CSE 763 Database Seminar

Herat Acharya

Towards Large Scale Integration: Building a MetaQuerier over Databases on the Web.

- Kevin Chen-Chuan Chang, Bin He and Zheng Zhang. (UIUC)

Few Slides and pictures are taken from the author's presentations on this paper.

Introduction

- **Deep Web:**

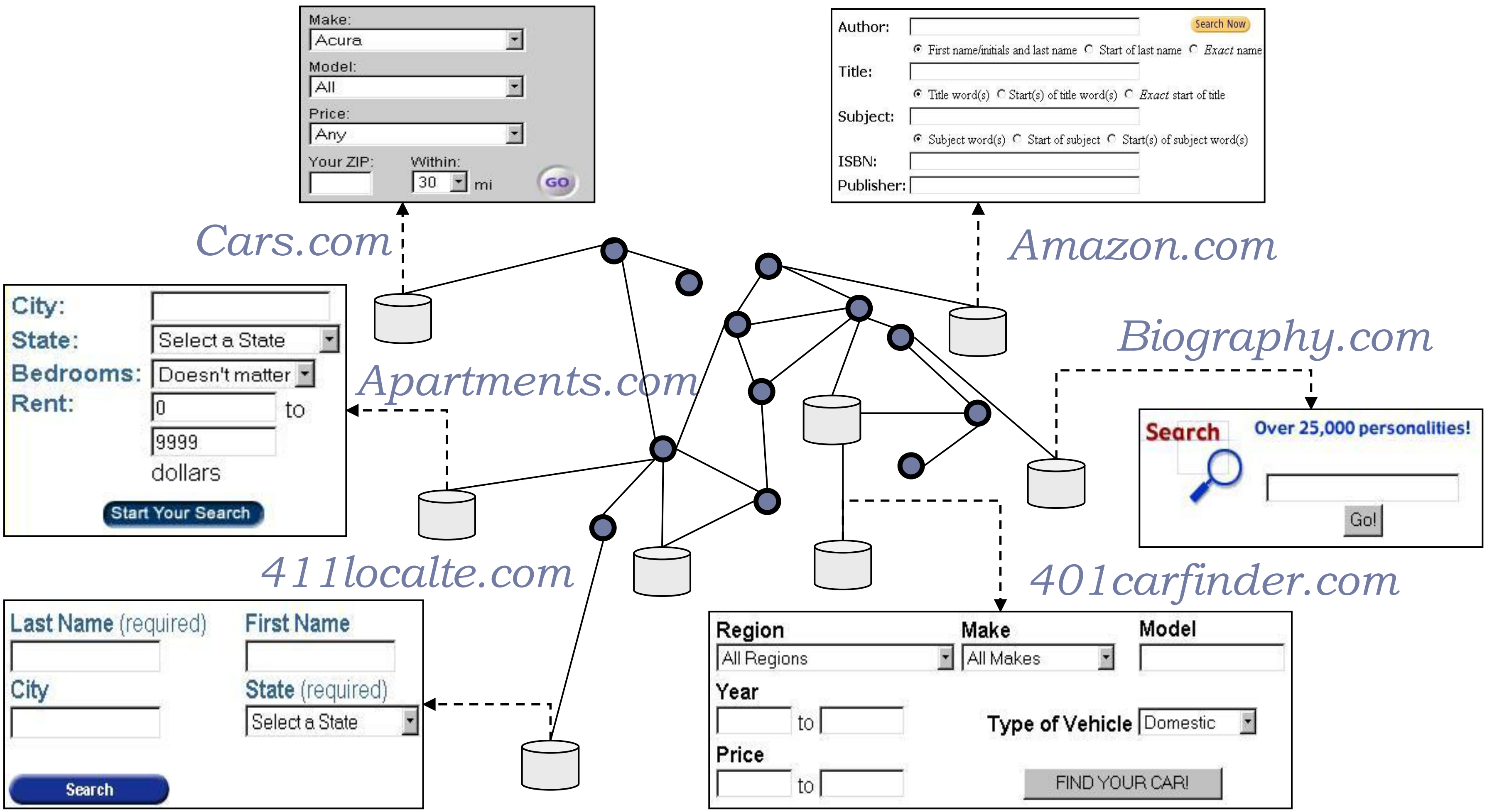
"The deep Web (also called Deepnet, the invisible Web, dark Web or the hidden Web) refers to World Wide Web content that is not part of the surface Web, which is indexed by standard search engines." – Wikipedia

- Since the structure data is hidden behind web forms, its inaccessible to search engine crawlers. For eg: Airline Tickets and Books website.
- *Finding sources:*
  - Wants to upgrade her car– Where can she study for her options? (cars.com, edmunds.com)
  - Wants to buy a house – Where can she look for houses in her town? (realtor.com)
  - Wants to write a grant proposal. (NSF Award Search)

    Wants to check for patents. (uspto.gov)

- *Querying sources:*
  - Then, she needs to learn the grueling details of querying

Introduction – Deep Web

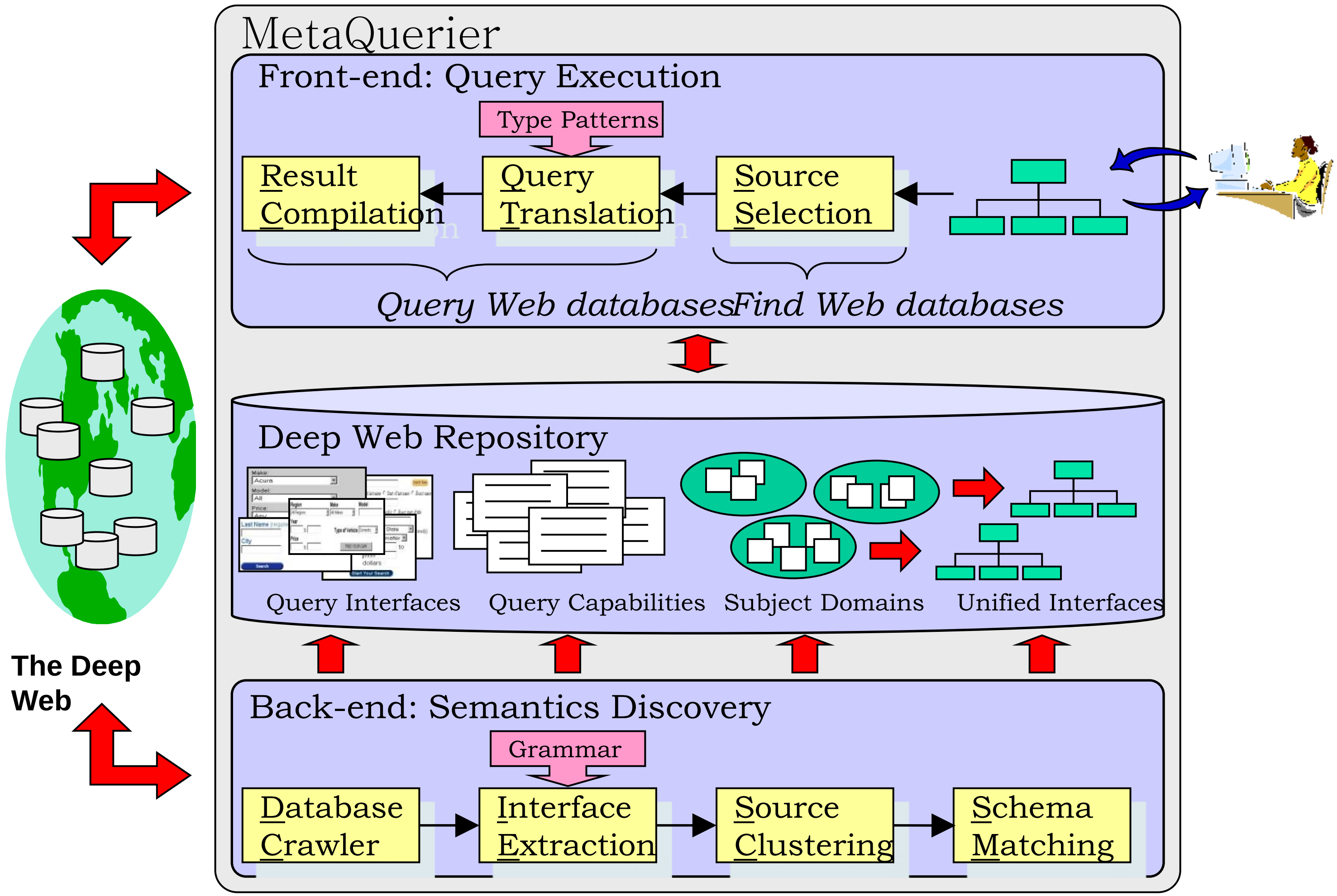


Goals and Challenges

- **Goals:**
  - **To make the Deep Web systematically accessible. This will help the users to find online databases useful for their queries.**
  - **To make the Deep Web uniformly usable. That is to make it user friendly so that the user can query databases with no or least prior knowledge of the system.**

