

Automatic Wrapper Generation

Kristina Lerman
University of Southern California

August 14, 2008

University of Southern California

1

Manual Wrapper Generation

- ⌘ Manual wrapper generation requires user to
 - ☒ Specify the schema of the information source
 - ☒ Single tuple
 - ☒ Nested type
 - ☒ List of tuples or nested types
 - ☒ Label data on several example HTML pages
 - ☒ Tedious, especially for lists
- ⌘ Goal of Automatic Wrapper Generation is to eliminate user intervention

August 14, 2008

University of Southern California

2

Overview

- ⌘ Methods for automatic wrapper creation and data extraction
 - ☒ Grammar Induction approach
 - ☒ *Towards Automatic Data Extraction from Large Web Sites*
 - ☒ Website structure-based approach
 - ☒ *AutoFeed: An Unsupervised Learning System for Generating Webfeeds*
 - ☒ *Using the Structure of Web Sites for Automatic Segmentation of Tables*
 - ☒ DOM-based:
 - ☒ Simile/Piggybank

August 14, 2008

University of Southern California

3

Grammar Induction Approach

- ⌘ Pages automatically generated by scripts that encode results of db query into HTML
 - ☒ Script = grammar
- ⌘ Given a set of pages generated by the same script
 - ☒ Learn the grammar of the pages
 - ☒ Wrapper induction step
 - ☒ Use the grammar to parse the pages
 - ☒ Data extraction step

August 14, 2008

University of Southern California

4

Limitations

- ⌘ Learnable grammars
 - ☒ Union-Free Regular Expressions (RoadRunner)
 - ☒ Variety of schema structure: tuples (with optional attributes) and lists of (nested) tuples
 - ☒ Does not efficiently handle disjunctions – pages with alternate presentations of the same attribute
 - ☒ Context-free Grammars (Hong&Clark paper)
 - ☒ Limited learning ability
- ⌘ User needs to provide a set of pages of the same type

August 14, 2008

University of Southern California

5

Website Structure-based Approach

- ⌘ Websites attempt to simplify user navigation and interaction with data by organizing how data is presented across the site
 - ☒ Hierarchical organization
 - ☒ List of results
 - ☒ Detail pages ...
 - ☒ Machine learning methods take advantage of structure to extract data

August 14, 2008

University of Southern California

6

Limitations

- ⌘ Performance depends on the choice of the learning algorithm
 - ☑ Noise can affect performance (Lerman et al paper)
 - ☑ Can combine different learning algorithms to improve performance (Autofeed)

August 14, 2008

University of Southern California

7

DOM-based Approach

- ⌘ Use Document Object Model tree of HTML pages to extract data
- ⌘ Works well, but on fairly simple and well-structured pages

August 14, 2008

University of Southern California

8

Towards Automatic Data Extraction from Large Web Sites by Crescenzi, Mecca, & Merialdo

August 14, 2008

University of Southern California

9

RoadRunner Overview

- ⌘ Automatically generates a wrapper from large structured web pages
- ⌘ Supports nested structures and lists
- ⌘ Efficient approach to large, complex pages with regular structure

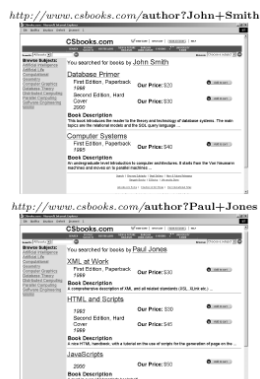
August 14, 2008

University of Southern California

10

Example Pages

- ⌘ Extracts the fields and hierarchical structure
- ⌘ Depends on well-structured HTML
- ⌘ Only extracts at the entire field level



August 14, 2008

University of Southern California

11

Extracted Result

The image shows a screenshot of the RoadRunner tool's output. It displays an XML schema with columns labeled A through F. The data extracted from the website is shown in a table format below the schema.

