WWW-2005 Tutorial

Web Content Mining

Bing Liu
Department of Computer Science
University of Illinois at Chicago (UIC)
liub@cs.uic.edu
http://www.cs.uic.edu/~liub

Introduction

- The Web is perhaps the single largest data source in the world.
- Web mining aims to extract and mine useful knowledge from the Web.
- A multidisciplinary field:
  - data mining, machine learning, natural language processing, statistics, databases, information retrieval, multimedia, etc.
- Due to the heterogeneity and lack of structure of Web data, mining is a challenging task.

The Web: Opportunities & Challenges

- Web offers an unprecedented opportunity and challenge to data mining
  - The amount of information on the Web is huge, and easily accessible.
  - The coverage of Web information is very wide and diverse. One can find information about almost anything.
  - Information/data of almost all types exist on the Web, e.g., structured tables, texts, multimedia data, etc.
  - Much of the Web information is semi-structured due to the nested structure of HTML code.
  - Much of the Web information is linked. There are hyperlinks among pages within a site, and across different sites.
  - Much of the Web information is redundant. The same piece of information or its variants may appear in many pages.

- The Web is noisy. A Web page typically contains a mixture of many kinds of information, e.g., main contents, advertisements, navigation panels, copyright notices, etc.
- The Web consists of surface Web and deep Web.
  - Surface Web: pages that can be browsed using a browser.
  - Deep Web: databases that can only be accessed through parameterized query interfaces.
- The Web is also about services. Many Web sites and pages enable people to perform operations with input parameters, i.e., they provide services.
- The Web is dynamic. Information on the Web changes constantly. Keeping up with the changes and monitoring the changes are important issues.
- Above all, the Web is a virtual society. It is not only about data, information and services, but also about interactions among people, organizations and automatic systems, i.e., communities.

Web mining

- Web mining generally consists of:
  - Web usage mining: the discovery of user access patterns from Web usage logs.
  - Web structure mining: the discovery of useful knowledge from the structure of hyperlinks.
  - **Web content mining**: **mining, extraction and integration of useful data, information and knowledge from Web page contents**.
- This tutorial focuses on *Web content mining*.

Tutorial topics

- Web content mining is still a large field.
- This tutorial introduces the following topics:
  - Structured data extraction
  - Sentiment classification, analysis and summarization of consumer reviews
  - Information integration and schema matching
  - Knowledge synthesis
  - Template detection and page segmentation
- All those topics have immediate applications.
- Classic topics such as page classification and clustering will not be discussed.

1. Structured Data Extraction

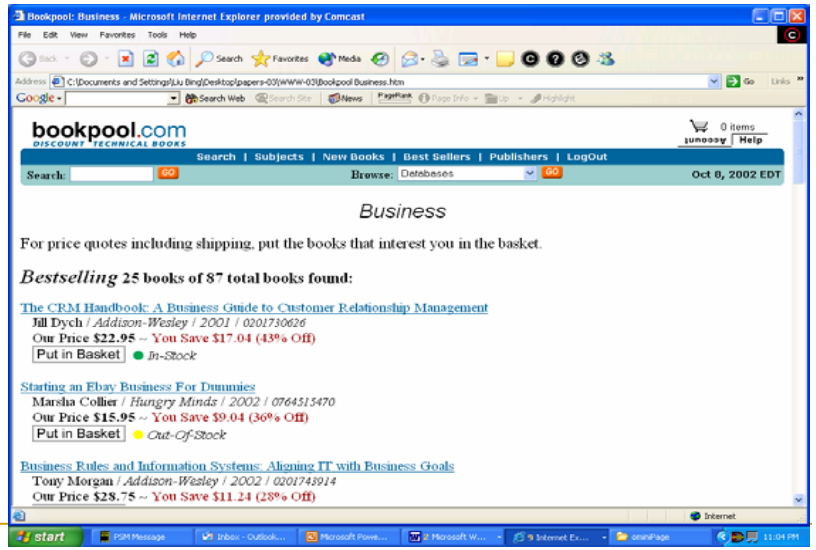
Wrapper induction

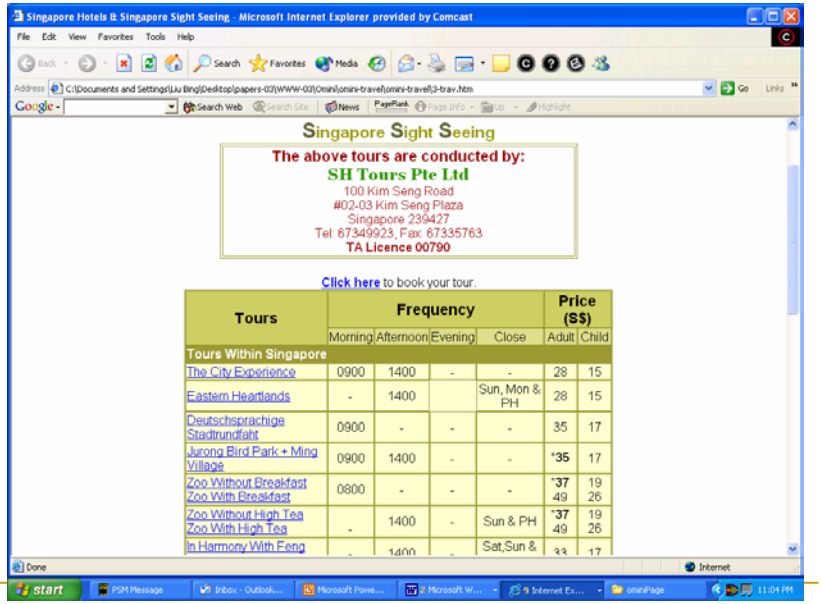
Automatic data extraction

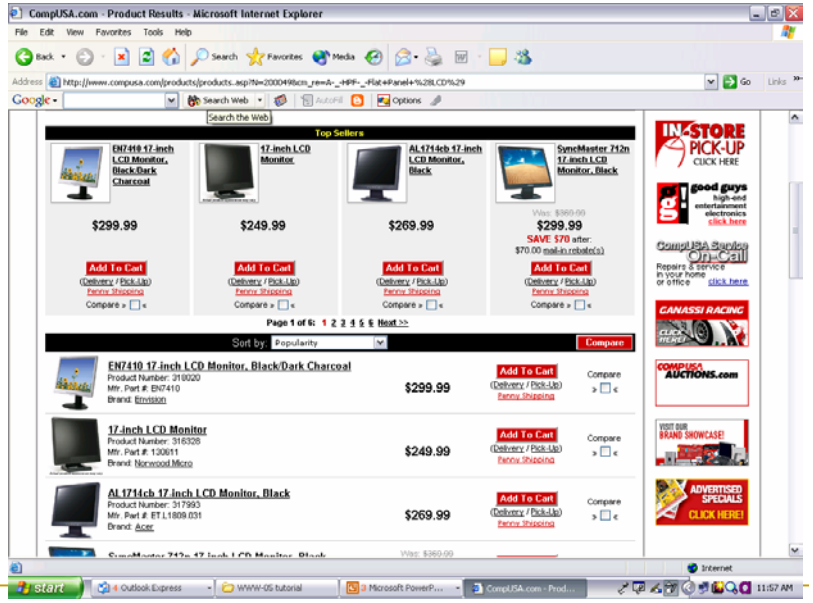
Introduction

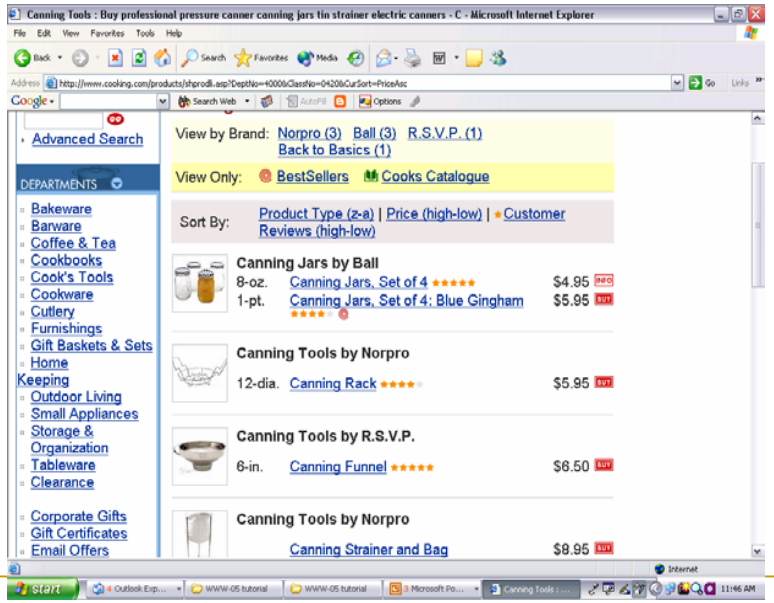
- A large amount of information on the Web is contained in regularly structured data objects.
  - which are data records retrieved from databases.
- Such Web data records are important because
  - they often present the essential information of their host pages, e.g., lists of products and services.
- Applications: integrated and value-added services, e.g.,
  - Comparative shopping, meta-search & query, etc.
- **We introduce:**
  - **Wrapper induction**
  - **automatic extraction**

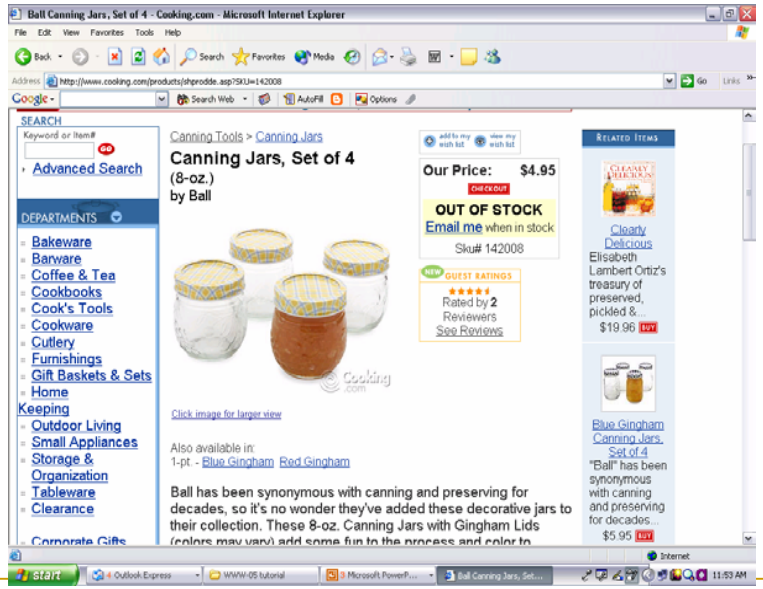
Some Example Pages











Road map

- Wrapper Induction
  - Given a set of manually labeled pages, a machine learning method is applied to learn extraction rules or patterns.
- Automatic data extraction
  - Given a set of positive pages, generate extraction patterns.
  - Given only a single page with multiple data records, generate extraction patterns.

Wrapper induction

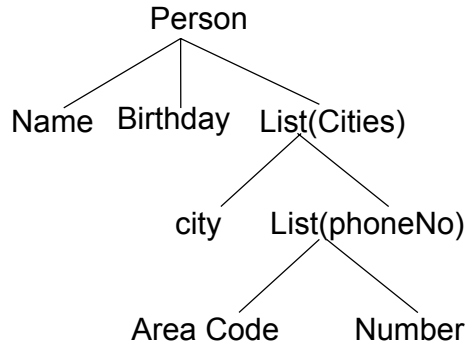
- Using machine learning to generate extraction rules.
  - The user marks the target items in a few training pages.
  - The system learns extraction rules from these pages.
  - The rules are applied to extract target items from other pages.
- Many wrapper induction systems, e.g.,
  - WIEN (Kushmerick et al, IJCAI-97),
  - Softmealy (Hsu and Dung, 1998),
  - Stalker (Muslea et al. Agents-99),
  - BWI (Freitag and McCallum, AAAI-00),
  - WL^2 (Cohen et al. WWW-02).
  - Thresher (Hogue and Karger, WWW-05)
- We will only focus on Stalker, which also has a commercial version, Fetch.

Stalker: A hierarchical wrapper induction system (Muslea et al. Agents-99)

- 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.
- Note: Thanks to Ion Muslea for some clarifications

Hierarchical extraction based on tree

Name: John Smith
Birthday: Oct 5, 1950
Cities:
  Chicago:
    (312) 378 3350
    (312) 755 1987
  New York:
    (212) 399 1987



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

An example from (Muslea et al. Agents-99)

E1: 513 Pico, Venice, Phone 1-800-555-1515
E2: 90 Colfax, Palms, Phone (800) 508-1570
E3: 523 1st St., LA, Phone 1-800-578-2293
E4: 403 La Tijera, Watts, Phone: (310) 798-0008

We want to extract area code.

