Web Mining
(網路探勘)

Structured Data Extraction
(結構化資料擷取)

1011WM09
TLMXM1A
Wed 8,9 (15:10-17:00) U705

Min-Yuh Day
戴敏育
Assistant Professor
專任助理教授
Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系

http://mail. tku.edu.tw/myday/
2012-11-28

課程大綱 (Syllabus)

週次	日期	內容（Subject/Topics）
1	101/09/12	Introduction to Web Mining (網路探勘導論)
2	101/09/19	Association Rules and Sequential Patterns (關聯規則和序列模式)
3	101/09/26	Supervised Learning (監督式學習)
4	101/10/03	Unsupervised Learning (非監督式學習)
5	101/10/10	國慶紀念日(放假一天)
6	101/10/17	Paper Reading and Discussion (論文研讀與討論)
7	101/10/24	Partially Supervised Learning (部分監督式學習)
8	101/10/31	Information Retrieval and Web Search (資訊檢索與網路搜尋)
9	101/11/07	Social Network Analysis (社會網路分析)

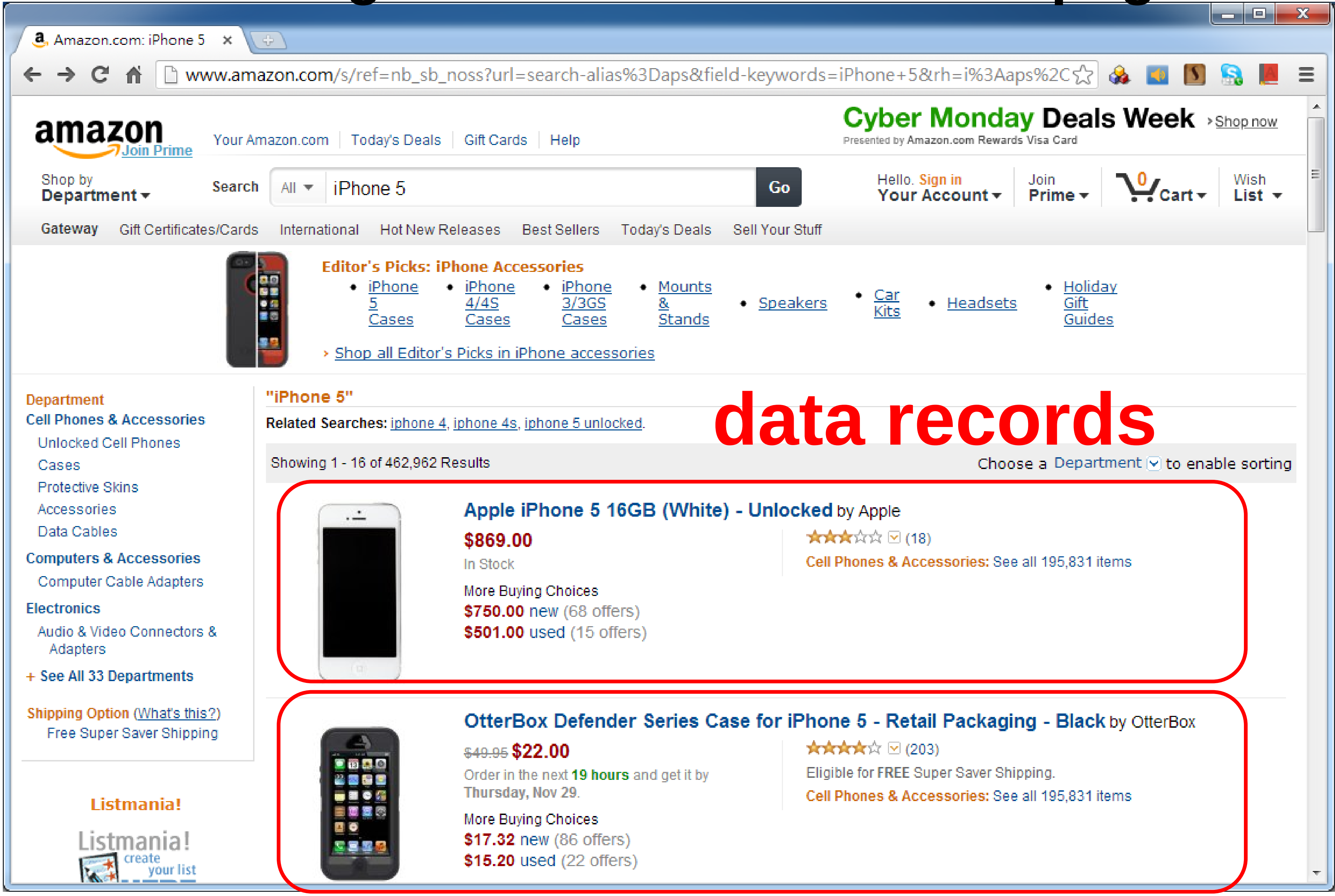
課程大綱 (Syllabus)

週次	日期	內容（Subject/Topics）
10	101/11/14	Midterm Presentation (期中報告)
11	101/11/21	Web Crawling (網路爬行)
12	101/11/28	Structured Data Extraction (結構化資料擷取)
13	101/12/05	Information Integration (資訊整合)
14	101/12/12	Opinion Mining and Sentiment Analysis (意見探勘與情感分析)
15	101/12/19	Paper Reading and Discussion (論文研讀與討論)
16	101/12/26	Web Usage Mining (網路使用挖掘)
17	102/01/02	Project Presentation 1 (期末報告1)
18	102/01/09	Project Presentation 2 (期末報告2)

Outline

- Web Information Extraction
  1. Manual Approach
     - Human programmer
  2. Wrapper Induction
     - Supervised learning approach
  3. Automatic Extraction
     - Unsupervised learning approach
- String Matching and Tree Matching

Extracting data records from web pages



Amazon.com: iPhone 5

www.amazon.com/s/ref=nb_sb_noss?url=search-alias%3Daps&field-keywords=iPhone+5&rh=i%3Aaps%2C

amazon Join Prime

Your Amazon.com | Today's Deals | Gift Cards | Help

Cyber Monday Deals Week › Shop now

Presented by Amazon.com Rewards Visa Card

Shop by Department | Search All | iPhone 5 | Go | Hello. Sign in Your Account | Join Prime | 0 Cart | Wish List

Gateway | Gift Certificates/Cards | International | Hot New Releases | Best Sellers | Today's Deals | Sell Your Stuff

Editor's Picks: iPhone Accessories

- iPhone 5 Cases
- iPhone 4/4S Cases
- iPhone 3/3GS Cases
- Mounts & Stands
- Speakers
- Car Kits
- Headsets
- Holiday Gift Guides

› Shop all Editor's Picks in iPhone accessories

Department

Cell Phones & Accessories

- Unlocked Cell Phones
- Cases
- Protective Skins
- Accessories
- Data Cables

Computers & Accessories

- Computer Cable Adapters

Electronics

- Audio & Video Connectors & Adapters

+ See All 33 Departments

Shipping Option (What's this?)

Free Super Saver Shipping

Listmania!

Listmania! create your list

"iPhone 5"

Related Searches: iphone 4, iphone 4s, iphone 5 unlocked.

Showing 1 - 16 of 462,962 Results

Choose a Department to enable sorting

data records

Apple iPhone 5 16GB (White) - Unlocked by Apple

$869.00

In Stock

(18)

Cell Phones & Accessories: See all 195,831 items

More Buying Choices

$750.00 new (68 offers)

$501.00 used (15 offers)

OtterBox Defender Series Case for iPhone 5 - Retail Packaging - Black by OtterBox

~~$49.95~~ $22.00

(203)

Order in the next 19 hours and get it by Thursday, Nov 29.

Eligible for FREE Super Saver Shipping.

Cell Phones & Accessories: See all 195,831 items

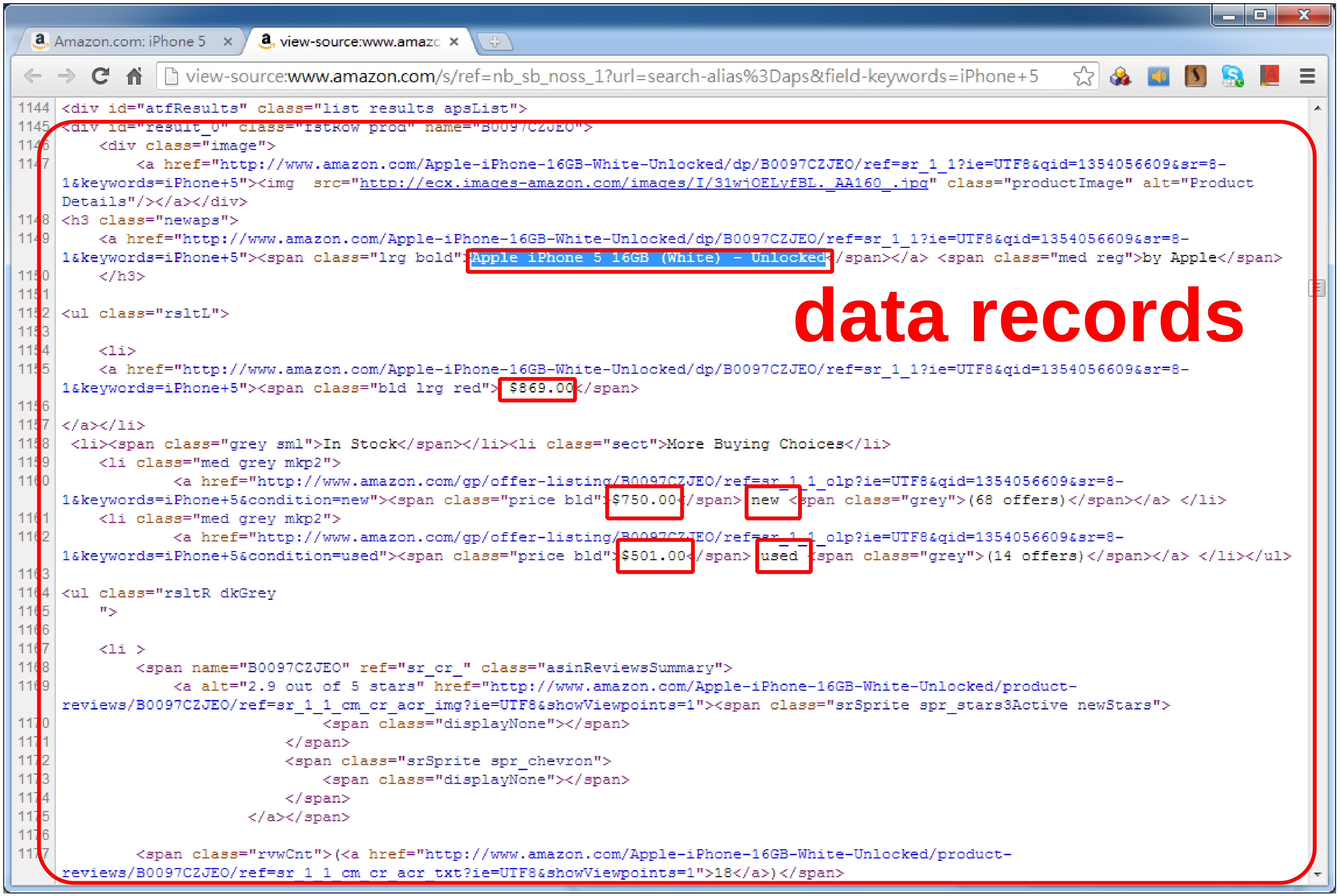
More Buying Choices

$17.32 new (86 offers)

$15.20 used (22 offers)

Source: http://www.amazon.com/

Data Extraction from Source Code of Web page



Amazon.com: iPhone 5
view-source:www.amazc
view-source:www.amazon.com/s/ref=nb_sb_noss_1?url=search-alias%3Daps&field-keywords=iPhone+5
data records

```
<div id="atfResults" class="list results apsList">
<div id="result_0" class="fstRow prod" name="B0097CZJEO">
    <div class="image">
        <a href="http://www.amazon.com/Apple-iPhone-16GB-White-Unlocked/dp/B0097CZJEO/ref=sr_1_1?ie=UTF8&qid=1354056609&sr=8-1&keywords=iPhone+5"><img  src="http://ecx.images-amazon.com/images/I/31wjOELvfBL._AA160_.jpg" class="productImage" alt="Product Details"/></a></div>
<h3 class="newaps">
    <a href="http://www.amazon.com/Apple-iPhone-16GB-White-Unlocked/dp/B0097CZJEO/ref=sr_1_1?ie=UTF8&qid=1354056609&sr=8-1&keywords=iPhone+5"><span class="lrg bold">Apple iPhone 5 16GB (White) - Unlocked</span></a> <span class="med reg">by Apple</span>
    </h3>

<ul class="rsltL">

    <li>
    <a href="http://www.amazon.com/Apple-iPhone-16GB-White-Unlocked/dp/B0097CZJEO/ref=sr_1_1?ie=UTF8&qid=1354056609&sr=8-1&keywords=iPhone+5"><span class="bld lrg red"> $869.00</span>

</a></li>
 <li><span class="grey sml">In Stock</span></li><li class="sect">More Buying Choices</li>
    <li class="med grey mkp2">
        <a href="http://www.amazon.com/gp/offer-listing/B0097CZJEO/ref=sr_1_1_olp?ie=UTF8&qid=1354056609&sr=8-1&keywords=iPhone+5&condition=new"><span class="price bld">$750.00</span> new <span class="grey">(68 offers)</span></a> </li>
    <li class="med grey mkp2">
        <a href="http://www.amazon.com/gp/offer-listing/B0097CZJEO/ref=sr_1_1_olp?ie=UTF8&qid=1354056609&sr=8-1&keywords=iPhone+5&condition=used"><span class="price bld">$501.00</span> used <span class="grey">(14 offers)</span></a> </li></ul>

<ul class="rsltR dkGrey
    ">

    <li >
        <span name="B0097CZJEO" ref="sr_cr_" class="asinReviewsSummary">
            <a alt="2.9 out of 5 stars" href="http://www.amazon.com/Apple-iPhone-16GB-White-Unlocked/product-reviews/B0097CZJEO/ref=sr_1_1_cm_cr_acr_img?ie=UTF8&showViewpoints=1"><span class="srSprite spr_stars3Active newStars">
                    <span class="displayNone"></span>
                </span>
                <span class="srSprite spr_chevron">
                    <span class="displayNone"></span>
                </span>
            </a></span>

        <span class="rvwCnt">(<a href="http://www.amazon.com/Apple-iPhone-16GB-White-Unlocked/product-reviews/B0097CZJEO/ref=sr_1_1_cm_cr_acr_txt?ie=UTF8&showViewpoints=1">18</a>)</span>
```

