Automatic Wrapper Generation

Craig Knoblock
University of Southern California

Automatic Data Extraction

- Learning wrappers for data extraction
  - Users specifies the schema of the information source
    - Single tuple
    - Nested type
      - List of tuples or nested types
  - Labels data on several example HTML pages
    - Tedious, especially for lists
- Goal of Automatic Data Extraction is to eliminate user intervention
  - Automatically generate wrappers to extract data from Web pages
  - *cf* Information Extraction: Automatically extracts data from text

Overview

- Methods for automatic wrapper generation 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*

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

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
    - Limited learning ability
- User needs to provide a set of pages of the same type

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

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

RoadRunner Overview

- Automatically generates a wrapper from large web pages
  - Pages of the same *class*
    - No dynamic content from javascript, ajax, etc
- Infers source schema
  - Supports nested structures and lists
  - Extracts data from pages
- Efficient approach to large, complex pages with regular structure

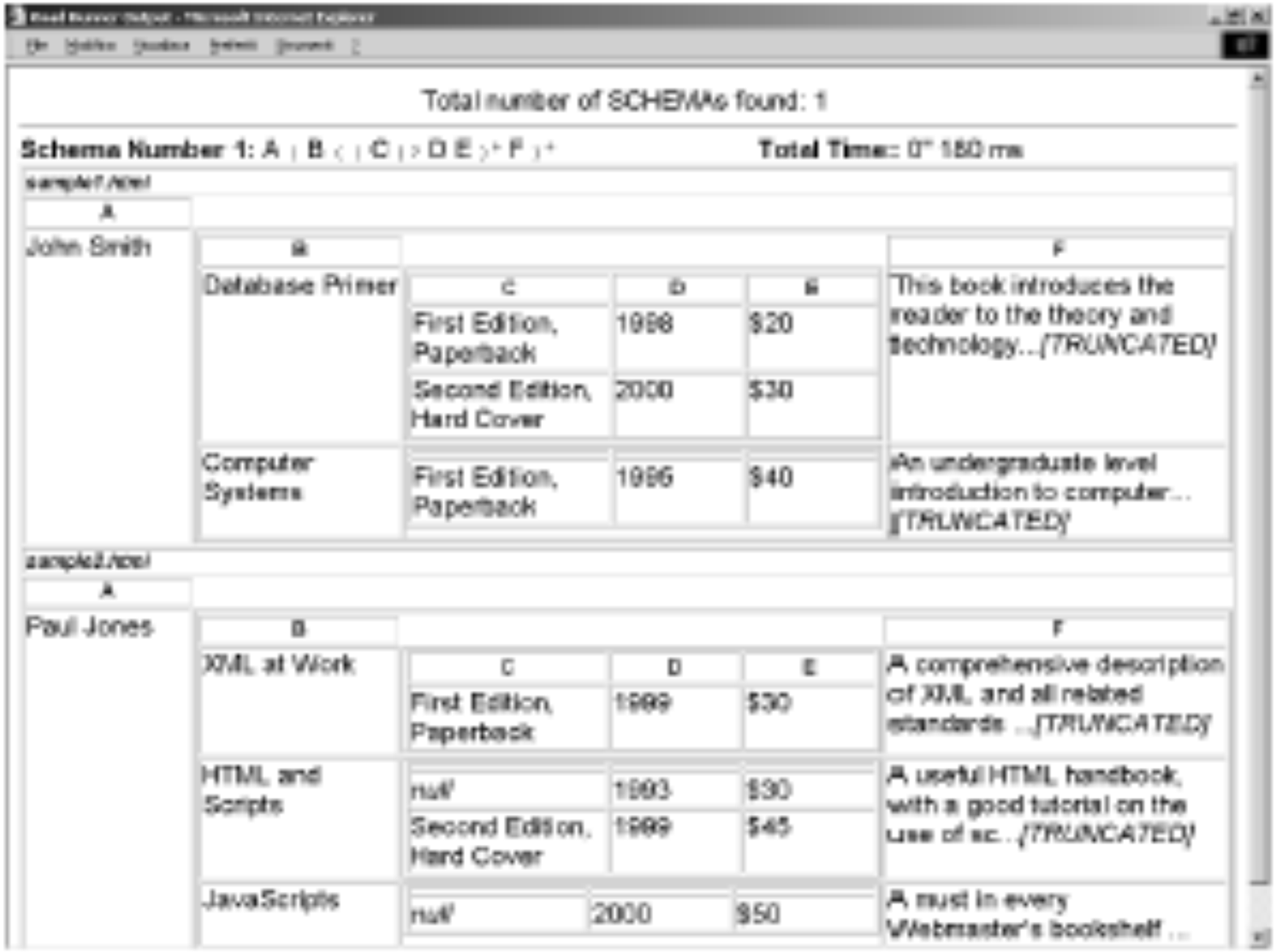
Example Pages

- Compares two pages at a time to find similarities and differences
- Infers nested structure (schema) of page
- Extracts fields

*http://www.csbooks.com/***author?John+Smith**

*http://www.csbooks.com/***author?Paul+Jones**

Extracted Result



Total number of SCHEMAs found: 1

Schema Number 1: A ‹ 1 › B ‹ 1 › C 1 › D E ›* F ›* **Total Time:** 0" 180 ms

sample1.html

A					
John Smith	B				F
	Database Primer	C	D	E	This book introduces the reader to the theory and technology...[TRUNCATED]
		First Edition, Paperback	1998	$20	
		Second Edition, Hard Cover	2000	$30	
	Computer Systems	First Edition, Paperback	1996	$40	An undergraduate level introduction to computer... [TRUNCATED]

sample2.html

A					
Paul Jones	B				F
	XML at Work	C	D	E	A comprehensive description of XML and all related standards ...[TRUNCATED]
		First Edition, Paperback	1999	$30	
	HTML and Scripts	null	1993	$30	A useful HTML handbook, with a good tutorial on the use of sc...[TRUNCATED]
		Second Edition, Hard Cover	1999	$45	
	JavaScripts	null	2000	$50	A must in every Webmaster's bookshelf ...