- Start rules:
  R1: SkipTo(()
  R2: SkipTo(-)
- End rules:
  R3: SkipTo())
  R4: SkipTo()

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.

Rule induction through an example

Training examples:
E1: 513 Pico, Venice, Phone 1-800-555-1515
E2: 90 Colfax, Palms, Phone (800) 508-1570
E3: 523 1st St., LA, Phone 1-800-578-2293
E4: 403 La Tijera, Watts, Phone: (310) 798-0008

We learn start rule for area code.

- Assume the algorithm starts with E2. It creates three initial candidate rules with first prefix symbol and two wildcards:
  - R1: SkipTo(()
  - R2: SkipTo(Punctuation)
  - R3: SkipTo(Anything)
- R1 is perfect. It covers two positive examples but no negative example.

Rule induction (cont …)

E1: 513 Pico, Venice, Phone 1-800-555-1515
E2: 90 Colfax, Palms, Phone (800) 508-1570
E3: 523 1st St., LA, Phone 1-800-578-2293
E4: 403 La Tijera, Watts, Phone: (310) 798-0008

- R1 covers E2 and E4, which are removed. E1 and E3 need additional rules.
- Three candidates are created:
  - R4: SkiptTo()
  - R5: SkipTo(HtmlTag)
  - R6: SkipTo(Anything)
- None is good. Refinement is needed.
- Stalker chooses R4 to refine, i.e., to add additional symbols, to specialize it.
- It will find R7: SkipTo(-), which is perfect.

Some other issues in wrapper learning

- Active learning
  - How to automatically choose examples for the user to label (Muslea et al, AAAI-99, AAAI-00)
- Wrapper verification
  - Check whether the current wrapper still work properly (Kushmerick, WWWJ-00)
- Wrapper maintenance
  - If the wrapper no longer works properly, is it possible to re-label automatically (Kushmerick AAAI-99; Lerman et al, JAIR-03)
- Wrapper verification and maintenance are still difficult problems.
  - Personally, I think wrapper verification is do-able in most cases.

Limitations of Supervised Learning

- Manual Labeling is labor intensive and time consuming, especially if one wants to extract data from a huge number of sites.
- Wrapper maintenance is very costly:
  - If Web sites change frequently
  - It is necessary to detect when a wrapper stops to work properly.
  - Any change may make existing extraction rules invalid.
  - Re-learning is needed, and most likely manual re-labeling as well.

Road map

- Wrapper Induction
  - Given a set of manually labeled pages, a machine learning method is applied to learn extraction rules or patterns.
- Automatic data extraction
  - Given a set of positive pages, generate extraction patterns.
  - Given only a single page with multiple data records, generate extraction patterns.

The RoadRunner System

(Crescenzi et al. VLDB-01)

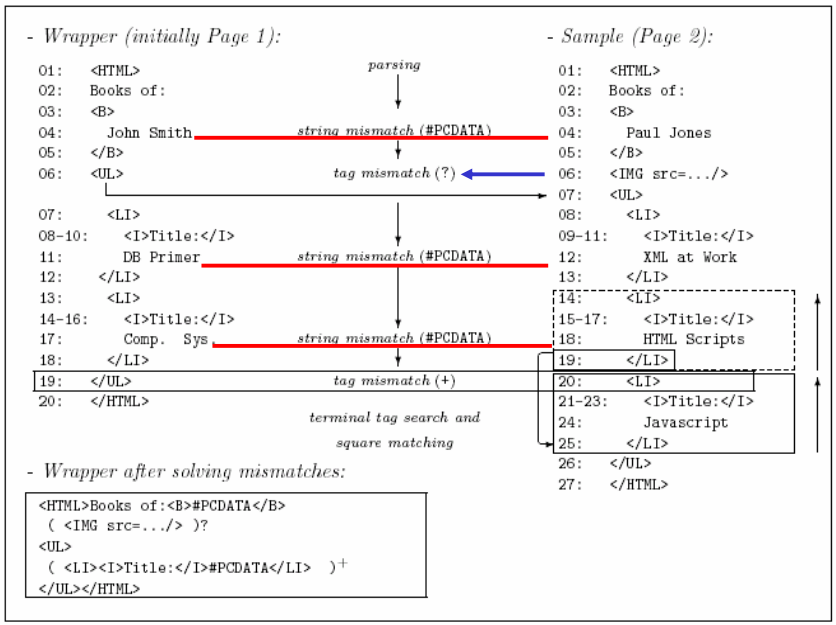
- Given a set of positive examples (multiple sample pages). Each contains one or more data records.
- From these pages, generate a wrapper as a union-free regular expression (i.e., no disjunction).
- Support nested data records.

The approach

- To start, a sample page is taken as the wrapper.
- The wrapper is then refined by solving mismatches between the wrapper and each sample page, which generalizes the wrapper.
  - A mismatch occurs when some token in the sample does not match the grammar of the wrapper.

Different types of mismatches and wrapper generalization

- Text string mismatches: indicate data fields (or items).
- Tag mismatches: indicate
  - optional elements, or
  - Iterators, list of repeated patterns
    - Mismatch occurs at the beginning of a repeated pattern and the end of the list.
    - Find the last token of the mismatch position and identify some candidate repeated patterns from the wrapper and sample by searching forward.
    - Compare the candidates with upward portion of the sample to confirm.



Computation issues

- The match algorithm is exponential in the input string length.
- Heuristic strategies are used to prune search:
  - Limit the space to explore, limit backtracking,
  - Pattern (iterator or optional) cannot be delimited on either side by an optional pattern (the expressiveness is reduced).
- Note: the multiple tree alignment algorithm in (Zhai and Liu WWW-05) can be used here instead by
  - treating each sample page as a data record (see slides 50-55 in this tutorial). It is much more efficient.

Compare with wrapper induction

- No manual labeling, but need a set of positive pages of the same template
  - which is not necessary for a page with multiple data records
- not wrapper for data records, but for pages.
  - A page often contains a lot of irrelevant information.

Issues of automatic extraction

- Hard to handle disjunctions.
- Hard to generate attribute names for the extracted data.
- extracted data from multiple sites need integration, manual or automatic.

The EXALG System

(Arasu and Garcia-Molina, SIGMOD-03)

- The same setting as for RoadRunner: need multiple input pages of the same template.

The approach:

Step 1: find sets of tokens (called equivalence classes) having the same frequency of occurrence in every page.

Step 2: expand the sets by differentiating "roles" of tokens using contexts. Same token in different contexts are treated as different tokens.

Step 3: build the page template using the equivalence classes based on what is in between two consecutive tokens, empty, data or list.

Road map

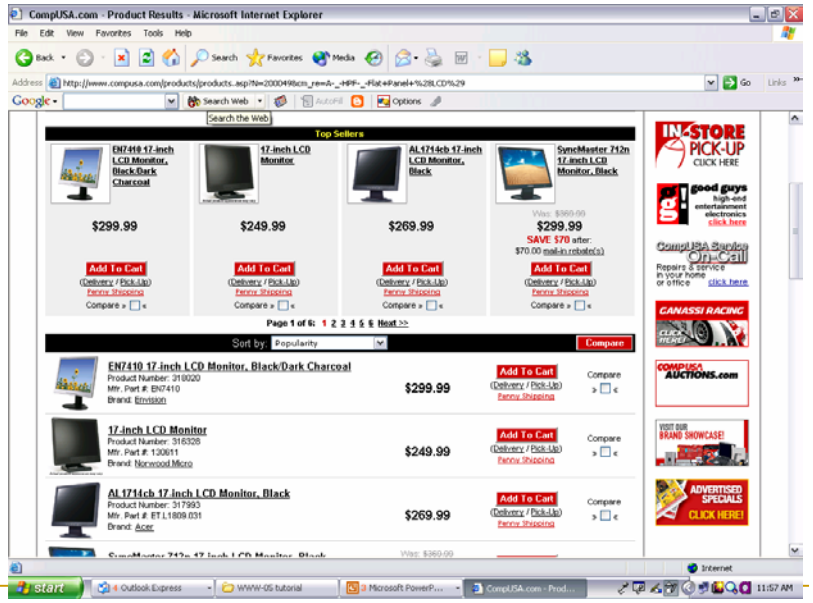
- Wrapper Induction
  - Given a set of manually labeled pages, a machine learning method is applied to learn extraction rules or patterns.
- Automatic data extraction
  - Given a set of positive pages, generate extraction patterns.
  - Given only a single page with multiple data records, generate extraction patterns.

Automatic data extraction

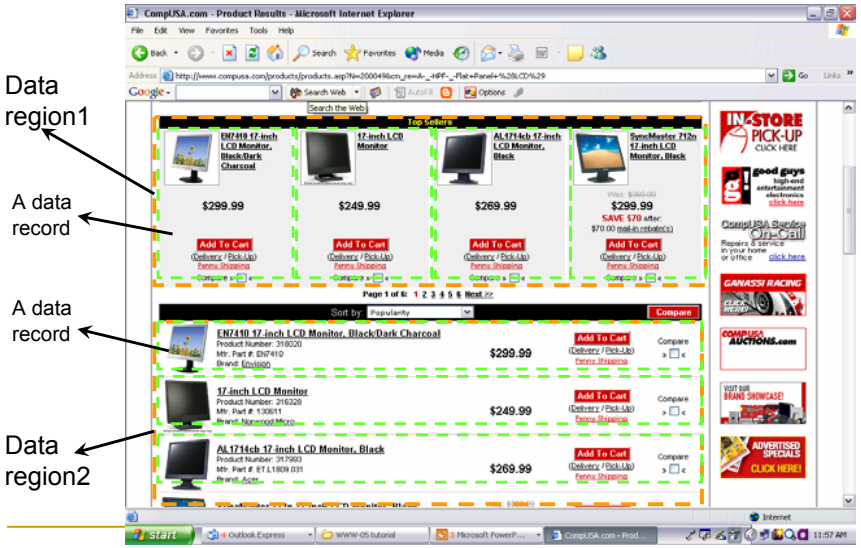
- Input: A single Web page with multiple data records (at least 2).
- Objective: Automatically (no human involvement)

  Step1: Identify data records in a page, and

  Step 2: align and extract data items from them



1. Identify data regions and data records



2. Align and extract data items (e.g., region1)

image 1	EN7410 17-inch LCD Monitor Black/Dark charcoal		$299.99		Add to Cart	(Delivery / Pick-Up)	Penny Shopping	Compare
image 2	17-inch LCD Monitor		$249.99		Add to Cart	(Delivery / Pick-Up)	Penny Shopping	Compare
image 3	AL1714 17-inch LCD Monitor, Black		$269.99		Add to Cart	(Delivery / Pick-Up)	Penny Shopping	Compare
image 4	SyncMaster 712n 17-inch LCD Monitor, Black	Was: $369.99	$299.99	Save $70 After: $70 mail-in-rebate(s)	Add to Cart	(Delivery / Pick-Up)	Penny Shopping	Compare

MDR: Mining Data Records

(Liu et al, KDD-03; Zhai and Liu, WWW-05)

- Given a single page with multiple data records, MDR extracts data records, but not data items (step 1)
- MDR is based on
  - two observations about data records in a Web page
  - a string matching algorithm (tree matching ok too)
- Considered both
  - contiguous
  - non-contiguous data records
- Nested Data Records: not discussed in the paper, but in the technical report. A new system can do this.

Two observations

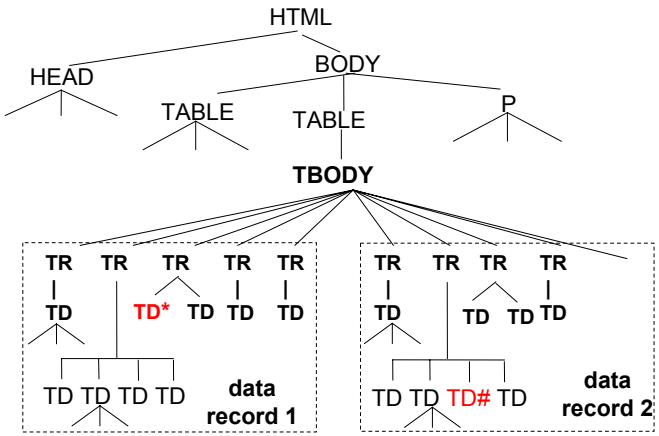
- A group of data records are presented in a contiguous region (a data region) of a page and are formatted using similar tags.
- A group of data records being placed in a data region are under one parent node and consists of children nodes.