Source: http://www.amazon.com/

Structured Data Extraction: Wrapper Generation

- Web information extraction
  - Information Extraction (IE)
  - Extracting target information items from Web pages
  - Two general problems
    - Extracting information from natural language text
    - Extracting structured data from Web pages
- Wrapper
  - A program for extracting structured data

Extracting Data records

- Extracting data records from web pages
  - Obtain and integrate data from multiple sources (Web sites and pages)
  - Provide value-added services
    - Customizable web information gathering
    - Comparative shopping
    - Meta-search
  - Examples
    - Products sold online
    - Product reviews
    - Job postings
    - Research publications
    - Forum discussions

Three Main Approaches for Information Extraction

1. Manual Approach
   – Human programmer
2. Wrapper Induction
   – Supervised learning approach
3. Automatic Extraction
   – Unsupervised learning approach

1. Manual Approach

- Observing a Web page and its source code
- Human programmer finds some patterns
  - Writes a program to extract the target data
- Manual approach is not scalable to a large number of Web sites

2. Wrapper Induction

- Supervised learning approach
- Semi-automatic
- A set of extraction rules is learned from a manually labeled pages or data records
- Extract target data items from other similarly formatted pages

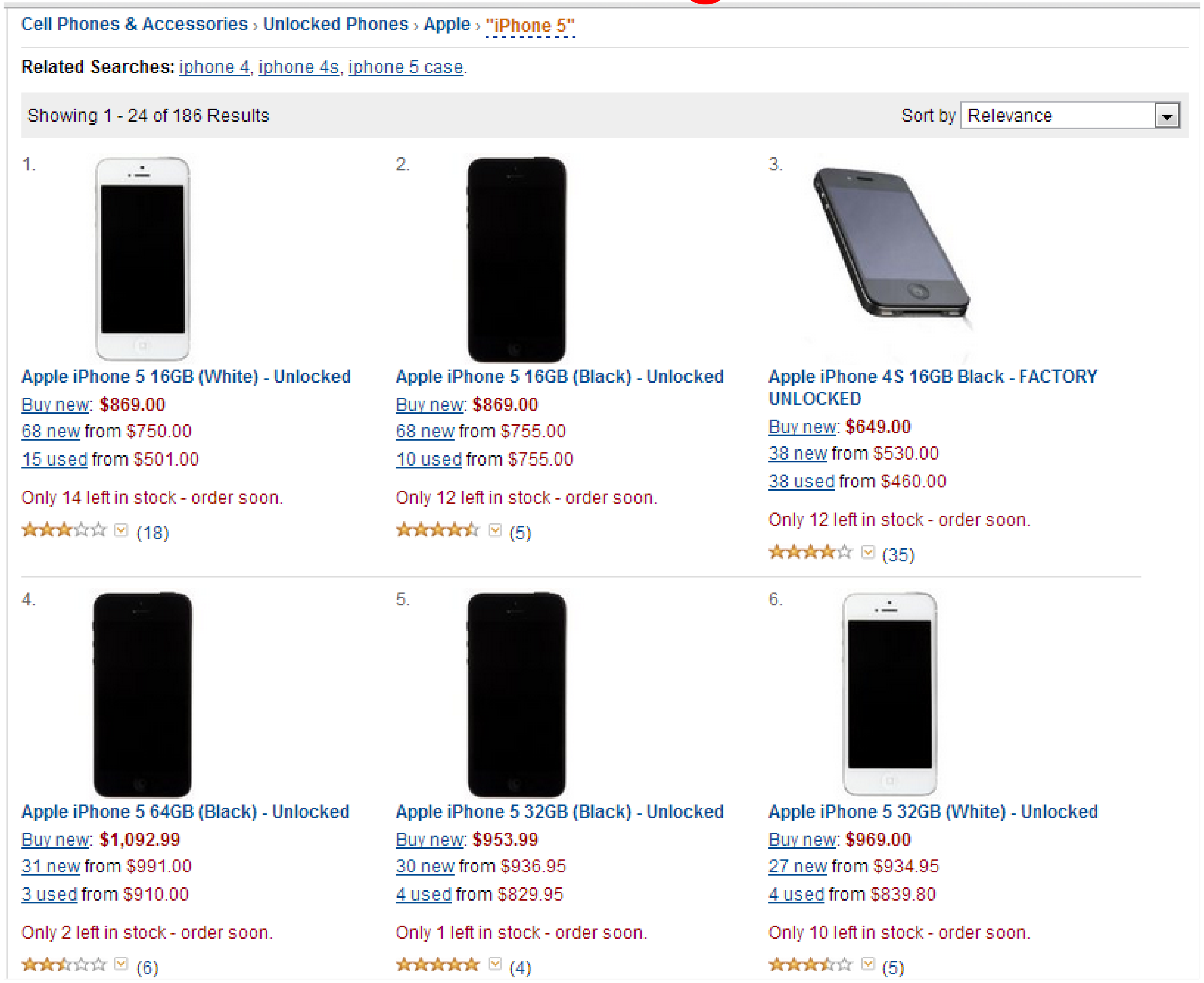
3. Automatic Extraction

- Unsupervised learning approach
- Automatically finds patterns or grammars from given a single or multiple web pages for data extraction
- Eliminates the manual labeling effort
- Scale up data extraction to a huge number of sites and pages

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.
  - A lot of related and unrelated information

List Page



Cell Phones & Accessories › Unlocked Phones › Apple › "iPhone 5"

Related Searches: iphone 4, iphone 4s, iphone 5 case.

Showing 1 - 24 of 186 Results

Sort by Relevance

1. **Apple iPhone 5 16GB (White) - Unlocked**
Buy new: **$869.00**
68 new from $750.00
15 used from $501.00
Only 14 left in stock - order soon.
★★★☆☆ (18)

2. **Apple iPhone 5 16GB (Black) - Unlocked**
Buy new: **$869.00**
68 new from $755.00
10 used from $755.00
Only 12 left in stock - order soon.
★★★★☆ (5)

3. **Apple iPhone 4S 16GB Black - FACTORY UNLOCKED**
Buy new: **$649.00**
38 new from $530.00
38 used from $460.00
Only 12 left in stock - order soon.
★★★★☆ (35)

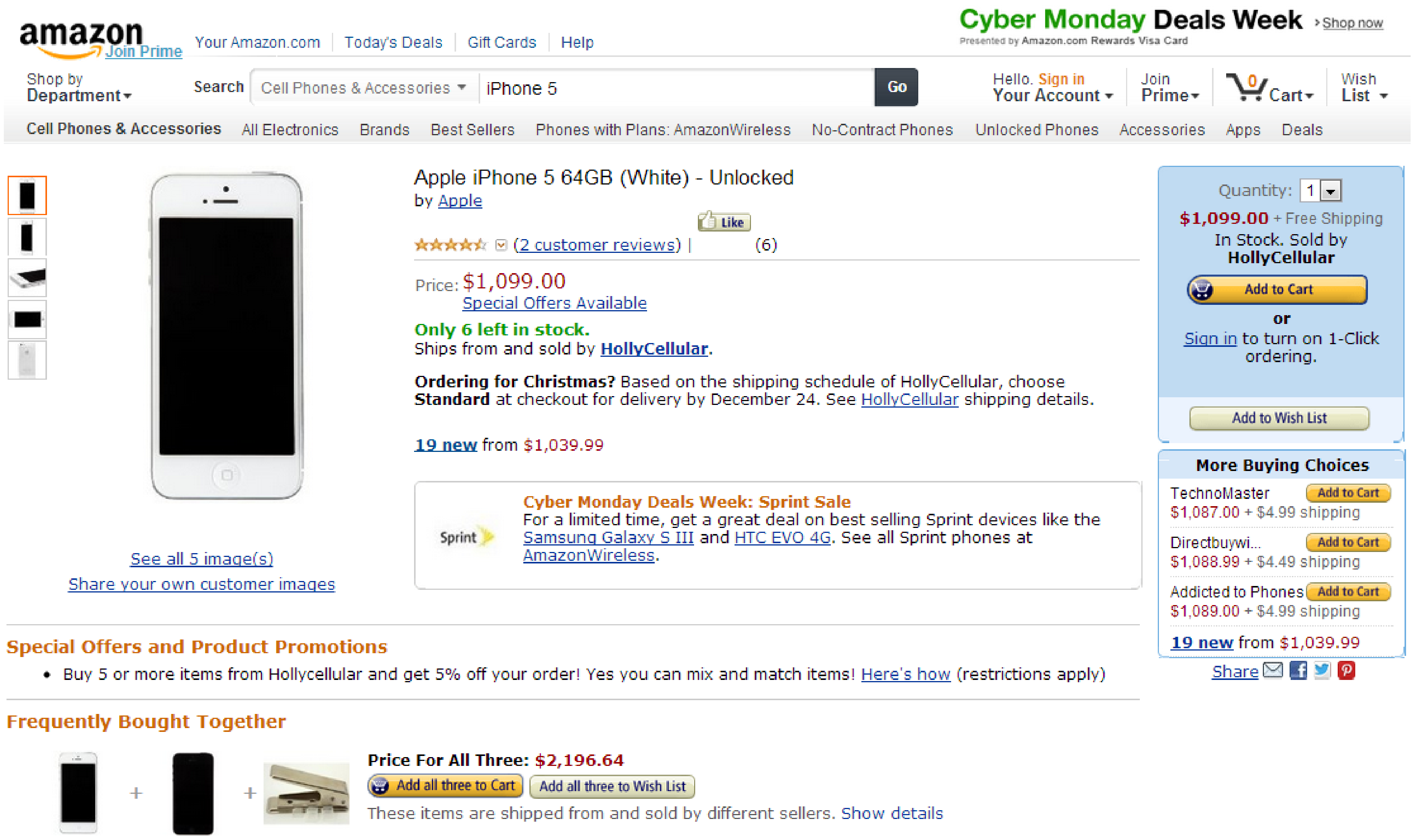
4. **Apple iPhone 5 64GB (Black) - Unlocked**
Buy new: **$1,092.99**
31 new from $991.00
3 used from $910.00
Only 2 left in stock - order soon.
★★☆☆☆ (6)

5. **Apple iPhone 5 32GB (Black) - Unlocked**
Buy new: **$953.99**
30 new from $936.95
4 used from $829.95
Only 1 left in stock - order soon.
★★★★★ (4)

6. **Apple iPhone 5 32GB (White) - Unlocked**
Buy new: **$969.00**
27 new from $934.95
4 used from $839.80
Only 10 left in stock - order soon.
★★★☆☆ (5)

Source: http://www.amazon.com/s/ref=sr_nr_p_n_feature_three_br_0?rh=n%3A2335752011%2Cn%3A%212335753011%2Cn%3A2407749011%2Ck%3AiPhone+5%2Cp_n_feature_three_browse-bin%3A2493008011&bbn=2407749011&keywords=iPhone+5&ie=UTF8&qid=1354059124&rnid=2420729011

Detail Page



amazon
Join Prime
Your Amazon.com | Today's Deals | Gift Cards | Help
Cyber Monday Deals Week › Shop now
Presented by Amazon.com Rewards Visa Card
Shop by Department
Search
Cell Phones & Accessories
iPhone 5
Go
Hello. Sign in Your Account
Join Prime
0 Cart
Wish List
Cell Phones & Accessories | All Electronics | Brands | Best Sellers | Phones with Plans: AmazonWireless | No-Contract Phones | Unlocked Phones | Accessories | Apps | Deals

Apple iPhone 5 64GB (White) - Unlocked
by Apple

(2 customer reviews) | Like (6)

Price: $1,099.00
Special Offers Available

Only 6 left in stock.
Ships from and sold by **HollyCellular**.

Ordering for Christmas? Based on the shipping schedule of HollyCellular, choose **Standard** at checkout for delivery by December 24. See HollyCellular shipping details.

19 new from $1,039.99

Cyber Monday Deals Week: Sprint Sale
For a limited time, get a great deal on best selling Sprint devices like the Samsung Galaxy S III and HTC EVO 4G. See all Sprint phones at AmazonWireless.

