Chapter 9: Structured Data Extraction

Supervised and unsupervised wrapper generation

Road map

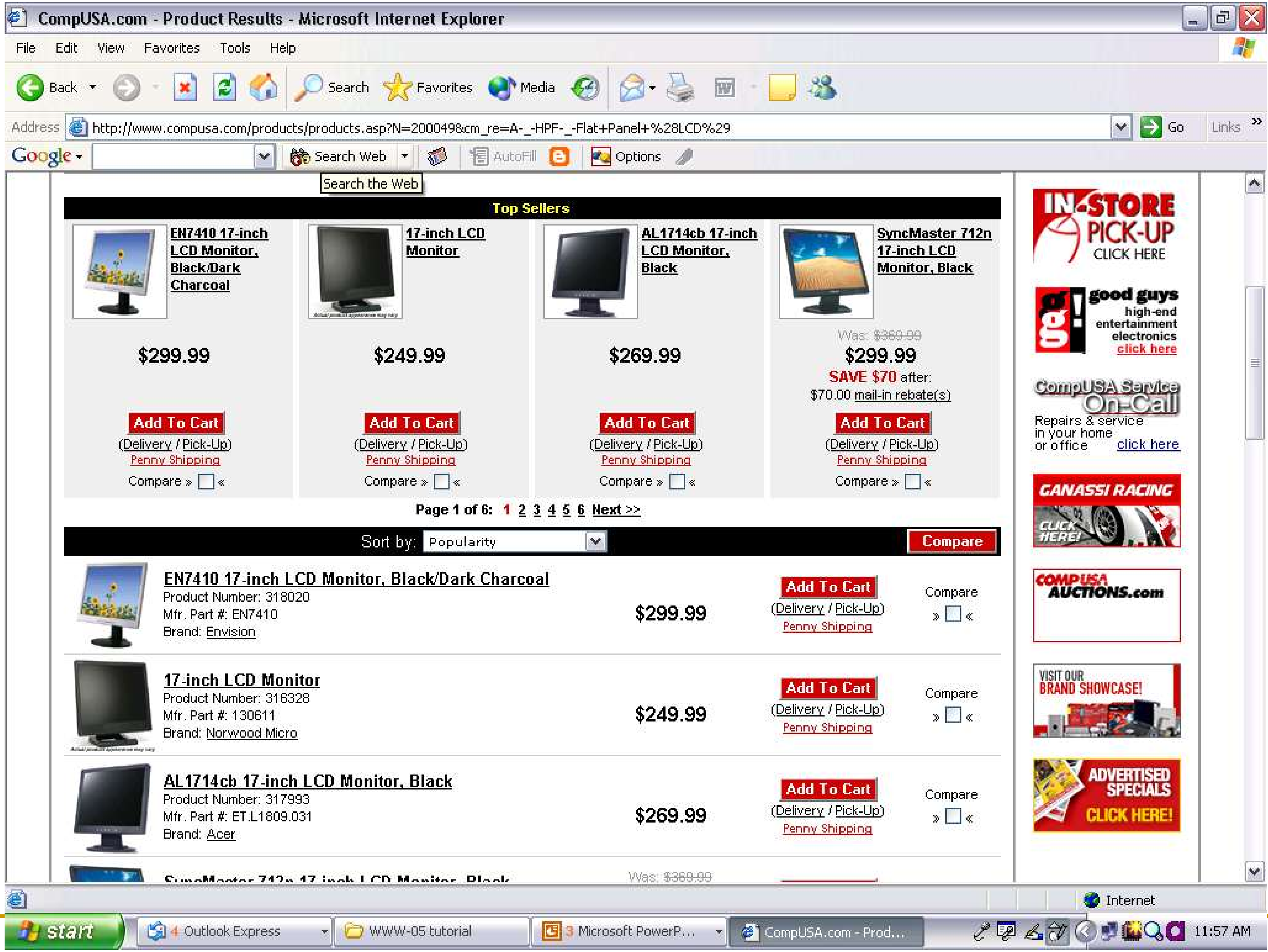
- **Introduction**
- Data Model and HTML encoding
- Wrapper induction
- Automatic Wrapper Generation: Two Problems
- String Matching and Tree Matching
- Multiple Alignments
- Building DOM Trees
- Extraction Given a List Page: Flat Data Records
- Extraction Given a List Page: Nested Data Records
- Extraction Given Multiple Pages
- Summary