Schema Number: 1: A	B	C	D	E	F
John Smith	Database Primer	First Edition, Paperback	1998	\$20	This book introduces the reader to the theory and technology...
		Second Edition, Hard Cover	2000	\$30	
	Computer Systems	First Edition, Paperback	1995	\$40	An undergraduate level introduction to computer...
Paul Jones	XML at Work	First Edition, Paperback	1999	\$30	A comprehensive description of XML and all related standards...
	HTML and Scripts	Second Edition, Hard Cover	1999	\$45	A useful HTML handbook with a good tutorial on the use of...
	JavaScripts		2000	\$50	A must in every Webmaster's bookshelf...

August 14, 2008

University of Southern California

12

Approach

- ⌘ Given a set of example pages
- ⌘ Generates a *Union-free Regular Expression (UFRE)*
 - ☑ RE without any disjunctions
 - ☑ List of tuples (possibly nested): (a, b, c)+
 - ☑ Optional attributes: (a)?
 - ☑ Strong assumption that usually holds
- ⌘ Find the least upper bounds on the RE lattice to generate a wrapper in *linear time*
- ⌘ Reduces to finding the least upper bound on two UFREs

August 14, 2008

University of Southern California

13

Matching/Mismatches

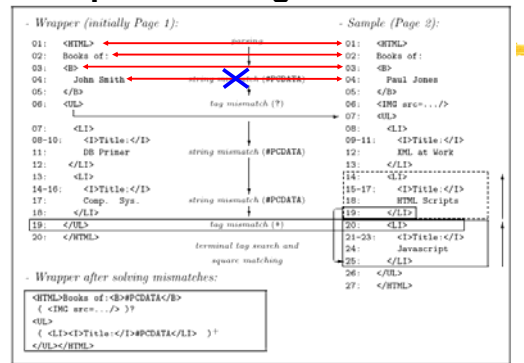
- Given a set of pages of the same type
- ⌘ Take the first page to be the wrapper (UFRE)
- ⌘ Match each successive sample page against the wrapper
- ⌘ Mismatches result in generalizations of the regular expression
- ⌘ Types of mismatches:
 - ☑ String mismatches
 - ☑ Tag mismatches

August 14, 2008

University of Southern California

14

Example Matching



August 14, 2008

University of Southern California

15

String Mismatches: Discovering Fields

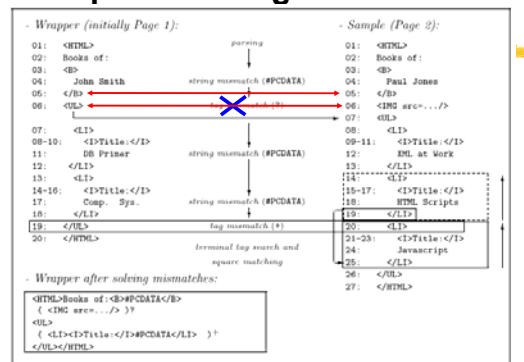
- ⌘ String mismatches are used to discover fields of the document
 - ⌘ Wrapper is generalized by replacing "John Smith" with #PCDATA
- <HTML>Books of: John Smith
 → <HTML> Books of: #PCDATA

August 14, 2008

University of Southern California

16

Example Matching



August 14, 2008

University of Southern California

17

Tag Mismatches: Discovering Optionals

- ⌘ First check to see if mismatch is caused by an iterator (described next)
- ⌘ If not, could be an optional field in wrapper or sample
- ⌘ Cross search used to determine possible optionals
- ⌘ Image field determined to be optional:
 - ☑ ()?