Union-Free Regular Expression (UFRE)

- Web page structure can be represented as *Union-Free Regular Expression* (UFRE)
  - UFRE is Regular Expressions without *disjunctions*
  - If *a* and *b* are UFRE, then the following are also UFREs
    - a.b
    - (a)+
    - (a)?

Union-Free Regular Expression (UFRE)

- Web page structure can be represented as *Union-Free Regular Expression* (UFRE)
  - UFRE is Regular Expressions without *disjunctions*
  - If *a* and *b* are UFRE, then the following are also UFREs
    - a.b → string fields
    - (a)+ → lists (possibly nested)
    - (a)? → optional fields
  - Strong assumption that usually holds

Approach

- Given a set of example pages
- Generate the *Union-Free Regular Expression* which contains example pages
- 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

Matching/Mismatches

Given a set of pages of the same type

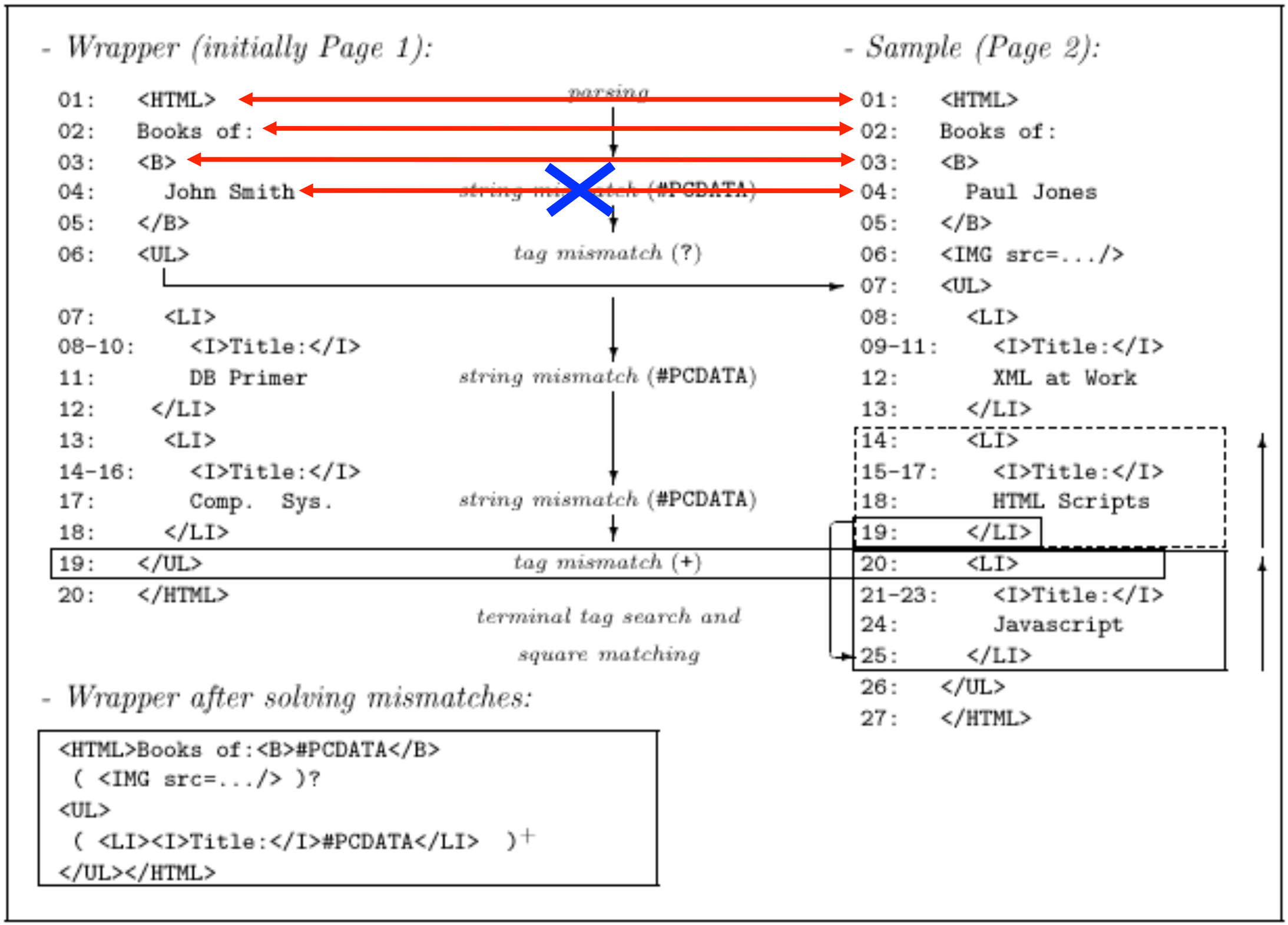
- Take the first page to be the *wrapper* (UFRE)
- Match each successive sample page against the wrapper
- *Mismatches* result in generalizations of wrapper
  - String mismatches
  - Tag mismatches

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 wrapper
  - String mismatches
    - Discover fields
  - Tag mismatches
    - Discover optional fields
    - Discover iterators