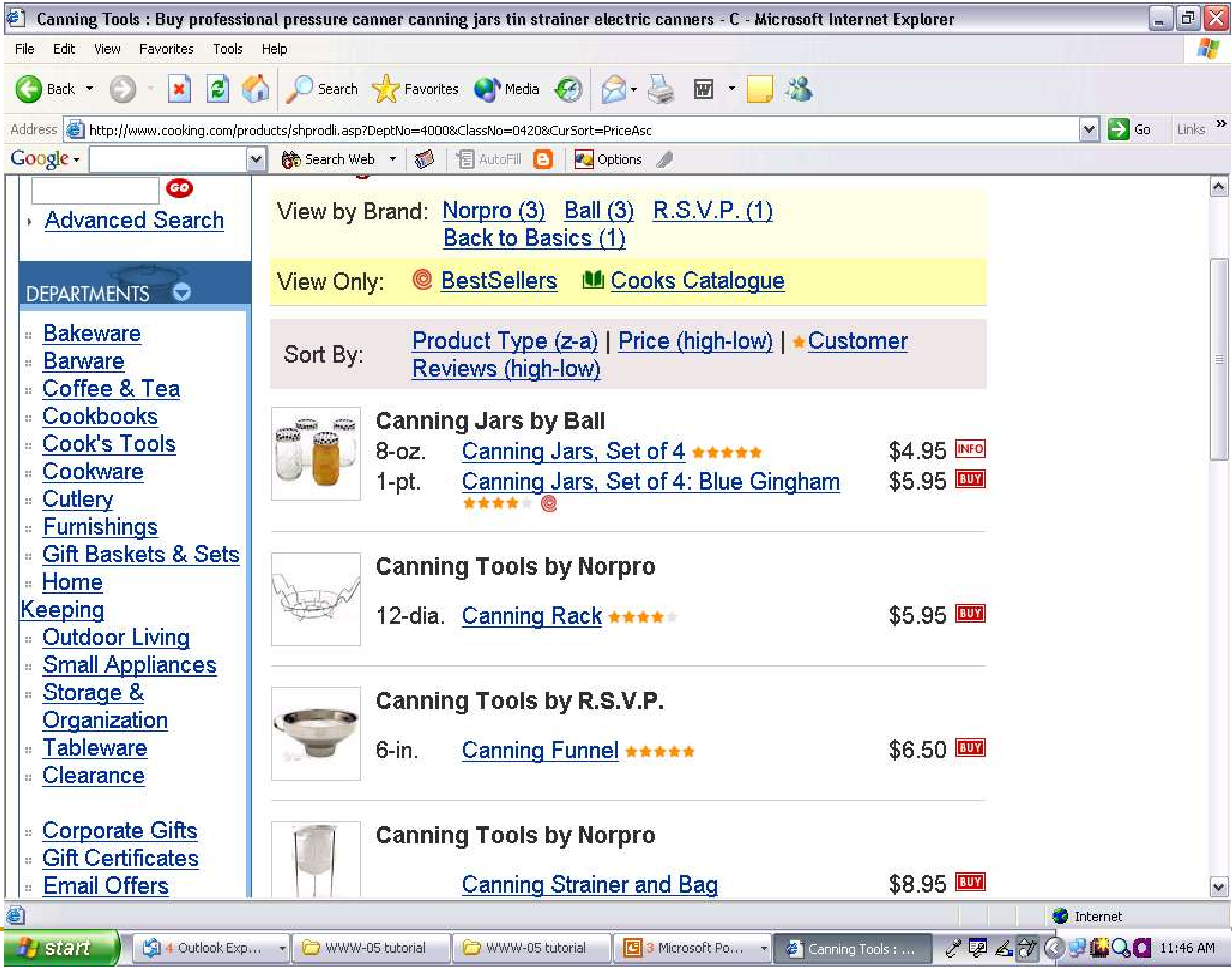
Introduction

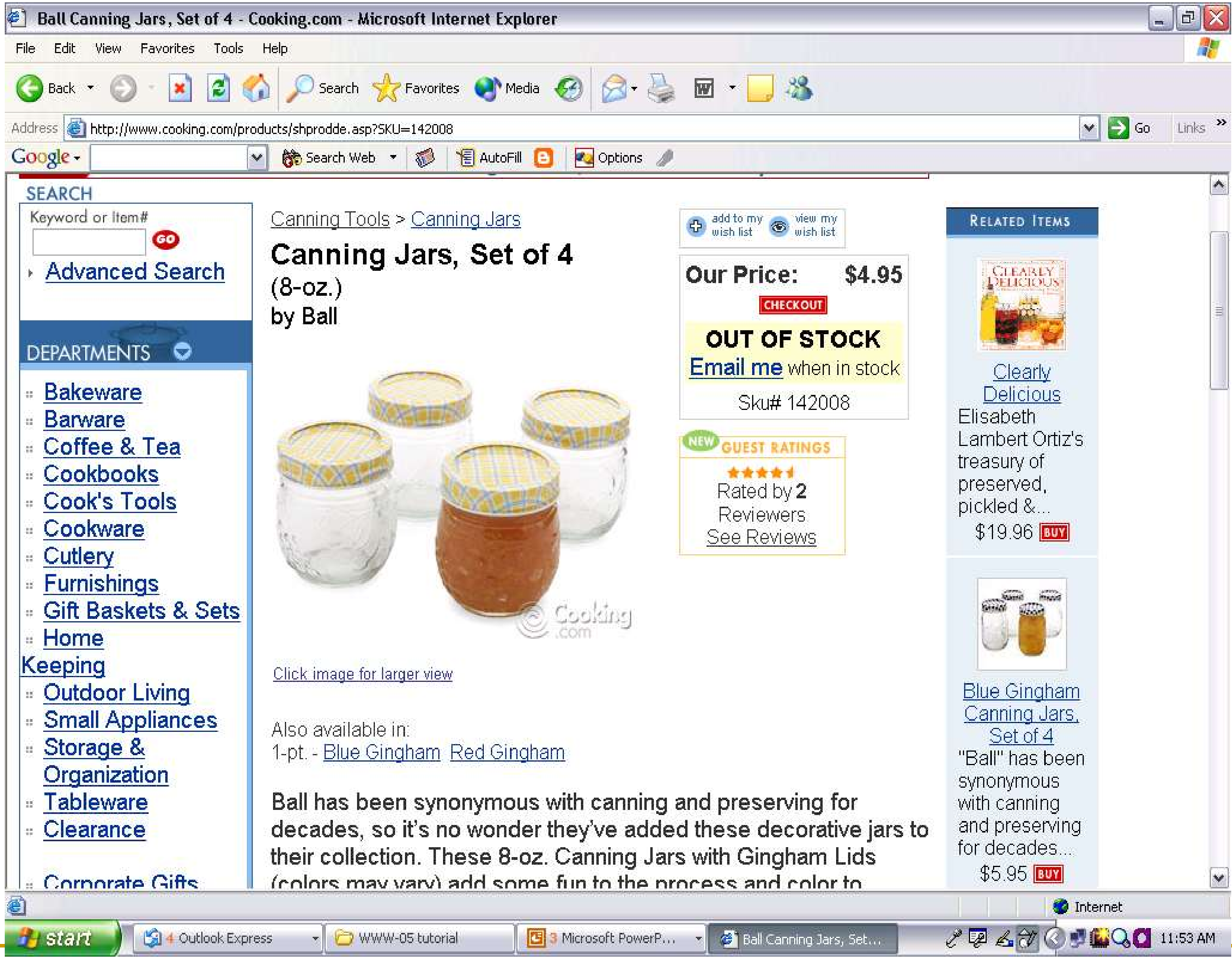
- A large amount of information on the Web is contained in regularly structured data objects.
  - often data records retrieved from databases.
- Such Web data records are important: lists of products and services.
- Applications: e.g.,
  - Comparative shopping, meta-search, meta-query, etc.
- Wrapper: A program for extracting structured data is usually called a wrapper.
- **We introduce:**
  - Wrapper induction (supervised learning)
  - automatic extraction (unsupervised learning)