- **Challenges:**
  - **The deep Web is a large collection of queryable databases and it is only increasing.**
  - **Requires the integration to be dynamic. Since the sources are proliferating and evolving on the web, this cannot be statistically configured.**
  - **The system is ad-hoc as the most of the time the user knows what is he searching for in structured databases.**
  - **Since the system is ad-hoc it must do on the fly integration.**

System architecture



Demo


Cazoodle
SEARCH, INTEGRATE, AND ORGANIZE - THE REAL WORLD.
About / Blog / Careers / CazoodleBot / Contact / News / Products
Search, integrate, and organize-- the real world. Cazoodle provides effective access to structured information on the Internet.
Apartment Search (Nationwide and all locations)
One Search, All Apartments, Entire Web!
Shopping Search for Electronics (All consumer electronic products)
A Tiny, Yet Powerful Shopping Engine for Electronics!
© 2010 Cazoodle

System architecture

- **Backend:**
  - Automatically collects Deep Web sources from the crawler.
  - Mines sources semantics from the collected sources.
  - Extracts query capabilities from interfaces.
  - Groups (or clusters) interfaces into subject domains.
  - Discovers semantic (schema) matching.
- **Deep Web Repository:**
  - The collected query interfaces and discovered semantics form the Deep Web Repository.
  - Exploited by the frontend to interact with the users.
  - Constructed on the fly.
- **Frontend:**
  - Used to interact with the users.
  - It has a hierarchy based on domain category which is automatically formed by source clustering in the backend.
  - User can choose the domain and query in that particular domain.

Subsystems

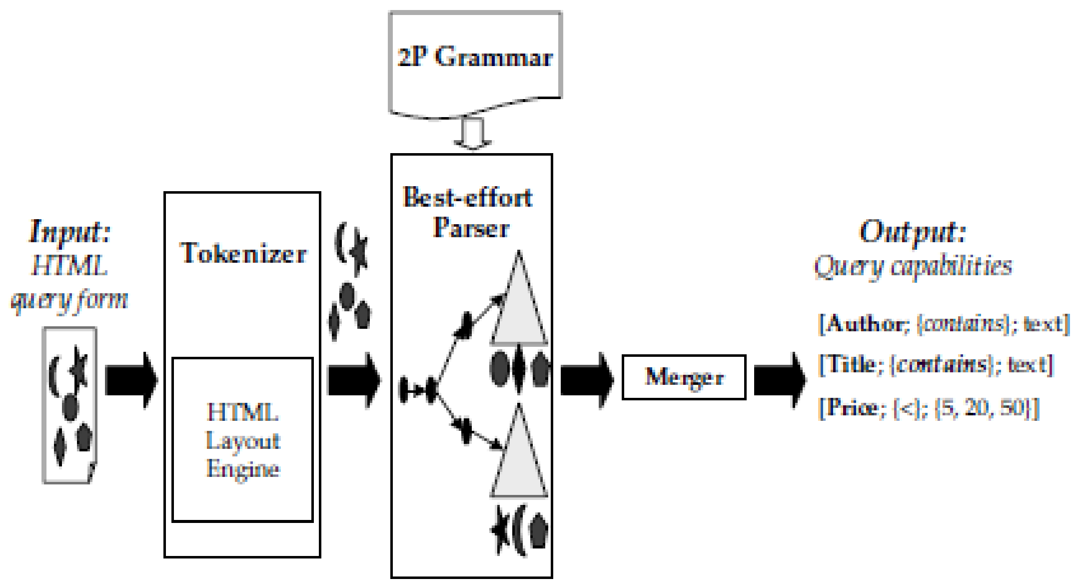
- **Database Crawler (DC):**
  - **Functionality:**
    - **Automatically discovers Deep Web databases, by crawling the web and identifying query interfaces.**
    - **Query interfaces are passed to interface extraction for source query capabilities.**
  - **Insight:**
    - **Building a focused crawler.**
    - **Survey shows that the web form(or query interface) is typically close to the root (or home page) of the Web site, which is called depth.**
    - **Statistics of 1,000,000 randomly generated IPs show that very few have depth more than 5 and 94% have depth of 3.**
  - **Approach:**
    - **Consists of 2 stages: Site collector and shallow crawler.**
    - **Site collector finds valid root pages or IPs that have Web Servers. There are large no. addresses and a fraction of them have Web servers. Crawling all addresses is inefficient.**
    - **Shallow crawler crawls the web server from the given root page. It has to crawl only starting few pages from the root page according to the statistics above.**

Subsystems

- **Interface Extraction (IE):**
  - **Functionality:**
    - **The IE subsystem extracts the query interface from the HTML format of the Web page.**
    - **Defines a set of constraint templates in the form of *[attribute;operator;value]* . IE extracts such contraints from a query interface.**
    - **For eg: S_1 :[title;contains;$v] , S_2 :[price range;between;$low,$high]**
  - **Insight:**
    - **Common query interface pattern in a particular domain.**
    - **Hence there exists a hidden syntax across holistic sources (Hypothesis).**
    - **Therefore this hypothesis transforms an interface into a visual language with a non-prescribed grammar. Hence it finally becomes a parsing problem.**
  - **Approach:**
    - **The HTML format is tokenized by the IE, these tokens are parsed and then merged into muiltiple parsed tress. This consists of a 2P grammar and best effort parser.**
    - **Human first examines varied interfaces and creates a 2P grammar. These consists of productions which capture hidden patterns in the forms.**
    - **Patterns might conflict thus its conventional precedence or priorities are also captured called as preferences.**

Subsystems

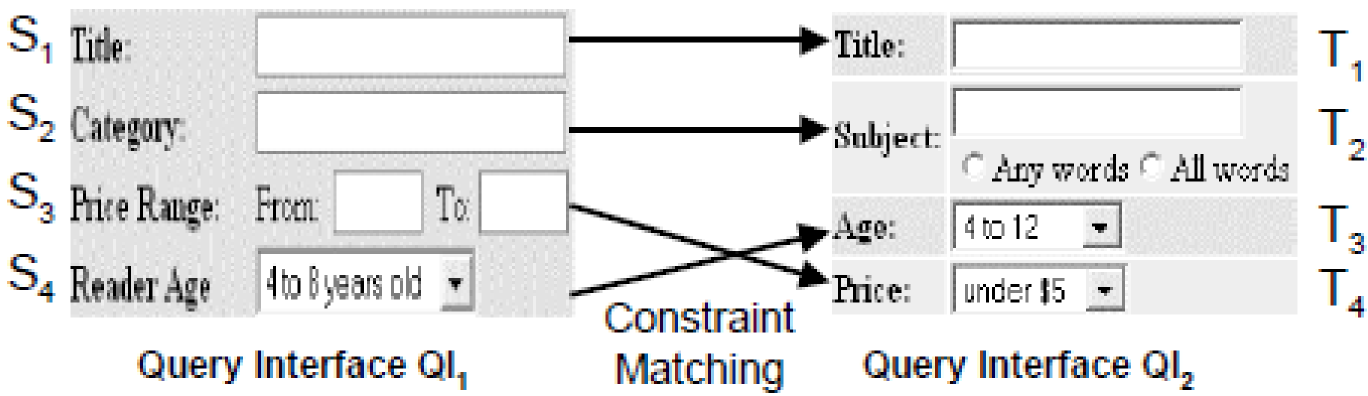
- **Approach: (contd.)**
  - **The hypothetical syntax is dealt by the best effort parser.**
  - **It prunes ambiguities by applying preferences from the 2P grammar and recognizes the structure and maximizes results by applying productions.**
  - **Since it merges multiple parse trees an error handling mechanism is also employed(to be seen in the later slides).**
  - **Merger parses all the parse trees to enhance the recall of the extraction.**



Subsystems

- **Schema Matching (SM):**
  - **Functionality:**
    - Extracts semantic matching among attributes from the extracted queries.
    - Complex matching is also considered. For eg: m attributes are matched with n attributes thus forming an m:n matching pattern.
    - Discovered matching are stored in Deep Web Repository to provide a unified user interface for each domain.
  - **Insight:**
    - Proposes an holistic schema matching that matches many schemas at same time.
    - Current implementation explores co-occurrence patterns of attributes for complex matching.
  - **Approach:**
    - A two step approach: data preparation and correlation mining.
    - The data extraction step cleans the extracted queries to be mined.
    - Correlation mining discovers correlation of attributes for complex matching schemas.