August 14, 2008

University of Southern California

18

Example Matching

- Wrapper (initially Page 1):

```

01: <HTML>
02: Books of:
03: <B>
04: John Smith
05: </B>
06: <UL>
07: <LI>
08-10: <IDTitle></ID>
11: </LI>
12: <LI>
13: <LI>
14-16: <IDTitle></ID>
17: Comp. Sys.
18: </LI>
19: </UL>
20: </HTML>

```

- Sample (Page 2):

```

01: <HTML>
02: Books of:
03: <B>
04: Paul Jones
05: </B>
06: <IMG src=... />
07: <UL>
08: <LI>
09-11: <IDTitle></ID>
12: EML at Work
13: </LI>
14: <LI>
15-17: <IDTitle></ID>
18: HTML Scripts
19: </LI>
20: <LI>
21-23: <IDTitle></ID>
24: Javascript
25: </LI>
26: </UL>
27: </HTML>

```

Diagram illustrating the matching process between the wrapper and sample HTML. Red arrows show string mismatches (PCDATA) and tag mismatches. A blue 'X' marks a terminal tag search and square matching failure. The final wrapper after solving mismatches is shown below.

- Wrapper after solving mismatches:

```

<HTML>Books of <B>#PCDATA</B>
( <IMG src=... /> )?
<UL>
( <LI><IDTitle></ID>#PCDATA</LI> )+
</UL></HTML>

```

August 14, 2008

University of Southern California

19

Tag Mismatches: Discovering Iterators

- ⌘ Assume mismatch is caused by repeated elements in a list
 - ☑ End of the list corresponds to last matching token:
 - ☑ Beginning of list corresponds to one of the mismatched tokens: or
 - ☑ These create possible "squares"
- ⌘ Match possible squares against earlier squares
- ⌘ Generalize the wrapper by finding all contiguous repeated occurrences:
 - ☑ (<IDTitle></ID>#PCDATA)+

August 14, 2008

University of Southern California

20

Example Matching

- Wrapper (initially Page 1):

```

01: <HTML>
02: Books of:
03: <B>
04: John Smith
05: </B>
06: <UL>
07: <LI>
08-10: <IDTitle></ID>
11: </LI>
12: <LI>
13: <LI>
14-16: <IDTitle></ID>
17: Comp. Sys.
18: </LI>
19: </UL>
20: </HTML>

```

- Sample (Page 2):

```

01: <HTML>
02: Books of:
03: <B>
04: Paul Jones
05: </B>
06: <IMG src=... />
07: <UL>
08: <LI>
09-11: <IDTitle></ID>
12: EML at Work
13: </LI>
14: <LI>
15-17: <IDTitle></ID>
18: HTML Scripts
19: </LI>
20: <LI>
21-23: <IDTitle></ID>
24: Javascript
25: </LI>
26: </UL>
27: </HTML>

```

Diagram illustrating the matching process. A red box highlights a specific mismatch in the sample HTML. The final wrapper after solving mismatches is shown below.

- Wrapper after solving mismatches:

```

<HTML>Books of <B>#PCDATA</B>
( <IMG src=... /> )?
<UL>
( <LI><IDTitle></ID>#PCDATA</LI> )+
</UL></HTML>

```

August 14, 2008

University of Southern California

21

Internal Mismatches

- ⌘ Generate *internal mismatch* while trying to match square against earlier squares on *the same page*
 - ☑ Solving internal mismatches yield further refinements in the wrapper
 - ☑ List of book editions
 - ☑ <I>Special!</I>

August 14, 2008

University of Southern California

22

Recursive Example

- Wrapper (initially Page 1):

```

01-05: <HTML>Books of: <B>John Smith</B>
06: <UL>
07: <LI>
08: Computer Systems
09: <B>
10: 1st Ed., 1998
11: </B>
12: </LI>
13: </UL>
14: <LI>
15: Database Primer
16: <B>
17: </B>
18: </LI>
19: 1st Ed., 1998
20-22: <I>Special!</I>
23: </LI>
24: </UL>
25: 2nd Ed., 2000
26: </LI>
27: </UL>
28-30: </HTML>

```

- Sample (Page 2):

```

01-05: <HTML>Books of: <B>Paul Jones</B>
06: <UL>
07: <LI>
08: EML at Work
09: <B>
10: 1st Ed., 1999
11: </B>
12: </LI>
13: </UL>
14: <LI>
15: </LI>
16: </HTML>

```

Diagram illustrating the recursive matching process. It shows external mismatches between the wrapper and sample HTML, and internal mismatches within the wrapper. The final wrapper after solving mismatches is shown below.