Two types of data rich pages

- List pages
  - Each such page contains one or more lists of data records.
  - Each list in a specific region in the page
  - Two types of data records: flat and nested
- Detail pages
  - Each such page focuses on a single object.
  - But can have a lot of related and unrelated information


CompUSA.com - Product Results - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.compusa.com/products/products.asp?N=200049&cm_re=A-_-HPF-_-Flat+Panel+%28LCD%29
Go Links
Search Web AutoFill Options
Search the Web
Top Sellers
EN7410 17-inch LCD Monitor, Black/Dark Charcoal
$299.99
Add To Cart
(Delivery / Pick-Up)
Penny Shipping
Compare
17-inch LCD Monitor
$249.99
AL1714cb 17-inch LCD Monitor, Black
$269.99
SyncMaster 712n 17-inch LCD Monitor, Black
Was: $369.99
$299.99
SAVE $70 after:
$70.00 mail-in rebate(s)
Page 1 of 6: 1 2 3 4 5 6 Next >>
Sort by: Popularity
Compare
EN7410 17-inch LCD Monitor, Black/Dark Charcoal
Product Number: 318020
Mfr. Part #: EN7410
Brand: Envision
$299.99
17-inch LCD Monitor
Product Number: 316328
Mfr. Part #: 130611
Brand: Norwood Micro
$249.99
AL1714cb 17-inch LCD Monitor, Black
Product Number: 317993
Mfr. Part #: ET.L1809.031
Brand: Acer
$269.99
Was: $369.99
IN-STORE PICK-UP CLICK HERE
good guys high-end entertainment electronics click here
CompUSA Service On-Call
Repairs & service in your home or office click here
GANASSI RACING CLICK HERE!
COMPUSA AUCTIONS.com
VISIT OUR BRAND SHOWCASE!
ADVERTISED SPECIALS CLICK HERE!
Internet
start
4 Outlook Express
WWW-05 tutorial
3 Microsoft PowerP...
CompUSA.com - Prod...
11:57 AM


Canning Tools : Buy professional pressure canner canning jars tin strainer electric canners - C - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.cooking.com/products/shprodli.asp?DeptNo=4000&ClassNo=0420&CurSort=PriceAsc
Go Links
Google Search Web AutoFill Options
GO
Advanced Search
DEPARTMENTS
Bakeware
Barware
Coffee & Tea
Cookbooks
Cook's Tools
Cookware
Cutlery
Furnishings
Gift Baskets & Sets
Home Keeping
Outdoor Living
Small Appliances
Storage & Organization
Tableware
Clearance
Corporate Gifts
Gift Certificates
Email Offers
View by Brand: Norpro (3) Ball (3) R.S.V.P. (1) Back to Basics (1)
View Only: BestSellers Cooks Catalogue
Sort By: Product Type (z-a) | Price (high-low) | Customer Reviews (high-low)
Canning Jars by Ball
8-oz. Canning Jars, Set of 4 $4.95 INFO
1-pt. Canning Jars, Set of 4: Blue Gingham $5.95 BUY
Canning Tools by Norpro
12-dia. Canning Rack $5.95 BUY
Canning Tools by R.S.V.P.
6-in. Canning Funnel $6.50 BUY
Canning Tools by Norpro
Canning Strainer and Bag $8.95 BUY
Internet
start
4 Outlook Exp...
WWW-05 tutorial
WWW-05 tutorial
3 Microsoft Po...
Canning Tools : ...
11:46 AM


Ball Canning Jars, Set of 4 - Cooking.com - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.cooking.com/products/shprodde.asp?SKU=142008
Go Links
Google Search Web AutoFill Options
SEARCH
Keyword or Item#
GO
Advanced Search
DEPARTMENTS
Bakeware
Barware
Coffee & Tea
Cookbooks
Cook's Tools
Cookware
Cutlery
Furnishings
Gift Baskets & Sets
Home Keeping
Outdoor Living
Small Appliances
Storage & Organization
Tableware
Clearance
Corporate Gifts
Canning Tools > Canning Jars
Canning Jars, Set of 4
(8-oz.)
by Ball
Click image for larger view
Also available in:
1-pt. - Blue Gingham Red Gingham
Ball has been synonymous with canning and preserving for decades, so it's no wonder they've added these decorative jars to their collection. These 8-oz. Canning Jars with Gingham Lids (colors may vary) add some fun to the process and color to
add to my wish list
view my wish list
Our Price: $4.95
CHECKOUT
OUT OF STOCK
Email me when in stock
Sku# 142008
NEW GUEST RATINGS
Rated by 2 Reviewers
See Reviews
RELATED ITEMS
Clearly Delicious
Elisabeth Lambert Ortiz's treasury of preserved, pickled &...
$19.96 BUY
Blue Gingham Canning Jars, Set of 4
"Ball" has been synonymous with canning and preserving for decades...
$5.95 BUY
Internet
start
4 Outlook Express
WWW-05 tutorial
3 Microsoft PowerP...
Ball Canning Jars, Set...
11:53 AM