Subsystems



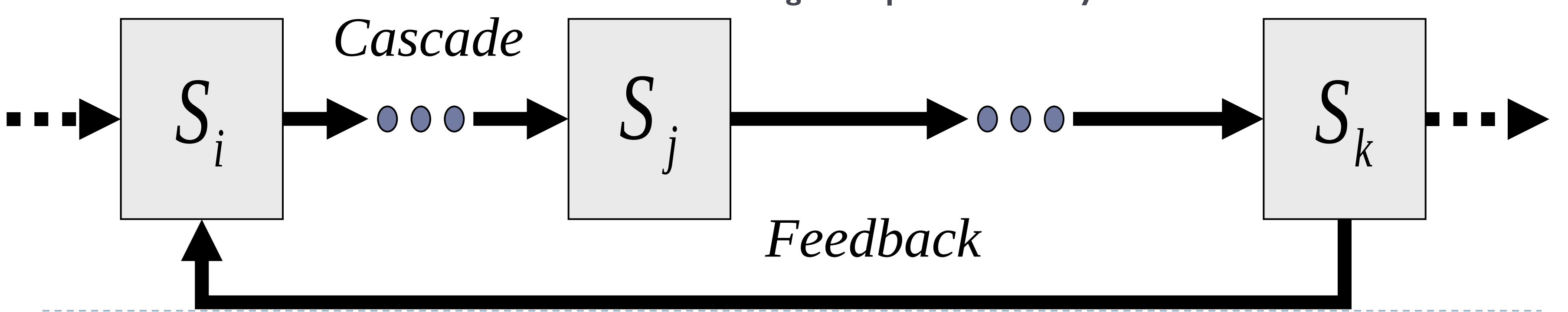
Example of Schema Matching

Putting Together: Integrating Subsystems

- **With just the single system integration, errors persist.**
- **Different interpretations of the same token may lead to conflicts. For eg: after a name field there is a label field with Mike. This is conflicting with the system as to what should it consider name or Mike.**

To increase the accuracy of the subsystems, authors propose 2 methods

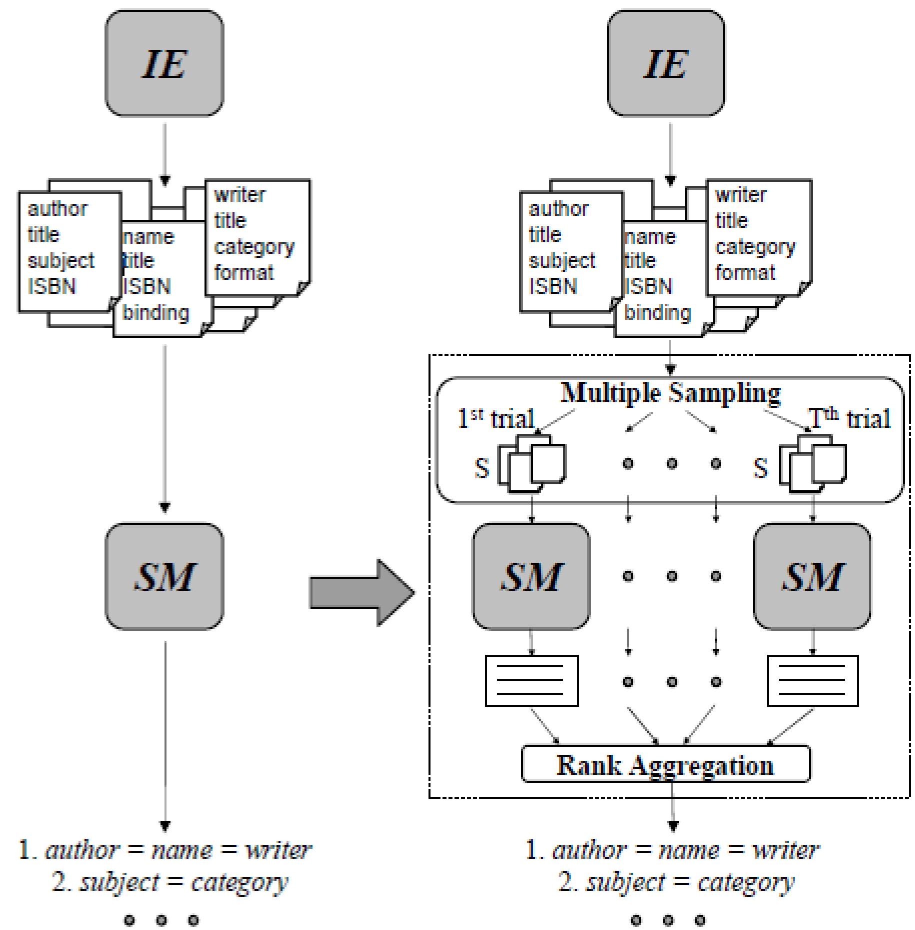
- **Ensemble cascading:**
  - **To sustain the accuracy of SM under imperfect input from IE.**
  - **Basically cascades many SM subsystems to achieve robustness.**
- **Domain feedback:**
  - **To take advantage of the information in latter subsystems.**
  - **This improves accuracy of IE.**
  - **Uses domain statistics from schema matching to improve accuracy.**



Putting Together: Integrating Subsystems

- **Ensemble Cascading:**
  - With just a single SM subsystem connected with IE, performance degrades with noisy input.
  - Hence we don't need all input schemas for matching .
  - Voting and sampling techniques are used to solve the problem.
  - First sampling is done and a subset of input schemas are chosen.
  - There are abundant schemas, hence its likely to contain correct schemas.
  - Sampling away some schemas many reduce noise as the set is small.
  - Multiple sampling is taken and given to rank aggression.
  - Rank aggression combines all schemas and does a majority voting
  - Majority voting involves selecting those inputs which frequently occur.
  - Foreg: author, title, subject, ISBN in a book site.

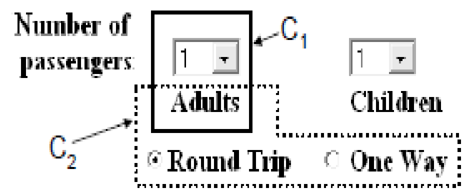
Putting Together: Integrating Subsystems



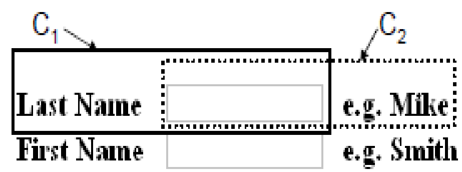
(a) The base framework
(b) The ensemble framework

Putting Together: Integrating Subsystems

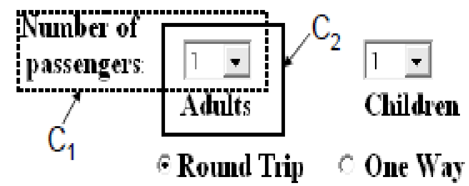
- **Domain Feedback:**



(a) Conflict 1 in query Interface QI_a



(b) Conflict 2 in query Interface QI_b



(c) Conflict 3 in query Interface QI_a

- **In Fig a:**

C_1 = [adults,equal,$val:{1,2,..}] and C_2 = [adults,equal,$val:{round-trip,oneway}]

They conflict because there system.

But by observing the distinctive patterns in other interfaces, it concludes adults is a numeric type.

- **Large amount of information to resolve conflicts are available from peer query interfaces in the same domain.**

Putting Together: Integrating Subsystems

- **Domain Feedback:**

Three domain statistics have been observed to effectively solve conflicts:

- ***Type of attributes:***

  Collects common type of attributes. For eg: when matching 2 schemas of Books domain Title is a common attribute.

- ***Frequency of attributes:***

  Frequency of the attributes occurring in the schema is taken into consideration. For eg: In airlines domain departure city, departure date, passengers, adults, children are frequently occurring attributes.

- ***Correlation of attributes:***

  Takes correlation of attributes within the group, i.e. attributes within the group are positively correlated and attribute across groups are negatively correlated.