See all 5 image(s)
Share your own customer images

Quantity: 1
$1,099.00 + Free Shipping
In Stock. Sold by **HollyCellular**
Add to Cart
or
Sign in to turn on 1-Click ordering.
Add to Wish List

More Buying Choices

TechnoMaster	$1,087.00 + $4.99 shipping
Directbuywi...	$1,088.99 + $4.49 shipping
Addicted to Phones	$1,089.00 + $4.99 shipping

19 new from $1,039.99

Share

Special Offers and Product Promotions

- Buy 5 or more items from Hollycellular and get 5% off your order! Yes you can mix and match items! Here's how (restrictions apply)

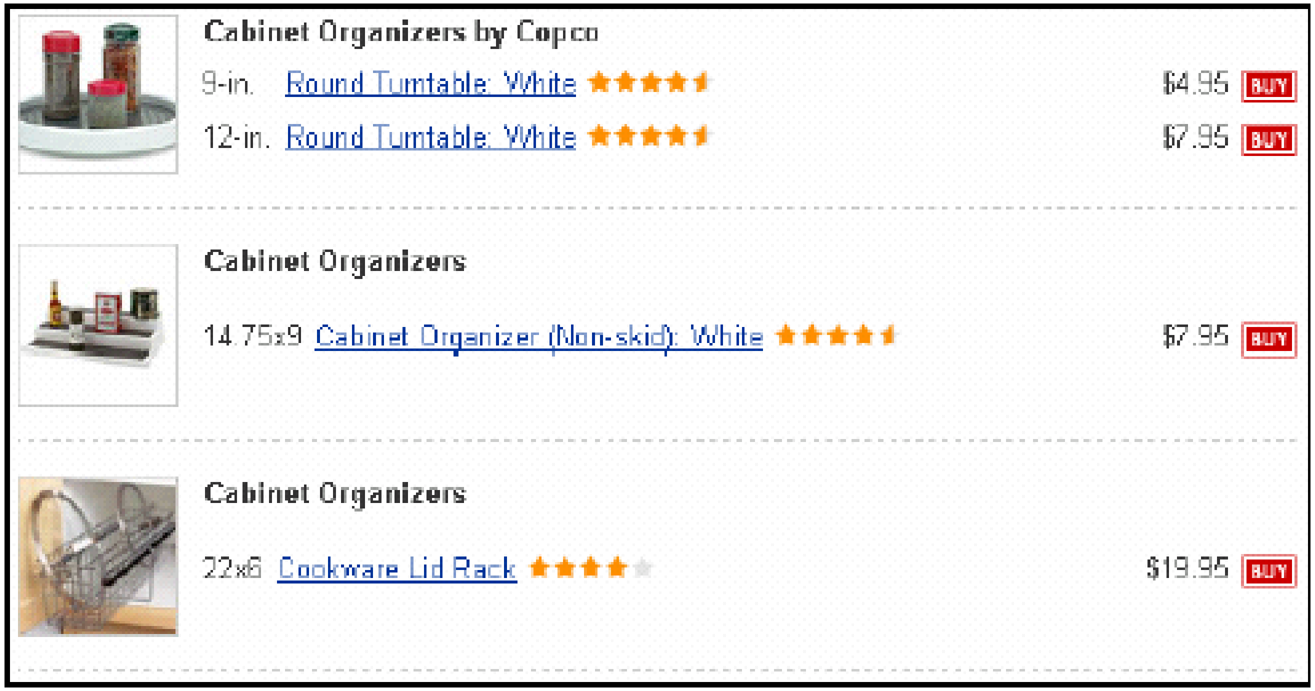
Frequently Bought Together

Price For All Three: $2,196.64
Add all three to Cart | Add all three to Wish List
These items are shipped from and sold by different sellers. Show details

Source: http://www.amazon.com/Apple-iPhone-64GB-White-Unlocked/dp/B0097CZR5A/ref=sr_1_9?s=wireless&ie=UTF8&qid=1354059147&sr=1-9&keywords=iPhone+5

Extraction Results

Input Web Page



Cabinet Organizers by Copco

9-in. Round Turntable: White ★★★★★ $4.95 BUY

12-in. Round Turntable: White ★★★★★ $7.95 BUY

Cabinet Organizers

14.75x9 Cabinet Organizer (Non-skid): White ★★★★★ $7.95 BUY

Cabinet Organizers

22x6 Cookware Lid Rack ★★★★ $19.95 BUY

(a). An example page segment

Output Data Table

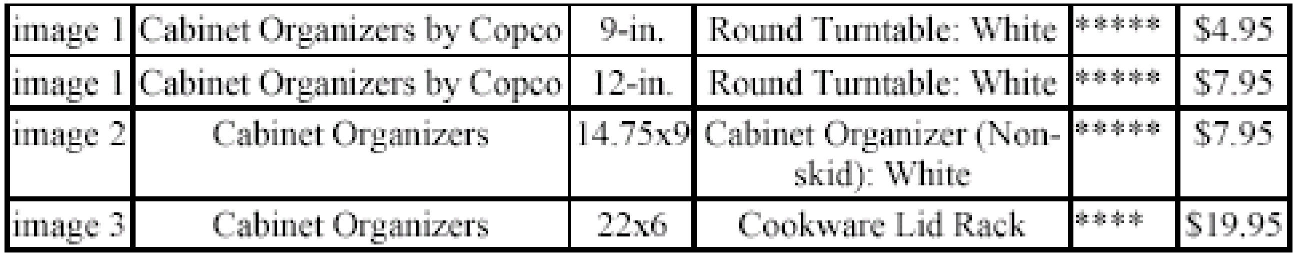
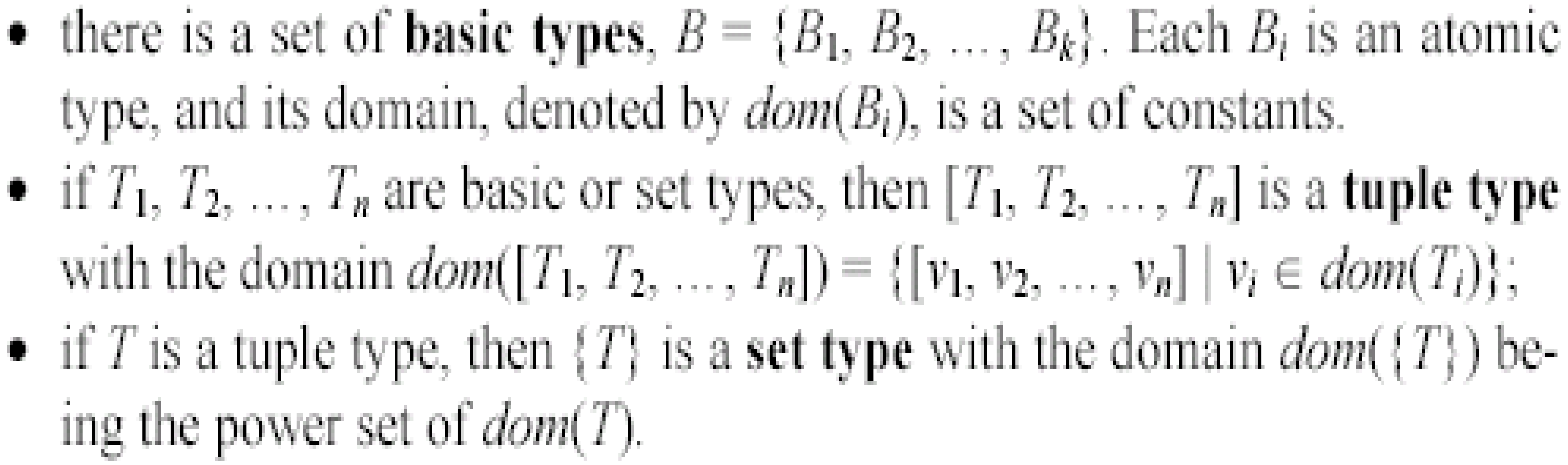


image 1	Cabinet Organizers by Copco	9-in.	Round Turntable: White	*****	$4.95
image 1	Cabinet Organizers by Copco	12-in.	Round Turntable: White	*****	$7.95
image 2	Cabinet Organizers	14.75x9	Cabinet Organizer (Non-skid): White	*****	$7.95
image 3	Cabinet Organizers	22x6	Cookware Lid Rack	****	$19.95

(b). Extraction results

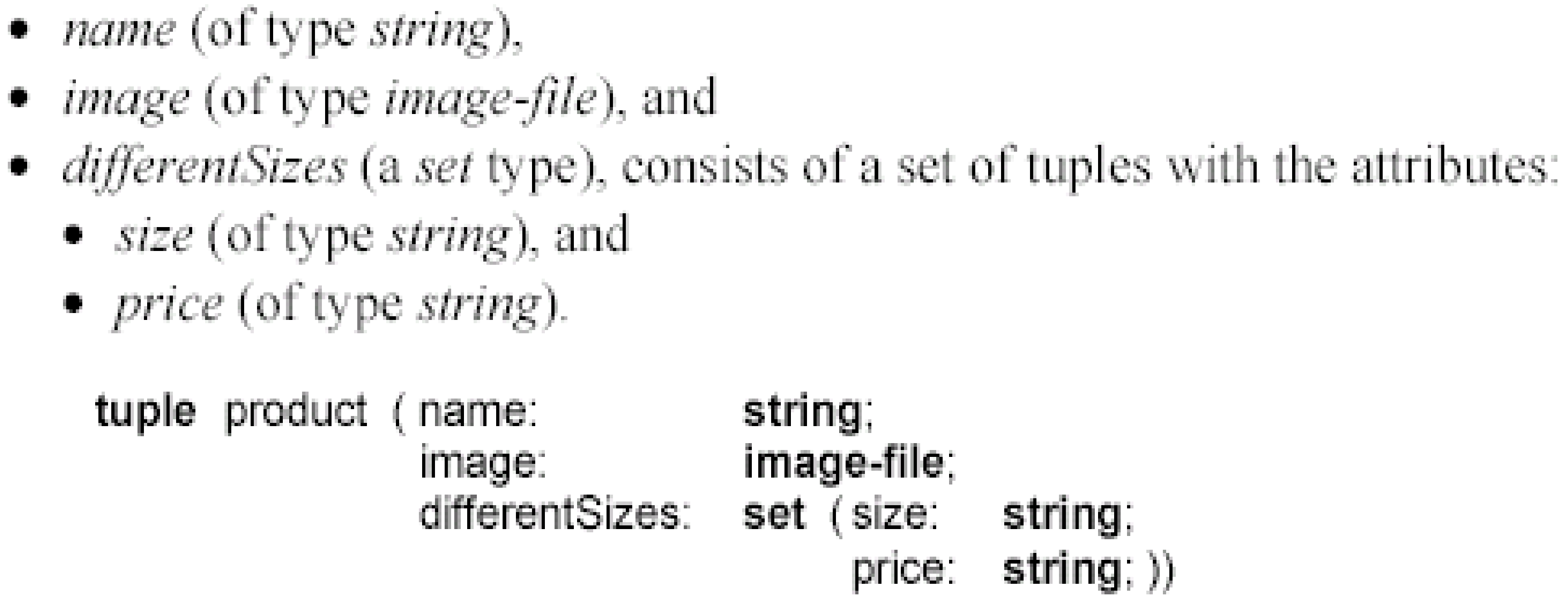
The data model

- 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

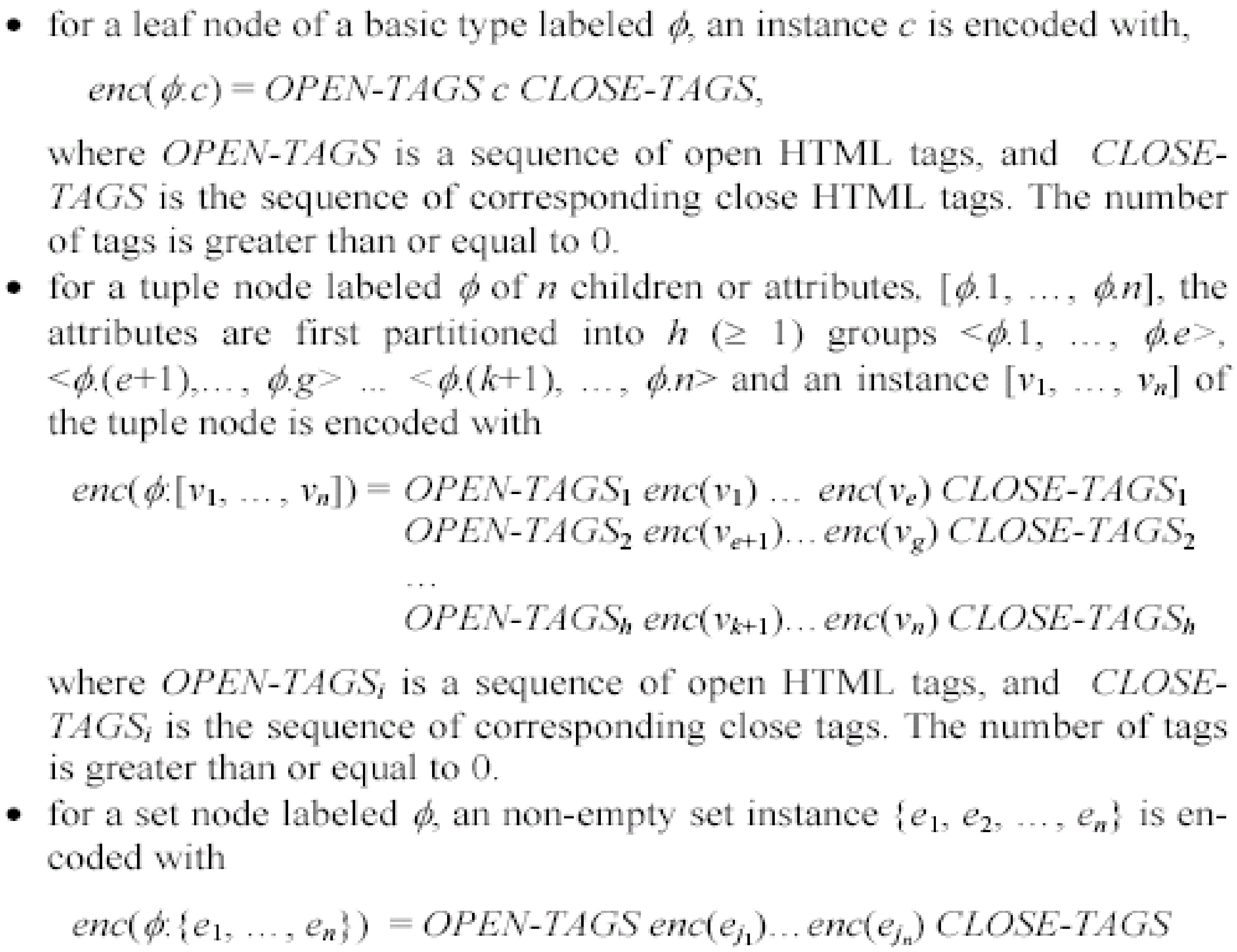
- An instance (constant) of a basic type is a leaf tree.
- A tuple instance [v_1, v_2, …, v_n] forms a tree rooted at a tuple node with *n* children or sub-trees representing attribute values v_1, v_2, …, v_n.
- A set instance {e_1, e_2, …, e_n} forms a set node with *n* children or sub-trees representing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

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

HTML Mark-Up Encoding of Data Instances

- 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-TAGS* is a sequence of open HTML tags, and *CLOSE-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 ($\geq$ 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