Dramatically reduce computation as they tell
- where a data record may start and end
- what string/tree matching should or should not be performed.

1. Apple iBook Notebook M8600LL/A (600-MHz PowerPC G3, 128 MB RAM, 20 GB hard drive)
Buy new: $1,194.00
Usually ships in 1 to 2 days
Customer Rating:
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)
Buy new: $2,399.99
Customer Rating:
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, ...

Tag (DOM) tree of the previous page



The approach

Given a page, three steps:
- Building the HTML Tag Tree
- Mining Data Regions
- Identifying Data Records

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

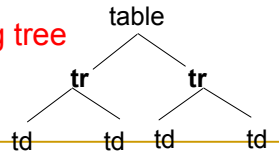
Building the tag tree for a page

- Fairly easy? Follow the nested blocks of the HTML tags in the page.
- Not easy to build a correct tree because
  - Erroneous tags
  - Unbalanced tags
  - Etc
  - Some problems are hard to fix.

Building tree based on visual cues

		left	right	top	bottom
1	<table>	100	300	200	400
2	<tr>	100	300	200	300
3	<td> … </td>	100	200	200	300
4	<td> … </td>	200	300	200	300
5	</tr>				
6	<tr>	100	300	300	400
7	<td> … </td>	100	200	300	400
8	<td> … </td>	200	300	300	400
9	</tr>				
10	</table>				

The tag tree



Mining data regions

- Find every data region with similar data records.

Definition: A *generalized node* (or a *node combination*) of length *r* consists of *r* ($r \geq 1$) nodes in the HTML tag tree with the following two properties:

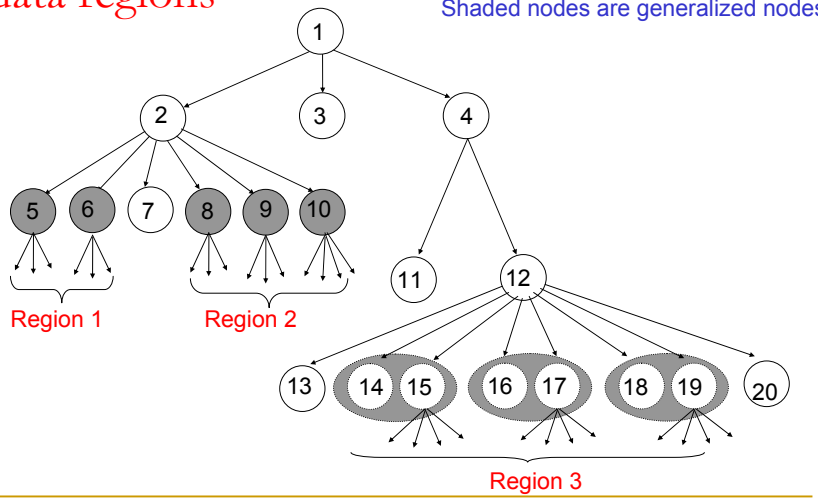
1. the nodes all have the same parent.
2. the nodes are adjacent.

Definition: A *data region* is a collection of two or more generalized nodes with the following properties:

1. the generalized nodes all have the same parent.
2. the generalized nodes are all adjacent.
3. adjacent generalized nodes are similar.

An illustration of generalized nodes and data regions

Shaded nodes are generalized nodes



Find data regions

- To find each data region, the algorithm needs to find the following.

1. Where does the first generalized node of the data region start?
   - try to start from each child node under a parent
2. How many tag nodes or components does a generalized node have?
   - we try: one node, two node, …, *K* node combinations

- The computation is actually not large due to the observations. Visual cues help to prune as well.

An example (incomplete)

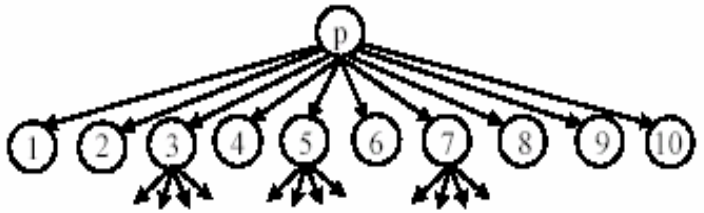


Figure 4: combination and comparison

Start from node 1: We compute the following string comparisons.

- (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9), (9, 10)
- (1-2, 3-4), (3-4, 5-6), (5-6, 7-8), (7-8, 9-10)
- (1-2-3, 4-5-6), (4-5-6, 7-8-9)

- **Using string comparison to find similarity**

Finding all data regions (Pre-order)

Algorithm FindDRs(*Node, K, T*) // Pre-order

```
1  if TreeDepth(Node) => 3 then
2     Node.DRs = IdenDRs(1, Node, K, T);
3     tempDRs = ∅;
4     for each Child ∈ Node.Children do
5        FindDRs(Child, K, T);
6        tempDRs = tempDRs ∪ UnCoveredDRs(Node, Child);
7     Node.DRs = Node.DRs ∪ tempDRs
```

- Use the string comparison results at each parent node to find similar children node combinations to find candidate generalized nodes and data regions for the parent node.
- Pre-order traversal: cannot find nested data records
- Change to post-order traversal: find nested data records

Identify Data Records

- A generalized node may not be a data record.
- Extra mechanisms are needed to identify true atomic objects (see the papers).
- Some highlights:
  - Contiguous
  - non-contiguous data records.

Name 1 Description of object 1	Name 2 Description of object 2
Name 3 Description of object 3	Name 4 Description of object 4

Name 1	Name 2
Description of object 1	Description of object 2
Name 3	Name 4
Description of object 3	Description of object 4

Is mining of data records useful?

- Data records enable object level search (rather than current page level search): E.g.,
  - if one can extract all the product data records on the Web, one can built a product search engine, by treating each data record/product as a Web page.
- Meta-search: re-ranking of search results from multiple search engines.
- Extract data items from data records and put them in tables for querying.

DEPTA: Extract Data from Data Records

(Zhai and Liu, WWW-05)

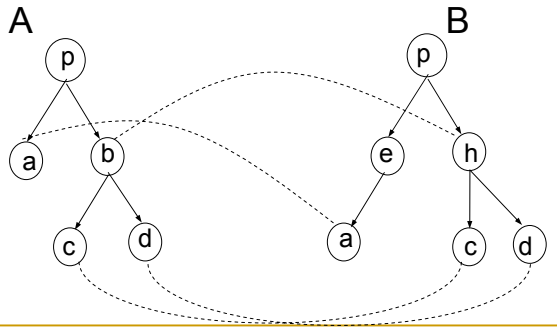
- Once a list of data records are identified, we can align and extract data items in them.
- Approaches (align multiple data records):
  - Multiple string alignment
    - Many ambiguities due to pervasive use of table related tags.
  - Multiple tree alignment (partial tree alignment)
    - Together with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

Tree matching (tree edit distance)

- 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 $n1$ and a tree B of size $n2$ is a set of ordered pairs (i, j), one from each tree, satisfying the following conditions for all $(i1, j1), (i2, j2) \in M$:
  - $i1 = i2$ *iff* $j1 = j2$;
  - $A[i1]$ is on the left of $A[i2]$ *iff* $B[j1]$ is on the left $B[j2]$;
  - $A[i1]$ is an ancestor of $A[i2]$ *iff* $B[j1]$ is an ancestor of $B[j2]$.

Intuitive idea

- The definition requires that
  - each node can appear no more than once in a mapping,
  - the order between sibling nodes are preserved, and
  - the hierarchical relation between nodes are also preserved.



An restrict version of tree matching (Yang 1991)

- No node replacement and no level crossing are allowed.
- Dynamic programming solution

Algorithm: Simple_Tree_Matching(*A*, *B*)

if the roots of the two trees *A* and *B* contain distinct symbols or have visual conflict /* a lot of pruning methods can be used here

then return (0);

else *m*:= the number of first-level sub-trees of *A*;

n:= the number of first-level sub-trees of *B*;

Initialization: *M*[*i*, 0]:= 0 for *i* = 0, …, *m*;

M[0, *j*] := 0 for *j* = 0, …, *n*;

for *i* = 1 to *m* **do**

for *j* = 1 to *n* **do**

M[*i*,*j*]:=max(*M*[*i*,*j*-1], *M*[*i*-1, *j*], *M*[*i*-1, *j*-1]+*W*[*i*, *j*]);

where *W*[*i*,*j*] = Simple_Tree_Matching(*Ai*, *Bj*)

return (*M*[*m*, *n*]+1)

Multiple tree alignment

- We need multiple alignment as we have multiple data records.
- Most multiple alignment methods work like hierarchical clustering, and require n^2 pair-wise matching.
  - Too expensive.
- Optimal alignment/matching is exponential.
- A partial tree matching algorithm is proposed in DEPTA to perform multiple tree alignment.

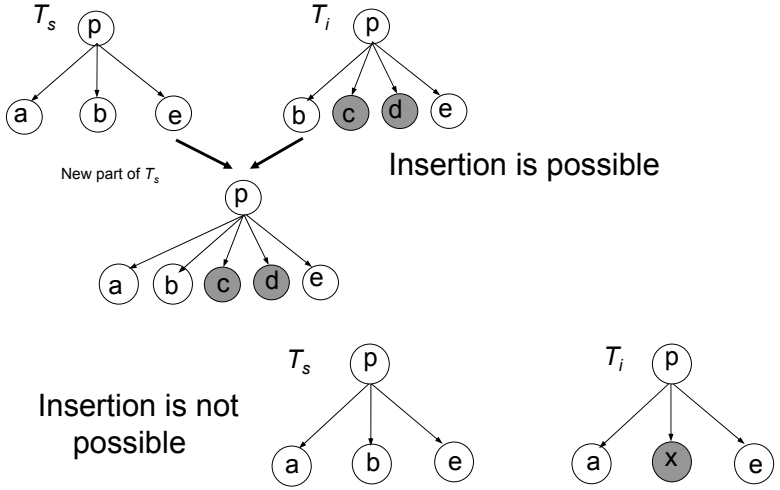
The Partial Tree Alignment approach

- Choose a seed tree: A seed tree, denoted by *Ts*, is picked with the maximum number of data items.
- Tree matching:

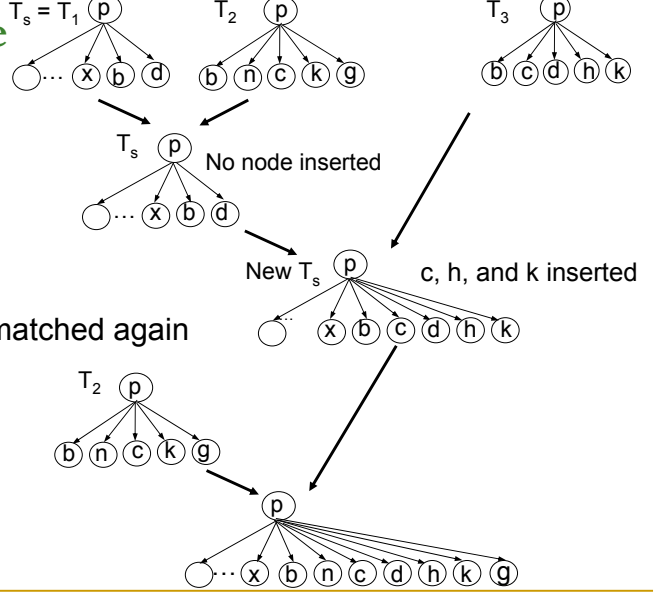
  For each unmatched tree *Ti* ($i \neq s$),
  - match *Ts* and *Ti*.
  - Each pair of matched nodes are linked (aligned).
  - For each unmatched node *nj* in *Ti* do
    - expand *Ts* by inserting *nj* into *Ts* if a position for insertion can be uniquely determined in *Ts*.

  The expanded seed tree *Ts* is then used in subsequent matching.

Illustration of partial tree alignment



A complete example



Output Data Table

	…	x	b	n	c	d	h	k	g
T_1	…	1	1			1			
T_2			1	1	1			1	1
T_3			1		1	1	1	1	

- The final tree may also be used to match and extract data from other similar pages.
- Disjunction can be dealt with (drop me an email if you are interested to know).