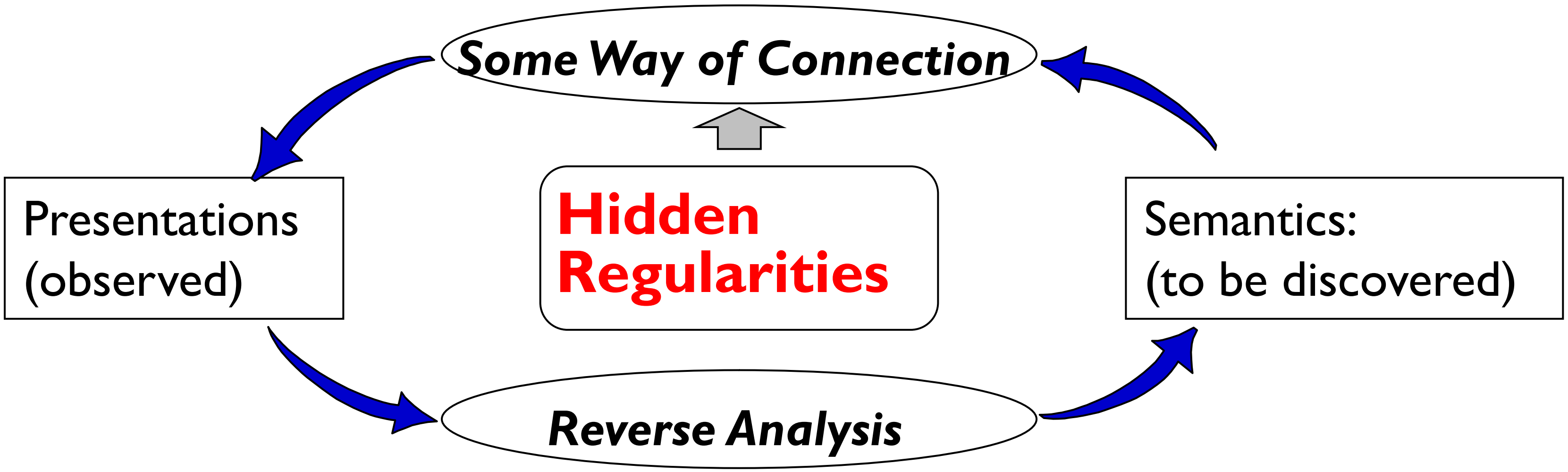
Unified Insight: Holistic Integration

- How it is done in MetaQuerier?
  - Its all about sematics discovery.
  - Take a holistic view to account for many sources together in integration
  - Globally exploit clues across all sources for resolving the ``semantics" of interest
  - A conceptually unifying framework.

- Proposed ways of Holistic Integration:
  - Hidden Regularities
  - Peer Majority

Unified Insight: Holistic Integration

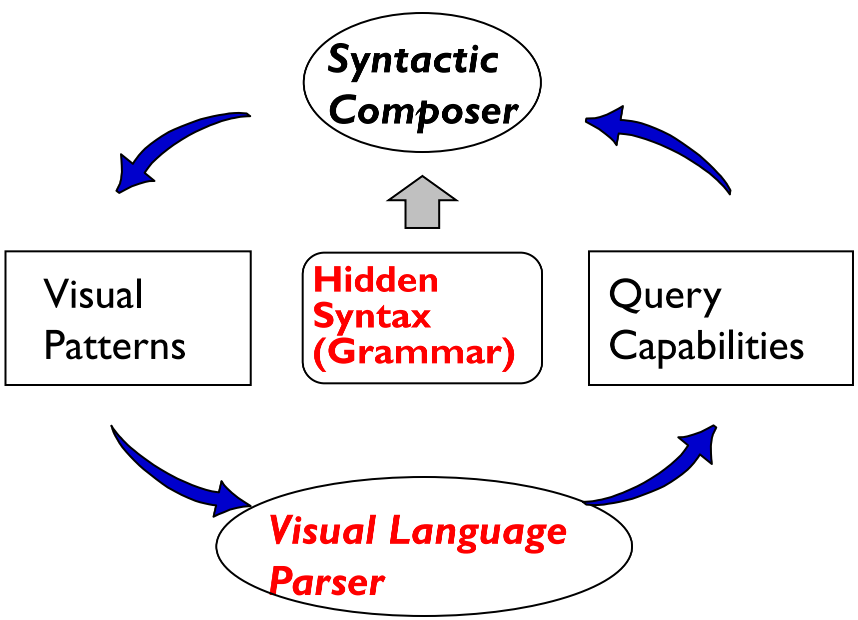
- Hidden Regularities:
  - Deals with finding hidden information that helps in sematics discovery.
  - For eg: For IE its hidden syntax and for SM its hidden schema.
  - *Shallow observable clues*: ``underlying'' semantics often relates to the ``observable'' presentations in some way of connection.
  - *Holistic hidden regularities*: Such connections often follow some implicit properties, which will reveal holistically across sources.
  - *Reverse analysis* has to be done which holistically analyzes shallow clues as guided by hidden regularities.



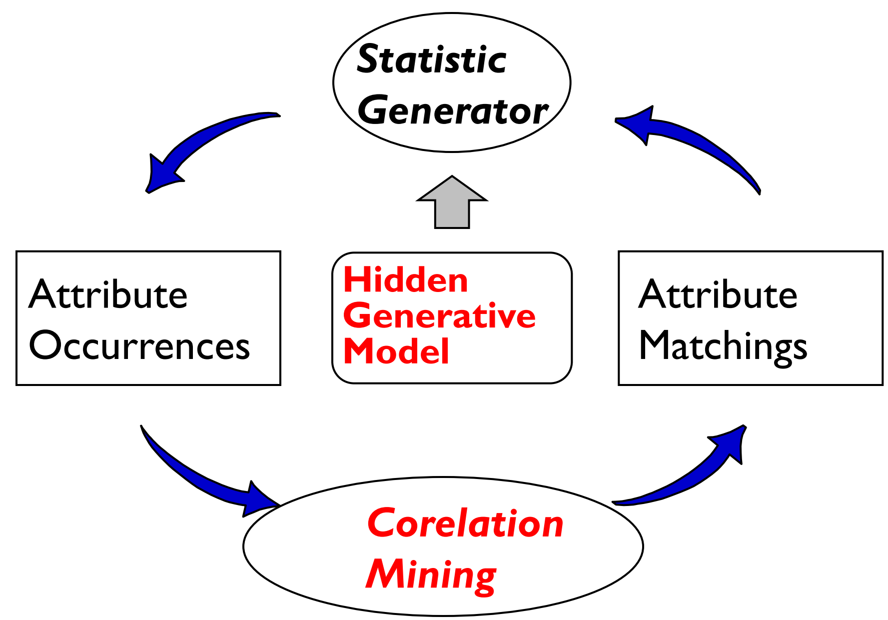
Unified Insight: Holistic Integration

Hidden Regularities (cont)

- Evidence 1: [SIGMOD04]
  Query Interface Understanding
  by *Hidden-syntax parsing*



- Evidence 2: [SIGMOD03, KDD04]
  Query Interfaces Matching
  by *Hidden-model discovery*



Unified Insight: Holistic Integration

Hidden Regularities (cont)

- Evidence 1: [SIGMOD04]

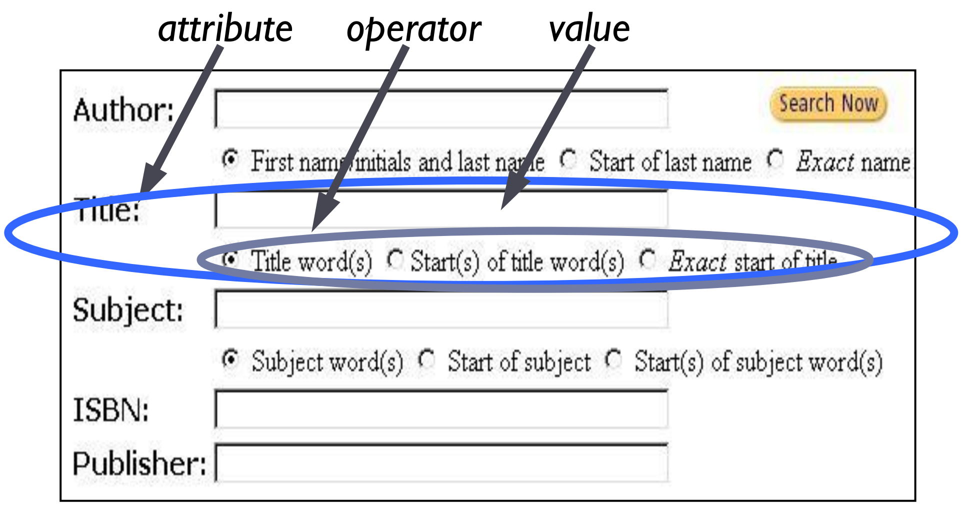
  Query Interface Understanding (IE)

