A Flexible Learning System for Wrapping Tables and Lists in HTML Documents

Cohen, Hurst, Jensen

WWW2002

Outline

- Wrapping and learning wrappers for HTML pages.
- Appropriate document representation for automated wrapper learning .
- A learning system for wrapper induction.
- Extracting from tables.
- Experiments.
- Conclusions.

Summary

- Repeated relation instances have similar or identical layout (efficient communication), though not necessarily identical encoding.
- The syntax of the encoding may not describe the syntax of the document element (better views of the document required).
- Instructing a learning system by examples requires capture of the users intentions (more examples reduce ambiguity, shared document idiom reduces examples).

WheezeBong.com:
Contact Info

Currently we have offices in three locations (original locations in bold):
•***Pittsburgh***, PA
•***Provo***, UT
•Honolulu, HI

```
<b><i>Pittsburgh</b></i>

<i><b>Provo</i></b>
```

WhizBang!

Wrapping and Learning Wrappers for HTML Pages.

- Aim – to be able to access structured (and relational) information on web pages programmatically, as if accessing a database.
- Input – some HTML and a query expression, Output – a set of relations.
- Simple solution – hand crafted scripts developed for each page; the query is implicit in the script.
- Ideal solution – machine generated programs produced with a minimum of human instruction; the query is elicited from the trainer.

Design Challenges

- Be future proof (tune using bias).
- Know when applicable and when broken.
- HTML –
  - There are many ways to express a particular layout (e.g. tabular layout can be achieved in a variety of ways).
  - Consistency in encoding is not required for visual consistency (e.g. ordering of text modifying tags).
- Inform the learning process in an intuitive and consistent manner.

WheezeBong.com:
Contact Info

Currently we have offices in two locations:
•Pittsburgh, PA
•Provo, UT
•Honolulu, HI

WhizBang!

Appropriate Document Representation

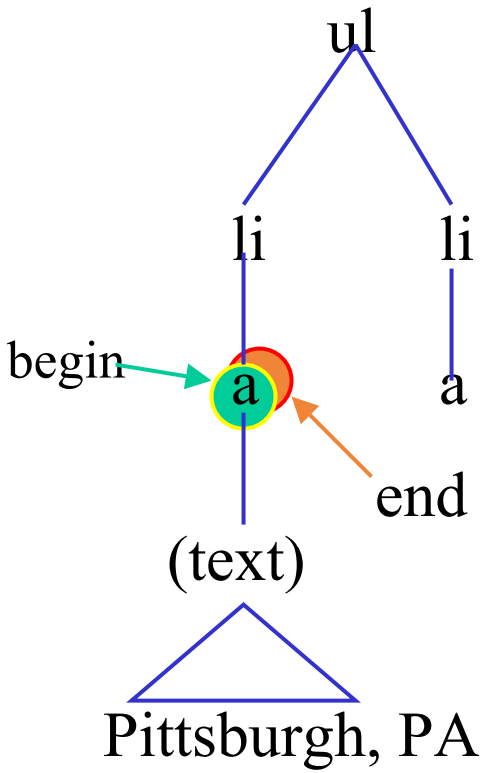
- The DOM is an obvious starting point.
- Other appropriate views of the document
  - Token sequence.
  - The result of certain normalizations of the DOM.
  - A representation of the layout produced by the HTML when rendered in a browser.
  - …
- Each document view is valid for learning certain types of concepts/document regularities.
- A learning system should be able to make use of multiple views and be extensible as new views are made available/demanded by new problems.

A Learning System for Wrapper Induction

- Premise : A wrapper learning system needs careful engineering
  - 6 hand-crafted languages in WIEN (Kushmeric AIJ2000)
  - 13 ordering heuristics in STALKER (Muslea et al AA1999)
- Approach : Architecture that facilitates hand-tuning the “bias” of the learner.
  - Bias is an ordered set of “builders”
  - Builders are simple “micor-learners”
  - A single master algorithm co-ordinates learning.

