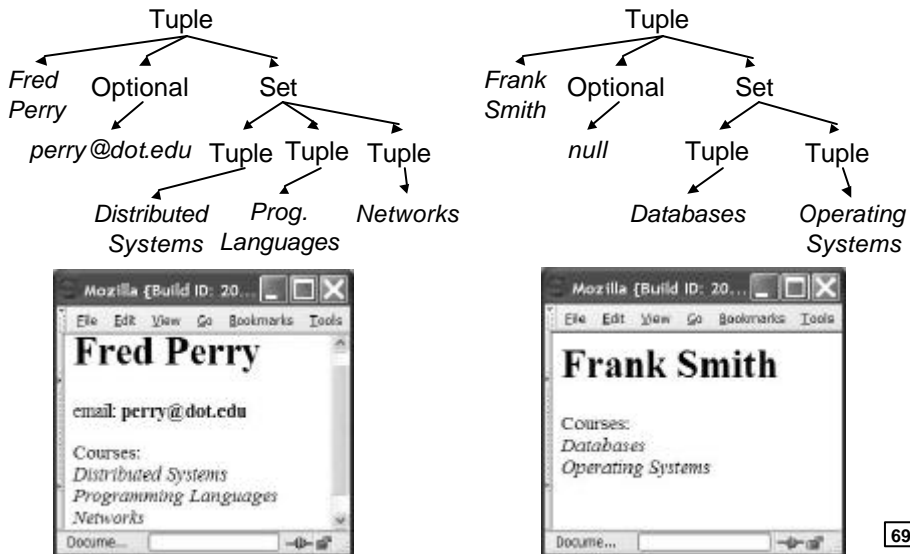


Prefix Markup-Encodings

- Rich set of instances
 - ⇒ basic richness: each leaf node has at least two distinct values
 - ⇒ set richness: each set node has instances of distinct cardinalities
 - ⇒ optional richness: for each optional node has at least one null and one not null instance
- It is possible to prove that each rich set is a characteristic sample for the corresponding prefix markup-language

68

Rich Set of Instances



69

Rich Set of Instances

- A simple probabilistic argument shows that in practice
 - ⇒ the probability of finding a rich set in a random sample is quite high
- Consider a type with k set types
 - ⇒ call p the probability that 2 pages have all instances of a set type of the same cardinality
 - ⇒ the probability to find 2 distinct cardinalities in n random pages is $(1-p^{n-1})$
 - ⇒ the probability that the n samples are set rich is $(1-p^{n-1})^k$ – if $k=5$, $n=10$, $p=50\%$ then 99%

70

Contributions of RoadRunner

- We have developed an unsupervised learning algorithm for identifying prefix-markup languages in the limit
 - ⇒ given a rich-sample of HTML pages, the algorithm correctly identifies the language, i.e., the wrapper
 - ⇒ the algorithm runs in polynomial time
- Note
 - ⇒ with respect to f -distinguishable languages, the algorithm is significantly different due to the different notion of characteristic sample

71

Prefix-Markup Languages

- This seems to represent a reasonable compromise between two aspects
- Expressibility of the grammar formalism
 - ⇒ we have been able to wrap many real-world sites like amazon.com, cnn.com, ebay.com etc.
- Effectiveness of the wrapper induction task
 - ⇒ it is possible to fully automate the step
 - ⇒ there are reasonable assumptions on the samples to inspect

72

Open Problems

- The algorithm needs multiple pages
 - ⇒ as any grammar inference algorithm
 - ⇒ how to wrap single pages ?
- The algorithm is not capable of inferring field names
 - ⇒ these are anonymous in the target wrapper
 - ⇒ some heuristics, but preliminary work
- Disjunctions are necessary in some cases
 - ⇒ this makes the learning much more complex
 - ⇒ a form of “non-disruptive” disjunction ?

73

References

74

References

Wrapper Induction - Finite State Techniques

- B. Adelberg. **NoDoSE - a tool for semi-automatically extracting structured and semistructured data from text documents.** In *SIGMOD98*.
- C. Hsu and M. Dung. **Generating Finite State Transducers for Semistructured Data Extraction from the Web.** *Information Systems 23, 1998*
- N. Kushmerick, D. S. Weld, and R. Doorenbos. **Wrapper induction for information extraction.** In *IJCAI'97*.
- N. Kushmerick. **Wrapper induction: Efficiency and Expressiveness.** *Artificial Intelligence 118, 2000*.
- N. Kushmerick. **Wrapper Verification.** *WWW Journal 3, 2000*.
- K. Lerman, S. Minton **Learning the Common Structure of Data** *Proc. of National Conference on Artificial Intelligence, 2000*
- I. Muslea, S. Minton, and C. A. Knoblock. **A hierarchical approach to wrapper induction.** In *Proceedings of the Third Annual Conference on Autonomous Agents, pages 190–197, 1999*.

75

References

Wrapper Induction - Other Approaches

- D. W. Embley, D. M. Campbell, Jiang Y. S., S. W. Liddle, Ng Y., D. Quass, and Smith R. D. **A conceptual-modeling approach to extracting data from the web.** In *ER98*.
- N. Kushmerick, B. Thomas **Adaptive Information Extraction: Core Technologies for Information Agents** *In Intelligent Information Agents R&D in Europe: An AgentLink Perspective, 2002*
<http://www.cs.ucd.ie/staff/nick/>
- I. Muslea **Extraction Patterns for Information Extraction Tasks: A Survey** *Proc. of AAAI Workshop on Machine Learning for Information Extraction, 1999*

76

References

Grammar Inference

- D. Angluin. **Inductive Inference of Formal Languages from positive Data.** In *Information and Control*, (45):117-135, 1980.
- D. Angluin. **Inference of Reversible Languages.** In *Journal of the ACM*, 29(3):741-765, 1982.
- H. Fernau. **On learning Function Distinguishable Languages.** Technical Report WSI-2000-13. *Wilhelm-Schickard-Institute fur Informatik*, 2000
- P. Garcia, E. Vidal **Inference of k-testable Languages in the Strict Sense and Application to Syntactic Pattern Recognition** *IEEE Trans. on Pattern Analysis and Machine Intelligence* 12, 1990
- E. M. Gold, **Language Identification in the limit.** *Information and Control* 10, 5 (1967), 447-474
- V. Radhakhishnan, G. Nagaraja **Inference of Regular Grammars via Skeletons** *IEEE Trans. on Systems, Man and Cybernetics* 17, 1987.

77

References

RoadRunner

- S. Grumbach, G. Mecca. **In Search of the Lost Schema.** In *ICDT'99*.
- V. Crescenzi, G. Mecca, P. Merialdo: **RoadRunner: Towards Automatic Data Extraction from Large Web Sites.** In *VLDB 2001: 109-118*
- V. Crescenzi, G. Mecca, P. Merialdo. **Automatic Web Information Extraction in the RoadRunner System.** Workshop DASWIS-2001 in conjunction with ER 2001
- V. Crescenzi, G. Mecca, P. Merialdo. **Wrapper Oriented Classification of Web Pages.** SAC 2002. March 10-14, Madrid, Spain

78