Example Matching



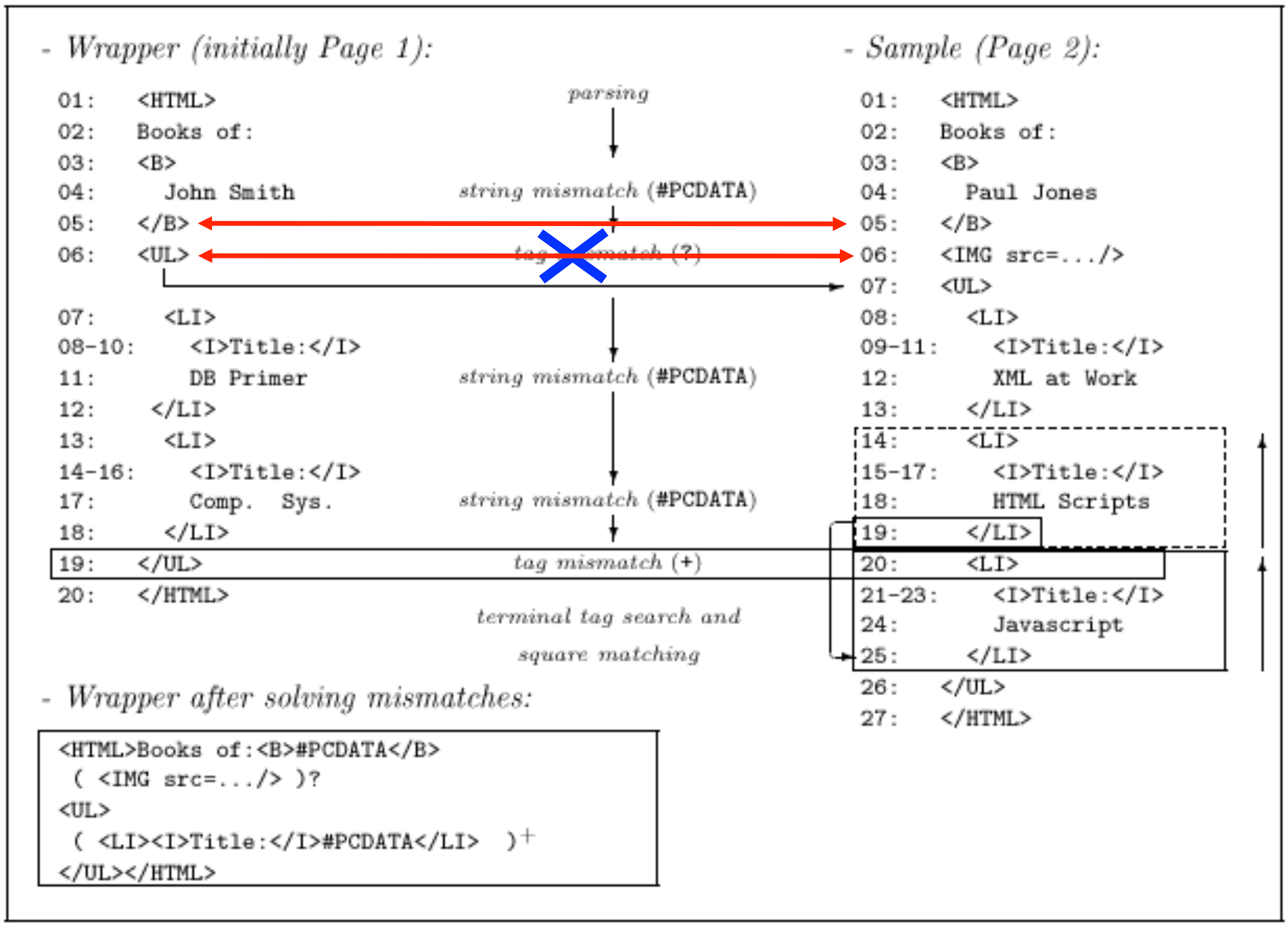
String Mismatches: Discovering Fields

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

<HTML>Books of: <B>John Smith

→ <HTML> Books of: <B>#PCDATA

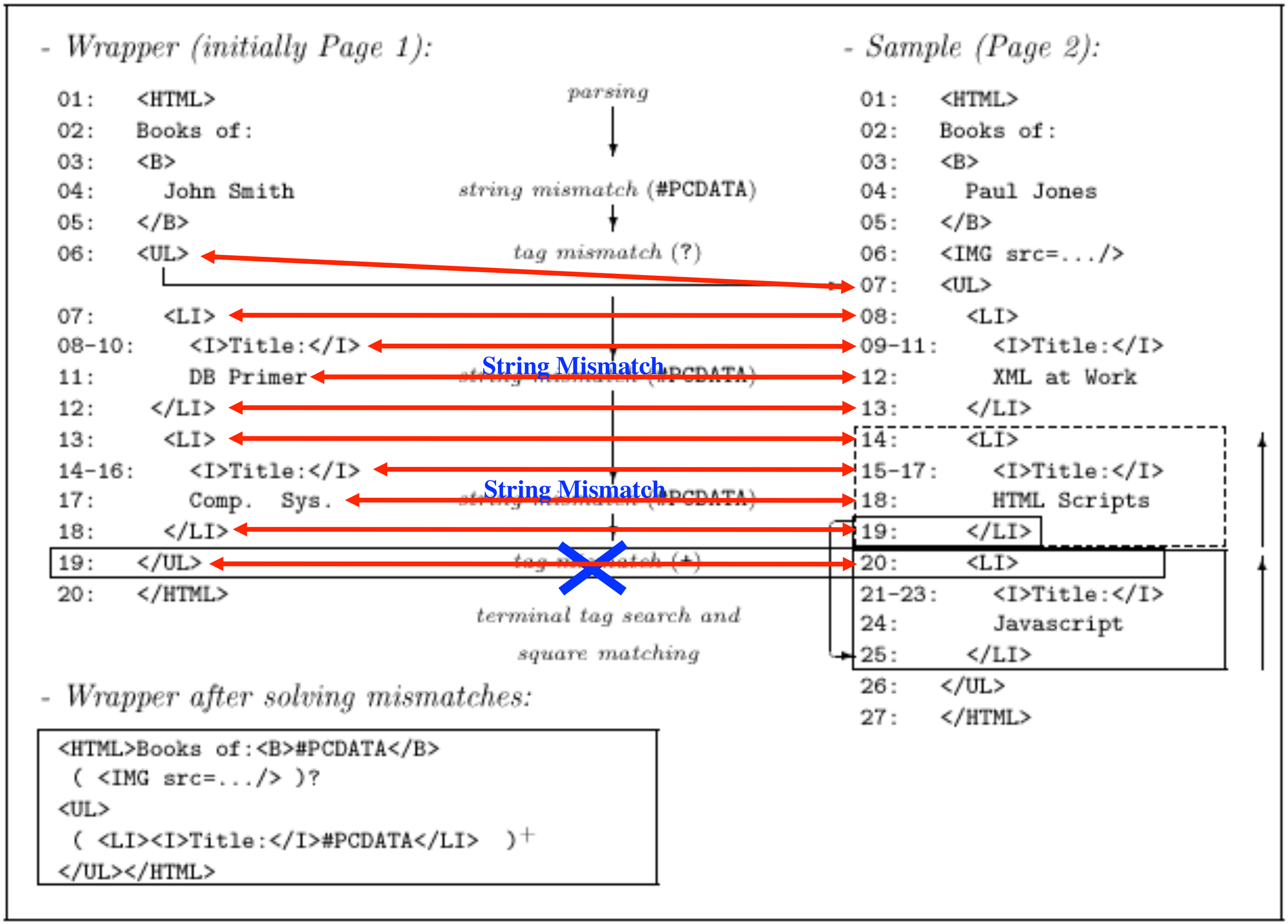
Example Matching



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:
  - ( )?