Extraction results

(a). An example page segment

image 1	Cabinet Organizers by Copco	9-in.	Round Turntable: White	*****	$4.95
image 1	Cabinet Organizers by Copco	12-in.	Round Turntable: White	*****	$7.95
image 2	Cabinet Organizers	14.75x9	Cabinet Organizer (Non-skid): White	*****	$7.95
image 3	Cabinet Organizers	22x6	Cookware Lid Rack	****	$19.95

(b). Extraction results

Road map

- Introduction
- **Data Model and HTML encoding**
- Wrapper induction
- Automatic Wrapper Generation: Two Problems
- String Matching and Tree Matching
- Multiple Alignments
- Building DOM Trees
- Extraction Given a List Page: Flat Data Records
- Extraction Given a List Page: Nested Data Records
- Extraction Given Multiple Pages
- Summary

The data model

- Most Web data can be modeled as **nested relations**
  - typed objects allowing nested sets and tuples.
- An **instance** of a type *T* is simply an element of *dom*(*T*).

- there is a set of **basic types**, $B = \{B_1, B_2, \ldots, B_k\}$. Each B_i is an atomic type, and its domain, denoted by $dom(B_i)$, is a set of constants.
- if $T_1, T_2, \ldots, T_n$ are basic or set types, then $[T_1, T_2, \ldots, T_n]$ is a **tuple type** with the domain $dom([T_1, T_2, \ldots, T_n]) = \{[v_1, v_2, \ldots, v_n] \mid v_i \in dom(T_i)\}$;
- if T is a tuple type, then $\{T\}$ is a **set type** with the domain $dom(\{T\})$ being the power set of $dom(T)$.

An example nested tuple type

- *name* (of type *string*),
- *image* (of type *image-file*), and
- *differentSizes* (a *set* type), consists of a set of tuples with the attributes:
  - *size* (of type *string*), and
  - *price* (of type *string*).

```
tuple  product  ( name:            string;
                  image:           image-file;
                  differentSizes:  set  ( size:   string;
                                          price:  string; ))
```

- Classic flat relations are of un-nested or flat set types.
- Nested relations are of arbitrary set types.

Type tree

- A basic type B_i is a leaf tree,
- A tuple type $[T_1, T_2, \ldots, T_n]$ is a tree rooted at a **tuple node** with n sub-trees, one for each T_i.
- A set type $\{T\}$ is a tree rooted at a **set node** with one sub-tree.

Note: attribute names are not included in the type tree.

We introduce a labeling of a type tree, which is defined recursively:

- If a set node is labeled ϕ, then its child is labeled $\phi.0$, a tuple node.
- If a tuple node is labeled ϕ, then its n children are labeled $\phi.1, \ldots, \phi.n$.

Instance tree

- An instance (constant) of a basic type is a leaf tree.
- A tuple instance $[v_1, v_2, \ldots, v_n]$ forms a tree rooted at a tuple node with n children or sub-trees representing attribute values $v_1, v_2, \ldots, v_n$.
- A set instance $\{e_1, e_2, \ldots, e_n\}$ forms a set node with n children or sub-trees representing the set elements $e_1, e_2, \ldots,$ and e_n.

Note: A tuple instance is usually called a **data record** in data extraction research.

HTML mark-up encoding of data

- There are no designated tags for each type as HTML was not designed as a data encoding language. Any HTML tag can be used for any type.
- For a tuple type, values (also called **data items**) of different attributes are usually encoded differently to distinguish them and to highlight important items.
- A tuple may be partitioned into several groups or sub-tuples. Each group covers a disjoint subset of attributes and may be encoded differently.