- Wrapper after solving mismatches:

```

<HTML>Books of <B>#PCDATA</B>
<UL><LI>#PCDATA</LI>
( <B>#PCDATA</B> )?
( <I>Special!</I> )?
</UL></HTML>

```

August 14, 2008

University of Southern California

23

Discussion

- ⌘ Assumptions:
 - ☑ Pages are well-structured
 - ☑ Want to extract at the level of entire fields
 - ☑ Structure can be modeled without disjunctions
- ⌘ Search space for explaining mismatches is huge
 - ☑ Uses a number of heuristics to prune space
 - ☑ Limited backtracking
 - ☑ Limit on number of choices to explore
 - ☑ Patterns cannot be delimited by optionals
 - ☑ Will result in pruning possible wrappers

August 14, 2008

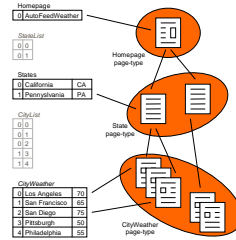
University of Southern California

24

AutoFeed: An Unsupervised Learning System for Generating Webfeeds by Gazen and Minton

Relational Model of a Web Site

- ☞ Sites are well structured to improve user experience
- ☞ **Generation:** Given relational data, scripts generate web site, e.g., weather site
- ☞ **Extraction** is opposite task: Given web site, find underlying relational data



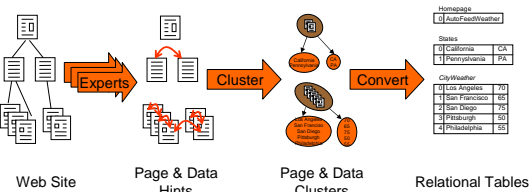
Overview of Approach

- ☞ Sites aim to be easy to understand and navigate
 - Many types of structure
 - ☑ Graph structure of site's links
 - ☑ URL naming scheme
 - ☑ Content of pages
 - ☑ HTML structure within page types, ...
- ☞ **Experts** focus on individual structures and output discoveries as *hints*
 - ☑ Experts are heterogeneous
 - ☑ Probabilistically combine experts (don't have to be correct all the time)

Hints

- ☞ Hints describe local structural similarities within *pages* or within *data*
- ☞ Hints help find relational structure of the site
 - ☑ Used to cluster pages and data

Overview



Page-level Experts

- ☞ **URL patterns** give clues about site structure
 - ☑ Similar pages have similar URLs, e.g.:
 - ☑ <http://www.bookpool.com/sm/0321349806>
 - ☑ <http://www.bookpool.com/sm/0131118269>
 - ☑ <http://www.bookpool.com/ss/L?pu=MN>
- ☞ **Page templates**
 - ☑ Similar pages contain common sequences of substrings

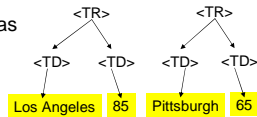


Data-level Experts

- ⌘ **Page layout** gives clues about relational structure
 - ☑ Similar items aligned vertically or horizontally, e.g.:
 - ☑ Coincidental alignment results in bad hints

List structure

- ☑ List rows are represented as repeating HTML structures



August 14, 2008

University of Southern California

31

Page and Data Similarity

- ⌘ Surface structure is often not helpful
 - ☑ e.g., page with an empty list and one with a long list will be found not similar
- ⌘ Instead, use local structural similarities
 - ☑ Experts output hints
 - ☑ Structure represented as hints
 - ☑ Clusters evaluated *probabilistically* using hints
 - ☑ Probabilistic representation gives flexible framework for combining possibly conflicting hints

August 14, 2008

University of Southern California

32

Probabilistic Evaluation

- ⌘ Find clustering that maximizes probability of observing hints:

$$P(\text{clustering}|\text{hints}) = P(\text{hints}|\text{clustering}) * P(\text{clustering}) / P(\text{hints})$$
- ⌘ Generative process for $P(\text{hints}|\text{clustering})$

