Automatic Wrapper Generation

Craig Knoblock
University of Southern California
Towards Automatic Data Extraction from Large Web Sites by Crescenzi, Mecca, & Merialdo
Overview

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

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

<table>
<thead>
<tr>
<th>ID</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John Smith</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This book introduces the reader to the theory and technology...</td>
</tr>
<tr>
<td></td>
<td>Computer Systems</td>
<td>First Edition, Paperback</td>
<td>1996</td>
<td>40</td>
<td></td>
<td>An undergraduate level introduction to computer...</td>
</tr>
<tr>
<td>2</td>
<td>Paul Jones</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XML at Work</td>
<td>First Edition, Paperback</td>
<td>1999</td>
<td>30</td>
<td></td>
<td>A comprehensive description of XML and all related standards...</td>
</tr>
<tr>
<td></td>
<td>HTML and Scripts</td>
<td>null</td>
<td>1993</td>
<td>90</td>
<td></td>
<td>A useful HTML handbook, with a good tutorial on the use of sc...</td>
</tr>
<tr>
<td></td>
<td>JavaScripts</td>
<td>null</td>
<td>2000</td>
<td>50</td>
<td></td>
<td>A must in every Webmaster's bookshelf...</td>
</tr>
</tbody>
</table>
Approach

- Given a set of example pages
- Generates a *Union-free Regular Expression (UFRE)*
  - RE without any disjunctions
  - Strong assumption that usually holds
- Find the least upper bounds on the RE lattice to generate a wrapper
- Reduces to finding the least upper bound on two UFREs
Matching/Mismatches

- Start with the first page and create a RE that defines the wrapper
- Match each successive sample against the wrapper
- Mismatches result in generalizations of the regular expression
- Types of mismatches:
  - String mismatches
  - Tag mismatches
Example Matching

- Wrapper (initially Page 1):
  01:  <HTML>
  02:  Books of:
  03:  <B>
  04:  John Smith
  05:  </B>
  06:  <UL>
  07:  <LI>
  08-10:  <I>Title:</I>
  11:  DB Primer
  12:  </LI>
  13:  <LI>
  14-16:  <I>Title:</I>
  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:  <I>Title:</I>
  12:  XML at Work
  13:  </LI>
  14:  <LI>
  15-17:  <I>Title:</I>
  18:  HTML Scripts
  19:  </LI>
  20:  </UL>
  21-23:  <I>Title:</I>
  24:  Javascript
  25:  </LI>
  26:  </UL>
  27:  </HTML>

- Wrapper after solving mismatches:

  <HTML>Books of:<B>#PCDATA</B>
  ( <IMG src=.../> )?
  <UL>
    ( <LI><I>Title:</I>#PCDATA</LI> )+
  </UL></HTML>
String Mismatches: Discovering Fields

- String mismatches are used to discover fields of the document.
- Wrapper is generalized by replacing “John Smith” with #PCDATA.

```html
<HTML>Books of: <B>John Smith
→ <HTML> Books of: <B>#PCDATA
```
Example Matching

- Wrapper (initially Page 1):

01:  <HTML>
02:  Books of:
03:  <B>
04:  John Smith
05:  </B>
06:  <UL>
07:  <LI>
08-10:  <I>Title:</I>
11:  DB Primer
12:  </LI>
13:  <LI>
14-16:  <I>Title:</I>
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:  <I>Title:</I>
12:  XML at Work
13:  </LI>
14:  <LI>
15-17:  <I>Title:</I>
18:  HTML Scripts
19:  </LI>
20:  </UL>
21-23:  <I>Title:</I>
24:  Javascript
25:  </LI>
26:  </UL>
27:  </HTML>

- Wrapper after solving mismatches:

```html
<HTML>
  Books of:<B>#PCDATA</B>
  (<IMG src=.../>)?
  <UL>
    (<LI><I>Title:</I>#PCDATA</LI>)+
  </UL></HTML>
```
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:
  \[\langle \text{img src=...}/\rangle?\]
Example Matching