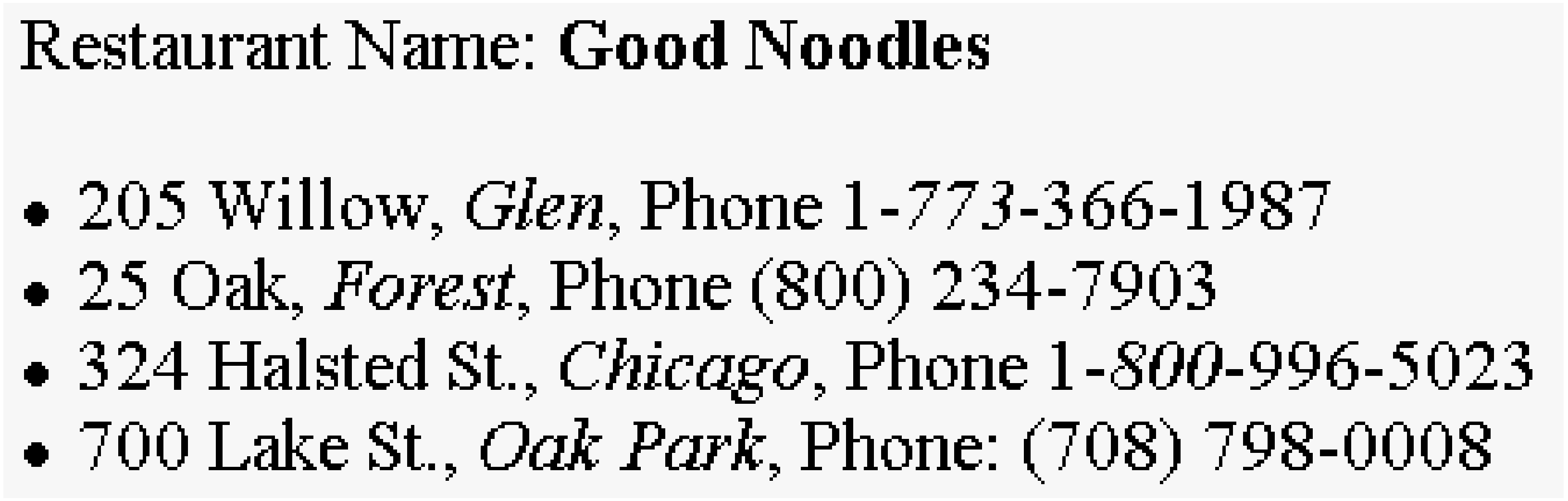
  $$\begin{aligned} enc(\phi{:}[v_1, \ldots, v_n]) = {} & OPEN\text{-}TAGS_1\ enc(v_1)\ \ldots\ enc(v_e)\ CLOSE\text{-}TAGS_1 \\ & OPEN\text{-}TAGS_2\ enc(v_{e+1}) \ldots enc(v_g)\ CLOSE\text{-}TAGS_2 \\ & \ldots \\ & OPEN\text{-}TAGS_h\ enc(v_{k+1}) \ldots enc(v_n)\ CLOSE\text{-}TAGS_h \end{aligned}$$

  where *OPEN-TAGS$_i$* is a sequence of open HTML tags, and *CLOSE-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

- Mark-up encoding covers all cases in Web pages.
  - each group of a tuple type can be further divided.
- Actual Web page the encoding may not be done by HTML tags alone.
  - Words and punctuation marks can be used as well.



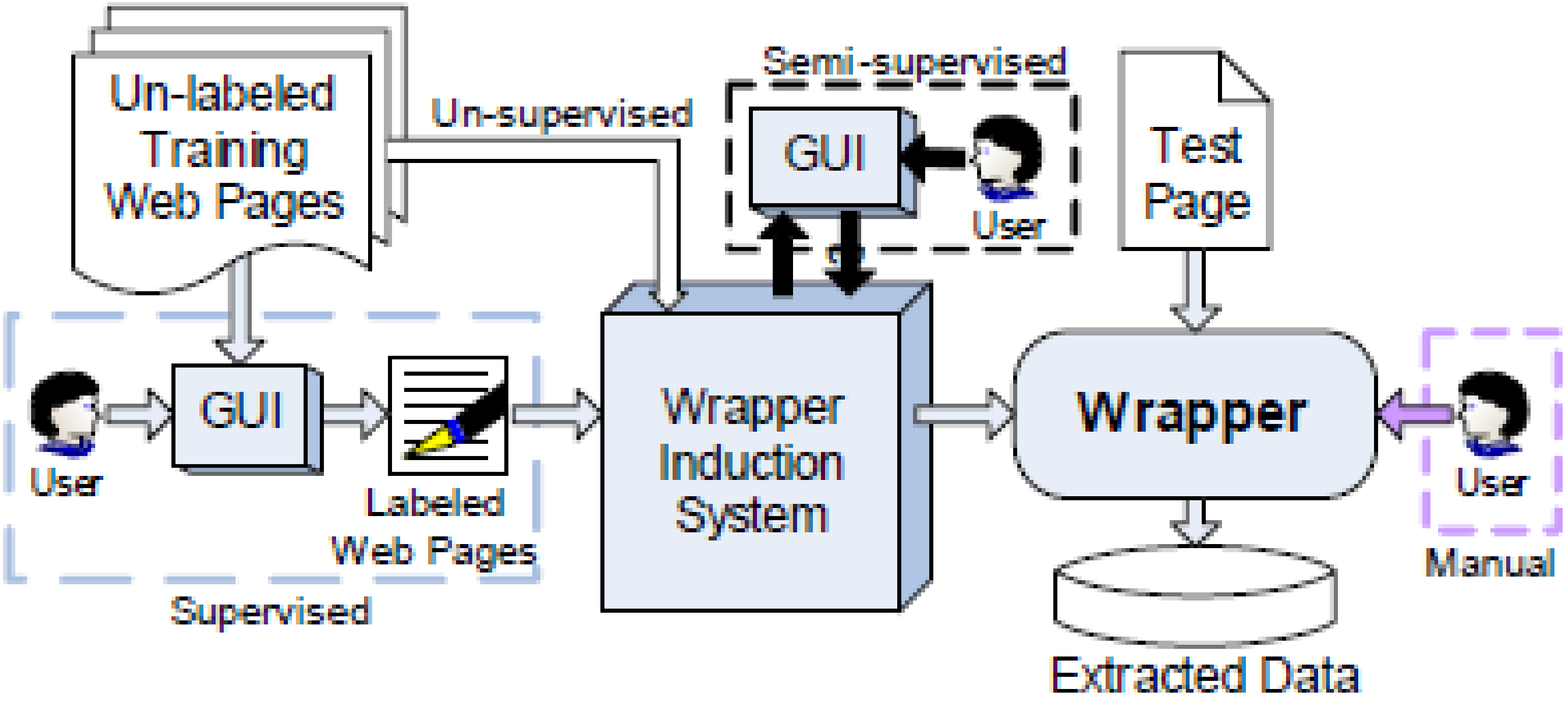
Restaurant Name: **Good Noodles**

- 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

Wrapper Induction

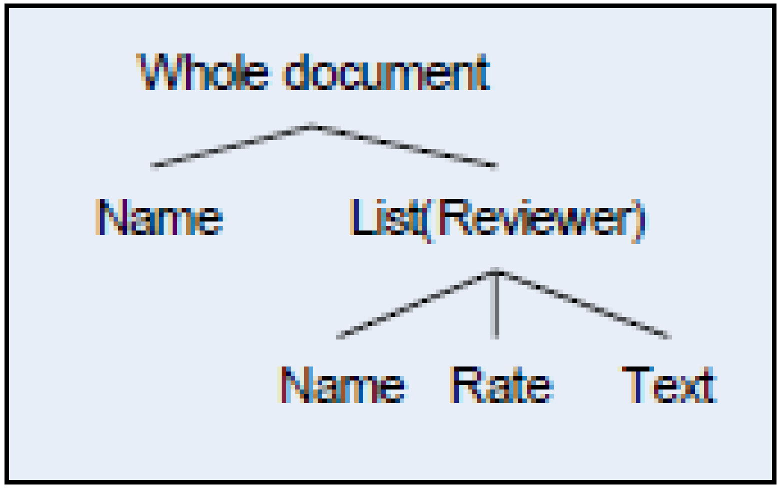
- Using machine learning to generate extraction rules.
  - The user marks the target items in a few training pages.
  - The system learns extraction rules from these pages.
  - The rules are applied to extract items from other pages.
- Many wrapper induction systems, e.g.,
  - WIEN (Kushmerick et al, 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).

A General View of Wrapper Induction (WI) systems.

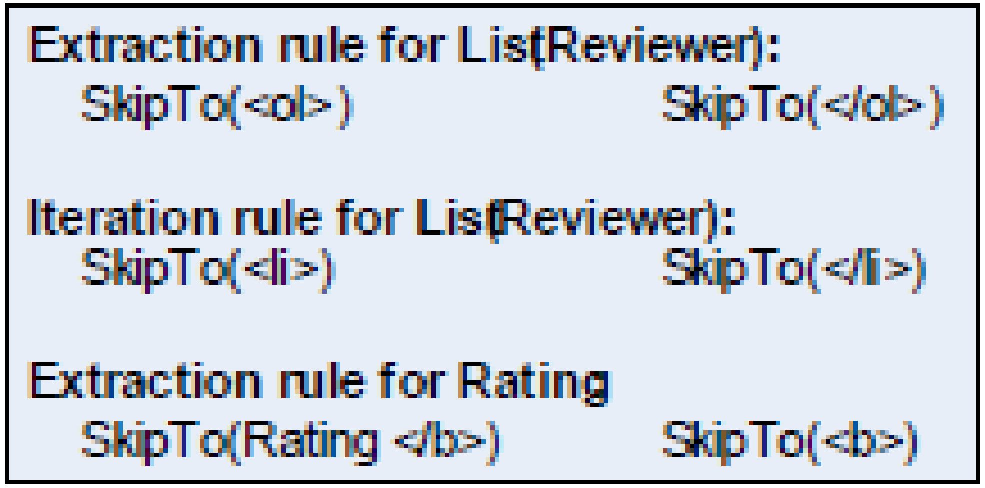


Source: Chang et al. (2006), A Survey of Web Information Extraction Systems

Example of an EC tree (a) and a Stalker extraction rule (b).



(a)


Extraction rule for List(Reviewer):
SkipTo(<ol>) SkipTo(</ol>)

Iteration rule for List(Reviewer):
SkipTo(<li>) SkipTo(</li>)

Extraction rule for Rating
SkipTo(Rating </b>) SkipTo(<b>)

(b)

EC tree

Stalker extraction rule

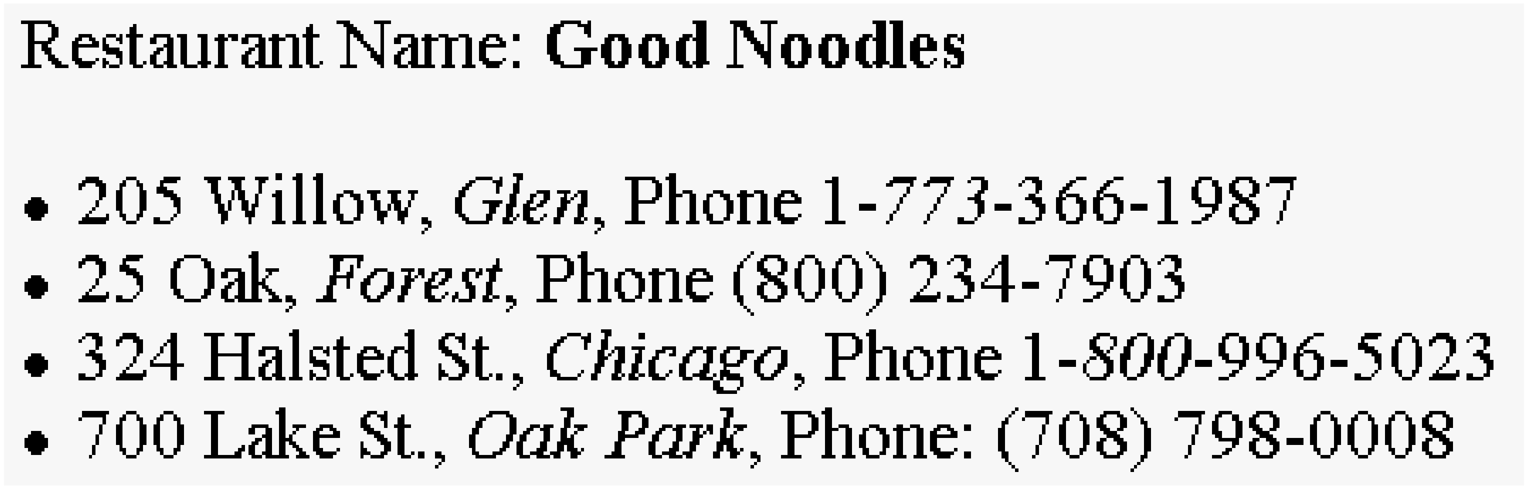
Source: Chang et al. (2006), A Survey of Web Information Extraction Systems

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.

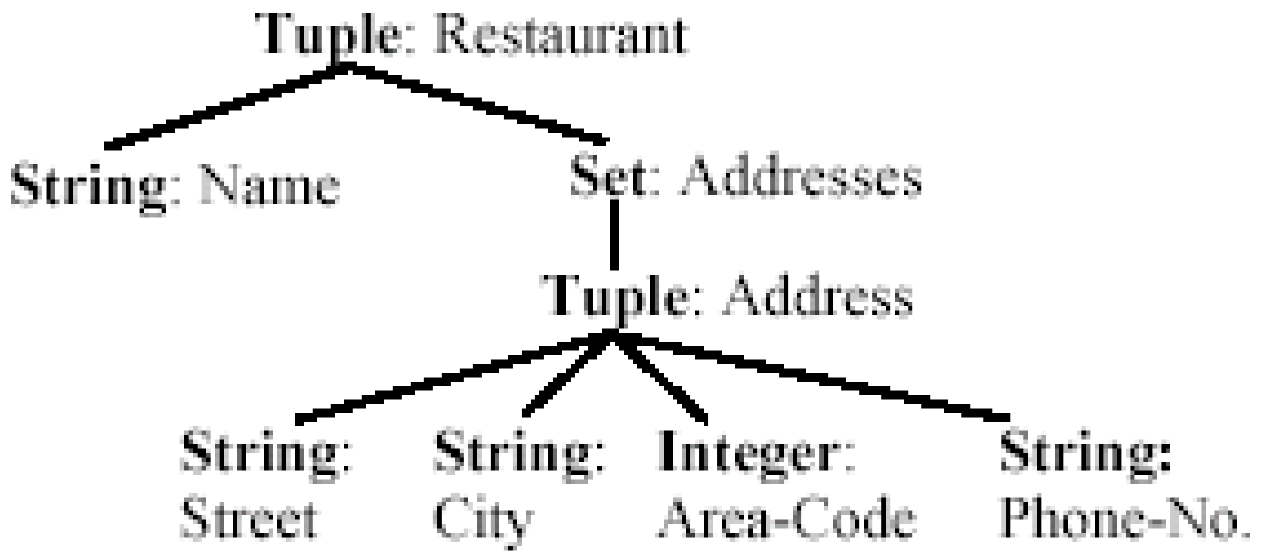
- Each target item is extracted using two rules
  - A start rule for detecting the beginning of the target item.
  - A end rule for detecting the ending of the target item.