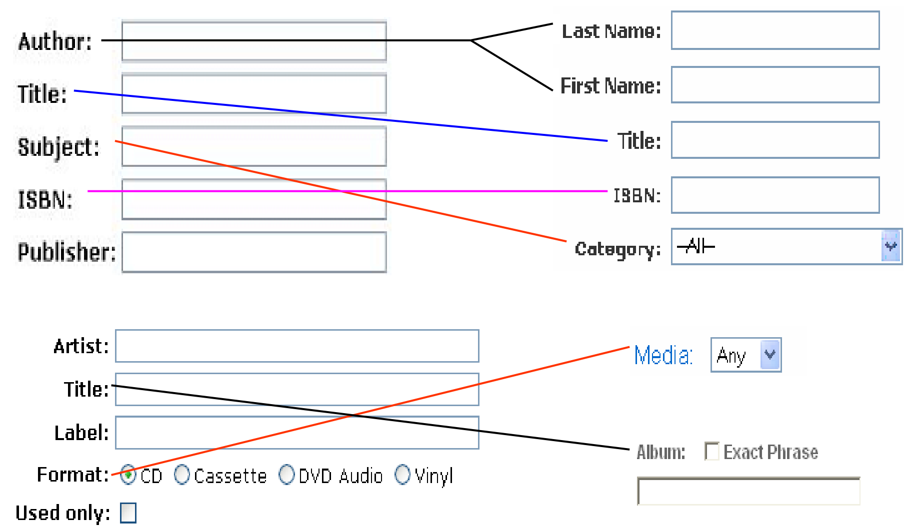
  Hidden-syntax parsing

- Evidence 2: [SIGMOD03, KDD04]

  Matching Query Interfaces (SM)

  Hidden-model discovery





Unified Insight: Holistic Integration

- Peer Majority (Error Correction):
  - Basically deals with gathering information from peers or neighboring subsystems for error correction.
  - This is based on following hypothesis:
    - *Reasonable base:* The base algorithm is reasonable. Its not perfect but errors are rare.
    - *Random samples:* Base algorithm can be executed over randomly generated samples.
  - *Foreg:*
    - *Ensemble Cascading:*

      SM enhances accuracy for matching query schemas. SM creates multiple samples of schemas by "downsampling" the original input, hence we create random samples and we assume that the algorithm for SM produces correct output. Thus we do majority voting which increases the accuracy of the system
    - *Domain Feedback:*

      This feature increases the accuracy of IE subsystem. The crawler is run for every interfaces, thereby creating multiple samples and we assume the base algorithm is reasonable. Feedback mechanism gathers statistics from all samples indicating majority.

Conclusions

- Problems in accessing structured databases on the Web.
- System architecture of MetaQuerier.
- How the systems are integrated holistically.
- What have we learnt while integrating the subsystems?

Entity Rank: Searching Entities Directly and Holistically

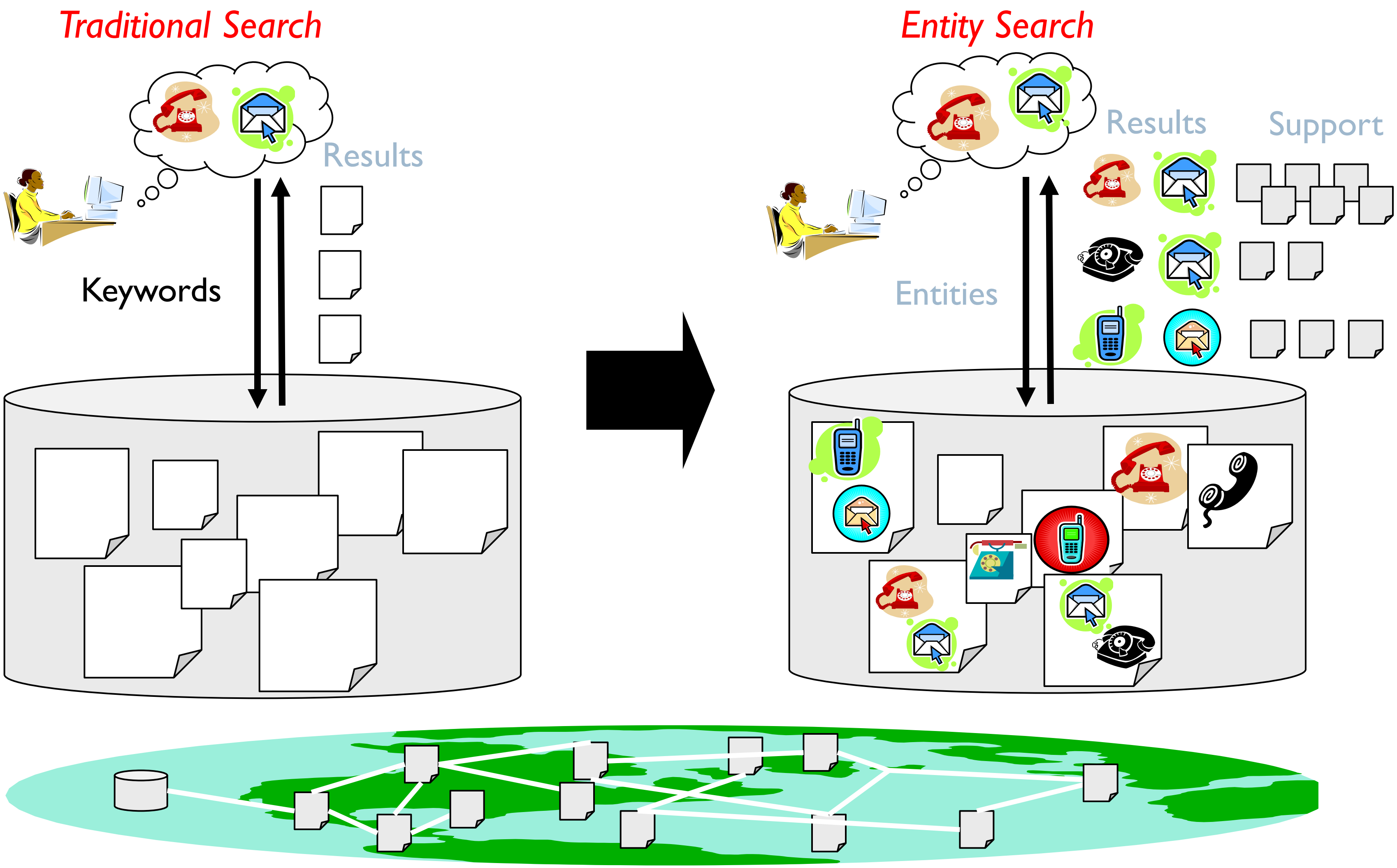
- Tao Cheng, Xifeng Yan, Kevin Chen-Chuan Chang. (UIUC)

Few Slides and pictures are taken from the author's presentations on this paper.

Entity Search - Introduction

- Focuses on data as an "Entity" rather than data as a document.
- Consider few scenarios:
  - *Scenario 1:* Amy want to find customer service "phone number" of Amazon.com. How does she go about finding it on the Web? Finding an entity such as a phone no. can be time consuming on the Web as Amy has to browse several pages to find one.
  - *Scenario 2:* Amy wants to apply for graduate schools. How can she find "professors" in "database" area of a particular school. Likewise she has to go through various departmental web pages to find what she wants.
  - *Scenario 3:* Amy wants to prepare for a seminar. How can she find a "pdf" of a "ppt" of a "research paper"?
  - *Scenario 4:* Now Amy wants to read a book. How can she find the exact "prices" and "cover images" of the books she likes to read without minimal effort?
- The problem of finding exactly what we want is addressed in the Entity Search.

Traditional Search Vs Entity Search



How does Entity Search work?

- As input, users describe what they are looking for.
- User can specify entity and keywords.
- To distinguish between entity and keywords user use "#".
- For eg:
  - Query Q1: ow(amazon customer service #phone)
  - Query Q2: (#professor #university #research="database")
  - Query Q3: ow(sigmod 2006 #pdf_file #ppt_file)
  - Query Q4: (#title="hamlet" #image #price)
- Context pattern: A target entity matches any instance of that entity type.
- Content restriction: How will results appear?

How does Entity Search work?

As an output they will directly get what they want.

Entities are matched holistically and are ordered according to their scores.

rank	phone number	score	urls
1	800-201-7575	0.9	amazon.com/support.htm myblog.org/shopping
2	800-988-0886	0.8	Dell.com/supportors
3	800-342-5283	0.6	xyz.com
4	206-346-2992	0.2	hp.com
....	....	....	....

rank	PDF	PPT	score	urls
1	sigmod6.pdf	sigmod6.ppt	0.8	db.com,sigmod.com
2	surajit21.pdf	surajit21.ppt	0.7	ms.com
....	....	....	....	....