- Wrapper (initially Page 1):

01:  <HTML>
02:  Books of:
03:  <B>
04:  John Smith
05:  </B>
06:  <UL>
07:  <LI>
08-10:  <I>Title:</I>
11:  DB Primer
12:  </LI>
13:  <LI>
14-16:  <I>Title:</I>
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:  <I>Title:</I>
12:  XML at Work
13:  </LI>
14:  <LI>
15-17:  <I>Title:</I>
18:  HTML Scripts
19:  </LI>
20:  </UL>
21-23:  <I>Title:</I>
24:  Javascript
25:  </LI>
26:  </UL>
27:  </HTML>

- Wrapper after solving mismatches:

<HTML>Books of:<B>#PCDATA</B>
( <IMG src=.../> )?
<UL>
 ( <LI><I>Title:</I>#PCDATA</LI> )+
</UL></HTML>
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
  - These create possible “squares”

- Match possible squares against earlier squares

- Generalize the wrapper by finding all contiguous repeated occurrences:
  - (`<LI><I>Title:</I>"#PCDATA</LI>`)+
Example Matching

- Wrapper (initially Page 1):
  01:  <HTML>
  02:  Books of:
  03:  <B>
  04:  John Smith  string mismatch (#PCDATA)
  05:  </B>
  06:  <UL>
  07:  <LI>
  08-10:  <!>Title:<!</I>
  11:  DB Primer  string mismatch (#PCDATA)
  12:  </LI>
  13:  </LI>
  14-16:  <!>Title:<!</I>
  17:  Comp. Sys.  string mismatch (#PCDATA)
  18:  </LI>
  19:  </UL>
  20:  </HTML>

  terminal tag search and
  square matching

  wrapping

- Sample (Page 2):
  01:  <HTML>
  02:  Books of:
  03:  <B>
  04:  Paul Jones
  05:  </B>
  06:  <IMG src=.../>
  07:  </UL>
  08:  <LI>
  09-11:  <!>Title:<!</I>
  12:  XML at Work
  13:  </LI>
  14:  </LI>
  15-17:  <!>Title:<!</I>
  18:  HTML Scripts
  19:  </LI>
  20:  </LI>
  21-23:  <!>Title:<!</I>
  24:  Javascript
  25:  </LI>
  26:  </UL>
  27:  </HTML>

- Wrapper after solving mismatches:

  <HTML>Books of:<B>#PCDATA</B>
  ( <IMG src=.../> )?
  <UL>
   ( <LI<!>Title:<</I>#PCDATA</LI> )+
  </UL> </HTML>
Recursive Example

<table>
<thead>
<tr>
<th>Wrapper (initially Page 1):</th>
<th>Sample (Page 2):</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-05: &lt;HTML&gt;Books of:&lt;B&gt;John Smith&lt;/B&gt;</td>
<td>01-05: &lt;HTML&gt;Books of:&lt;B&gt;Paul Jones&lt;/B&gt;</td>
</tr>
</tbody>
</table>
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
Automatic Extraction of Data from Lists and Tables in Web Sources by Lerman et al.
Borders Suggests

Your search results include the following suggested title(s).

1. Daughter of Fortune ~ Allende, Isabel / Peden, Margaret Sayers ~ Hardcover ~ 1999
2. Daughter of Fortune, Unabridged ~ Allende, Isabel / Peden, Margaret Sayers / Brown, Blair ~ Audio Cassette ~ 1999

Complete Results

- Re-sort your results by: Title | Author

In Stock/Available for Advance Order

- Daughter of Fortune ~ In stock - ships in 24 hours
  Allende, Isabel / Peden, Margaret Sayers ~ Hardcover ~ 1999
  Our Price: $13.00 ~ You Save: $13.00 (50%)

- Daughter of Fortune, Unabridged ~ In stock - ships in 24 hours
  Allende, Isabel / Peden, Margaret Sayers / Brown, Blair ~ Audio Cassette ~ 1999
  Our Price: $27.97
Exploiting Structure of Data Sources