HTML encoding (cont …)

- for a leaf node of a basic type labeled ϕ, an instance c is encoded with,

  $enc(\phi.c) = OPEN\text{-}TAGS\ c\ CLOSE\text{-}TAGS$,

  where $OPEN\text{-}TAGS$ is a sequence of open HTML tags, and $CLOSE\text{-}TAGS$ is the sequence of corresponding close HTML tags. The number of tags is greater than or equal to 0.
- for a tuple node labeled ϕ of n children or attributes, $[\phi.1, \ldots, \phi.n]$, the attributes are first partitioned into h (≥ 1) groups $<\phi.1, \ldots, \phi.e>$, $<\phi.(e+1), \ldots, \phi.g>$ ... $<\phi.(k+1), \ldots, \phi.n>$ and an instance $[v_1, \ldots, v_n]$ of the tuple node is encoded with

  $$\begin{aligned} enc(\phi{:}[v_1, \ldots, v_n]) = \ & OPEN\text{-}TAGS_1\ enc(v_1) \ldots enc(v_e)\ CLOSE\text{-}TAGS_1 \\ & OPEN\text{-}TAGS_2\ enc(v_{e+1}) \ldots enc(v_g)\ CLOSE\text{-}TAGS_2 \\ & \ldots \\ & OPEN\text{-}TAGS_h\ enc(v_{k+1}) \ldots enc(v_n)\ CLOSE\text{-}TAGS_h \end{aligned}$$

  where $OPEN\text{-}TAGS_i$ is a sequence of open HTML tags, and $CLOSE\text{-}TAGS_i$ is the sequence of corresponding close tags. The number of tags is greater than or equal to 0.
- for a set node labeled ϕ, an non-empty set instance $\{e_1, e_2, \ldots, e_n\}$ is encoded with

  $$enc(\phi{:}\{e_1, \ldots, e_n\}) = OPEN\text{-}TAGS\ enc(e_{j_1}) \ldots enc(e_{j_n})\ CLOSE\text{-}TAGS$$

More on HTML encoding

- By no means, this mark-up encoding covers all cases in Web pages.
  - In fact, each group of a tuple type can be further divided.
- We must also note that in an actual Web page the encoding may not be done by HTML tags alone.
  - Words and punctuation marks can be used as well.

Restaurant Name: **Good Noodles**

- 205 Willow, *Glen*, Phone 1-*773*-366-1987
- 25 Oak, *Forest*, Phone (800) 234-7903
- 324 Halsted St., *Chicago*, Phone 1-*800*-996-5023
- 700 Lake St., *Oak Park*, Phone: (708) 798-0008

Road map

- Introduction
- Data Model and HTML encoding
- **Wrapper induction**
- Automatic Wrapper Generation: Two Problems
- String Matching and Tree Matching
- Multiple Alignments
- Building DOM Trees
- Extraction Given a List Page: Flat Data Records
- Extraction Given a List Page: Nested Data Records
- Extraction Given Multiple Pages
- Summary

Wrapper induction

- Using machine learning to generate extraction rules.
  - The user marks the target items in a few training pages.
  - The system learns extraction rules from these pages.
  - The rules are applied to extract items from other pages.
- Many wrapper induction systems, e.g.,
  - WIEN (Kushmerick et al, IJCAI-97),
  - Softmealy (Hsu and Dung, 1998),
  - Stalker (Muslea et al. Agents-99),
  - BWI (Freitag and Kushmerick, AAAI-00),
  - WL^2 (Cohen et al. WWW-02).
- We will only focus on Stalker, which also has a commercial version, Fetch.