Data Hints

$$P(I) \times \prod_{i \in I} p_i \text{ where } p_i = \begin{cases} \frac{1}{\text{count}_i} \times \frac{1}{\binom{c_i}{2}} & \text{if } m_i \text{ is a matched pair of tokens} \\ \frac{1}{\text{count}_i} & \text{if } m_i \text{ is an unmatched token} \end{cases}$$

Page Hints

$$\frac{1}{\text{count}_p} \times \frac{1}{\binom{c_p}{2}}$$

August 14, 2008

University of Southern California

33

Clustering Algorithm

- ⌘ Leader-follower algorithm
 - ☑ Process items one at a time
 - ☑ If distance to nearest cluster < threshold add item to it
 - ☑ Else create new cluster
 - ☑ "distance" between page and page-cluster is determined by
 - change in $Prob(\text{clustering}|\text{hints})$
 - ☑ Prevents one large cluster, b/c smaller probabilities assigned to hints in larger clusters

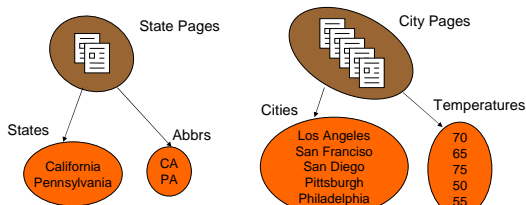
August 14, 2008

University of Southern California

34

Clustering

- ⌘ Cluster pages and data
- ⌘ Page-clusters are parents of data-clusters
- ⌘ For example:



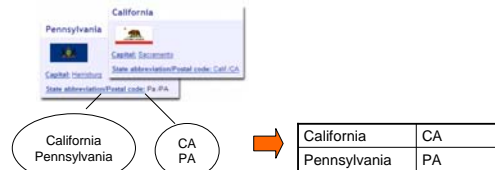
August 14, 2008

University of Southern California

35

Clusters to Tables

- ⌘ Base tables are built from clusters that contain a single tuple per page.
 - ☑ Each cluster becomes a column.
 - ☑ Each row represents a page.
- ⌘ Lists constructed from clusters not used for base table



August 14, 2008

University of Southern California

36

Evaluation

- ⌘ AutoFeed evaluated against manually built wrappers
 - ☐ But can only evaluate part of output
- ⌘ For each target column in the wrapper
 - ☐ Find matching AutoFeed column
 - ☐ Count relevant retrieved values
 - ☐ Calculate precision and recall

Target Column
from Supervised Wrapper

Column Extracted
by AutoFeed

Los Angeles	70
San Francisco	65
San Diego	75
Pittsburgh	50
Philadelphia	55

Mostly Cloudy	Hi
San Francisco	65
San Diego	75
Pittsburgh	50

Retrieved&Relevant=3
Retrieved=4
Relevant=5
Precision=3/4
Recall=3/5

August 14, 2008

University of Southern California

37

Experiments and Results

- ⌘ Domains
 - ☐ Extract product name, manufacturer, price, etc., from online retail sites (Buy.com, CompUSA, etc.)
 - ☐ Extract authors, titles, URLs, from journals (DMTCS, JAIR, etc.)
 - ☐ Extract job id, position, location from job listings (50 Forbes 500 companies)
- ⌘ Good results
 - ☐ Data extracted correctly from many of the sites
- ⌘ Some problems:
 - ☐ Extracting larger fields, e.g. "Price: \$19.95"
 - ☐ Over-general & over-specific clusters

Field	RR	Ret.	Rel.	Precision	Recall
Field	81	81	81	100%	100%
item no.	100	100	103	97%	97%
model no.	66	68	71	96%	93%
name	97	100	103	97%	94%
price	77	88	103	85%	75%
authors	1173	1197	1212	98%	97%
title	1173	1188	1212	99%	97%
URL	528	528	1212	100%	44%
position	1391	1391	1453	100%	96%
req. id	1278	1278	1422	100%	90%
location	1302	1302	1445	100%	90%