Web pages containing lists and tables contain explicit and implicit structure

- Structure of pages
  - Pages generated from a template
  - Locate table or list, identify data

- Structure of data
  - Data in the same column/field is of the same type

- Structure of tables or lists
  - Sequence of columns repeated in every row/tuple
Our Approach

To extract data from lists or tables and assign it to rows and columns:

- Extract all data from lists or tables
  - Using information about page structure
- Identify columns
  - Classify data based on a set of content and layout features
- Identify rows
  - Each row = data tuple
  - Grammar induction on a sequence of class labels
Learning Page Structure

Web source generates pages from a template and fills it with results of a query

- Deducing page template
  - Given two or more example pages, deduce the template used to generate them
  - Find list or table on the page
    - Look for repeated sequences of HTML tags and text that mark column and row separators

- Use page structure information to extract all data from the table or list
Learning Data Structure

Data in the same columns is of the same type
e.g. author name, book price

Separate data into columns

❖ Each extract is described by a set of features
   ▲ Layout features
       separators = adjacent HTML tags
   ▲ Content features
       describes the format of data*

❖ Cluster extracts using AutoClass

* DataPro (Lerman & Minton, 2000)
AutoClass*

- AutoClass is a mixture model-based unsupervised classification algorithm
  - Finds the optimal number of classes, and
  - Finds the best assignment of data instances to classes

- AutoClass is flexible – allows data instances to be described by continuous, discrete, and binary values simultaneously

- Each data type will be in a different class

* AutoClass (Hanson, Stutz and Cheeseman, 1991)
Learn Table Structure

The sequence of columns in each row is a string of some regular language.
Learn the table grammar from examples.
- Table data elements are arranged sequentially.
- Attribute to each data element an AutoClass-assigned column label.
- Grammar induction on several tables.

Challenges:
- Missing columns
- AutoClass errors
Grammar Induction Algorithm*

- Start with a prefix tree that accepts all strings
- Merge nodes $\rightarrow$ minimum FSA
  - Merge two nodes $i$ and $j$ if
    - Incoming arcs correspond to the same symbol, and
    - At least one of the outgoing arcs from each node correspond to the same symbol
  - Determinize FSA
    - All descendants have at most one outgoing transition corresponding to a given symbol
- Extract all cycles from the FSA
- Partition the list into rows using cycles

*Adapted from ALERGIA (Carrasco & Oncina, 1994)
Extracting from a Web Page

(a) Daughter of Fortune
(b) In stock - ships in 24 hours
(c) Allende, Isabel
(d) Hardcover
(e) 1999
(f) Our Price
(g) 13.00
(h) You Save
(i) 13.00 (50%)

(a) Daughter of Fortune, Unabridged
(b) In stock - ships in 24 hours
(c) Allende, Isabel
(c) Peden, Margaret Sayers
(c) Brown, Blair
(d) Audio Cassette
(e) 1999
(f) Our Price
(g) 27.97
Grammar and Tuples

\[(abc*defghi?)^*\]