Stalker: A hierarchical wrapper induction system

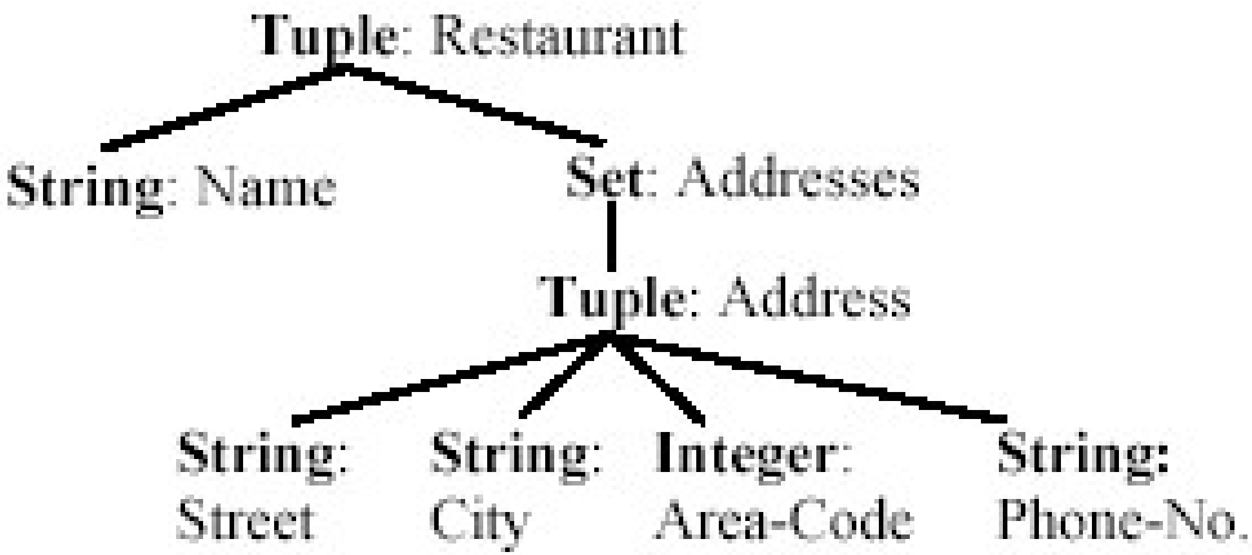
- Hierarchical wrapper learning
  - Extraction is isolated at different levels of hierarchy
  - This is suitable for nested data records (embedded list)
- Each item is extracted independent of others.

- Each target item is extracted using two rules
  - A start rule for detecting the beginning of the target item.
  - A end rule for detecting the ending of the target item.