Hierarchical representation: type tree



Restaurant Name: **Good Noodles**

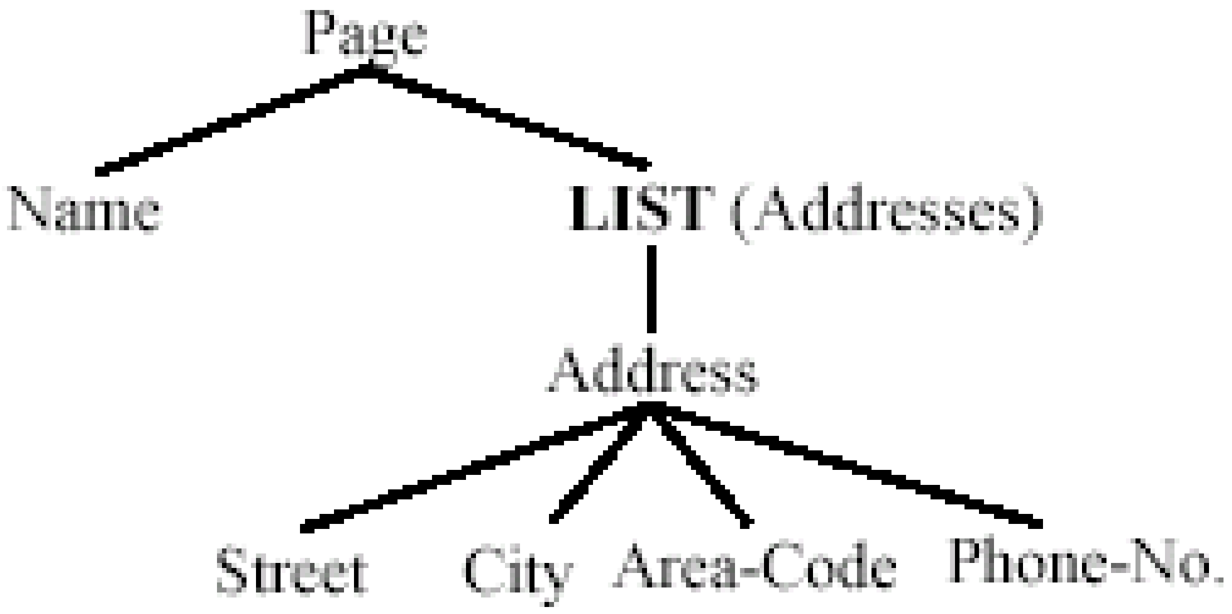
- 205 Willow, *Glen*, Phone 1-*773*-366-1987
- 25 Oak, *Forest*, Phone (800) 234-7903
- 324 Halsted St., *Chicago*, Phone 1-*800*-996-5023
- 700 Lake St., *Oak Park*, Phone: (708) 798-0008



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

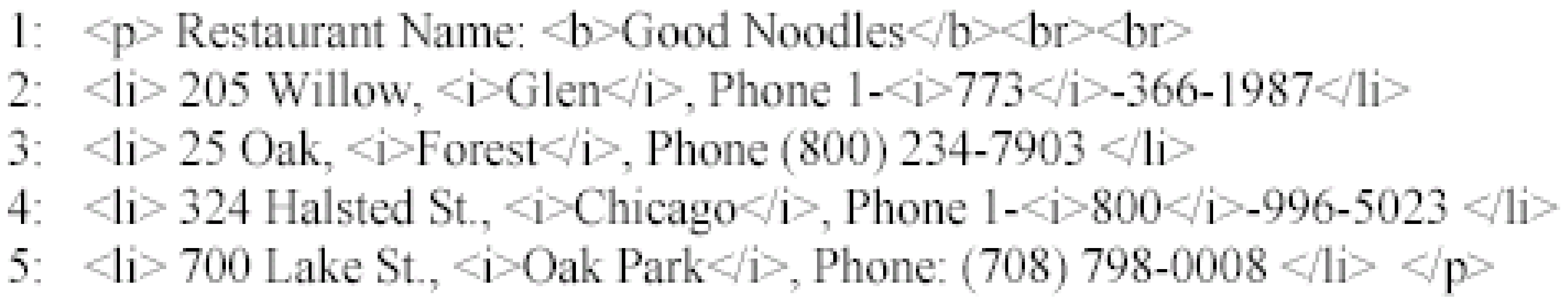
An example

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

  R1: *SkipTo*(<b>) // 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 <b> tag. <b> is a landmark.
- Similarly, to identify the end of the restaurant name, we use:

  R2: *SkipTo*(</b>) // end rule



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

Rules are not unique

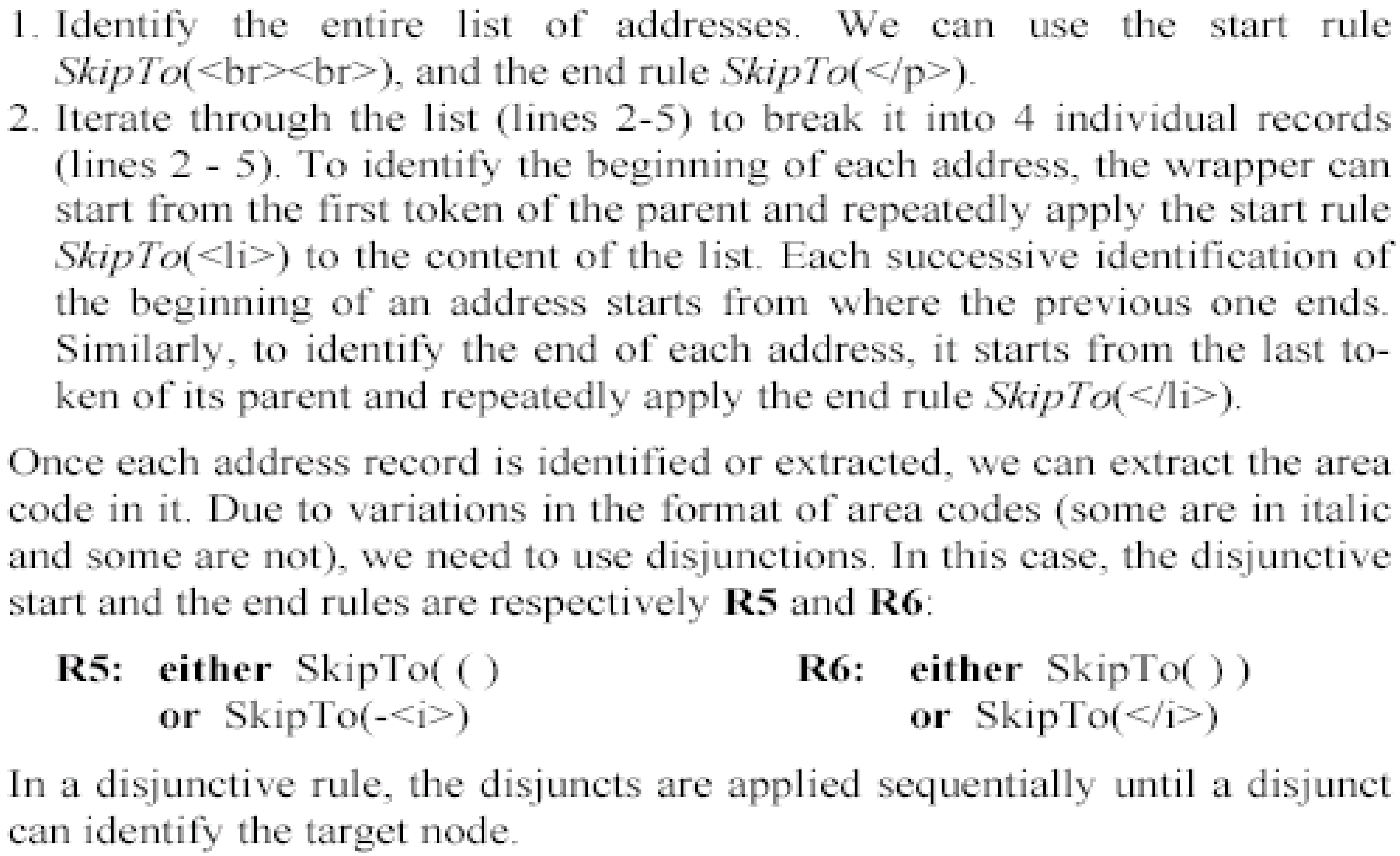
- Note that a rule may not be unique. For example, we can also use the following rules to identify the beginning of the name:

  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*(</li>).

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**:

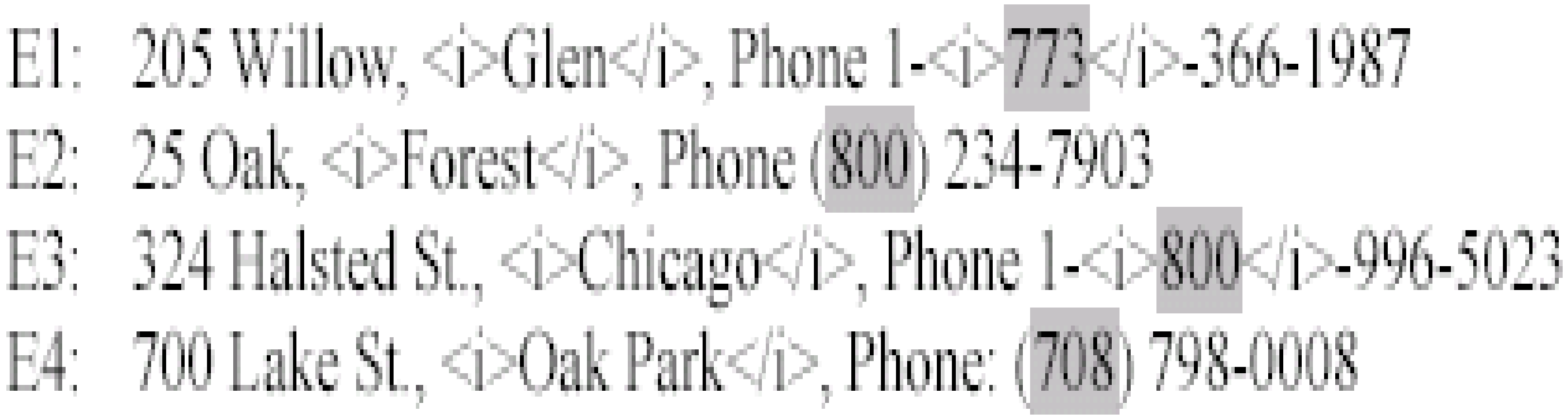
```
R5:  either SkipTo( ( )                R6:  either SkipTo( ) )
     or  SkipTo(-<i>)                       or  SkipTo(</i>)
```

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.

Example: Extract area codes



E1: 205 Willow, <i>Glen</i>, Phone 1-<i>773</i>-366-1987
E2: 25 Oak, <i>Forest</i>, Phone (800) 234-7903
E3: 324 Halsted St., <i>Chicago</i>, Phone 1-<i>800</i>-996-5023
E4: 700 Lake St., <i>Oak Park</i>, Phone: (708) 798-0008

Fig. 9. Training examples: four addresses with labeled area codes

Example

- For the example E2 of Fig. 9, the following candidate disjuncts are generated:

  D1:SkipTo(**(**)

  D2:SkipTo(_*Punctuation*_)

- D1 is selected by BestDisjunct
- D1 is a perfect disjunct.
- The first iteration of LearnRule() ends. E2 and E4 are removed

The next iteration of LearnRule

- The next iteration of LearnRule() is left with E1 and E3.
- LearnDisjunct() will select E1 as the Seed Two candidates are then generated:

  D3: SkipTo(**<i>**)

  D4: SkipTo(*_HtmlTag_*)
- Both these two candidates match early in the uncovered examples, E1 and E3. Thus, they cannot uniquely locate the positive items.
- Refinement is needed.