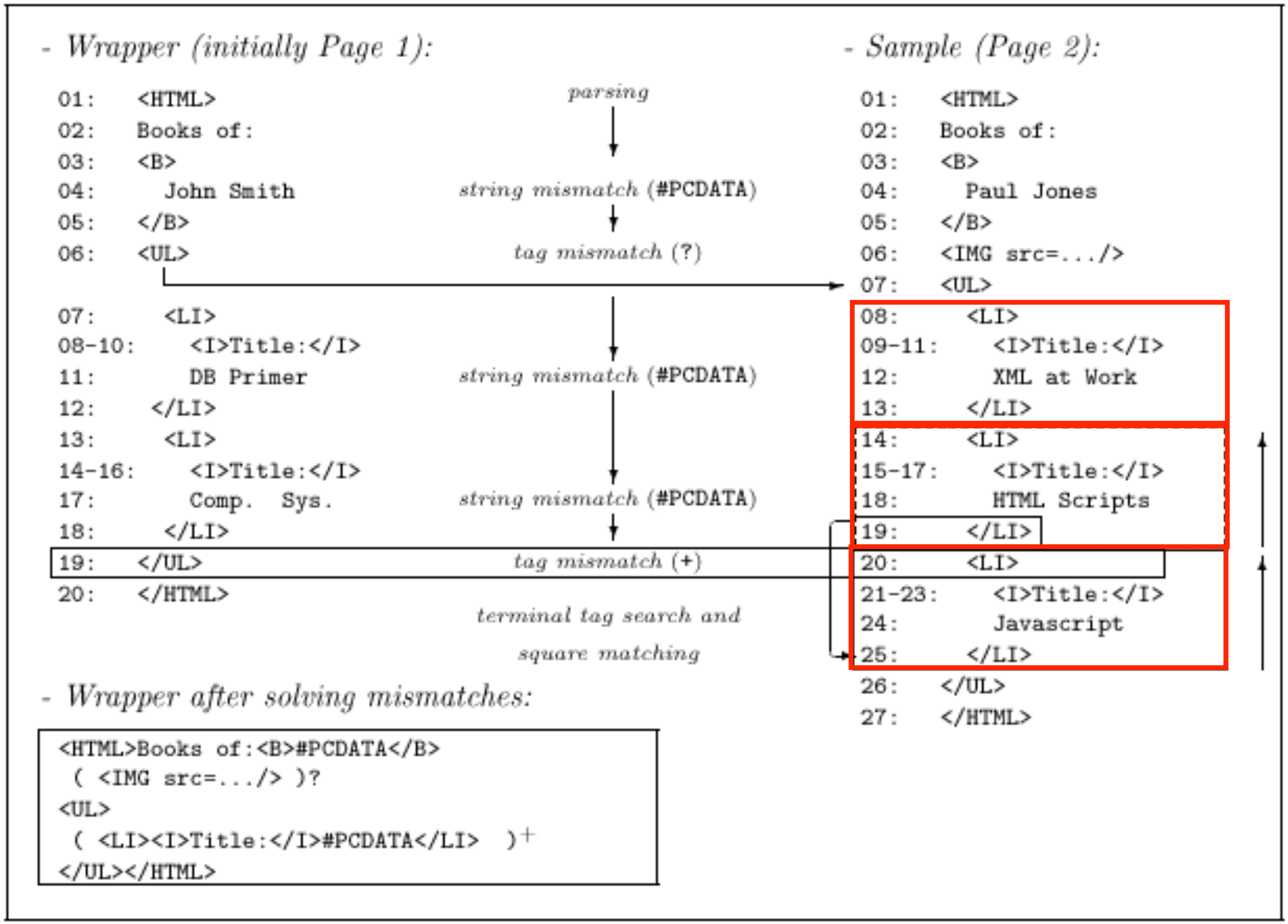
Example Matching



Tag Mismatches: Discovering Iterators

- Assume mismatch is caused by repeated elements in a list
  - End of the list corresponds to last matching token: </LI>
  - Beginning of list corresponds to one of the mismatched tokens: <LI> or </UL>
    - 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>)+