Some other systems and techniques

- IEPAD (Chang & Lui WWW-01), DeLa (Wang & Lochovsky WWW-03)
  - These systems treat a page as a long string, and find repeated substring patterns.
  - They often produce multiple patterns (rules). Hard to decide which is correct.
- RoadRunner (Crescenzi et al, VLDB-01), EXALG(Arasu & Garcia-Molina SIGMOD-03), (Lerman et al, SIGMOD-04).
  - Require multiple pages to find patterns.
  - Which is not necessary for pages with multiple records.
- (Zhao et al, WWW-04)
  - It extracts data records in one area of a page.

Limitations and issues

- Not for a page with only a single data record
- not able to generate attribute names (label) for the extracted data (yet!)
- extracted data from multiple sites need integration.
- In general, automatic integration is hard.
- It is, however, possible in each specific application domain, e.g.,
  - products sold online.
    - need “product name”, “image”, and “price”.
    - identify only these three fields may not be too hard.
  - Job postings, publications, etc …

2. Sentiment Classification, Analysis and Summarization of Consumer Reviews

How to classify reviews according to sentiments?
What exactly did consumers praise or complain?
How to summarize reviews?

Word-of-mouth on the Web

- The Web has dramatically changed the way that consumers express their opinions.
- One can post reviews of products at merchant sites, Web forums, discussion groups, blogs
- Techniques are being developed to exploit these sources to help companies and individuals to gain market intelligence info.
- Benefits:
  - Potential Customer: No need to read many reviews
  - Product manufacturer: market intelligence, product benchmarking

Road Map

- Sentiment classification
  - Whole reviews
  - Sentences
- Consumer review analysis
  - Going inside each sentence to find what exactly consumers praise or complain.
    - Extraction of product features commented by consumers.
    - Determine whether the comments are positive or negative (semantic orientation)
    - Produce a feature based summary (not text summarization).

Sentiment Classification of Reviews

- Classify reviews (or other documents) based on the overall sentiment expressed by the authors, i.e.,
  - Positive or negative
  - Recommended or not recommended
- This problem has been mainly studied in natural language processing (NLP) community.
- The problem is related but different from traditional text classification, which classifies documents into different topic categories.

Unsupervised review classification (Turney ACL-02)

- Data: reviews from epinions.com on automobiles, banks, movies, and travel destinations.
- The approach: Three steps
- Step 1:
  - Part-of-speech tagging
  - Extracting two consecutive words (two-word phrases) from reviews if their tags conform to some given patterns, e.g., (1) JJ, (2) NN.

- Step 2: Estimate the semantic orientation of the extracted phrases
  - Use Pointwise mutual information

$$PMI(word_1, word_2) = \log_2\left(\frac{P(word_1 \wedge word_2)}{P(word_1)P(word_2)}\right)$$

  - Semantic orientation (SO):

    SO(phrase) = PMI(phrase, "excellent")
    - PMI(phrase, "poor")

  - Using AltaVista NEAR operator to do search to find the number of hits to compute PMI and SO.

- Step 3: Compute the average SO of all phrases
  - classify the review as recommended if average SO is positive, not recommended otherwise.

- Final classification accuracy:
  - automobiles - 84%
  - banks - 80%
  - movies - 65.83
  - travel destinations - 70.53%

Sentiment classification using machine learning methods (Pang et al, EMNLP-02)

- The paper applied several machine learning techniques to classify movie reviews into positive and negative.
- Three classification techniques were tried:
  - Naïve Bayes
  - Maximum entropy
  - Support vector machine
- Pre-processing settings: negation tag, unigram (single words), bigram, POS tag, position.
- SVM: the best accuracy 83% (unigram)

Review classification by scoring features

(Dave, Lawrence and Pennock, WWW-03)

- It first selects a set of features F = f_1, f_2, ……
- Score the features
  - C and C' are classes

$$score(f_i) = \frac{P(f_i \mid C) - P(f_i \mid C')}{P(f_i \mid C) + P(f_i \mid C')}$$

- Classification of a review d_j (using sign):

$$class(d_j) = \begin{cases} C & eval(d_j) > 0 \\ C' & eval(d_j) < 0 \end{cases}$$

$$eval(d_j) = \sum_i score(f_i)$$

Evaluation

- The paper presented and tested many methods to select features, to score features, …
- The technique does well for review classification with accuracy of 84-88%
- It does not do so well for classifying review sentences, max accuracy = 68% even after removing hard and ambiguous cases.
- Sentence classification is much harder.

Other related works

- Estimate semantic orientation of words and phrases (Hatzivassiloglou and Wiebe COLING-00, Hatzivassiloglou and McKeown ACL-97; Wiebe, Bruce and O'Hara, ACL-99).
- Generating semantic timelines by tracking online discussion of movies and display a plot of the number positive and negative messages (Tong, 2001).
- Determine subjectivity and extract subjective sentences, e.g., (Wilson, Wiebe and Hwa, AAAI-04; Riloff and Wiebe, EMNLP-03)
- Mining product reputation (Morinaga et al, KDD-02).
- Classify people into opposite camps in newsgroups (Agrawal et al WWW-03).
- More …

Road Map

- Sentiment classification
  - Whole reviews
  - Sentences
- Consumer review analysis
  - Going inside each sentence to find what exactly consumers praise or complain.
    - Extraction of product features commented by consumers.
    - Determine whether the comments are positive or negative (semantic orientation)
    - Produce a feature based summary (not text summarization)

Mining and summarizing reviews

- Sentiment classification is useful. But
  - can we go inside each sentence to find what exactly consumers praise or complain about?

That is,

- Extract product features commented by consumers.
- Determine whether the comments are positive or negative (semantic orientation)
- Produce a *feature based summary* (not text summary).

- In online shopping, more and more people are writing reviews online to express their opinions
  - A lot of reviews…

- Time consuming and tedious to read all the reviews

- Benefits:
  - Potential Customer: No need to read many reviews
  - Product manufacturer: market intelligence, product benchmarking

Different Types of Consumer Reviews

(Hu and Liu, KDD-04; Liu et al WWW-05)

Format (1) - Pros and Cons: The reviewer is asked to describe Pros and Cons separately. C|net.com uses this format.

Format (2) - Pros, Cons and detailed review: The reviewer is asked to describe Pros and Cons separately and also write a detailed review. Epinions.com uses this format.

Format (3) - free format: The reviewer can write freely, i.e., no separation of Pros and Cons. Amazon.com uses this format.

Note: Professional reviews not included.

Feature Based Summarization

- Extracting product features (called Opinion Features) that have been commented on by customers.
- Identifying opinion sentences in each review and deciding whether each opinion sentence is positive or negative.
- Summarizing and comparing results.

Note: a wrapper can be used to extract reviews from Web pages as reviews are all regularly structured.

The Problem Model

Product feature:

- product component, function feature, or specification

Model: Each product has a finite set of features,

- $F = \{f_1, f_2, \ldots, f_n\}$.
- Each feature f_i in F can be expressed with a finite set of words or phrases W_i.
- Each reviewer j comments on a subset S_j of F, i.e., $S_j \subseteq F$.
- For each feature $f_k \in F$ that reviewer j comments, he/she chooses a word/phrase $w \in W_k$ to represent the feature.
- The system does not have any information about F or W_i beforehand.

- This simple model covers most but not all cases.

Example 1: Format 3

GREAT Camera., Jun 3, 2004
Reviewer: **jprice174** from Atlanta, Ga.

I did a lot of research last year before I bought this camera... It kinda hurt to leave behind my beloved nikon 35mm SLR, but I was going to Italy, and I needed something smaller, and digital.

The pictures coming out of this camera are amazing. The 'auto' feature takes great pictures most of the time. And with digital, you're not wasting film if the picture doesn't come out. …

….

Feature based Summary:

Feature1: **picture**
Positive: 12

- The pictures coming out of this camera are amazing.
- Overall this is a good camera with a really good picture clarity.

…
Negative: 2

- The pictures come out hazy if your hands shake even for a moment during the entire process of taking a picture.
- Focusing on a display rack about 20 feet away in a brightly lit room during day time, pictures produced by this camera were blurry and in a shade of orange.

Feature2: **battery life**
…

Example 2: Format 2

My SLR is on the shelf
by shortstop24, Aug 09 '03

Pros: Great photos, easy to use, good manual, many options, takes videos
Cons: Battery usage; included software could be improved; included 16MB is stingy.

I had never used a digital camera prior to purchasing the Canon A70. I have always used a SLR (Minol ...

Read the full review

Example 3: Format 1

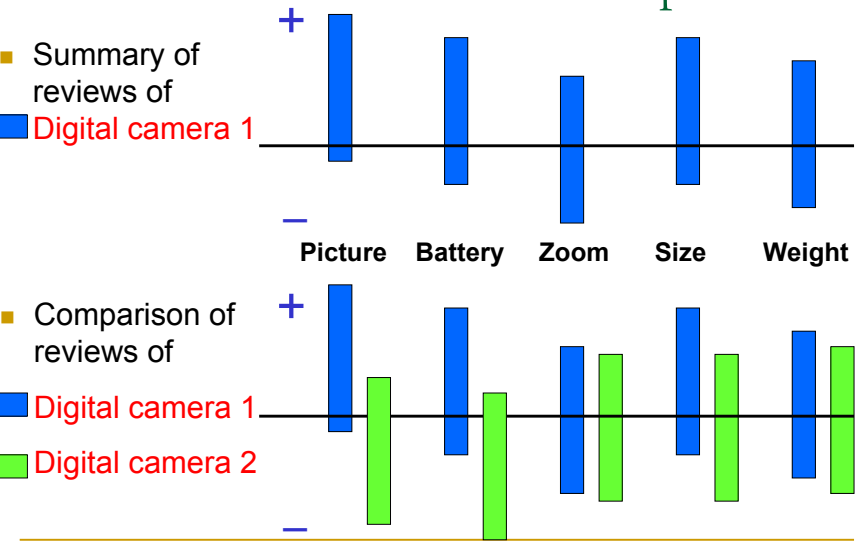
User rating
Perfect
10
out of 10

"It is a great digitbal still camera for this century"
September 1, 2004

Pros:
It's small in size, and the rotatable lens is great. It's very easy to use, and has fast response from the shutter. The LCD has increased from 1.5 in to 1.8, which gives bigger view. It has lots of modes to choose from in order to take better pictures.

Cons:
It almost has no cons, it would be better if the LCD is bigger and it's going to be best if the model is designed to a smaller size.

Visual Summarization & Comparison



Analyzing Reviews of formats 1 and 3

(Hu and Liu, KDD-04)

- Reviews are usually full sentences
- "The pictures are very clear."
  - Explicit feature: picture
- "It is small enough to fit easily in a coat pocket or purse."
  - Implicit feature: size
- Frequent and infrequent features
  - Frequent features (commented by many users)
  - Infrequent features

Step 1: Mining product features

1. Part-of-Speech tagging - in this work, features are nouns and nouns phrases (which is not sufficient!).
2. Frequent feature generation (unsupervised)
   - Association mining to generate candidate features
   - Feature pruning.
3. Infrequent feature generation
   - Opinion word extraction.
   - Find infrequent feature using opinion words.

Part-of-Speech tagging

- Segment the review text into sentences.
- Generate POS tags for each word.
- Syntactic chunking recognizes boundaries of noun groups and verb groups.

<S> <NG><W C='PRP' L='SS' T='w' S='Y'> ***I*** </W> </NG> <VG> <W C='VBP'> ***am*** </W><W C='RB'> ***absolutely*** </W></VG> <W C='IN'> ***in*** </W> <NG> <W C='NN'> ***awe*** </W> </NG> <W C='IN'> ***of*** </W> <NG> <W C='DT'> ***this*** </W> <W C='NN'> ***camera*** </W></NG><W C='.'> **.** </W></S>

Frequent feature identification

- Frequent features: those features that are talked about by many customers.
- Use association (frequent itemset) Mining
  - Why use association mining?
    - Different reviewers tell different stories (irrelevant)
    - When people discuss the product features, they use similar words.
    - Association mining finds frequent phrases.
- Note: only nouns/noun groups are used to generate frequent itemsets (features)

Compactness and redundancy pruning

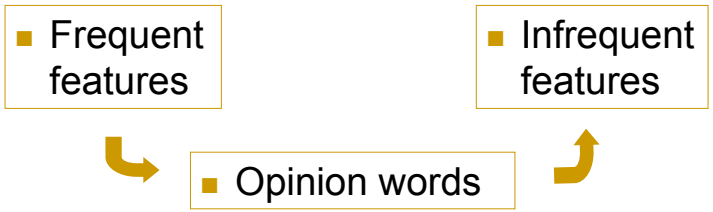
Not all candidate frequent features generated by association mining are genuine features.