August 14, 2008

University of Southern California

38

AutoFeed Conclusions

- ⌘ Promising approach:
 - ☐ Multiple experts for multiple structures
 - ☐ Common language for collecting and combining evidence
- ⌘ Principle applicable to beyond web extraction
 - ☐ Interpreting complex environments
- ⌘ Need to improve prototype system:
 - ☐ More experts
 - ☐ Better ways to combine evidence
 - ☐ Confidence scores on hints
 - ☐ Cannot-link hints

August 14, 2008

University of Southern California

39

Using the Structure of Web Sites for Automatic Segmentation of Tables by Lerman et al.

August 14, 2008

University of Southern California

40

Exploiting Structure of Web Sites

- ⌘ Most Web sites that allow user to access databases present results in dynamically generated pages
 - ☐ Web tables
- ⌘ Web sites are very structured in terms of
 - ☐ Organization of the site
 - ☐ Layout of pages
 - ☐ Content of data
- ⌘ Exploit this structure for automatic information extraction

August 14, 2008

University of Southern California

41

Structure of Web Sites

Entry page → List pages → Detail pages



August 14, 2008

University of Southern California

42

Structure of Pages and Data



August 14, 2008

University of Southern California

43

- ⌘ Pages are generated from a template
- ⌘ Data in the same "column" is of the same type
 - ☐ E.g., each listing starts with NAME, followed by ADDRESS, CITY, STATE, etc.

Underlying Structure is not Always Clear



August 14, 2008

University of Southern California

44

- ⌘ Variability of real-world data may obscure the underlying structure
 - ☐ Missing columns
 - ☐ "List Price" and "You save"
 - ☐ Formatting
 - ☐ Content

Problem Overview

Automatically, efficiently extract records from Web tables

Given a set of list and detail pages...

- ☐ Segment list data using information from detail pages
 - ☐ Logic based approach
 - Based on Constraint Satisfaction Problems (CSP)
 - Encode relations between data on list and detail pages as logical constraints and solve them
 - ☐ Probabilistic inference approach
 - Learns a model from data
 - Record segmentation is an assignment that maximizes the likelihood of data given the model

August 14, 2008

University of Southern California

45

Identify Table and Extract Data



August 14, 2008

University of Southern California

46

Web sources generate list pages from a template and fill them with query results

- ⌘ Deduce page template
 - ☐ Given two or more example pages, derive the template used to generate them
 - ☐ Table data and formatting tags are not part of the template
- ⌘ Heuristic: largest slot contains the table
- ⌘ Extract table data
 - ☐ Extract contiguous sequences of tokens from the largest page slot

Record Segmentation Basics (1)

- ⌘ List and detail pages present two views of the same record
 - ☐ Some overlapping fields
- ⌘ Each detail page is a distinct record
- ⌘ Assumption: Web tables are laid out horizontally
 - ☐ Each record is in a separate row
 - ☐ Order in which extracts appear in the text stream of list page is the same order they appear in the table

August 14, 2008

University of Southern California

47

Record Segmentation Basics (2)



August 14, 2008

University of Southern California

48

For each extract E_i , record all detail pages on which it appears

E_1 : John Smith	r1, r2
E_2 : 221 Was...erloo	r1
E_3 : New Hol...43145	r1
E_4 : (740) 335-5555	r1, r2
E_5 : John Smith	r1, r2
E_6 : 221R Was...erloo	r2
E_7 : Washing...43160	r2
E_8 : (740) 335-5555	r1, r2
E_9 : George W. Smith	r3
E_{10} : Findlay, ... 45840	r3
E_{11} : (419) 423-1212	r3

Record Segmentation Basics (3)

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}	E_{11}
r_1	1	1	1	1	1			1			
r_2	1			1	1	1	1	1			
r_3									1	1	1