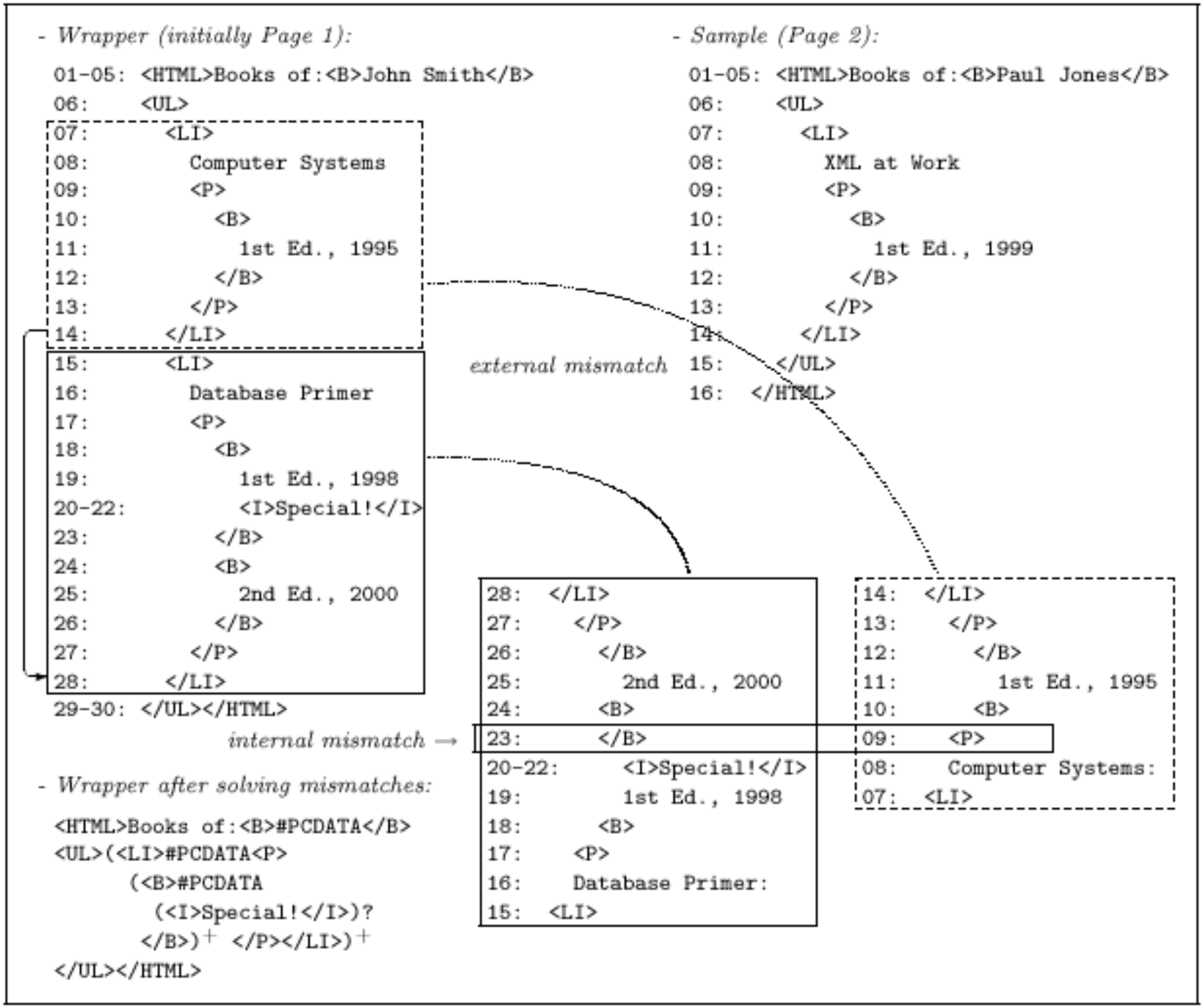
Example Matching



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>

Recursive Example



Discussion

- Assumptions:
  - Pages are well-structured
  - Structure can be modeled by UFRE (no 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