Refinement

- To specialize a disjunct by adding more **terminals** to it.
- A **terminal** means a token or one of its matching wildcards.
- We hope the refined version will be able to uniquely identify the positive items in some examples without matching any negative item in any example in *E*.
- Two types of refinement
  - Landmark refinement
  - Topology refinement

Landmark refinement

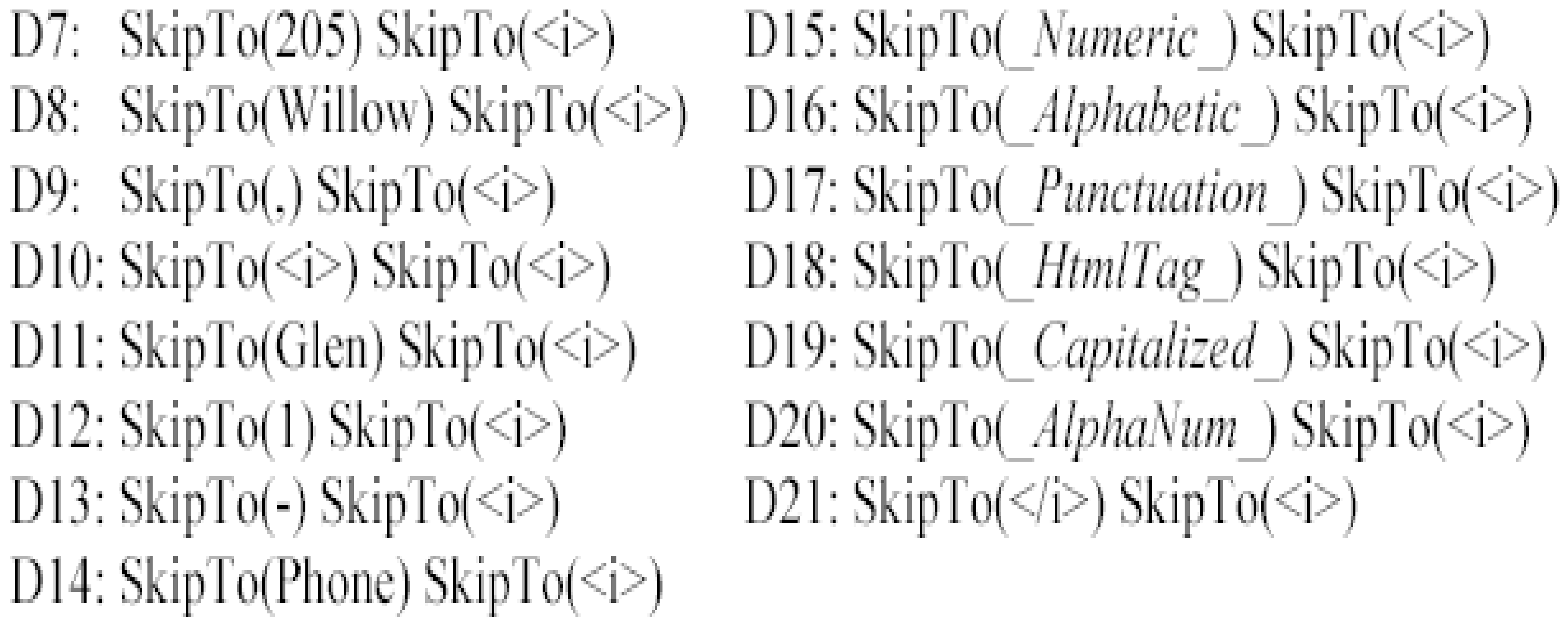
- Landmark refinement: Increase the size of a landmark by concatenating a terminal.
  - E.g.,

D5: SkipTo(- <i>)

D6: SkipTo(_*Punctuation*_ <i>)

Topology refinement

- Topology refinement: Increase the number of landmarks by adding 1-terminal landmarks, i.e., *t* and its matching wildcards



D7: SkipTo(205) SkipTo(<i>)
D8: SkipTo(Willow) SkipTo(<i>)
D9: SkipTo(.) SkipTo(<i>)
D10: SkipTo(<i>) SkipTo(<i>)
D11: SkipTo(Glen) SkipTo(<i>)
D12: SkipTo(1) SkipTo(<i>)
D13: SkipTo(-) SkipTo(<i>)
D14: SkipTo(Phone) SkipTo(<i>)
D15: SkipTo(_*Numeric*_) SkipTo(<i>)
D16: SkipTo(_*Alphabetic*_) SkipTo(<i>)
D17: SkipTo(_*Punctuation*_) SkipTo(<i>)
D18: SkipTo(_*HtmlTag*_) SkipTo(<i>)
D19: SkipTo(_*Capitalized*_) SkipTo(<i>)
D20: SkipTo(_*AlphaNum*_) SkipTo(<i>)
D21: SkipTo(</i>) SkipTo(<i>)

Refining, specializing

```
Procedure Refine($D$, $Seed$)
1  $D$ is a consecutive landmarks $(l_0, l_1, \ldots, l_n)$;
2  $TopologyRefs \leftarrow LandmarkRefs \leftarrow \emptyset$;
3  for $i = 1$ to $n$ do        // $t_0$ or $t_1$ below may be null
4      for each sequence $s = t_0\, l_i\, t_1$ before the target item in $Seed$ do
5          $LandmarkRefs \leftarrow LandmarkRefs \cup \{(l_0, \ldots, l_{i-1}, t_0\, l_i, \ldots, l_n)\} \cup$
               $\{(l_0, \ldots, l_{i-1}, x\, l_i, \ldots, l_n) \mid x$ is a wildcard that matches $t_0\}$
               $\cup\ \{(l_0, \ldots, l_i\, t_1, l_{i+1}, \ldots, l_n)\} \cup$
               $\{(l_0, \ldots, l_i\, x, l_{i+1}, \ldots, l_n) \mid x$ is a wildcard that matches $t_1\}$
6      for each token $t$ between $l_{i-1}$ and $l_i$ before the target item in $Seed$ do
7          $ToplogyRefs \leftarrow TopologyRefs \cup \{(l_0, \ldots, l_i, t, l_{i+1}, \ldots, l_n)\} \cup$
               $\{(l_0, \ldots, l_i, x, l_{i+1}, \ldots, l_n)\} \mid x$ is a wildcard that matches $t\}$
8  return $TopologyRefs \cup LandmarkRefs$
```

Fig. 12. This function refines a disjunct to generate more specialized candidates

The final solution

- We can see that **D5**, **D10, D12, D13**, **D14**, **D15**, **D18** and **D21** match correctly with E1 and E3 and fail to match on E2 and E4.
- Using BestDisjunct in Fig. 13, **D5** is selected as the final solution as it has longest last landmark (- <i>).
- **D5** is then returned by **LearnDisjunct()**.
- Since all the examples are covered, LearnRule() returns the disjunctive (start) rule either **D1** or **D5**

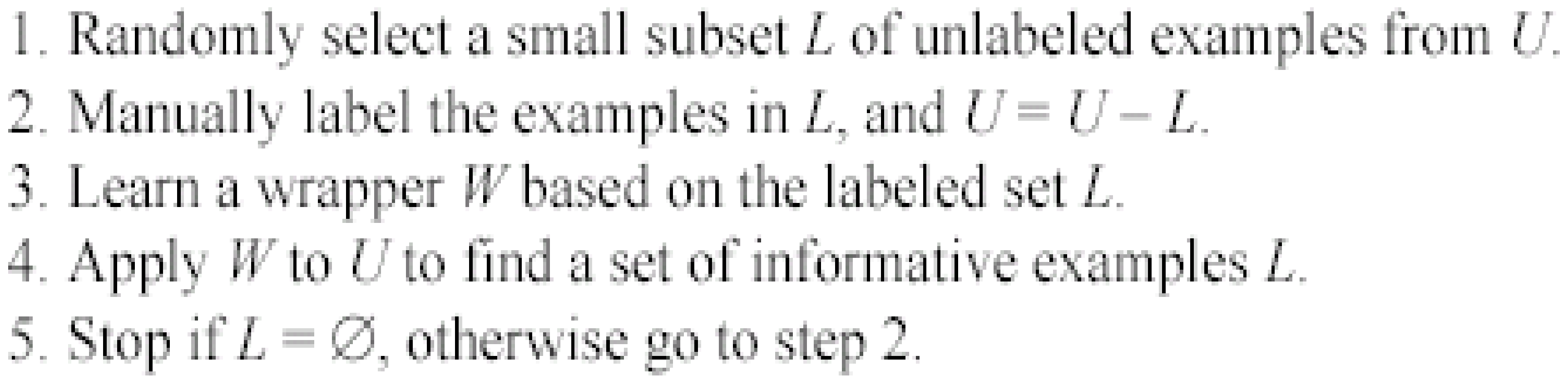
R7: either *SkipTo*(( )
or *SkipTo*(- <i>)

Identifying Informative Examples

- Wrapper learning needs manual labeling of training examples.
- To ensure accurate learning, a large number of training examples are needed.
- Manual labeling labor intensive and time consuming.
- Is it possible to automatically select (unlabelled) examples that are informative for the user to label.
  - Clearly, examples of the same formatting are of limited use.
  - Examples that represent exceptions are informative as they are different from already labeled examples.

Active learning

- help identify informative unlabeled examples in learning automatically.



```
1. Randomly select a small subset $L$ of unlabeled examples from $U$.
2. Manually label the examples in $L$, and $U = U - L$.
3. Learn a wrapper $W$ based on the labeled set $L$.
4. Apply $W$ to $U$ to find a set of informative examples $L$.
5. Stop if $L = \emptyset$, otherwise go to step 2.
```

Active learning: co-testing

- Co-testing exploits the fact that there are often multiple ways of extracting the same item.
- The system can learn different rules, **forward** and **backward rules**, to locate the same item.
  - *forward rules*
    - consume tokens from the beginning of the example to the end.
  - *backward rules*
    - consume tokens from the end of the example to the beginning.

Co-testing (cont …)

- Given an unlabeled example, both the forward rule and backward rule are applied.
- If the two rules disagree on the beginning of a target item in the example, this example is given to the user to label.
- Intuition: When the two rules agree, the extraction is very likely to be correct.
  - When the two rules do not agree on the example, one of them must be wrong.
  - By giving the user the example to label, we obtain an informative training example.

Wrapper maintenance

- Wrapper verification: If the site changes, does the wrapper know the change?
- Wrapper repair: If the change is correctly detected, how to automatically repair the wrapper?
- One way to deal with both problems is to learn the characteristic patterns of the target items.
- These patterns are then used to monitor the extraction to check whether the extracted items are correct.

Wrapper maintenance (cont …)

- Re-labeling: If they are incorrect, the same patterns can be used to locate the correct items assuming that the page changes are minor formatting changes.
- Re-learning: re-learning produces a new wrapper.

- Difficult problems: These two tasks are extremely difficult because it often needs contextual and semantic information to detect changes and to find the new locations of the target items.
- Wrapper maintenance is still an active research area.

Automatic wrapper generation

- Two main shortcomings of Wrapper induction (supervised):
  - It is unsuitable for a large number of sites due to the manual labeling effort.
  - Wrapper maintenance is very costly. The Web is a dynamic environment. Sites change constantly. Since rules learnt by wrapper induction systems mainly use formatting tags, if a site changes its formatting templates, existing extraction rules for the site become invalid.

Unsupervised learning is possible

- Automatic extraction is possible because data records (tuple instances) in a Web site are usually encoded using a very small number of fixed templates.
- It is possible to find these templates by mining repeated patterns.

Templates as regular expressions

- A regular expression can be naturally used to model the HTML encoded version of a nested type.
- Given an alphabet of symbols *Σ* and a special token "*#text*" that is not in *Σ*,
  - a *regular expression* over *Σ* is a string over *Σ* ∪ {*#text*, *, ?, |, (,)} defined as follows:

Regular expressions

- The empty string ε and all elements of $\Sigma \cup \{\#text\}$ are regular expressions.
- If A and B are regular expressions, then AB, $(A|B)$ and $(A)^?$ are regular expressions, where $(A|B)$ stands for A or B and $(A)^?$ stands for $(A|\varepsilon)$.
- If A is a regular expression, $(A)^*$ is a regular expression, where $(A)^*$ stands for ε or A or AA or ...

We also use $(A)+$ as a shortcut for $A(A)^*$, which can be used to model the set type of a list of tuples. $(A)^?$ indicates that A is optional. $(A|B)$ represents a disjunction.

Regular expressions and extraction

- Regular expressions are often employed to represent templates (or encoding functions).
- However, templates can also be represented as string or tree patterns as we will see later.
- Extraction:
  - Given a regular expression, a nondeterministic finite-state automaton can be constructed and employed to match its occurrences in string sequences representing Web pages.
  - In the process, data items can be extracted, which are text strings represented by *#text*.

Some useful algorithms

- The key is to finding the encoding template from a collection of encoded instances of the same type.
- A natural way to do this is to detect repeated patterns from HTML encoding strings.
- String edit distance and tree edit distance are obvious techniques for the task.

String edit distance

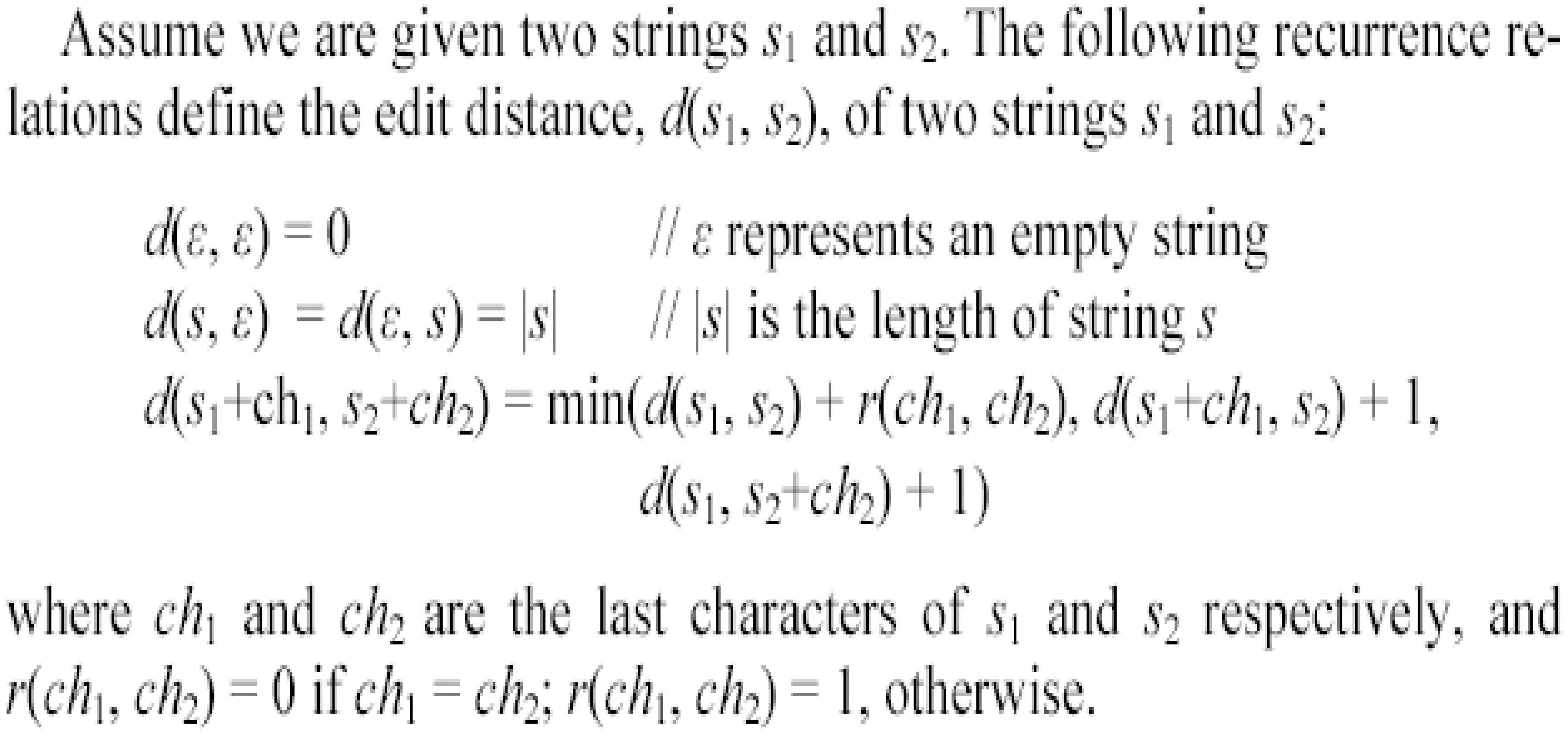
- String edit distance (Levenshtein distance):
  - The most widely used string comparison technique.
- The **edit distance** of two strings, *s*1 and *s*2, is defined as the minimum number of *point mutations* required to change *s*1 into *s*2, where a point mutation is one of:
  - (1) change a letter,
  - (2) insert a letter, and
  - (3) delete a letter.