<table>
<thead>
<tr>
<th></th>
<th>Daughter of Fortune</th>
<th>Daughter of Fortune, Unabridged</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>In stock</strong></td>
<td>In stock</td>
<td>hours</td>
</tr>
<tr>
<td><strong>- ships in 24</strong></td>
<td>- ships in 24</td>
<td>hours</td>
</tr>
<tr>
<td><strong>Allende, Isabel</strong></td>
<td>Peden, Margaret</td>
<td>Peden, Margaret</td>
</tr>
<tr>
<td><strong>Sayers</strong></td>
<td>Sayers</td>
<td>Sayers</td>
</tr>
<tr>
<td><strong>Hard cover</strong></td>
<td>199 9</td>
<td>199 9</td>
</tr>
<tr>
<td><strong>Our Price</strong></td>
<td>13.0 0</td>
<td>27.9 7</td>
</tr>
<tr>
<td><strong>You Save</strong></td>
<td>13.0 0 (50 %)</td>
<td></td>
</tr>
<tr>
<td><strong>Audio Cassette</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>source</th>
<th>pages</th>
<th>extracts</th>
<th>columns</th>
<th>classes</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>airport</td>
<td>4</td>
<td>370</td>
<td>4</td>
<td>5</td>
<td>correct tuples</td>
</tr>
<tr>
<td>Blockbuster</td>
<td>4</td>
<td>663</td>
<td></td>
<td>11</td>
<td>no tuples extracted</td>
</tr>
<tr>
<td>Borders books</td>
<td>4</td>
<td>186</td>
<td>9</td>
<td>9</td>
<td>correct tuples</td>
</tr>
<tr>
<td>Cuisinenet restaurants</td>
<td>3</td>
<td>535</td>
<td>17</td>
<td>15</td>
<td>no tuples extracted</td>
</tr>
<tr>
<td>RestaurantRow</td>
<td>4</td>
<td>273</td>
<td>14</td>
<td>14</td>
<td>correct tuples</td>
</tr>
<tr>
<td>Yahoo people whitepages</td>
<td>3</td>
<td>126</td>
<td>8</td>
<td>8</td>
<td>correct tuples</td>
</tr>
<tr>
<td>Yahoo quote</td>
<td>3</td>
<td>259</td>
<td>13</td>
<td>13</td>
<td>18/20 tuples correct</td>
</tr>
<tr>
<td>Whitepages</td>
<td>3</td>
<td>73</td>
<td>9</td>
<td>5</td>
<td>correct tuples</td>
</tr>
<tr>
<td>MapQuest driving dirs</td>
<td>3</td>
<td>83</td>
<td>5</td>
<td>5</td>
<td>tuples begin in the middle of the rows</td>
</tr>
<tr>
<td>hotel</td>
<td>4</td>
<td>163</td>
<td>6</td>
<td>6</td>
<td>correct tuples</td>
</tr>
<tr>
<td>CitySearch restaurants</td>
<td>4</td>
<td>204</td>
<td>4</td>
<td>6</td>
<td>correct tuples</td>
</tr>
<tr>
<td>car rental</td>
<td>4</td>
<td>161</td>
<td>8+</td>
<td>9</td>
<td>correct tuples</td>
</tr>
<tr>
<td>boston restaurants</td>
<td>4</td>
<td>174</td>
<td>4+</td>
<td>6</td>
<td>correct tuples</td>
</tr>
</tbody>
</table>
Discussion

- Goal: Automatic labeling of data for wrapper creation
- Accurate extraction of data from semi-structured Web pages
- Initial results are encouraging: correct extraction from 10 out of 14 sources
- Future challenge: extracting from a single page containing a list or table
Using Grammatical Inference to Automate Information Extraction by Hong & Clark
Stochastic Grammars and Complexity

Let $G$ be a stochastic context-free grammar with productions and associated probabilities given by:

$$
X_1 \rightarrow w_{11} \mid w_{12} \mid \ldots \mid w_{1,m_1} \quad [P_{11}, P_{12}, \ldots, P_{1,m_1}]
$$

$$
X_2 \rightarrow w_{21} \mid w_{22} \mid \ldots \mid w_{2,m_2} \quad [P_{21}, P_{22}, \ldots, P_{2,m_2}]
$$

$$
\vdots
$$

$$
X_n \rightarrow w_{n1} \mid w_{n2} \mid \ldots \mid w_{n,m_n} \quad [P_{n1}, P_{n2}, \ldots, P_{n,m_n}]
$$

(1)

Following Cook et al. [7], we define the complexity $C(G)$ as:

$$
C(G) = \sum_{i=1}^{n} \sum_{j=1}^{m_n} -\log P_{ij} + c(w_{ij})
$$
Searching the Space of Grammars

Start with the exact grammar for a page:

Search in the space of grammar transformations minimizing complexity

- **Substitution**
  - \( X_1 \rightarrow ab, X_2 \rightarrow cd \)
  - Add \( Y \rightarrow s \) and replace \( s \) with \( Y \)