AutoFeed: An Unsupervised Learning System for Generating Webfeeds

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

Homepage

0	AutoFeedWeather

StateList

0	0
0	1

States

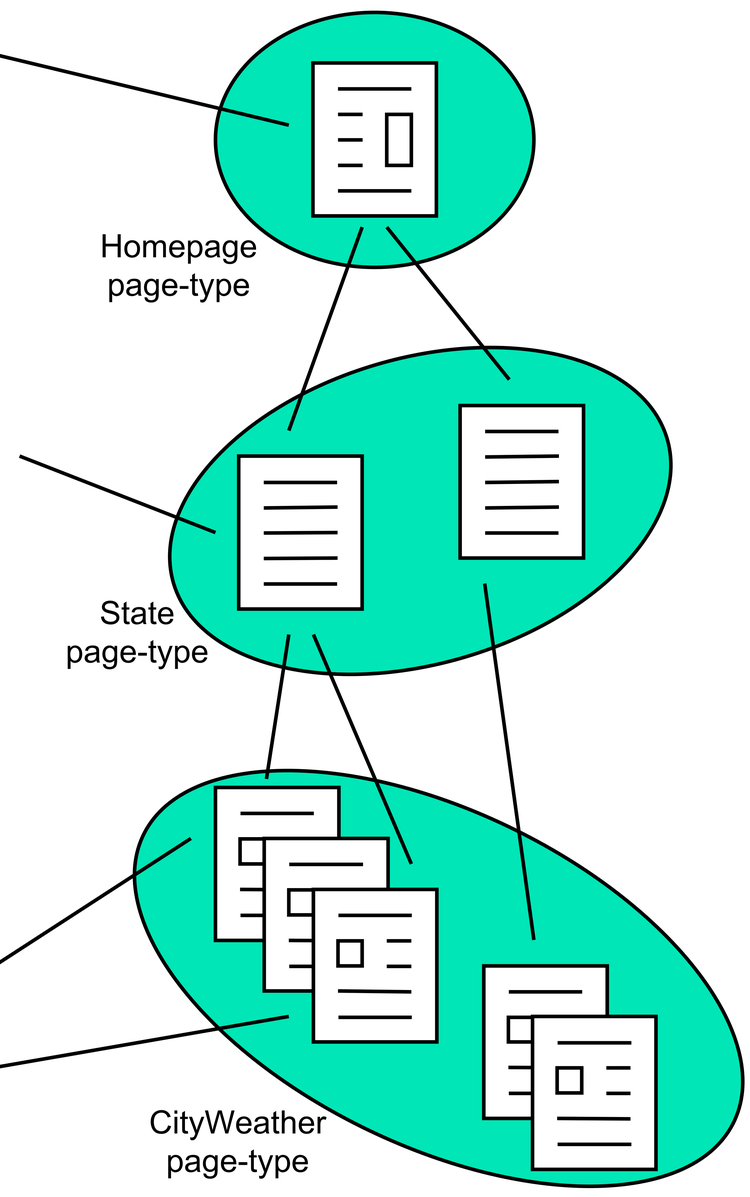
0	California	CA
1	Pennyslvania	PA

CityList

0	0
0	1
0	2
1	3
1	4

CityWeather

0	Los Angeles	70
1	San Francisco	65
2	San Diego	75
3	Pittsburgh	50
4	Philadelphia	55



Chicken ‘n Egg Problem

- If we could pick out a set of pages of the same class, we could learn their grammar and extract data
- But how do we pick the right pages in the first place?