An Example of Levenshtein distance

- The Levenshtein distance between "kitten" and "sitting" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:
  - **k**itten → **s**itten (substitution of "s" for "k")
  - sitt**e**n → sitt**i**n (substitution of "i" for "e")
  - sittin → sittin**g** (insertion of "g" at the end).

Source: http://en.wikipedia.org/wiki/Levenshtein_distance

String edit distance (definition)



Assume we are given two strings s_1 and s_2. The following recurrence relations define the edit distance, $d(s_1, s_2)$, of two strings s_1 and s_2:

$$
\begin{aligned}
&d(\varepsilon, \varepsilon) = 0 && // \ \varepsilon \text{ represents an empty string} \\
&d(s, \varepsilon) = d(\varepsilon, s) = |s| && // \ |s| \text{ is the length of string } s \\
&d(s_1+\mathrm{ch}_1, s_2+ch_2) = \min(d(s_1, s_2) + r(ch_1, ch_2), d(s_1+ch_1, s_2) + 1, \\
&\qquad d(s_1, s_2+ch_2) + 1)
\end{aligned}
$$

where ch_1 and ch_2 are the last characters of s_1 and s_2 respectively, and $r(ch_1, ch_2) = 0$ if $ch_1 = ch_2$; $r(ch_1, ch_2) = 1$, otherwise.

An example

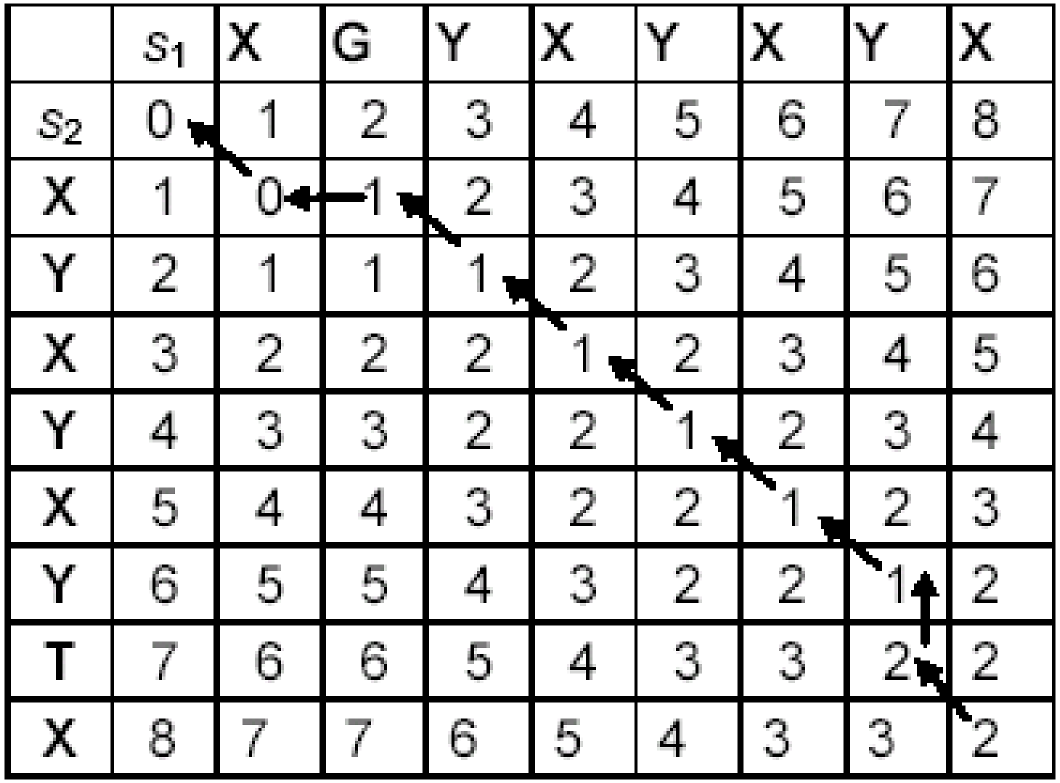
Example 1: We want to compute the edit distance and find the alignment of the following two strings:

s_1:	X	G	Y	X	Y	X	Y	X
s_2:	X	Y	X	Y	X	Y	T	X

The edit distance matrix and back trace path

alignment

s_1:	X	G	Y	X	Y	X	Y	–	X
s_2:	X	–	Y	X	Y	X	Y	T	X



	s_1	X	G	Y	X	Y	X	Y	X
s_2	0	1	2	3	4	5	6	7	8
X	1	0	1	2	3	4	5	6	7
Y	2	1	1	1	2	3	4	5	6
X	3	2	2	2	1	2	3	4	5
Y	4	3	3	2	2	1	2	3	4
X	5	4	4	3	2	2	1	2	3
Y	6	5	5	4	3	2	2	1	2
T	7	6	6	5	4	3	3	2	2
X	8	7	7	6	5	4	3	3	2

Tree Edit Distance

- Tree edit distance between two trees *A* and *B* (*labeled ordered rooted trees*) is the cost associated with the minimum set of operations needed to transform *A* into *B*.
- The set of operations used to define tree edit distance includes three operations:
  - node removal,
  - node insertion, and
  - node replacement.

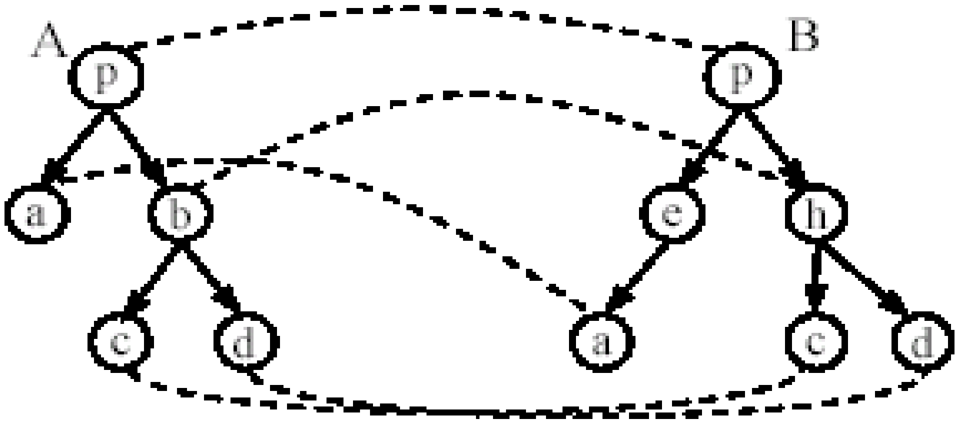
  A cost is assigned to each of the operations.

Definition

Let X be a tree and let $X[i]$ be the ith node of tree X in a preorder walk of the tree. A *mapping* M between a tree A of size n_1 and a tree B of size n_2 is a set of ordered pairs (i, j), one from each tree, satisfying the following conditions for all $(i_1, j_1), (i_2, j_2) \in M$:

(1) $i_1 = i_2$ *iff* $j_1 = j_2$;
(2) $A[i_1]$ is on the left of $A[i_2]$ *iff* $B[j_1]$ is on the left $B[j_2]$;
(3) $A[i_1]$ is an ancestor of $A[i_2]$ *iff* $B[j_1]$ is an ancestor of $B[j_2]$.

Intuitively, the definition requires that each node appears no more than once in a mapping and the order among siblings and the hierarchical relation among nodes are both preserved. Fig. 16 shows a mapping example.



Multiple alignment

- Pairwise alignment is not sufficient because a web page usually contain more than one data records.
- We need multiple alignment.
- Two techniques
  - Center Star method
  - Partial tree alignment.

Building DOM trees

- DOM (Document Object Model) tree (tag tree) building from HTML pages is a necessary step for data extraction.
- Using Tags Alone
  - Most HTML tags work in pairs. Within each corresponding tag-pair, there can be other pairs of tags, resulting in a nested structure.
  - Building a DOM tree from a page using its HTML code is thus natural.
- In the tree, each pair of tags is a **node**, and the nested tags within it are the **children** of the node.

Two steps to build a tree

- **HTML code cleaning**:
  - Some tags do not require closing tags (e.g., <li>, <hr> and <p>) although they have closing tags.
  - Additional closing tags need to be inserted to ensure all tags are balanced.
  - Ill-formatted tags need to be fixed. One popular program is called **Tidy**, which can be downloaded from http://tidy.sourceforge.net/.
- **Tree building**:
  - simply follow the nested blocks of the HTML tags in the page to build the DOM tree. It is straightforward.

Building Tree Using Tags & Visual cues

- Correcting errors in HTML can be hard.
- There are also dynamically generated pages with scripts.
- Visual information comes to the rescue.
- As long as a browser can render a page correct, a tree can be built correctly.
  - Each HTML element is rendered as a rectangle.
  - Containments of rectangles representing nesting.

An example

```
<table>
  <tr>
      <td>…</td>
      <td>…</td>
  </tr>
  <tr>
      <td>…</td>
      <td>…</td>
  </tr>
</table>
```

left	right	top	bottom
100	300	200	400
100	300	200	300
100	200	200	300
200	300	200	300
100	300	300	400
100	200	300	400
200	300	300	400

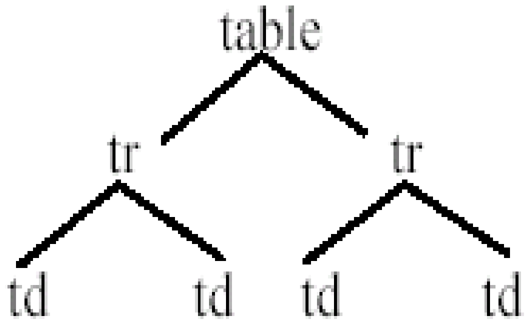


Fig. 23. A HTML code segment, boundary coordinates and the resulting tree

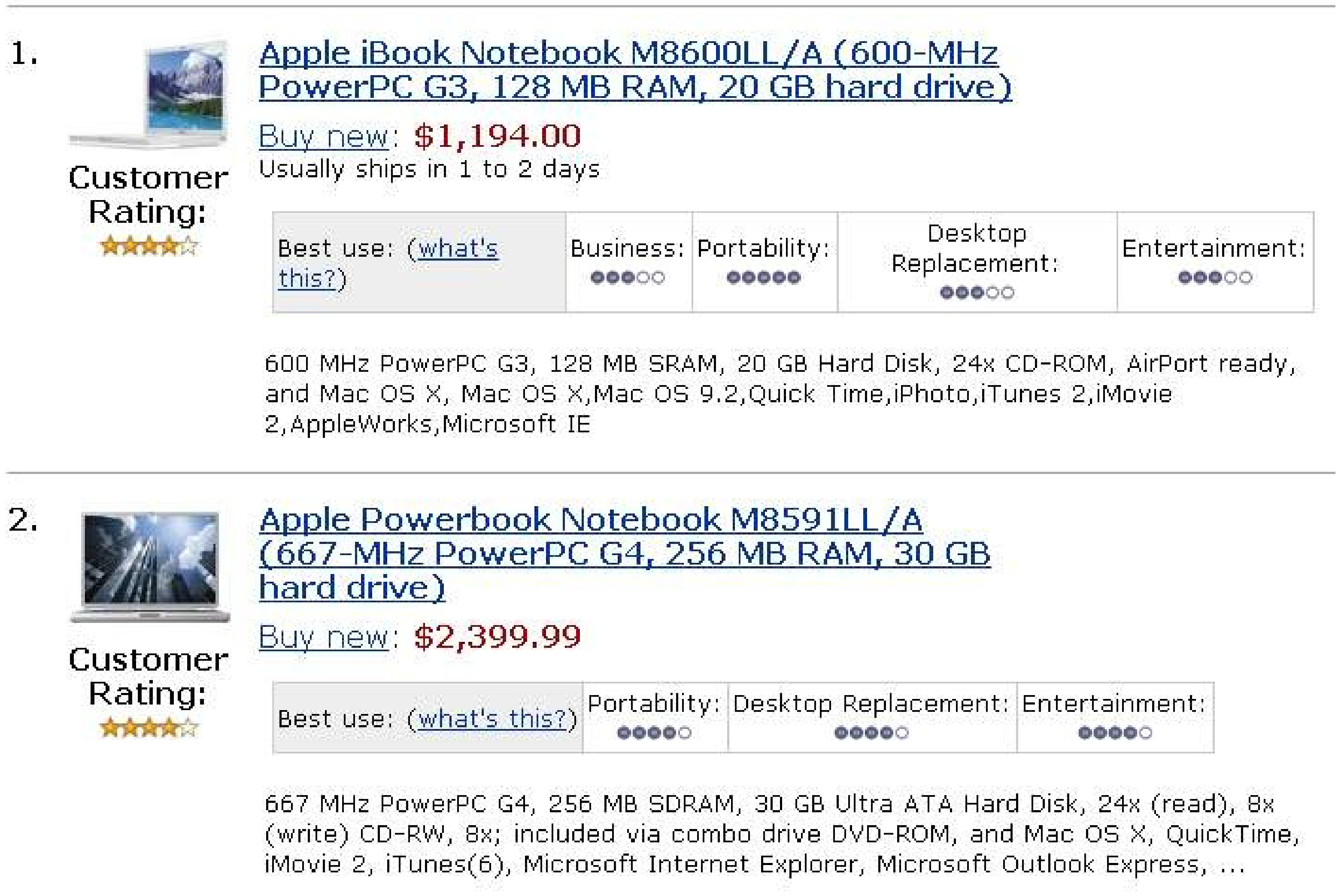
Extraction Given a List Page: Flat Data Records

- Given a single list page with multiple data records,
  - Automatically segment data records
  - Extract data from data records.
- Since the data records are flat (no nested lists), string similarity or tree matching can be used to find similar structures.
  - Computation is a problem
  - A data record can start anywhere and end anywhere

Two important observations

- **Observation 1**: A group of data records that contains descriptions of a set of similar objects are typically presented in a contiguous region of a page and are formatted using similar HTML tags. Such a region is called a **data region**.
- **Observation 2**: A set of data records are formed by some child sub-trees of the same parent node.