- **Disjunction**
  - \( X \rightarrow vs \mid ws \)
  - Add \( Y \rightarrow v \mid w \) and replace rule with \( X \rightarrow Ys \)

- **Normalization**
  - Merge redundant alternatives \( X \rightarrow s \mid s \)
  - Drop productions that are inaccessible
Example Search

Start with language of balanced parens up to length 6

\[ X \rightarrow () | ()() | ()() | ()()() | ()(()) | (()()) | (()()) | (((()))) \]

[0.5,0.125,0.125,0.0625,0.03125,0.03125,0.0625,0.0625]

Replace () with X

\[ X \rightarrow () | XX | (X) | XXX | X(X) | (X)X | (XX) | ((X)) \]

Replace (X) with X

\[ X \rightarrow () | XX | (X) | XXX | XX | XX | (XX) | (X) \]

Merge redundant alternatives, sum associated probabilities

\[ X \rightarrow () | XX | (X) | XXX | (XX) \]

Replace XX with X

\[ X \rightarrow () | XX | (X) | XX | (X) \]

Merge

\[ X \rightarrow () | XX | (X) \]

[0.5,0.25,0.25]
Example Web Page

Southwest London

**New Kings Road, SW6**
£120,000
A quiet and secluded studio flat with a garden situated at the rear of this Victorian converted building in the heart of Fulham. Contact: 020-7222-3322.

**Addison Gardens, SW14**
£124,000
A particularly quiet and conveniently located studio flat with the great benefit of direct access to wonderful 'hidden' communal gardens. Contact: 020 7431 1020.

**Cheval Court, SW15**
£120,000
A refurbished first floor studio apartment with southerly views over gardens in a purpose built block with off street parking. Contact: 020–8879–7922.

**Jeffreys Road, SW4**
£130,000
A good sized lower ground floor flat which would be ideal for a first time buyer. The flat has a spacious sitting room which further benefits from having direct access to its own west facing patio. Jeffreys Road runs between Larkhall Lane and Clapham Road. The closest tube station is Stockwell with transport links to Victoria, Patio. 1 reception, 1 bedroom, 1 bathroom. Leasehold. Contact: 0800–919–308.

April 2001
Grammar Inference

- Convert pages into the alphabet:
  - \{HTML tag types\} U \{text\}
  - Discard HTML attributes

- Given
  - `<hr><a href=mailto:sales@a.com>Company A</a>`
  - `<hr><a href=mailto:help@b.com>Company B</a>`

- Maps to:
  - `hr a text /a`
Grammar Inference (cont.)

<html><head>
<title>Listings in Southwest London</title>
</head>
<body>
<h1>Southwest London</h1>
<table border=1 width=100%>
<tr><td><b>New Kings Road, SW6</b><br>
£120,000<br>
A quiet and secluded studio flat with a garden situated at the rear of this Victorian converted building in the heart of Fulham. Contact: 020-722-3322.
</td></tr>
</table>

S → html head title text /title /head body h1 text /h1
   table T /table p address text /address /body /html
T → TT | tr td U b text /b br text br text /td /tr
U → e | img br.
Domain-Specific Phase

- Manually write a set of domain specific rules to extract the fields:
  - Number price = (£|£|L){\d} where \d = number
  - String telephone = {[0-9]+-[0-9]+-[0-9]+}
  - Boolean garden = garden|yard
- Rules are applied at each level to find a nonterminal where there are no duplicates and they are not empty
Wrapper Generation Phase

- Apply domain rules to the appropriate text
- Can write additional rules to further refine the text:
  - String address = 1 : {.*}, ;; extract string to first comma in first text field
  - String description = 3 ;; third text field
Discussion

- Simple and elegant approach
- Doesn’t solve the labeling as claimed
  - Need to write domain specific rules to find the data
  - Need to write regular expressions to further decompose text fields
- No results on coverage of the approach
- In general, it is easy to come up with an algorithm that works on a few examples
- Difficult to achieve widespread coverage, which is an advantage of the learning approach