Homepage

0	AutoFeedWeather

StateList

0	0
0	1

States

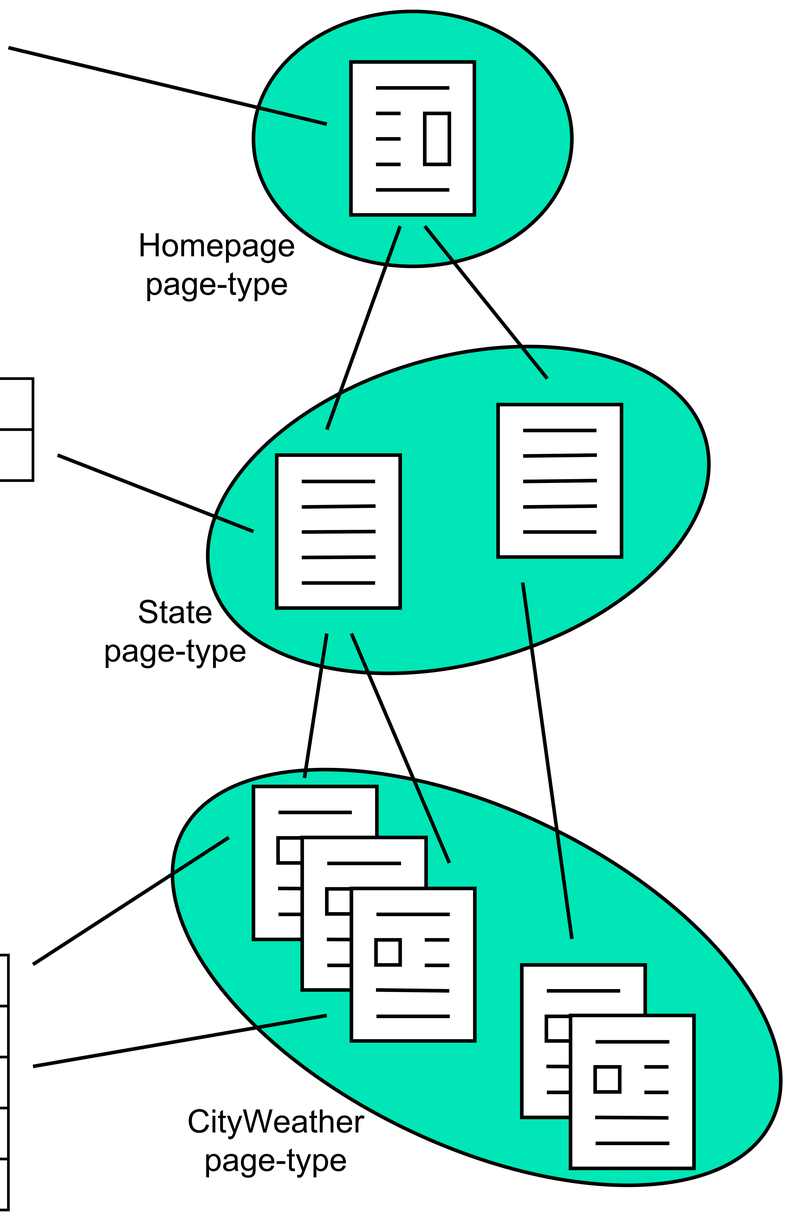
0	California	CA
1	Pennyslvania	PA

CityList

0	0
0	1
0	2
1	3
1	4

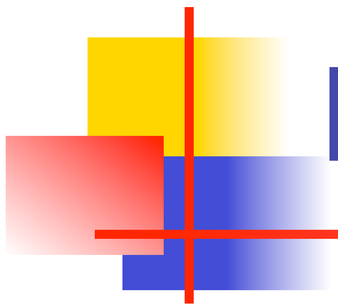
CityWeather

0	Los Angeles	70
1	San Francisco	65
2	San Diego	75
3	Pittsburgh	50
4	Philadelphia	55



Overview of Approach

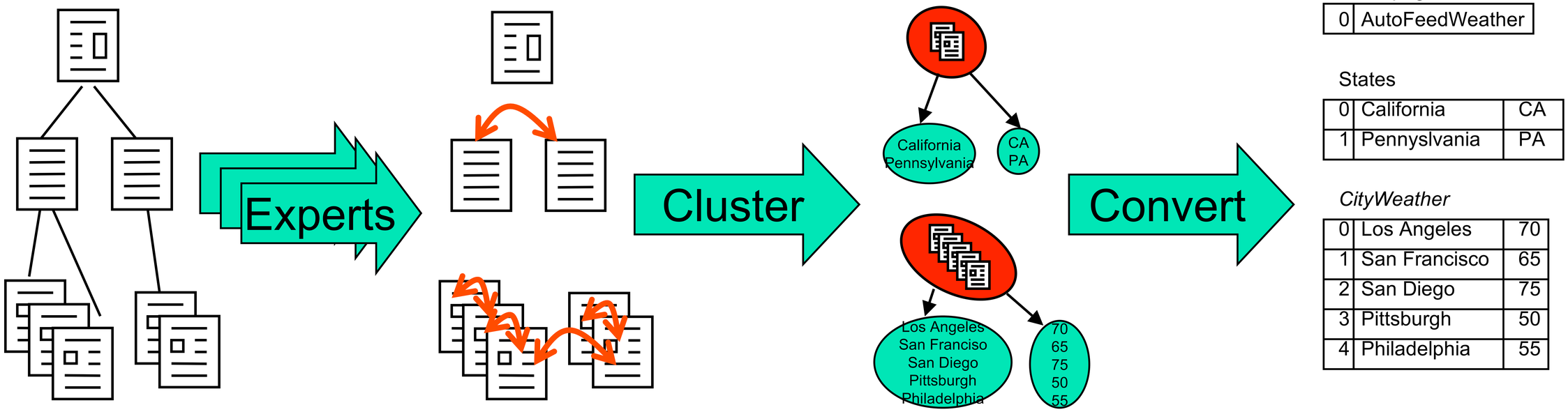
- Many types of structure within site
  - Graph structure of site's links
  - URL naming scheme
  - Content of pages
  - HTML structure within page types, …
- **Experts** focus on individual structures and output discoveries as *hints*
  - Experts are heterogeneous
  - Probabilistically combine experts (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
- Simultaneously cluster pages and data using hints

Overview



Homepage

0	AutoFeedWeather

States

0	California	CA
1	Pennyslvania	PA

CityWeather

0	Los Angeles	70
1	San Francisco	65
2	San Diego	75
3	Pittsburgh	50
4	Philadelphia	55

Web Site

Page & Data Hints

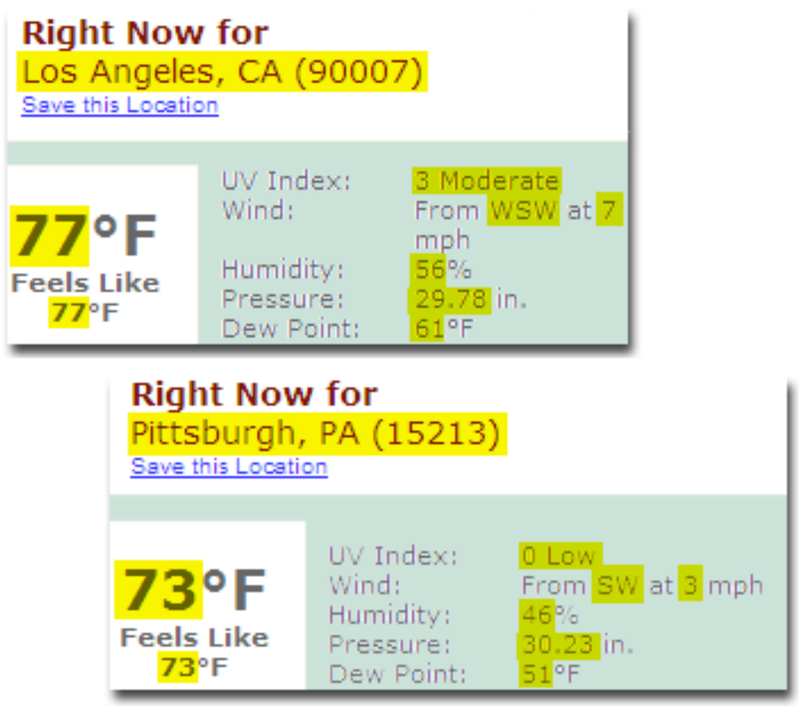
Page & Data Clusters

Relational Tables

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

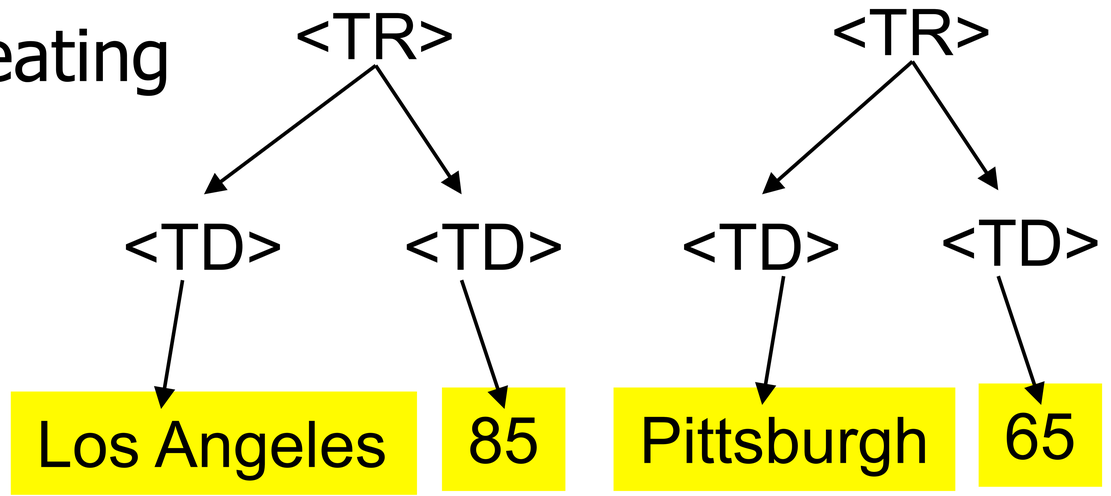
- **Page templates**
  - Similar pages contain common sequences of substrings