A Learning System for Wrapper Induction

- A span is a subset of the document defined by a start and end point in the DOM.



A Learning System for Wrapper Induction

- A predicate is a binary relation on spans : *p(s1, s2)* means that *s2* is extracted from *s1*.
- Membership can be tested:
  - Given (s1, s2) is p(s1, s2) true?
- Predicates can be executed:
  - EXECUTE(p, s1) is the set of s2 for which p(s1, s2) is true.
- Example:
  - p(s1, s2) iff s2 are the tokens below an `li` in s1
  - EXECUTE(p, s1) extracts:
    - Pittsburgh, PA
    - Provo, UT
    - Honolulu, HI
  - p(s1, s2) iff s2 starts with 'P'…

WheezeBong.com:
Contact Info

Currently we have offices in three locations:
•Pittsburgh, PA
•Provo, UT
•Honolulu, HI

WhizBang!

A Learning System for Wrapper Induction

- Predicates are implemented by simple languages.
- $L_{bracket}$: p is defined by a pair of strings *[l, r]* and $p_{[l, r]}(s1, s2)$, is true iff s2 is preceded by l and followed by r.
- EXECUTE($p_{[for, Induction]}$, s1) = {“Wrapper”}
- $L_{tagpath}$: p is defined by a sequence of tags {t0, t1, …, tk} and $p_{\{t0, t1, \ldots, tk\}}$ (s1, s2) is true iff s1 and s2 correspond to DOM nodes and s2 is reached from s1 by following a path ending in t0, t1, …tk.

A Learning System for Wrapper Induction

- For each language L there is a builder B_L which implements a few simple operations:
  - LGG(positive examples of p(s1, s2)): least general p in L that covers all the positive examples.
    - For $L_{bracket}$, longest common prefix and suffix of the examples.
  - REFINE(p, examples): a set of p's that cover some but not all of the examples.
    - For $L_{tagpath}$, extend the path with one additional tag that appears in the examples.

The Learning Algorithm

- Inputs
  - An ordered set of builders (order defines bias).
  - Positive examples of the predicates to be learned (negative examples can be inferred).
- Algorithm
  - Compute the LGG of positive examples for each builder.
  - If any LGG is consistent with the implicit negative data, then return it (break ties in favour of earlier builders).
  - Otherwise, execute the best LGG to get explicit negative examples, then apply a FOIL-like learning algorithm using LGG and REFINE to create “features”.

Extraction from Tables

Actresses			
Lucy	Lawless	Images	Links
Angelina	Jolie	Images	Links
…	…	…	…
Singers			
Madonna		Images	Links
Brittany	Shakespeare	Images	Links
…	…	…	…

- How can we express “links to pages about singers”?

Extraction from Tables

- Classify HTML TABLE nodes as true tables or non tables.
- Render each table into a logical representation using an extended version of the HTML table rendering algorithm.
- Record the true logical table position of each cell (not limited to TD/TH nodes).
- Determine the geometric role of each cell – {cut-in, normal}.

Extraction from Tables : Deriving an Abstract Model

- Start with the standard table rendering algorithm.
- Attempt to “inline” complex geometric blocks:
  - Nested tables
  - Lists
  - BR separated lines
  - Plain text

Tel:	123 456
Fax:	123 567

```
<table>
  <tr>
    <td>Tel:</td><td>123 456</td>
  </tr>
  <tr>
    <td>Fax:</td><td>123 567</td>
  <tr>
</table>
```

```
<table>
  <tr>
    <td>Tel:<BR>Fax:</td>
  </tr>
  <tr>
    <td>123 456<BR>123 567</td>
  </tr>
</table>
```

WhizBang!