- Compactness pruning: remove those non-compact feature phrases:
  - compact in a sentence
    - "I had searched a digital camera for months." -- compact
    - "This is the best digital camera on the market." -- compact
    - "This camera does not have a digital zoom." – not compact
- *p-support* (pure support).
  - manual (sup = 12), manual mode (sup = 5)
    - p-support of manual = 7
  - life (sup = 5), battery life (sup = 4)
    - p-support of life = 1

  set a minimum *p-support* value to do pruning.
  - life will be pruned while manual will not, if minimum p-support is 4.

Infrequent features generation

- How to find the infrequent features?
- Observation: one opinion word can be used to describe different objects.
  - "The pictures are absolutely amazing."
  - "The software that comes with it is amazing."



Step 2: Identify Orientation of an Opinion Sentence

- Use dominant orientation of opinion words (e.g., adjectives) as sentence orientation.
- The semantic orientation of an adjective:
  - positive orientation: desirable states (e.g., beautiful, awesome)
  - negative orientation: undesirable states (e.g., disappointing).
  - no orientation. e.g., external, digital.
- Using a seed set to grow a set of positive and negative words using WordNet,
  - synonyms,
  - antonyms.

Feature extraction evaluation

Product name	Frequent features (association mining)		Compactness pruning		Redundancy pruning		**Infrequent feature identification**	
	Recall	Precision	Recall	Precision	Recall	Precision	**Recall**	**Precision**
Digital camera1	0.671	0.552	0.658	0.634	0.658	0.825	**0.822**	**0.747**
Digital camera2	0.594	0.594	0.594	0.679	0.594	0.781	**0.792**	**0.710**
Cellular phone	0.731	0.563	0.716	0.676	0.716	0.828	**0.761**	**0.718**
Mp3 player	0.652	0.573	0.652	0.683	0.652	0.754	**0.818**	**0.692**
DVD player	0.754	0.531	0.754	0.634	0.754	0.765	**0.797**	**0.743**
Average	**0.68**	**0.56**	**0.67**	**0.66**	**0.67**	**0.79**	**0.80**	**0.72**

Table 1: Recall and precision at each step of feature generation

Opinion sentence extraction (Avg): Recall: 69.3% Precision: 64.2%

Opinion orientation accuracy: 84.2%

Analyzing Reviews of Format 2: Pros and Cons (Liu et al, WWW-05)

- Pros and Cons: Short phrases or incomplete sentences.

My SLR is on the shelf

by shortstop24, Aug 09 '03

Pros: Great photos, easy to use, good manual, many options, takes videos
Cons: Battery usage; included software could be improved; included 16MB is stingy.

I had never used a digital camera prior to purchasing the Canon A70. I have always used a SLR (Minol ...

Read the full review

Product feature extraction

- An important observation:

  Each sentence segment contains at most one product feature. Sentence segments are separated by ',', '.', 'and', 'but', 'however'.
- Pros in previous page have 5 segments.
  - great photos <photo>
  - easy to use <use>
  - good manual <manual>
  - many options <option>
  - takes videos <video>

Approach: extracting product features

- Supervised learning: Class Association Rules (Liu et al 1998).
- Extraction based on learned language patterns.
- Product Features
  - Explicit and implicit features
    - battery usage <battery>
    - included software could be improved <software>
    - included 16MB is stingy <16MB> ⇒ <memory>
  - Adjectives and verbs could be features
    - Quick ⇒ speed, heavy ⇒ weight
    - easy to use, does not work

The process

- Perform Part-Of-Speech (POS) tagging

great photos	<JJ> great <NN> [feature]
easy to use	<JJ> easy <TO> to <VB> [feature]

- Use n-gram to produce shorter segments
- Data mining: Generate language patterns, e.g.,
  - <JJ> [don't care] <NN> [feature]
- Extract features by using the language patterns.
  - "nice picture" => "picture"

(Data mining can also be done using Class Sequential Rules)

Generating extraction patterns

- **Rule generation**
  - <NN>, <JJ> → [feature]
  - <VB>, easy, to → [feature]
- **Considering word sequence**
  - <JJ>, <NN> → [feature]
  - <NN>, <JJ> → [feature] (pruned, low support/confidence)
  - easy, to, <VB> → [Feature]
- **Generating language patterns, e.g., from**
  - <JJ>, <NN> → [feature]
  - easy, to, <VB> → [feature]

  to
  - <JJ> <NN> [feature]
  - easy to <VB> [feature]

Feature extraction using language patterns

- **Length relaxation**: A language pattern does not need to match a sentence segment with the same length as the pattern.
- **Ranking of patterns**: If a sentence segment satisfies multiple patterns, use the pattern with the highest confidence.
- **No pattern applies**: use nouns or noun phrases.

For other interesting issues, look at the paper

Feature Refinement

- Correct some mistakes made during extraction.
- Two main cases:
  - Feature conflict: two or more candidate features in one sentence segment.
  - Missed feature: there is a feature in the sentence segment but not extracted by any pattern.
- E.g., "slight hum from subwoofer when not in use."
  - "hum" or "subwoofer"? how does the system know this?
  - Use candidate feature "subwoofer" (as it appears elsewhere):
    - "subwoofer annoys people."
    - "subwoofer is bulky."
- An iterative algorithm can be used to deal with the problem by remembering occurrence counts.

Experiment Results: Pros

- Data: reviews of 15 electronic products from epinions.com
- Manually tagged: 10 training, 5 testing

Pros	Patterns only		Frequent-noun strategy		**Frequent-term strategy**	
	Recall	Prec.	Recall	Prec.	**Recall**	**Prec.**
data1	0.878	0.880	0.849	0.861	**0.922**	**0.876**
data2	0.787	0.804	0.798	0.821	**0.894**	**0.902**
data3	0.782	0.806	0.758	0.782	**0.825**	**0.825**
data4	0.943	0.926	0.939	0.926	**0.942**	**0.922**
data5	0.899	0.893	0.878	0.881	**0.930**	**0.923**
Avg.	**0.857**	**0.862**	**0.844**	**0.854**	**0.902**	**0.889**

Experiment Results: Cons

Cons	Patterns only		Frequent-noun strategy		**Frequent-term strategy**	
	Recall	Prec	Recall	Prec	**Recall**	**Prec**
data1	0.900	0.856	0.867	0.848	**0.850**	**0.798**
data2	0.795	0.794	0.808	0.804	**0.860**	**0.833**
data3	0.677	0.699	0.834	0.801	**0.846**	**0.769**
data4	0.632	0.623	0.654	0.623	**0.681**	**0.657**
data5	0.772	0.772	0.839	0.867	**0.881**	**0.897**
Avg.	**0.755**	**0.748**	**0.801**	**0.788**	**0.824**	**0.791**

Summary

- Automatic opinion analysis has many applications.
- Some techniques have been proposed.
- However, the current work is still preliminary.
  - Other supervised or unsupervised learning should be tried. Additional NLP is likely to help.
- Much future work is needed: Accuracy is not yet good enough for industrial use, especially for reviews in full sentences.
- Analyzing blogspace is also an promising direction (Gruhl et al, WWW-04).
- Trust and distrust on the Web is an important issue too (Guha et al, WWW-04)

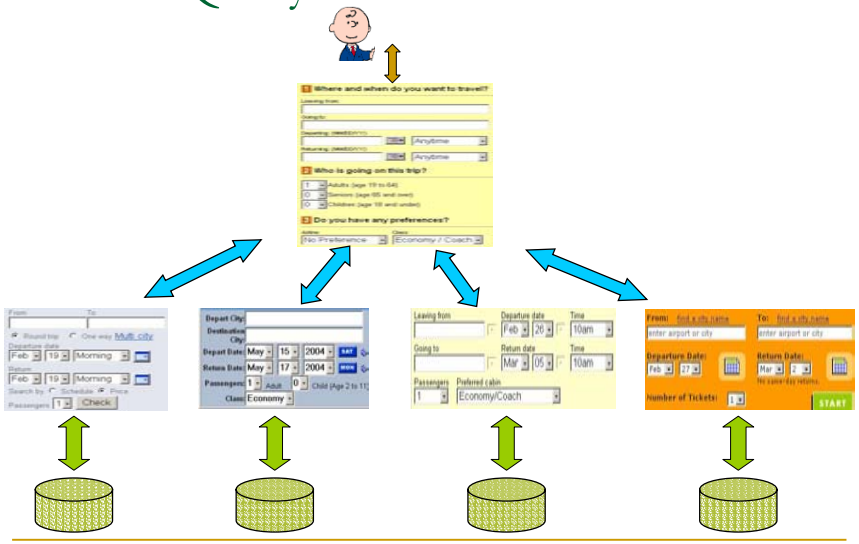
3. Web Information Integration and Schema Matching

Thanks to Kevin Chang, AnHai Doan, Ji-Rong Wen and Clement Yu for permission to use some of their slides.

Web query interface integration

- Many integration tasks,
  - Integrating Web query interfaces (search forms)
  - Integrating ontologies (taxonomy)
  - Integrating extracted data
  - Integrating textual information
  - …
- We only introduce integration of query interfaces.
  - Many web sites provide forms to query deep web
  - Applications: meta-search and meta-query

Global Query Interface



Constructing global query interface (QI)

- A unified query interface:
  - Conciseness - Combine semantically similar fields over source interfaces
  - Completeness - Retain source-specific fields
  - User-friendliness – Highly related fields are close together

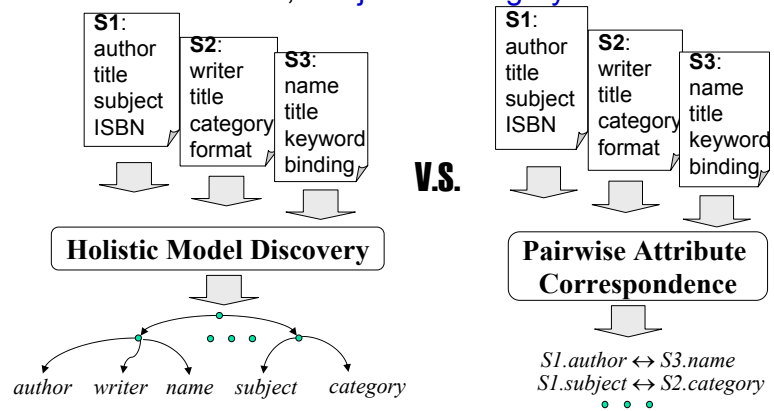
- Two-phrased integration
  - ***Interface Matching*** – Identify semantically similar fields

  - Interface Integration – Merge the source query interfaces

Hidden Model Discovery (He and Chang, SIGMOD-03)

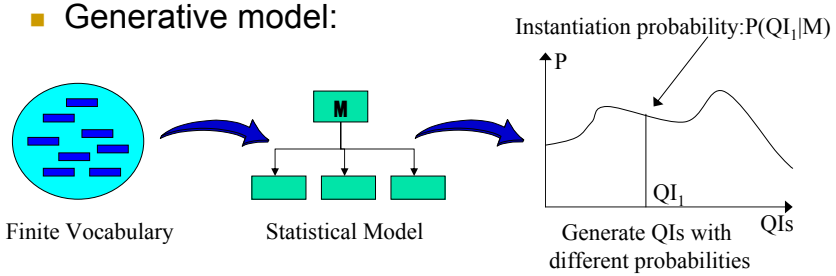
- Discover synonym attributes

Author – Writer, Subject – Category

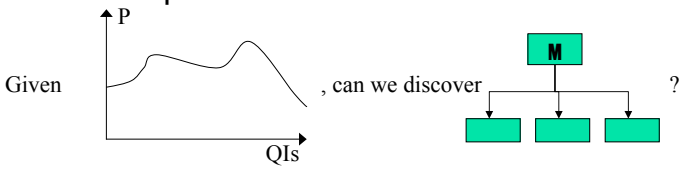


A hidden schema model exists?