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



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 found *probabilistically* using hints
    - Probabilistic representation gives flexible framework for combining possibly conflicting hints

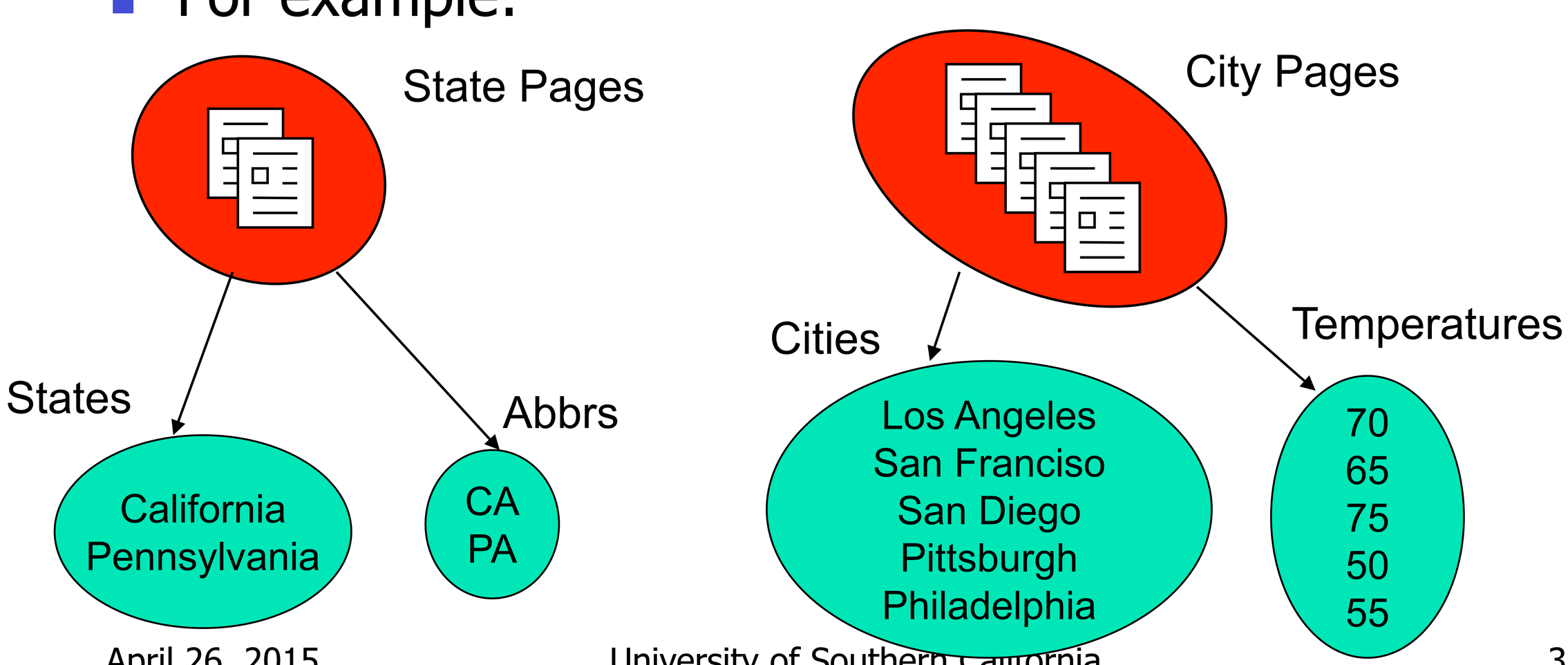
Probabilistic Evaluation

- Find clustering that maximizes probability of observing hints:

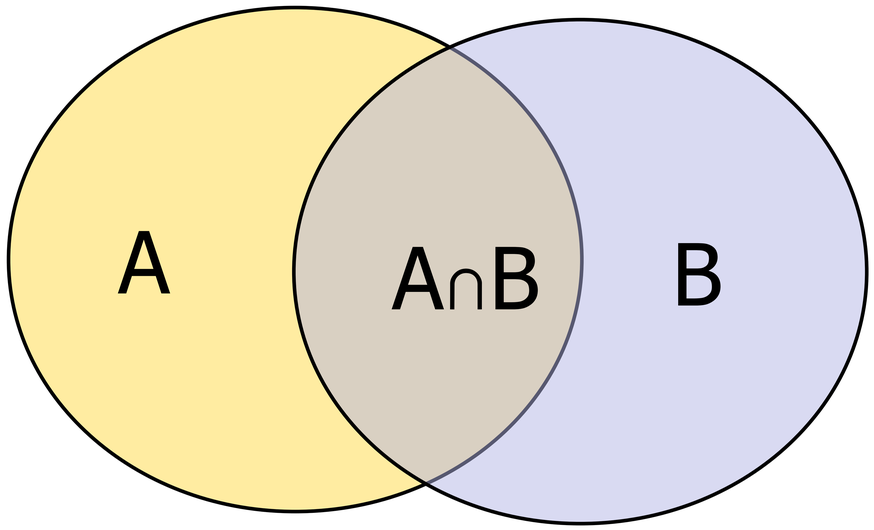
$$P(clustering|hints) = P(hints|clustering)*P(clustering)/P(hints)$$

Clustering

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



Evaluation



Retrieved Results

Relevant Results

Precision = A∩B/A

Recall = A∩B/B

Evaluation – another view

	Actual Positive	**Actual Negative**
Retrieved Positive	TP	FP
Retrieved Negative	FN	TN

Precision = TP/(TP+FP)

Recall = TP/(TP+FN)

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
Retail	mfg	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%
Journals	authors	1173	1197	1212	98%	97%
	title	1173	1188	1212	99%	97%
	URL	528	528	1212	100%	44%
Jobs	position	1391	1391	1453	100%	96%
	req. id	1278	1278	1422	100%	90%
	location	1302	1302	1445	100%	90%

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