Extraction from Tables

Actresses [cut-in cell] {0,0 – 3, 0}			
Lucy	Lawless	Images	Links
Angelina	Jolie	Images	Links
…	…	…	…
Singers [cut-in cell] {0,4 – 3,4}			
Madonna		Images	Links
Brittany	Shakespeare	Images	Links
…	…	…	…

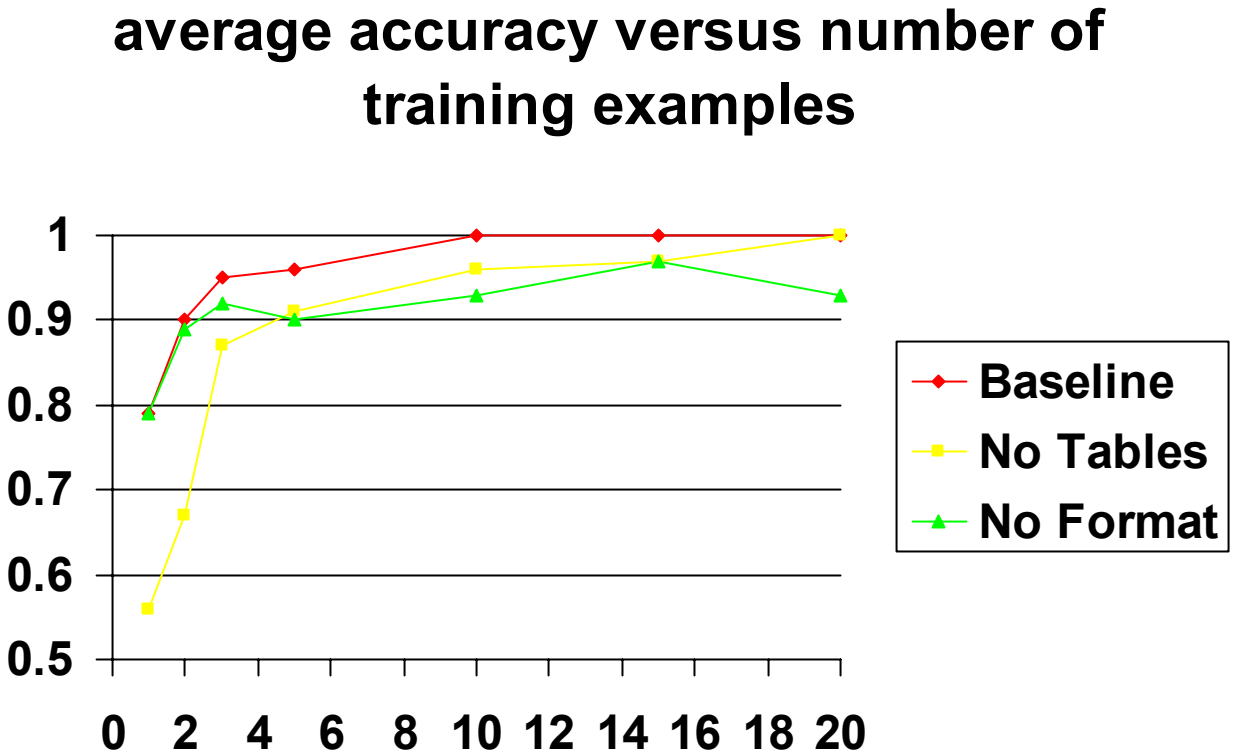
- Element name, words in last cut-in. (e.g. table cells where the last cut-in was "singers")
- Tagpath builder with awareness of co-ordinates. (e.g. table cells with y = 5).

WhizBang!

Experiments : real world wrapping problems

Problem	WL2	Problem	WL2
JOB1	3	CLASS1	1
JOB2	1	CLASS2	3
JOB3	1	CLASS3	3
JOB4	2	CLASS4	3
JOB5	2	CLASS5	6
JOB6	9	CLASS6	3
JOB7	4		
Median	2	Median	3

Experiments : real world wrapping problems



WhizBang!

Conclusions

- Wrapper learners need tuning. Structuring the bias space provides a principled approach to tuning.
- Builders let one mix generalization strategies based on different views of the document:
  - DOM
  - Token sequence
  - Table model
  - …
- Multiple views of the document improve performance by allowing builders to express concepts in appropriate language.
- (*The TABLE element is a good example of a specification that failed – cf Keynote talk, TBL*).