- Generative model:

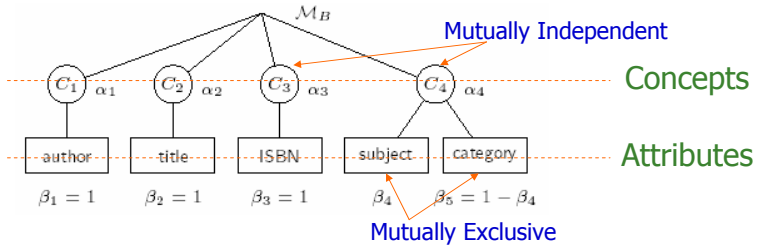


- Now the problem is:



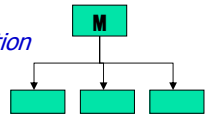
The Model Structure

- Goal: capture synonym relationship
- Two-level model structure



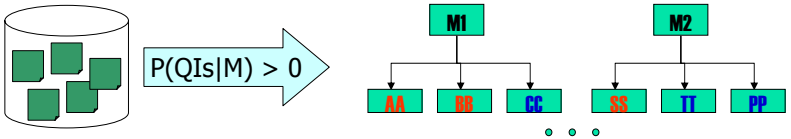
Statistical schema matching

1. Define the abstract Model structure M to solve a *target question*

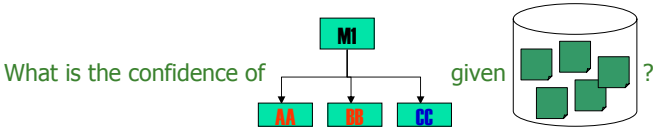


P(QI|M) = ...

2. Given QIs, Generate the model candidates



3. Select the candidate with highest confidence: hypothesis testing



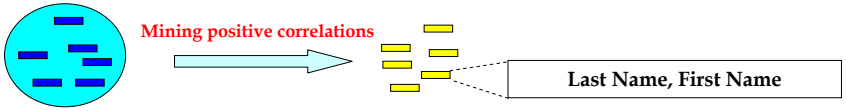
Schema matching as correlation mining

(He and Chang, KDD-04)

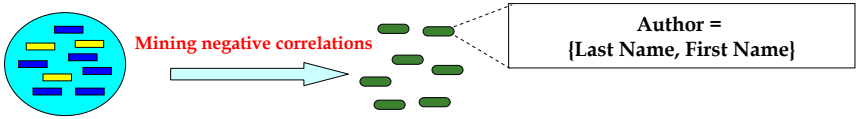
Across many sources:

- Synonym attributes are negatively correlated
  - synonym attributes are semantically alternatives.
  - thus, *rarely co-occur* in query interfaces
- Grouping attributes with positive correlation
  - grouping attributes semantically complement
  - thus, *often co-occur* in query interfaces
- A correlation data mining problem

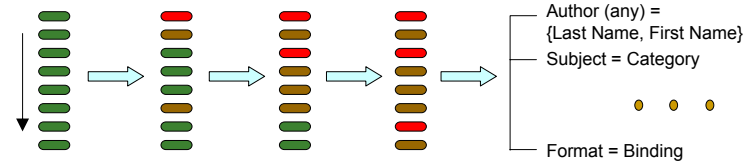
1. Positive correlation mining as potential groups



2. Negative correlation mining as potential matchings



3. Matching selection as model construction



Correlation measure --- *H*-measure

- H-measure $H = f_{01}f_{10}/(f_{+1}f_{1+})$

	A_p	$\neg A_p$	
A_q	f_{11}	f_{10}	f_{1+}
$\neg A_q$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	f_{++}

Ignore the co-absence

	A_p	$\neg A_p$	
A_q	5	5	10
$\neg A_q$	5	85	90
	10	90	100

Less positive correlation
$H = 0.25$

	A_p	$\neg A_p$	
A_q	55	20	75
$\neg A_q$	20	5	25
	75	25	100

More positive correlation
$H = 0.07$

Differentiate the subtlety of negative correlations

	A_p	$\neg A_p$	
A_q	1	49	50
$\neg A_q$	1	1	2
	2	50	52

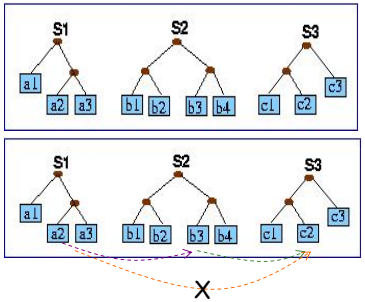
A_p as rare attributes
and $H = 0.49$

	A_p	$\neg A_p$	
A_q	1	25	26
$\neg A_q$	25	1	26
	26	26	52

No rare attributes
and $H = 0.92$

A clustering approach to schema matching
(Wu et al. SIGMOD-04)

- Hierarchical modeling
- Bridging effect
  - "a2" and "c2" might not look similar themselves but they might both be similar to "b3"
- 1:m mappings
  - Aggregate and is-a types
- User interaction helps in:
  - learning of matching thresholds
  - resolution of uncertain mappings



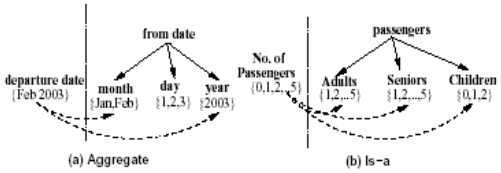
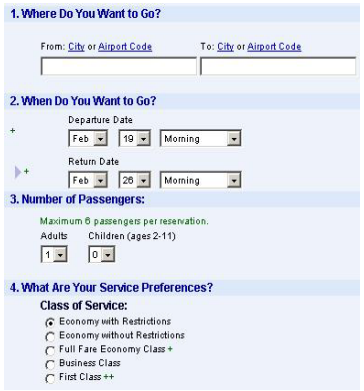
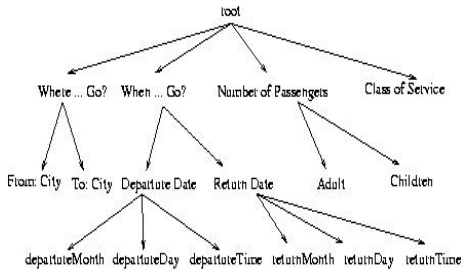


Figure 3: 1:m mappings

Hierarchical Modeling



Source Query Interface

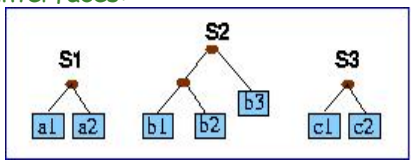


Ordered Tree Representation

Capture: ordering and grouping of fields

Find 1:1 Mappings via Clustering

Interfaces:



Initial similarity matrix:

	a_1	a_2	b_1	b_2	b_3	c_1	c_2
a_1	0	0	.9	0	.85	0	0
a_2	0	0	.15	0	0	0	0
b_1	.9	.15	0	0	0	.8	0
b_2	0	0	0	0	0	.1	.6
b_3	.85	0	0	0	0	0	0
c_1	0	0	0	.1	0	0	0
c_2	0	0	.8	.6	0	0	0

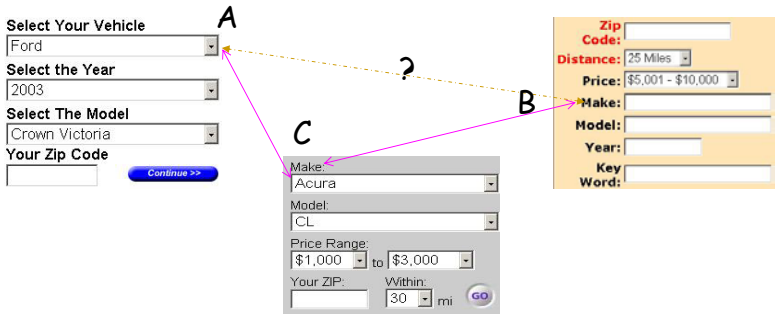
After one merge:

	a_2	b_2	b_3	c_1	c_2	$\{a_1, b_1\}$
a_2	0	0	0	0	0	0
b_2	0	0	0	.1	.6	0
b_3	0	0	0	0	0	0
c_1	0	.1	0	0	0	.8
c_2	0	.6	0	0	0	0
$\{a_1, b_1\}$	0	0	0	.8	0	0

- Similarity functions
  - linguistic similarity
  - domain similarity

…, final clusters: {{a1,b1,c1}, {b2,c2},{a2},{b3}}

"Bridging" Effect

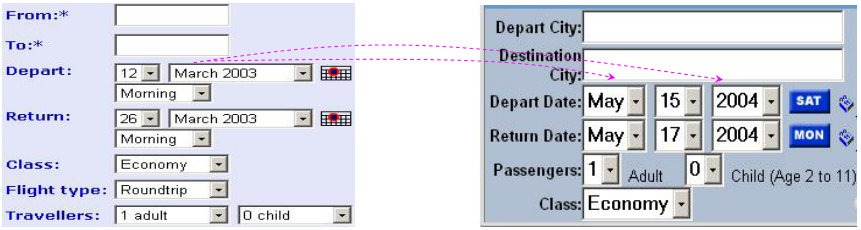


Observations:

- It is difficult to match "vehicle" field, A, with "make" field, B
- But A's instances are similar to C's, and C's label is similar to B's
- Thus, C might serve as a "bridge" to connect A and B!

Note: Connections might also be made via labels

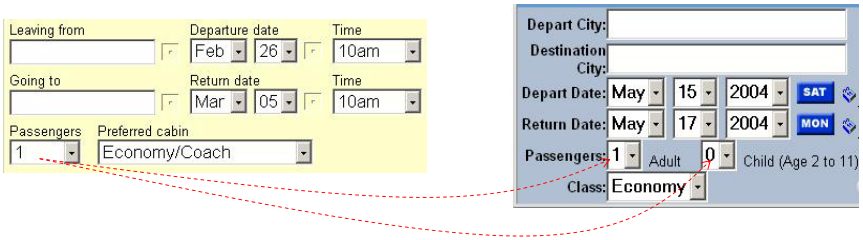
Complex Mappings



Aggregate type – contents of fields on the many side are part of the content of field on the one side

Commonalities – (1) field proximity, (2) parent label similarity, and (3) value characteristics

Complex Mappings (Cont'd)

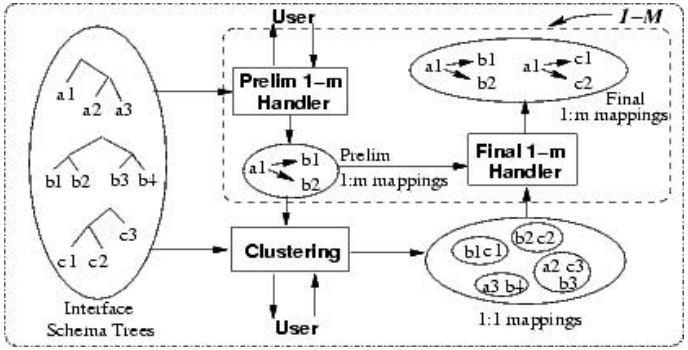


Is-a type – contents of fields on the many side are sum/union of the content of field on the one side

Commonalities – (1) field proximity, (2) parent label similarity, and (3) value characteristics

Complex Mappings (Cont'd)

- Final 1-m phase infers new mappings:



Preliminary 1-m phase: a1 → (b1, b2)
Clustering phase: b1 → c1, b2 → c2
Final 1-m phase: a1 → (c1, c2)

Further enhancements

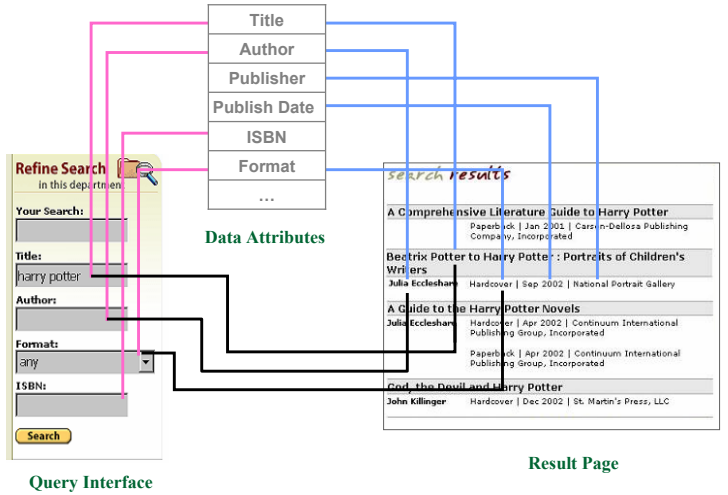
- Interactive learning to set parameter thresholds.
- Interactive resolution of uncertain mappings
  - Resolve potential homonyms
  - Determine potential synonyms
  - Determine potential 1:m mappings

Instance-based matching via query probing

(Wang et al. VLDB-04)

- Both query interfaces and returned results (called instances) are considered in matching.
  - Assume a global schema (GS) is given and a set of instances are also given.
  - The method uses each instance value (IV) of every attribute in GS to probe the underlying database to obtain the count of IV appeared in the returned results.
  - These counts are used to help matching.
- It performs matches of
  - Interface schema and global schema,
  - result schema and global schema, and
  - interface schema and results schema.