Hierarchical representation: type tree

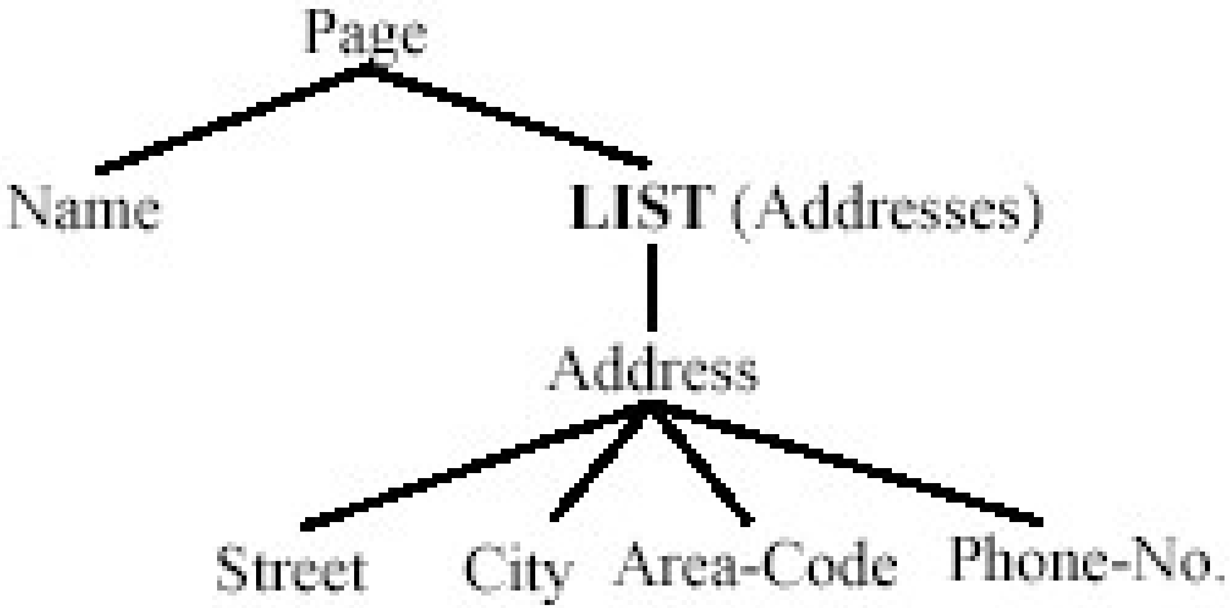
Restaurant Name: **Good Noodles**

- 205 Willow, *Glen*, Phone 1-*773*-366-1987
- 25 Oak, *Forest*, Phone (800) 234-7903
- 324 Halsted St., *Chicago*, Phone 1-*800*-996-5023
- 700 Lake St., *Oak Park*, Phone: (708) 798-0008



Data extraction based on EC tree

- The extraction is done using a tree structure called the *EC* tree (**embedded catalog tree**).
- The *EC* tree is based on the type tree above.



- To extract each target item (a node), the wrapper needs a rule that extracts the item from its parent.

Extraction using two rules

- Each extraction is done using two rules,
  - **a start rule** and **a end rule**.
- The start rule identifies the beginning of the node and the end rule identifies the end of the node.
  - This strategy is applicable to both leaf nodes (which represent data items) and list nodes.
- For a list node, **list iteration rules** are needed to break the list into individual data records (tuple instances).

Rules use landmarks

- The extraction rules are based on the idea of **landmarks**.
  - Each landmark is a sequence of *consecutive* tokens.
- Landmarks are used to locate the beginning and the end of a target item.
- Rules use landmarks

An example

- Let us try to extract the restaurant name “Good Noodles”. Rule R1 can to identify the beginning :

  R1: *SkipTo*() // start rule

- This rule means that the system should start from the beginning of the page and skip all the tokens until it sees the first tag. is a landmark.
- Similarly, to identify the end of the restaurant name, we use:

  R2: *SkipTo*() // end rule

```
1: <p> Restaurant Name: <b>Good Noodles</b><br><br>
2: <li> 205 Willow, <i>Glen</i>, Phone 1-<i>773</i>-366-1987</li>
3: <li> 25 Oak, <i>Forest</i>, Phone (800) 234-7903 </li>
4: <li> 324 Halsted St., <i>Chicago</i>, Phone 1-<i>800</i>-996-5023 </li>
5: <li> 700 Lake St., <i>Oak Park</i>, Phone: (708) 798-0008 </li> </p>
```

Rules are not unique

- Note that a rule may not be unique. For example, we can also use the following rules to identify the beginning of the name:

  R3: *SkiptTo*(Name _*Punctuation*_ _*HtmlTag*_)

or **R4**: *SkiptTo*(Name) *SkipTo*(<b>)

- **R3** means that we skip everything till the word "Name" followed by a punctuation symbol and then a HTML tag. In this case, "Name _*Punctuation*_ _*HtmlTag*_" together is a landmark.
  - _*Punctuation*_ and _*HtmlTag*_ are **wildcards**.

Extract area codes

1. Identify the entire list of addresses. We can use the start rule *SkipTo*(

), and the end rule *SkipTo*(</p>).
2. Iterate through the list (lines 2-5) to break it into 4 individual records (lines 2 - 5). To identify the beginning of each address, the wrapper can start from the first token of the parent and repeatedly apply the start rule *SkipTo*(<li>) to the content of the list. Each successive identification of the beginning of an address starts from where the previous one ends. Similarly, to identify the end of each address, it starts from the last token of its parent and repeatedly apply the end rule *SkipTo*().

Once each address record is identified or extracted, we can extract the area code in it. Due to variations in the format of area codes (some are in italic and some are not), we need to use disjunctions. In this case, the disjunctive start and the end rules are respectively **R5** and **R6**:

R5: **either** SkipTo(( )
or SkipTo(-<i>)

R6: **either** SkipTo( ))
or SkipTo(</i>)

In a disjunctive rule, the disjuncts are applied sequentially until a disjunct can identify the target node.

Learning extraction rules

- Stalker uses sequential covering to learn extraction rules for each target item.
  - In each iteration, it learns a perfect rule that covers as many positive examples as possible without covering any negative example.
  - Once a positive example is covered by a rule, it is removed.
  - The algorithm ends when all the positive examples are covered. The result is an ordered list of all learned rules.