Observations of extracts on detail pages add valuable information for record segmentation

Second record can be

- $E_1E_3E_6E_7E_8$
- $E_4E_5E_6E_7$
- $E_4E_5E_6E_7$
- $E_6E_7E_8$
- $E_3E_6E_7E_8$
- E_6E_7

CSP Approach to Record Segmentation

In CSP, problems are stated as logical expressions over variables

Pseudo-boolean (PB) representation

- Variables are 0-1, constraints can be inequalities
- Solution is assignment that minimizes inequality constraints

Encode record segmentation problem in PB representation

Assignment variable x_j

$x_j=1$ when E_j is assigned to r_j

$x_j=0$ when E_j is no part of r_j

Information from detail pages imposes constraints

- Structure constraints
- Position constraints

Structure Constraints

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}	E_{11}
r_1	1	1	1	1	1			1			
r_2	1			1	1	1	1	1			
r_3									1	1	1

Uniqueness constraint

- Every extract E_j belongs to exactly one record r_j
- $\sum_j x_{ij} = 1$

Consecutiveness constraint

- Only contiguous blocks of extracts can be assigned to the same record

Structure Constraints

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}	E_{11}
r_1	1	1	1	1	1			1			
r_2	1			1	1	1	1	1			
r_3									1	1	1

Uniqueness constraint

- Every extract E_j belongs to exactly one record r_j

Consecutiveness constraint

- Only contiguous blocks of extracts can be assigned to the same record
- $x_{ij} + x_{kj} \leq 1$ when there is $n, k < n < i$, s.t. $x_{nj} = 0$

Position Constraints

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}	E_{11}
p^{730}_1	1				1						
p^{772}_1		1									
p^{812}_1			1								
p^{840}_1				1				1			
p^{536}_2	1				1						
p^{578}_2				1				1			
p^{608}_2						1					
p^{642}_2							1				

Position constraint

- No two extracts assigned to same record can appear in the same position on the detail page

$pos_j(E_i) = pos_j(E_k)$, then E_i and E_k cannot be assigned to same record j

Constraints are expressed mathematically and solved using integer optimization

Probabilistic Approach to Record Segmentation

Record segmentation as probabilistic inference

No labeled training examples

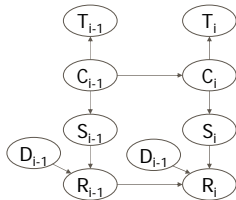
Factor the problem for efficient learning

Bootstrap the learning algorithm with information from detail pages

Structure constrain the problem further with global parameters such as record length

Probabilistic Model for Record Extraction: Variables

- ⌘ Observed variables
 - ⊠ $T=(T_1, \dots, T_n)$ token types of extract E_i
 - ⊠ $D=(D_1, \dots, D_n)$ detail pages on which E_i was observed
- ⌘ Unobserved variables
 - ⊠ $R=(R_1, \dots, R_n)$ record id
 - ⊠ $C=(C_1, \dots, C_n)$ column label
 - ⊠ $S=(S_1, \dots, S_n)$: S_i =true if E_i is the start of a new record; false otherwise
- ⌘ Dependencies
 - ⊠ Given by arrows, eg, $P(C_i|C_{i-1})$
- ⌘ Segmentation
 - ⊠ find values for R and C given T, D variables: $\text{argmax}_{P(R,C|T,D)}$



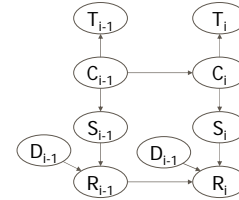
August 14, 2008

University of Southern California

55

Probabilistic Model for Record Extraction: Dependencies

- ⌘ $P(T_i|C_i)$: token type of E_i depends on column
- ⌘ $P(C_i|C_{i-1})$: column label of E_i depends on previous column label (eg, *NAME* followed by *ADDRESS*, sometimes by *STATE*)
- ⌘ $P(S_i|C_i)$: new record starts with a given column (eg, *NAME*)
- ⌘ $P(R_i|R_{i-1}, D_i, S_i)$: record number of E_i depends on record number of previous extract, whether it starts a new record, and detail pages on which it was observed.