Query interface and result page



Summary

- Information integration is an active research area.
- Industrial activities are vibrant.
- We only introduced a few techniques for integrating query interfaces and underlying database schemas.
- Another area of research is Web ontology matching See (Noy and Musen, AAAI-00; Agrawal and Srikant, WWW-01; Doan et al. WWW-02; Zhang and Lee, WWW-04).
- Finally, database schema matching is a prominent research area in the database community as well. See (Doan and Halevy, AI Magazine 2005) for a short survey.

4. Knowledge Synthesis

According to Dictionary.com,

Synthesis: The combining of separate elements or substances to form a coherent whole

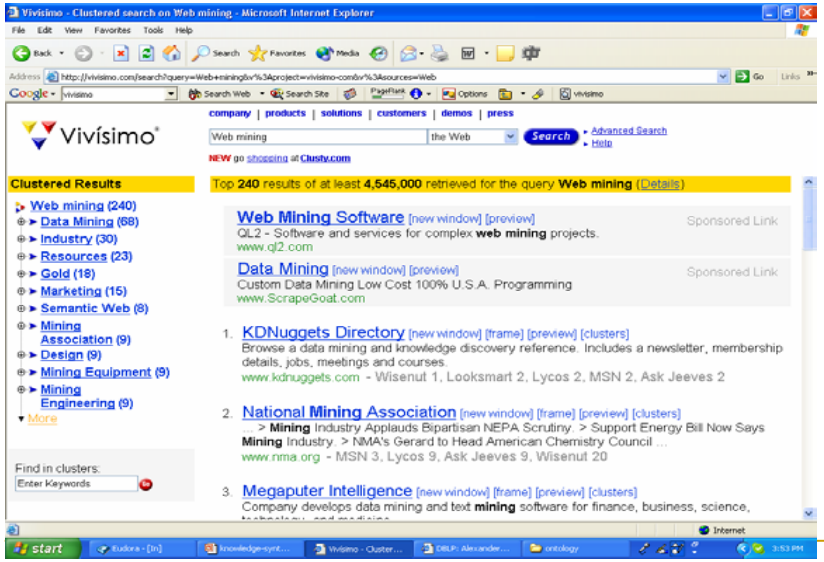
Web Search

- Web search paradigm:
  - Given a query, a few words
  - A search engine returns a ranked list of pages.
  - The user then browses and reads the pages to find what s/he wants.
- Sufficient
  - if one is looking for a specific piece of information, e.g., homepage of a person, a paper.
- Not sufficient for
  - open-ended research or exploration, for which more can be done.

Search results clustering

- The aim is to produce a taxonomy to provide navigational and browsing help by
  - organizing search results (snippets) into a small number of hierarchical clusters.
- Several researchers have worked on it.
  - E.g., Zamir & Etzioni, WWW-1998; Vaithyanathan & Dom, ICML-1999; Leuski & Allan, RIAO-00; Zeng et al. SIGIR-04; Kummamuru et al. WWW-04.
- Some search engines already provide categorized results, e.g., vivisimo.com, northernlight.com
- Note: Ontology learning also uses clustering to build ontologies (e.g., Maedche and Staab, 2001).

Vivisimo.com results for "web mining"



Going beyond search results clustering

- Search results clustering is well known and is in commercial systems.
  - Clusters provide browsing help so that the user can focus on what he/she really wants.
- Going beyond: Can a system provide the "complete" information of a search topic? I.e.,
  - Find and combine related bits and pieces
  - to provide a coherent picture of the topic.
- We discuss only one case study. There are many other applications.

Knowledge synthesis: a case study

(Liu, Chee and Ng, WWW-03)

- Motivation: traditionally, when one wants to learn about a topic,
  - one reads a book or a survey paper.
  - With the rapid expansion of the Web, this habit is changing.
- Learning in-depth knowledge of a topic from the Web is becoming increasingly popular.
  - Web's convenience
  - Richness of information, diversity, and applications
  - For emerging topics, it may be essential - no book.
- Can we mine "a book" from the Web on a topic?
  - Knowledge in a book is well organized: the authors have painstakingly synthesize and organize the knowledge about the topic and present it in a coherent manner.

An example

- Given the topic "data mining", can the system produce the following, a concept hierarchy?
  - Classification
    - Decision trees
      - … (Web pages containing the descriptions of the topic)
    - Naïve bayes
      - …
    - …
  - Clustering
    - Hierarchical
    - Partitioning
    - K-means
    - ….
  - Association rules
  - Sequential patterns
  - …

The Approach:

Exploiting information redundancy

- **Web information redundancy**: many Web pages contain similar information.
- Observation 1: If some phrases are mentioned in a number of pages, they are likely to be important concepts or sub-topics of the given topic.
- This means that we can use data mining to find concepts and sub-topics:
  - What are candidate words or phrases that may represent concepts of sub-topics?

Each Web page is already organized

- Observation 2: The content of each Web page is already organized.
  - Different levels of headings
  - Emphasized words and phrases
- They are indicated by various HTML emphasizing tags, e.g., <H1>, <H2>, <H3>, <B>, <I>, etc.
- We utilize existing page organizations to find a global organization of the topic.
  - Cannot rely on only one page because it is often incomplete, and mainly focus on what the page authors are familiar with or are working on.

Using language patterns to find sub-topics

- Certain syntactic language patterns express some relationship of concepts.
- The following patterns represent hierarchical relationships, concepts and sub-concepts:
  - *Such as*
  - *For example (e.g.,)*
  - *Including*
- E.g., "*There are many clustering techniques (e.g., hierarchical, partitioning, k-means, k-medoids).*"

Put them together

1. Crawl the set of pages (a set of given documents)
2. Identify important phrases using
   1. HTML emphasizing tags, e.g., *<h1>,…,<h4>, <b>, , <big>, <i>, , <u>, <li>, <dt>*.
   2. Language patterns.
3. Perform data mining (frequent itemset mining) to find frequent itemsets (candidate concepts)
   - Data mining can weed out peculiarities of individual pages to find the essentials.
4. Eliminate unlikely itemsets (using heuristic rules).
5. Rank the remaining itemsets, which are main concepts.

Additional techniques

- Segment a page into different sections.
  - Find sub-topics/concepts only in the appropriate sections.
- Mutual reinforcements:
  - Using sub-concepts search to help each other
- …
- Finding definition of each concept using syntactic patterns (again)
  - {is | are} [*adverb*] {called | known as | defined as} {*concept*}
  - {*concept*} {refer(s) to | satisfy(ies)} …
  - {*concept*} {is | are} [*determiner*] …
  - {*concept*} {is | are} [*adverb*] {being used to | used to | referred to | employed to | defined as | formalized as | described as | concerned with | called} …

Some concepts extraction results

Data Mining
Clustering
Classification
Data Warehouses
Databases
Knowledge Discovery
Web Mining
Information Discovery
Association Rules
Machine Learning
Sequential Patterns

Web Mining
Web Usage Mining
Web Content Mining
Data Mining
Webminers
Text Mining
Personalization
Information Extraction
Semantic Web Mining
XML
Mining Web Data

Classification
Neural networks
Trees
Naive bayes
Decision trees
K nearest neighbor
Regression
Neural net
Sliq algorithm
Parallel algorithms
Classification rule learning
ID3 algorithm
C4.5 algorithm
Probabilistic models

Clustering
Hierarchical
K means
Density based
Partitioning
K medoids
Distance based methods
Mixture models
Graphical techniques
Intelligent miner
Agglomerative
Graph based algorithms

Some recent work on finding **classes and instances** using syntactic patterns

- As we discussed earlier, syntactic language patterns do convey some semantic relationships.
- Earlier work by Hearst (Hearst, SIGIR-92) used patterns to find concepts/sub-concepts relations.
- WWW-04 has two papers on this issue (Cimiano, Handschuh and Staab 2004) and (Etzioni et al 2004).
  - apply lexicon-syntactic patterns such as those discussed 5 slides ago and more
  - Use a search engine to find concepts and sub-concepts (class/instance) relationships.

PANKOW (Cimiano, Handschuh and Staab WWW 04)

- Objective: Annotate a certain entity in a Web page with a corresponding concept. I.e.,
  - PANKOW categorizes instances into given concept classes, e.g., is "Japan" a "country" or a "hotel"?
- The linguistic patterns used are (the first 4 are from (Hearst SIGIR-92)):

1: <concept>s such as <instance>
2: such <concept>s as <instance>
3: <concepts>s, (especially|including)<instance
4: <instance> (and|or) other <concept>s
5: the <instance> <concept>
6: the <concept> <instance>
7: <instance>, a <concept>
8: <instance> is a <concept>

The steps

- Given a proper noun (instance), it is introduced together with given ontology concepts into the linguistic patterns to form hypothesis phrases, e.g.,
  - Proper noun: Japan
  - Given concepts: country, hotel.
  - ⇒ "Japan is a country", "Japan is a hotel" // pattern 8
- All the hypothesis phrases are sent to Google.
- Counts from Google are collected

Categorization step

- The system sums up the counts for each instance and concept pair (i:instance, c:concept, p:pattern).

$$count(i,c) = \sum_{p \in P} count(i,c,p)$$

- The candidate proper noun (instance) is given to the highest ranked concept(s):

$$R = \{(i,c_i) \mid i \in I, c_i = \arg\max_{c \in C} count(i,c)\}$$

  - I: instances, C: concepts
- Result: Categorization was reasonably accurate.
- Recent work C-PANKOW (WWW-2005) also considers the context in which the entity appears.

KnowItAll (Etzioni et al WWW-04)

- Basically uses a similar approach of linguistic patterns and Web search, but with a different objective: to extract class and instance relationships. I.e.,
  - Given a class, find all instances.
- KnowItAll has a sophisticated mechanisms to assess the probability of every extraction, using Naïve Bayesian classifiers.

Syntactic patterns used in KnowItAll

NP1 {",") "such as" NPList2
NP1 {",") "and other" NP2
NP1 {",") "including" NPList2
NP1 {",") "is a" NP2
NP1 {",") "is the" NP2 "of" NP3
"the" NP1 "of" NP2 "is" NP3
…

Main Modules of KnowItAll

- Extractor: generate a set of extraction rules for each class and relation from the language patterns. E.g.,
  - "NP1 such as NPList2" indicates that each NP in NPList1 is a instance of class NP1. "He visited cities such as Tokyo, Paris, and Chicago".
  - KnowItAll will extract three instances of class CITY.
- Search engine interface: a search query is automatically formed for each extraction rule. E.g., "cities such as". KnowItAll will
  - search using a number of search engines
  - Download the returned pages
  - Apply extraction rule to appropriate sentences.
- Assessor: Each extracted candidate is assessed to check its likelihood for being correct.

Summary

- Knowledge synthesis is becoming important as we move up the information food chain.
- The questions is: Can a system provide a coherent and complete picture about a search topic rather than only bits and pieces?
- Key: Exploiting information redundancy on the Web
  - Using syntactic patterns, existing page organizations, and data mining.
- More research is needed.

5. Template Detection and Page Segmentation

Introduction

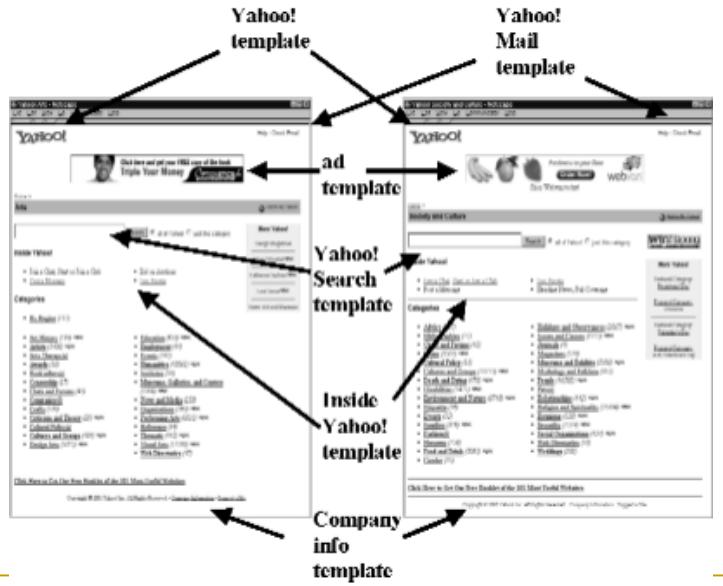
- Most web sites, especially commercial sites, use well designed templates.
  - A templatized page is one among a number of pages sharing a common look and feel.
- A templatized page typically contains many blocks:
  - Main content blocks
  - Navigation blocks
  - Service blocks,
  - Advertisements, etc.
- Each block is basically an (micro) information unit.
- Due to diverse information in the blocks, templatized pages affect search ranking, IR and DM algorithms.