An example



1. **Apple iBook Notebook M8600LL/A (600-MHz PowerPC G3, 128 MB RAM, 20 GB hard drive)**

Customer Rating: ★★★★☆

Buy new: $1,194.00
Usually ships in 1 to 2 days

Best use: (what's this?)	Business:	Portability:	Desktop Replacement:	Entertainment:
	●●●○○	●●●●●	●●●○○	●●●○○

600 MHz PowerPC G3, 128 MB SRAM, 20 GB Hard Disk, 24x CD-ROM, AirPort ready, and Mac OS X, Mac OS X,Mac OS 9.2,Quick Time,iPhoto,iTunes 2,iMovie 2,AppleWorks,Microsoft IE

2. **Apple Powerbook Notebook M8591LL/A (667-MHz PowerPC G4, 256 MB RAM, 30 GB hard drive)**

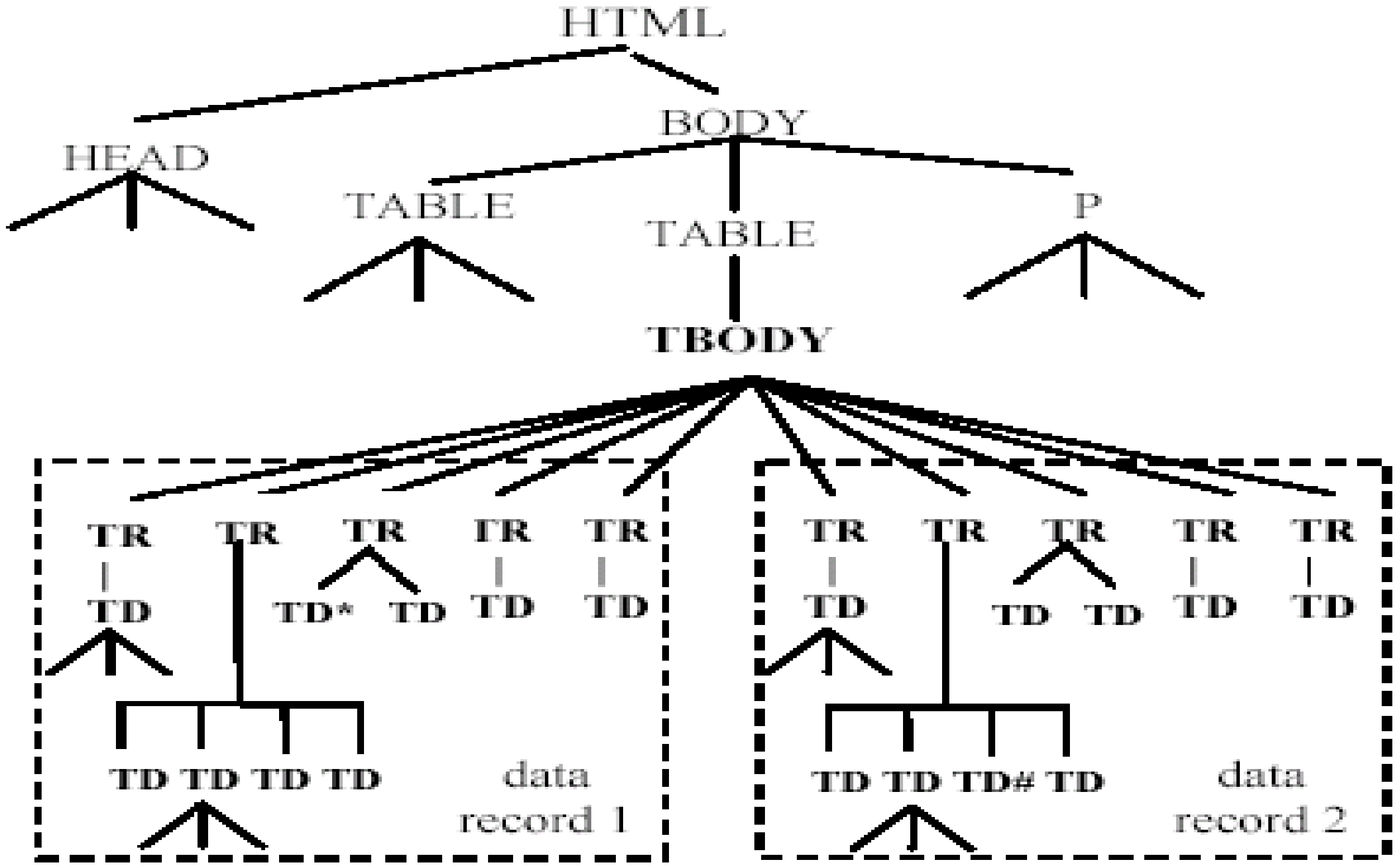
Customer Rating: ★★★★☆

Buy new: $2,399.99

Best use: (what's this?)	Portability:	Desktop Replacement:	Entertainment:
	●●●●○	●●●●○	●●●●○

667 MHz PowerPC G4, 256 MB SDRAM, 30 GB Ultra ATA Hard Disk, 24x (read), 8x (write) CD-RW, 8x; included via combo drive DVD-ROM, and Mac OS X, QuickTime, iMovie 2, iTunes(6), Microsoft Internet Explorer, Microsoft Outlook Express, ...

The DOM tree


HTML
HEAD
BODY
TABLE
TABLE
P
TBODY
TR
TD
TR
TD TD TD TD
TR
TD*
TD
TR
TD
TR
TD
data record 1
TR
TD
TR
TD TD TD# TD
TR
TD
TD
TR
TD
TR
TD
data record 2

The Approach

Given a page, three steps:

- Building the HTML Tag Tree
  - – Erroneous tags, unbalanced tags, etc
- Mining Data Regions
  - – Spring matching or tree matching
- Identifying Data Records

Rendering (or visual) information is very useful in the whole process

Extract Data from Data Records

- Once a list of data records is identified, we can align and extract data items from them.
- Approaches (align multiple data records):
  - Multiple string alignment
    - Many ambiguities due to pervasive use of table related tags.
  - Multiple tree alignment (partial tree alignment)
    - Together with visual information is effective

Generating extraction patterns and data extraction

- Once data records in each data region are discovered, we align them to produce an extraction pattern that can be used to extract data from the current page and also other pages that use the same encoding template.
- P**artial tree alignment algorithm** is just for the purpose.
- Visual information can help in various ways

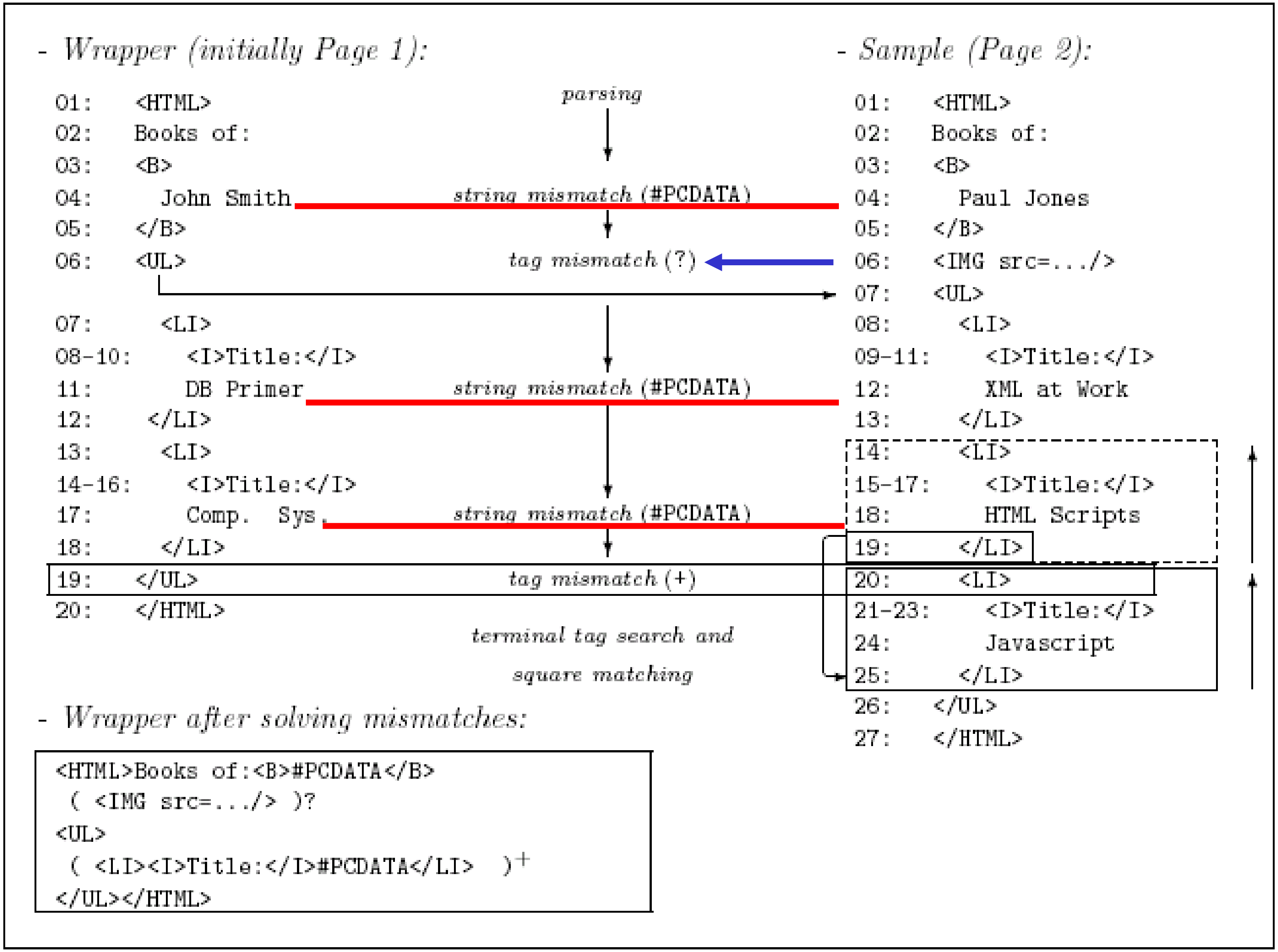
Extraction Given a List Page: Nested Data Records

- The most general case
  - Nested data records
- Problem with the previous method
  - not suitable for nested data records, i.e., data records containing nested lists.
  - Since the number of elements in the list of each data record can be different, using a fixed threshold to determine the similarity of data records will not work.

Solution idea

- The problem, however, can be dealt with as follows.
  - Instead of traversing the DOM tree top down, we can traverse it post-order.
  - This ensures that nested lists at lower levels are found first based on repeated patterns before going to higher levels.
  - When a nested list is found, its records are **collapsed** to produce a single template.
  - This template replaces the list of nested data records.
- When comparisons are made at a higher level, the algorithm only sees the template. Thus it is treated as a flat data record.

A wrapper generation example



- *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:        <LI>
21-23:        <I>Title:</I>
24:           Javascript
25:        </LI>
26:     </UL>
27:     </HTML>
```

parsing
string mismatch (#PCDATA)
tag mismatch (?)
string mismatch (#PCDATA)
string mismatch (#PCDATA)
tag mismatch (+)
terminal tag search and square matching

- *Wrapper after solving mismatches:*

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

Summary

Wrapper induction

- Advantages:
  - Only the target data are extracted as the user can label only data items that he/she is interested in.
  - Due to manual labeling, there is no integration issue for data extracted from multiple sites as the problem is solved by the user.
- Disadvantages:
  - It is not scalable to a large number of sites due to significant manual efforts. Even finding the pages to label is non-trivial.
  - Wrapper maintenance (verification and repair) is very costly if the sites change frequently.

Summary (cont ...)

Automatic extraction

- Advantages:
  - It is scalable to a huge number of sites due to the automatic process.
  - There is little maintenance cost.
- Disadvantages:
  - It may extract a large amount of unwanted data because the system does not know what is interesting to the user. Domain heuristics or manual filtering may be needed to remove unwanted data.
  - Extracted data from multiple sites need integration, i.e., their schemas need to be matched.

Summary (cont…)

- In terms of extraction accuracy, it is reasonable to assume that wrapper induction is more accurate than automatic extraction. However, there is no reported comparison.
- Applications
  - Wrapper induction should be used in applications in which the number of sites to be extracted and the number of templates in these sites are not large.
  - Automatic extraction is more suitable for large scale extraction tasks which do not require accurate labeling or integration.

References

- Bing Liu (2011) , “Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data,” 2nd Edition, Springer. http://www.cs.uic.edu/~liub/WebMiningBook.html
- Chang, C.-H., Kayed, M., Girgis, M. R., and Shaalan, K. F (2006), “A Survey of Web Information Extraction Systems,” IEEE Transactions on Knowledge and Data Engineering (TKDE), Vol. 18, No. 10, pp. 1411–1428.