The Problem: Entity Search

- Not like finding documents on the Web. The system must be made "entity-aware".
- We consider $E = \{E_1, E_2, \ldots E_n\}$ as a set of entities over a document collection $D = \{D_1, D_2, \ldots D_n\}$
- Since entity search is a contextual search it lets the user specify patterns (α) , i.e. how they may appear in certain pattern in collection D.
- The output is ranked as m-ary entity tuples in the form of $t = \{e_1, e_2, \ldots, e_n\}$.
- The measure of how t matches the query q is denoted by a query score as:

$$Score(q(t)) = Score(\alpha(e_1, e_2, \ldots e_m, k_1, k_2, \ldots k_l))$$

Where q(t) is the measure of how t appears according to the tuple pattern α across various documents

Characteristic I – Contextual

Appearance of keywords and entity instances might be different. There are 2 factors – Pattern and Proximity

Characteristic II – Uncertainty

Entity extraction is always not perfect and its extraction confidence probability must be captured.

Steve Lawrence, Luis Gravano: Learning to find answers to questions on the Web. ACM Trans. Internet Techn. 4(2): 129-162 (2004)

Luis Gravano, Amélie Marian: Optimizing Top-k Selection Queries over Multimedia Repositories. IEEE Trans. Knowl. Data Eng. 16(8): 992-1009 (2004)

Xantrex Technologies XPower Plus 400-Watt Inverter 851-0400
Xantrex (May 14, 2003)
Average Customer Review: (58)
In Stock

List Price: ~~$59.99~~

Characteristic III – Holistic

A specific entity may occur multiple times in many pages. Every instance of the entity must be aggregated.

Amazon.com: The Death of Customer Service

By Antoine du Rocher

SAN FRANCISCO, 28 December 2003—Customer service, small surprise, has been one of the casualties of America's drive towards cost-cutting in the age of e-business. The movement of customer service call centers off-shore is one-upped by companies like Amazon.com, which increasingly are hiding their customer service telephone numbers and other contact information, in order to prevent dissatisfied customers from calling in for service at all.

Amazon US Customer Service
1.800.201.7575 (Toll free, US and Canada)
1.206.346.2992 or 1.206.266.2992 (Outside US and Canada)
1.877.586.3230 (Canada only)

Gregory (Grisha) Chockler
Research Staff Member,
IBM Haifa Research Laboratory

During 2003-2005, I was a postdoctoral associate with the Theory of Distributed Systems group, MIT/ CSAIL

Ph.D., CS and Eng. School, The Hebrew University of Jerusalem, Israel, 2003.

AMAZON.COM customer service phone number (US): (800) 201-7575

Digging up buried info, like how to quit AOL

August 17, 2006

By Jim Rossman / The Dallas Morning News

Jim Rossman is your Tech Adviser offering advice and tips for computer hardware and programs. Helpful links are included. Jim Rossman is technical manager for Macintosh support for Belo Corp.

Reaching eBay, Amazon

While I'm at it, another hard-to-find phone number is the customer service line for eBay.

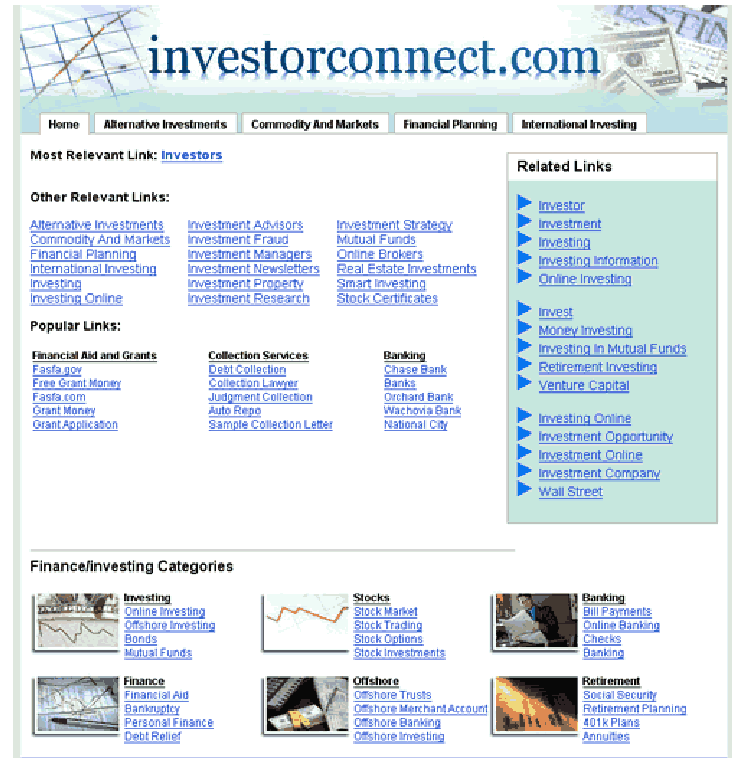
The main number for eBay is 1-888-749-3229. Once connected, press 2 for customer service.

Another handy phone number for eBay users is the customer care number for PayPal, 1-888-221-1161.

I'll throw in one more — Amazon's number is 1-800-201-7575.

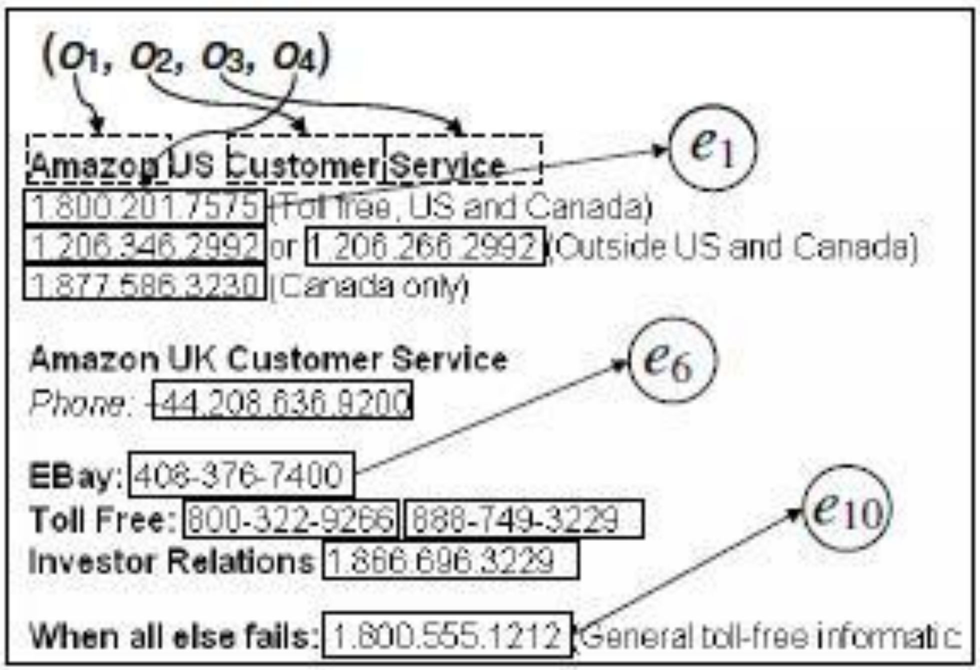
Characteristic IV – Discriminative

Entity instances matched on more popular pages should be ranked higher than instances matched on lesser popular pages.



Characteristic V – Associative

- An entity instance must not be accidental.
- Hence we must carefully calibrate to purify the associations we get.



The Impression Model - Theoretically

- Assuming:
  - No time constraints
  - Unlimited resources

- For query Q1 = ("amazon customer service", #phone), collection over Web say D.
  - Dispatch an *observer* to repeatedly access Web D.
  - Collects all evidence for potential answer.
  - Examines the document d for any instance of #phone near the keyword.
  - Forms a judgment of how good the matches are and due to unlimited memory he remembers every judgment.
  - Stops when he gets sufficient evidences and calculates the score.