Frequent pagelet (template) detection
(Bar-Yossef and Rajagopalan, WWW-02)

- Templates as frequent pagelets.
  - A pagelet is a self-contained logical region within a page that has a well defined topic or functionality (Chakrabarti, WWW-01).

Definition 1 [Pagelet - semantic definition] A pagelet is a region of a web page that (1) has a single well-defined topic or functionality; and (2) is not nested within another region that has exactly the same topic or functionality.

Definition 2 [**Pagelet - syntactic definition**] An HTML element in the parse tree of a page p is a pagelet if (1) none of its children contains at least k (=3) hyperlinks; and (2) none of its ancestor elements is a pagelet.



Templates as a collection of pages

Definition 3 [Template - semantic definition] A template is a collection of pages that (1) share the same look and feel and (2) are controlled by a single authority.

Definition 4 [**Template - syntactic definition**] A template is a collection of pagelets $p_1,\ldots,p_k$ that satisfies the following two requirements:

- *$C(p_i) = C(p_j)$ for all $1 \le i \ne j \le k$. // $C(p)$ is content of p*
- *$O(p_1),\ldots,O(p_k)$ form an undirected connected (graph) component. (O is the page owning p).*

- Content equality or similarity is determined using the shingle technique in (Broder et al, WWW-97).

The algorithm (given a set of linked pages G)

(1) Select all the pagelet shingles in PAGELETS that have at least two occurrences. Call the resulting table TEMPLATE_SHINGLES. These are the shingles of the re-occurring pagelets.

(2) Extract from PAGELETS only the pagelets whose shingle occurs in TEMPLATE_SHINGLES. Call the resulting table TEMPLATE_CANDIDATES. These are all the pagelets that have multiple occurrences in *G*.

(3) For every shingle *s* that occurs in TEMPLATE_SHINGLES define G_s to be the shingle's group: all the pages that contain pagelets whose shingle is *s*. By joining TEMPLATE_CANDIDATES and LINKS find for every *s* all the links between pages in G_s. Call the resulting relation TEMPLATE_LINKS.

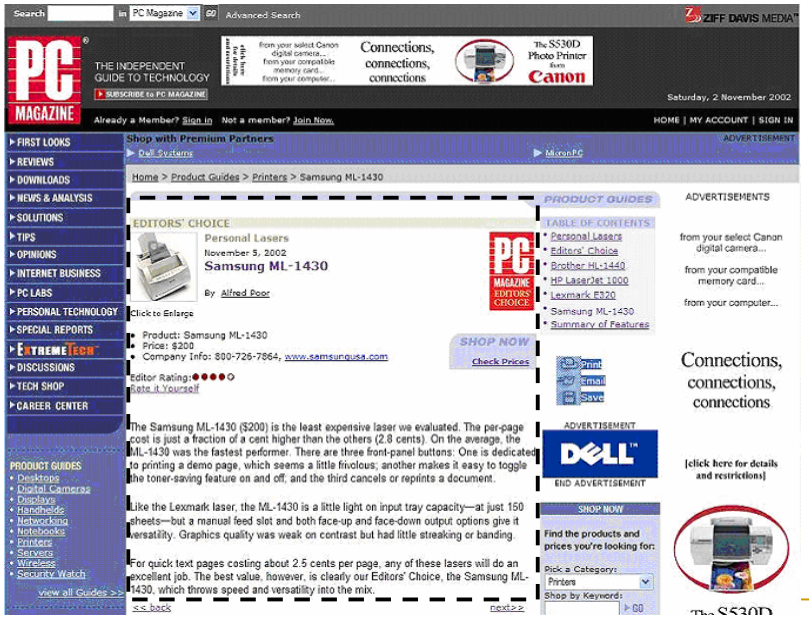
(4) Enumerate the shingles *s* in TEMPLATE_SHINGLES. For each one, load into main memory all the links between pages in G_s.

(5) Use a BFS algorithm to find all the undirected connected components in G_s. Each such component is either a template or a singleton. Output the component if it is not a singleton.

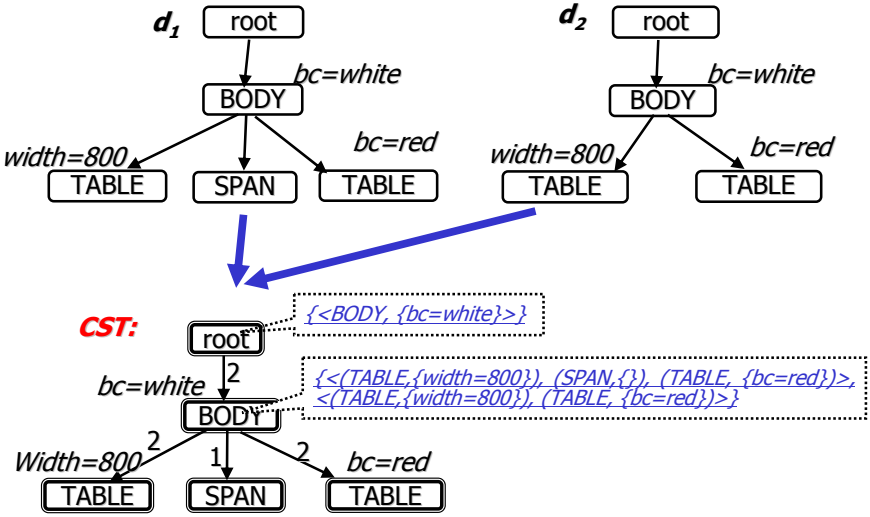
Templates as frequent DOM trees

(Yi and Liu, KDD-03 and IJCAI-03)

- Pages with similar "look and feel" are basically reflected by their similar DOM trees.
  - Similar layout or presentation Style
- Given a set of pages, the method merges their DOM trees to build a Compressed Structure Tree (CST).
  - Merge similar branches and
  - Split on differences
- The final CST represents a set of templates.
  - The algorithm uses CST for Web page cleaning and find main content blocks



Compressed Structure Tree



Merging trees (or pages)

- *Element node: E* = (*Tag*, *Attr*, *TAGs*, *STYLEs*, *CHILDs*)
  - *Tag* — tag name. E.g., TABLE, IMG;
  - *Attr* — display attributes of *Tag*.
  - *TAGs* — actual tag nodes
  - *STYLEs* — presentation styles
  - *CHILDs* — pointers to child element nodes
- Many techniques and their combinations can be used in merging:
  - HTML tags
  - Visual information, e.g., size, location, and background color.
  - Tree edit distance and alignment
  - Text contents at the leave nodes.
  - A combination of the above methods.

Finding the main content and noisy blocks

- Inner Node Importance: diversity of presentation styles

$$NodeImp(E) = \begin{cases} -\sum_{i=1}^{l} p_i \log_m p_i & if \quad m > 1 \\ 1 & if \quad m = 1 \end{cases} \qquad (1)$$

  - l = |*E.STYLEs*|, m = |*E.TAGs*|
  - p_i — percentage of tag nodes (in *E.TAGs*) using the *i-th* presentation style
- Leaf Node Importance: diversity of contents

$$NodeImp(E) = \frac{\sum_{i=1}^{N}(1 - H_E(a_i))}{N} = 1 - \frac{\sum_{i=1}^{N} H_E(a_i)}{N} \qquad (2)$$

  - N — number of features in E
  - a_i — a feature of content in E
  - $(1\text{-}H_E(a_i))$ — information contained in a_i

Identify the main content blocks or weighting features (words)

- Similarly, features or words can also be evaluated ($H_E(a_i)$).
- Based on the computation, one can combine the evaluations in various ways to:
  - Remove noisy blocks, e.g., navigation, site description, etc.
  - Weight features or words for data mining, as a feature selection mechanism based on the site structure of the pages.

Automatically extracting Web news
(Reis et al, WWW-04)

- The setting is similar to (Yi & Liu, KDD-03).
- Given a set of crawled pages, find the template patterns to identify news articles in the pages.
  - It first generates clusters of pages that share the same templates.
    - Distance measure is based on tree edit distance
  - Each cluster is then generalized into an extraction tree by tree matching,
  - Pattern: A regular expression for trees.

Learning Block Importance Models
(Song et al, WWW-04)

- Different blocks in a page are not equally important.
- Web designers tend to organize page contents to:
  - give prominence to important items and
  - deemphasize unimportant parts

  with features, e.g., position, size, color, word, image, link, etc.
- A block importance model is a function that maps from features to importance of each block.
  - Blocks are categorizes into a few importance levels.
  - A machine learning method is used to learn the model.
- Block segmentation is done using a visual-based method (Cai et al. APWeb-03).

Machine learning and user study

- Feature engineering
  - Spatial features:*BlockCenterX, BlockCenterY, BlockRectWidth, BlockRectHeight*
  - Content features: {ImgNum, ImgSize, LinkNum, LinkTextLength, InnerTextLength, InteractionNum, InteractionSize, FormNum, FormSize}
- Learning methods: SVM and Neural networks
  - SVM performs better
- A user study is also done showing that there is a general agreement of block importance.

Using Link Analysis to Rank blocks
(Yin and Lee, WWW 04)

- Given a Web page p arrived from a link S,
  - it first builds a graph similar to the Web link structure. Each node of the graph is a basic element of the page.
  - It then uses the PageRank algorithm to rank the basic elements.
  - Finally, It merges elements to form rectangular blocks according to their rankings, etc
- Application: Mobile Web browsing

Figure 1. Original HTML page

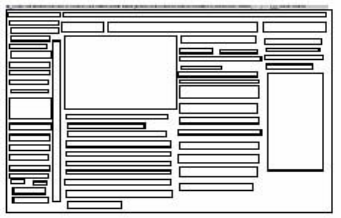


Figure 2. Decompose the original HTML pag

Build the graph and perform ranking

- The user enters a web page from a link S with anchor text.
  - Each basic element in the page is linked to S with an weight, which is computed based on type, size, location, shape and content similarity to the anchor text of S.
- Relationships between two basic elements are also represented with weighted edges.
  - The weight is a function of attributes of the two elements, such as word similarity and physical proximity of the elements within the page.
- Ranking of elements (PageRank):

$$PR^t(i) = (1-d) + d \sum_{(j,i)\in E} PR^{t-1}(j)/C(i)$$

Fragment (or blocks) detection

(Ramaswamy et al WWW 04)

- As in (Bar-Yossef and Rajagopalan, WWW-02), this paper also uses the shingling method in (Broder et al, WWW-97).
- Its block (called fragment in the paper) segmentation is more sophisticated, Based on AF-tree (augmented fragment tree), which is a compact DOM tree with
  - text formatting tags removed and
  - shingle values (encoding) attached to nodes.
- The method detects *Shared Fragments* and *Lifetime-Personalization based Fragments*

Detecting shared fragments

- Given a set of AF-trees, it uses the following to detect shared fragments in a set of pages.
  - **Minimum Fragment Size***(MinFragSize):* This parameter specifies the minimum size of the detected fragment.
  - **Sharing Factor***(ShareFactor):* This indicates the minimum number of pages that should share a segment in order for it to be declared a fragment.
  - **MinimumMatching Factor***(MinMatchFactor):* This parameter specifies the minimum overlap between the SubtreeShingles to be considered as a shared fragment.

Applications of Page Segmentation

- Removing noise or identifying main content blocks of a page, e.g., for information retrieval and data mining (Lin and Ho, KDD-02; Yi & Liu, IJCAI-03; Yi, Liu & Li, KDD-03; Reis et al, WWW-04; etc).
- Information unit-based or block-based Web search (e.g., Li et al, CIKM 02; Cai et al, SIGIR-04; Bar-Yossef and Rajagopalan, WWW-02).
- Browsing on small mobile devices (Gupta et al, WWW-03; Ying and Lee WWW-04).
- Cost-effective caching (Ramaswamy et al, WWW-04).
- Information extraction as discussed earlier.
- Etc.

Conclusions

- This tutorial introduced several topics of Web content mining:
  - Structured data extraction
  - Sentiment classification, analysis and summarization of consumer reviews
  - Information integration and schema matching
  - Knowledge synthesis
  - Template detection and page segmentation
- The coverage is by no means exhaustive.
- Research is only beginning. A lot to be done …
- References at:
  - http://www.cs.uic.edu/~liub/WebContentMining.html