August 14, 2008

University of Southern California

56

Learning the Model

- ⌘ Constrain the problem further
 - ⊠ Bootstrap
 - Detail pages provide initial guesses for parameters
 - ⊠ $P(R_i=r_i)$
 - ⊠ Evidence about where records start: $P(S_i=\text{true})=1$
 - ⊠ Token types of columns $P(T_i|C_i)$
 - ⊠ Structure
 - ⊠ Table has π columns specified by the underlying database schema
 - ⊠ However, not every record will have an attribute for every field, i.e., not every record has π fields
 - ⊠ Number of fields in a record estimated from data

August 14, 2008

University of Southern California

57

Learning the Model

Initial guess for record assignment $P(R_i)$

$P(R_i=r_i)$	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}	E_{11}
r_1	1/2	1	1	1/2	1/2			1/2			
r_2	1/2			1/2	1/2	1	1	1/2			
r_3									1	1	1

Initial guess for record start $P(S_i)$

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}	E_{11}
$P(S_i)$	1	1		1	1	1			1		

Initial guess for length of records

π_k	2	3	4	5	6	7	...				
$P(\pi_k)$	2/14	6/14	4/14	2/14	0	0	0				

August 14, 2008

University of Southern California

58

Learning Algorithm

- ⌘ Use EM to implement the inference algorithm
 1. Initial guess for π_k for each record j
 2. For each potential record, update $P(C_i|T_i, C_{i-1})$
 3. Update $P(S_i|C_i)$
 4. Update $P(R_i|R_{i-1}, D_i, S_i)$
- Result is the most likely assignment of data to R and C = record segmentation

August 14, 2008

University of Southern California

59

Validation

- ⌘ Input data
 - ⊠ list and detail pages from 12 sites in domains: *book sellers*, *property tax*, *white pages*, *corrections*
- ⌘ Metrics
 - ⊠ $P = \text{Cor} / (\text{Cor} + \text{InCor} + \text{NonRecords})$
 - ⊠ $R = \text{Cor} / (\text{Cor} + \text{UnsegRecords})$
 - ⊠ $F = 2PR / (P + R)$
- ⌘ Results
 - ⊠ CSP approach: $P=0.85$, $R=0.84$, $F=0.84$
 - ⊠ Probabilistic approach: $P=0.74$, $R=0.99$, $F=0.85$
 - ⊠ Good performance for an automatic algorithm!

August 14, 2008

University of Southern California

60

Discussion of Results

- ⌘ CSP approach is very reliable on clean data, but sensitive to errors in data source
 - ☑ Attribute has one value on list page and another on detail page
- ⌘ Probabilistic approach tolerates inconsistencies and is more expressive
- ⌘ Combination of two techniques may be more robust

August 14, 2008

University of Southern California

61

Comparison with RoadRunner

- ⌘ RoadRunner System (Crescenzi et al, 2001)
 - ☑ Automatically learns the page and table template by exploiting similarities in page layout (HTML tags)
 - ☑ Uses the template to automatically extract data
 - ☑ Does not allow for disjunctions
 - ☑ Disjunctions are necessary to represent alternative layout instructions for the same field



August 14, 2008

University of Southern California

62

Conclusion

- ⌘ Domain-independent approach for automatically extracting and segmenting data from Web tables
- ⌘ Approach leverages additional information provided by Web site structure
 - ☑ Logic based approach
 - ☑ Information provided by detail pages encoded as constraints and solved to obtain record segmentation
 - ☑ Probabilistic inference approach
 - ☑ Information provided by detail pages and table structure represented as a probabilistic model
 - ☑ Use inference to learn proper segmentation
- ⌘ Validated approach on 12 Web sites from diverse information domains
 - ☑ Efficient, accurate performance, $F=0.85$ and $F=0.84$

August 14, 2008

University of Southern California

63