The Impression Model - Theoretically

- *Access layer:* For accessing each document .
- *Recognition layer:* While searching the document, it recognizes any tuple present.
- *Association Probability:* Signifies the relevance of the tuple.
- At some time $\tau = T$ the observer may have sufficient trials. At that point his impression stabilizes.
- The Access probability is p(d) ie probability that observer visits a document d.
- Hence over T trials d will appear T x p(d) times
- Thus if T is sufficiently large association probability of q(t) over entire collection D will be :

$$p(q(t)|D) = \lim_{T \to \infty} \frac{\sum_{\tau=1}^{T} p(q(t)|d^{\tau})}{T} = \sum_{d \in D} p(d) \cdot p(q(t)|d)$$

The Impression Model – The naïve observer

- Treats all documents uniformly.
- *Access layer:* Views each document equally with uniform probability ie

  $$p(d) = \frac{1}{n}, \forall d \in D \quad \text{(where } |D| = n\text{)}$$

- *Recognition layer:* The observer accesses p(q(t)|d) by document co-occurrence for all entity and keywords specified in q(t) ie p(q(t)|d) = 1 if they occur 0 otherwise.
- *Overall Score Thus the overall score is given by:*

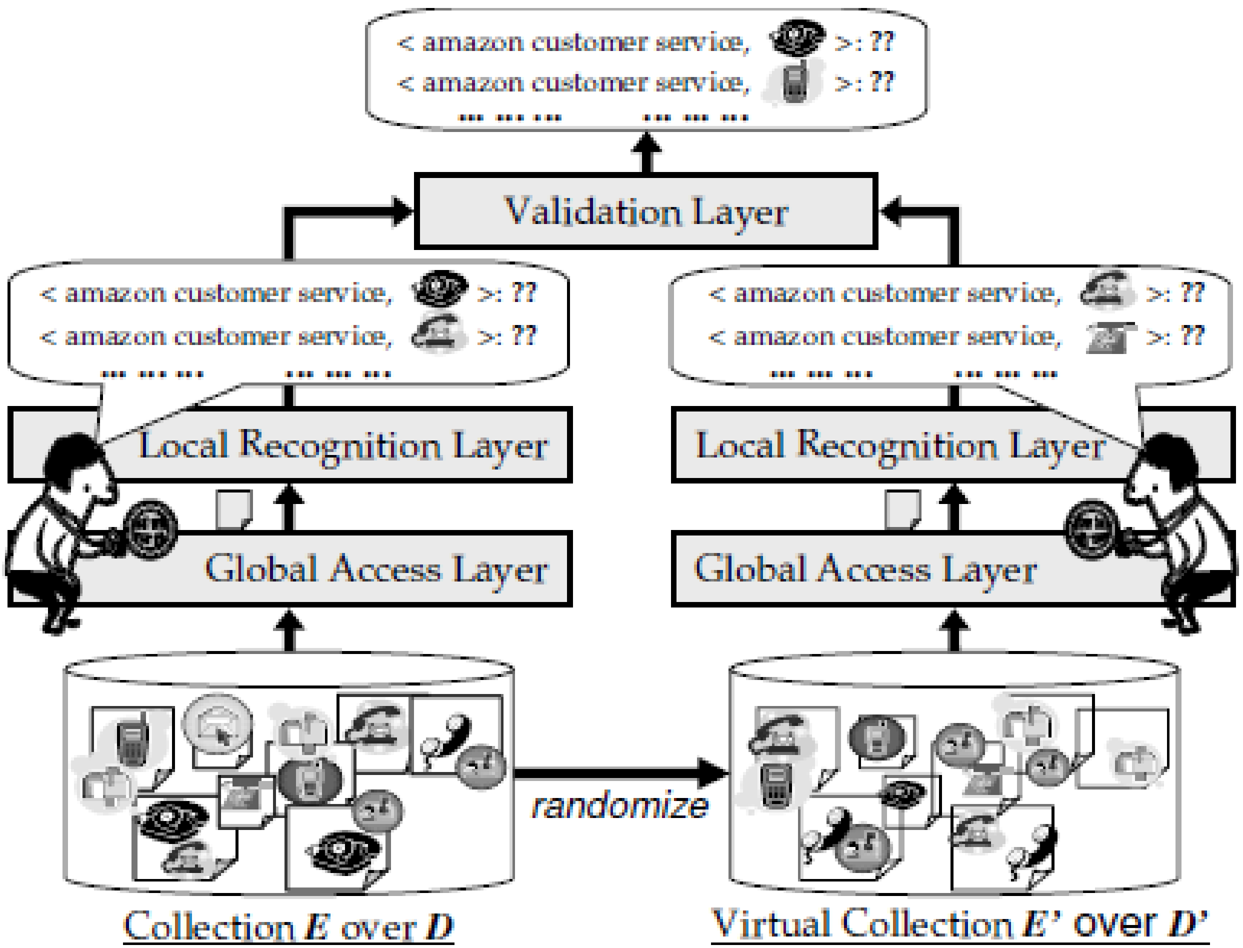
  $$Score(q(t)) = \sum_{d \in D} \frac{1}{n} \cdot \begin{cases} 1 & \text{if } q(t) \in d \\ 0 & \text{otherwise} \end{cases} = \frac{1}{n} C(q(t)),$$

- Limitations :
  - Does not discriminate sources.
  - Not aware of entity uncertainty and contextual patterns
  - A validation layer is lacking.

Entity Rank - Concretely

- A new virtual observer is introduced who will perform the observation job over a randomized version of D say D' .
- A validation layer to compares the impression of real observer with that of virtual observer.
- Defines 3 layers:
  - Access layer (Global Aggregation)
  - Recognition layer (Local Assessment)
  - Validation Layer (Hypothesis Testing)

Entity Rank - Concretely



Access Layer – Global Aggregation

- Defines how the observer selects the documents.
- Discriminates the documents searched by their "quality".
- Measure of quality depends on document collection ie its structure – for web documents the notion of popularity metric is chosen.
- *Random walk model:* It defines p(d), which is the probability of visiting a document d.
- It used PageRank method to find out the popularity metric ie p(d) = PR[d].

Recognition Layer – Local Assessment

- Defines how observer examines a document d locally for a tuple.
- This layer determines $p(q(t)|d)$ ie how query tuple $q(t)$ in the form of $\alpha(e_1,e_2,\ldots.e_m,k_1,\ldots.k_l)$ holds true given d.
- Each entity or a keyword may appear many times. They combine all the instance as described : $\gamma(o_1,o_2,\ldots..o_n)$.
- Hence , $$p(q(t)|d) = \max_{\gamma \text{ is a tuple occurrence of } q(t)} p(\alpha(\gamma))$$

Where $$p(\alpha(\gamma)) \;=\; (\prod_{e_i \in \gamma} e_i.conf) \times p_{context}(\alpha(\gamma))$$

- Next, to define context operator $p_{context}$ ie how γ occurs in a way matching α in terms of context.
- Its done in 2 steps:
  - Boolean pattern analysis
  - Probabilistic proximity analysis.

Recognition Layer – Local Assessment

- Boolean pattern analysis:
  - Its defined as α_B which returns 1 or 0 whether some pattern is satisfied or not.
  - For eg: doc($o_1, o_2, \ldots . o_m$) objects must occur in the same document.
- Probabilistic proximity analysis:
  - Defines α_P , how well the proximity between objects match the desired tuple.
  - The closer they appear to each other the more relevant they are as a tuple (span proximity model).

$$\alpha_P(\gamma) \equiv p(\gamma \text{ is a tuple} | s), \text{or simply } p(\gamma \,| s).$$

$$p(\gamma \,| s) = \frac{p(\gamma)}{p(s)} p(s|\gamma) \propto p(s|\gamma)$$ (by applying Bayes' Theorem)

$$p(q(t)|d) = \max_{\gamma} \prod_{e_i \in \gamma} e_i.conf \times \alpha_B(\gamma) \times p(s|\gamma)$$

Validation Layer – Hypothesis Testing

- Validates the significance of the impression.
- Suggested null hypothesis to validate thereby simulating a virtual observer.
- Create a randomize version of D say D'.
- First we randomly search entities and keywords in D' with same probability of appearing in any document of D.
- Thus probability of entity/keyword belonging to d' is:

$$p(e_i \in d') = \sum_{e_i \in d, d \in D} p(d); \quad p(k_j \in d') = \sum_{k_j \in d, d \in D} p(d)$$

- Probability that a tuple belonging to entire collection D' is

$$\begin{aligned} p(q(t)|D') &= \sum_{d' \in D' \text{and } q(t) \in d'} p(d') \times p(q(t)|d') \\ &= p(q(t)|d') \times \sum_{d' \in D' \text{and } q(t) \in d'} p(d') \\ &= p(q(t)|d') \times p(q(t) \in d'). \end{aligned}$$

- $p(q(t) \in d')$ is the probability of t appearing in some document d'. Its defined by:

$$p(q(t) \in d') = \prod_{j=1}^{m} p(e_j \in d') \prod_{i=1}^{l} p(k_i \in d')$$

Validation Layer – Hypothesis Testing

- Next we define a probability of tuple t in d'

$$p(q(t)|d') = (\prod_{j=1}^{m} \overline{e_j.conf}) \times p_{context}(q(t)|d')$$

- The contextual probability is defined by :

$$p_{context}(q(t)|d') = \overline{p}(q(t)|s) = \frac{\sum_s p(q(t)|s)}{|s|}$$

- Putting all these equations together we get p_r
- Now we should compare p_r with p_o . Using *G-Test* we compare these 2 values. The score is given by

$$Score(q(t)) = 2(p_o log\frac{p_o}{p_r} + (1 - p_o) log\frac{1 - p_o}{1 - p_r})$$

- Higher the G-Test score the more likely that entity instances t appear with keyword k. Here $p_o, p_r << 1$.

$$Score(q(t)) \propto p_o \cdot log\frac{p_o}{p_r}$$

Entity Rank – Scoring Function

- **Query:** $q(\langle E_1, \ldots, E_m \rangle) = \alpha(E_1, \ldots, E_m, k_1, \ldots, k_l)$ **over** $\mathcal{D}$
- **Result:** $\forall\, t \in E_1 \times \cdots E_m$: Rank all t by computing $Score(q(t))$ as follows.

$$\textbf{(1)} \quad Score(q(t)) = \boxed{p_o \cdot \log \frac{p_o}{p_r}}, \textbf{ where}$$

$$\textbf{(2)} \quad p_o \equiv p(q(t)|D) = \boxed{\sum_{d \in D} \mathbf{PR}[d]} \times \boxed{\max_{\gamma}\Big(\prod_{e_i \in \gamma} e_i.conf \times \alpha_B(\gamma) \times p(s|\gamma)\Big)}$$

$$\textbf{(3)} \quad p_r \equiv p(q(t)|D') = \prod_{j=1}^{m}\Big(\sum_{e_j \in d, d \in D} p(d)\Big) \times \prod_{i=1}^{l}\Big(\sum_{k_i \in d, d \in D} p(d)\Big) \times \prod_{j=1}^{m} \overline{e_j.conf} \times \frac{\sum_s p(q(t)|s)}{|s|}$$

Validation

Global Aggregation

Local Recognition

Entity Rank – Algorithm

The EntityRank Algorithm: *Actual Execution of Entity Search.*

Given: $L(E_i), L(k_j)$: Ordered lists for all the entity and keywords.

Input: $q = \alpha(E_1, \ldots, E_m, k_1, \ldots, k_l)$.

0: Load inverted lists: $L(E_1), \ldots, L(E_m), L(k_1), \ldots, L(k_l)$;

/* *intersecting lists by document number*

1: **For** each doc d in the intersection of all lists

2: Use pattern α to instantiate tuples; /* *matching*

3: **For** each instantiated tuple t in document d

4: **Calculate** $p(q(t)|d)$; /* *Section 4.2*

5: **For** each instantiated tuple t in the whole process

6: **calculate** $p(q(t)|D) = \sum_d p(q(t)|d)p(d)$; /* *observed probability*

7: **output** $Score(\mathrm{q(t)}) = p(q(t)|D) \log \frac{p(q(t)|D)}{p(q(t)|D')}$; /* *Section 4.3*

Experimental Setup

- **Corpus:** General crawl of the Web(Aug, 2006), around 2TB with 93M pages.
- **Entities:** Phone (8.8M distinctive instances) Email (4.6M distinctive instances)
- **System:** A cluster of 34 machines

Comparing Entity Rank with Various Approaches

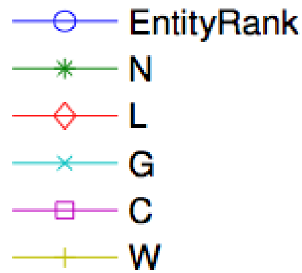
	Contextual	Uncertain	Holistic	Discriminative	Associative
Naïve			✓		
Local	✓	✓			
Global			✓	✓	
Combine	✓	✓	✓		
Without	✓	✓	✓	✓	
EntityRank	✓	✓	✓	✓	✓

Example Query Results

Query	Telephone	ER	L	N	G	C	W
Citibank Customer Service	800-967-2400	**1**	4	7	43	**1**	**1**
New York DMV	800-342-5368	**2**	**2**	213	882	5	3
Amazon Customer Service	800-201-7575	**1**	**1**	52	83	**1**	**1**
Ebay Customer Service	888-749-3229	**1**	7	859	118	2	13
Thinkpad Customer Service	877-338-4465	5	12	249	127	19	**4**
Illinois IRS	800-829-3676	**1**	**1**	157	697	3	2
Barnes & Noble Customer Service	800-422-7717	**1**	2	2158	1141	7	**1**

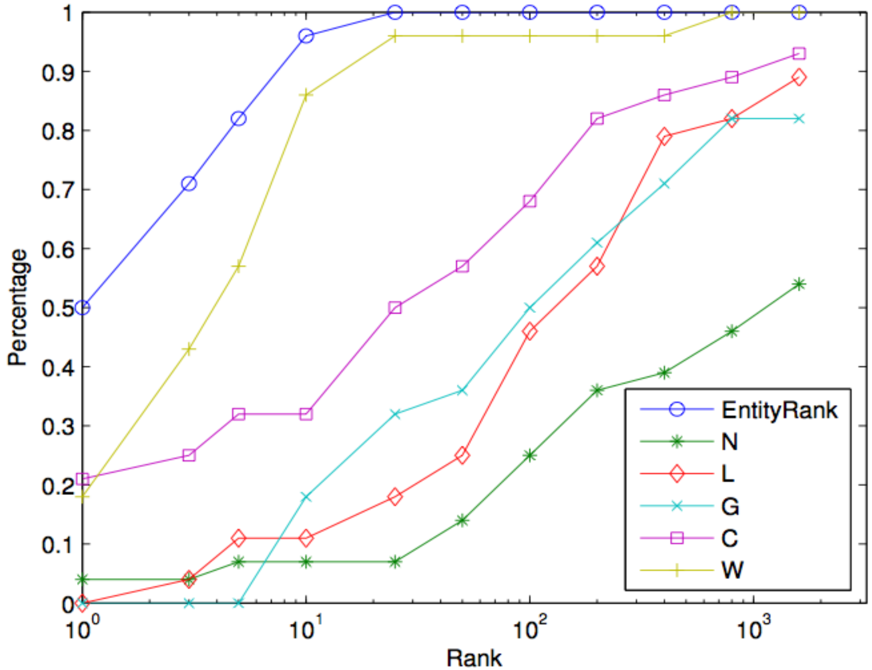
Query	Email	ER	L	N	G	C	W
Bill Gates	bgates@microsoft.com	**4**	44	2502	376	21	23
Oprah Winfrey	oprah@aol.com	**2**	6	745	80	4	3
Elvis Presley	elvis@icomm.com	**5**	56	1106	267	20	8
Larry Page	larrypage@google.com	**8**	24	9968	26932	12	11
Arnold Schwarzenegger	governor@governor.ca.gov	**4**	45	165	169	5	6

Comparison – Query Results

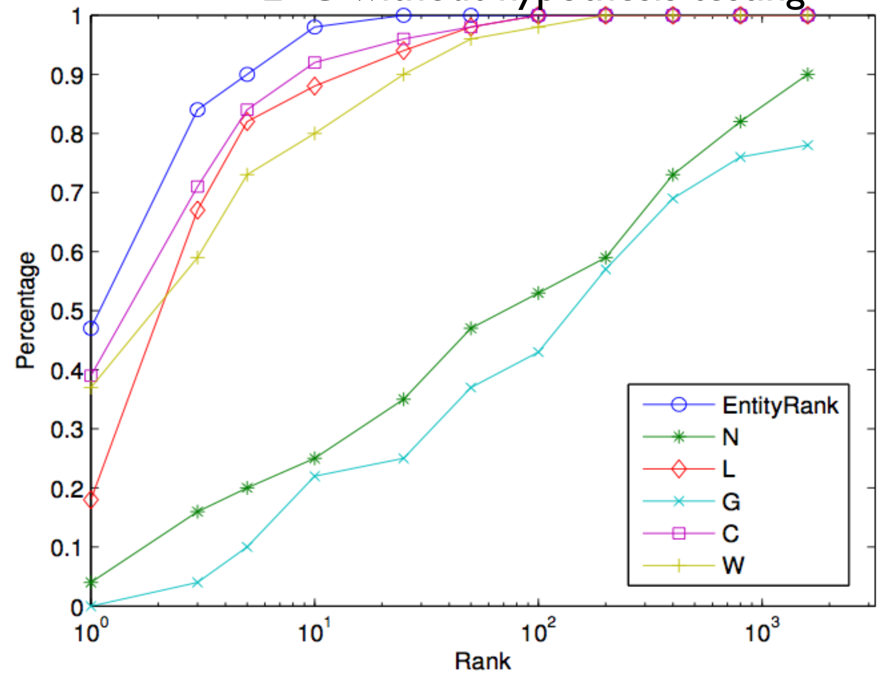


EntityRank
Naïve approch
Local only
Global only
Combine L by simple summation
L+G without hypothesis testing

%Satisfied Queries at #Rank



Query Type I:
Phone for Top-30 Fortune500 Companies



Query Type II:
Email for 51 of 88 SIGMOD07 PC

Conclusions

- Formulate the entity search problem
- Study and define the characteristics of entity search
- Conceptual Impression Model and concrete EntityRank framework for ranking entities
- An online prototype with real Web corpus

Questions???

Thank You!!!!