Cost-Driven Exploration of Faceted Query Results

Abhijith Kashyap
Vagelis Hristidis
Michalis Petropoulos

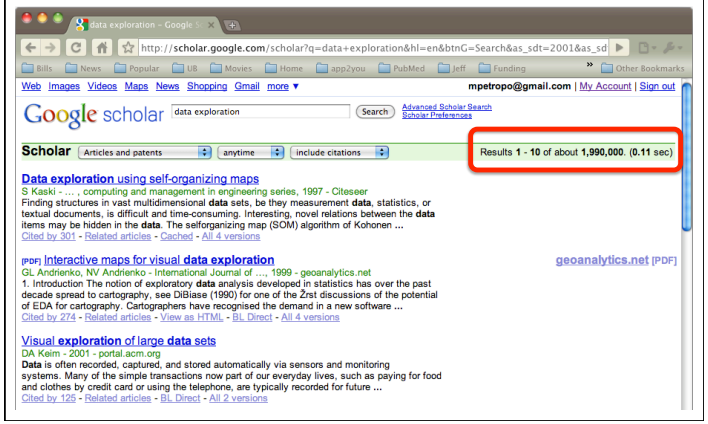
Data Exploration

- Large and complex datasets are commonplace nowadays.
  - Product Catalogs (Amazon, eBay, …)
  - Publications (Google Scholar, CiteSeer, DBLP, …)
  - Gene/Protein Databases (PubMed)
- *Exposing* such datasets to users is *challenging*
  - Users have a hard time querying these datasets and understanding the results.
  - Research in Data Exploration aims at *easing this pain.*

Data Exploration Tasks

- In the simplest case, a user:
  - Issues a Query
    - Query Formulation
      (Data Exploration *buzzwords in blue*)
  - Browse the returned results.
    - Result Navigation

Example: Google Scholar



Data Exploration Challenges

- Users have difficulty in formulating queries
  - Unfamiliarity with underlying data or its structure.
  - Many queries are *underspecified*
- Information Overload: Most queries issued against such datasets return a large number of results.
  - Users have trouble navigating large resultsets looking for results that satisfy their information need

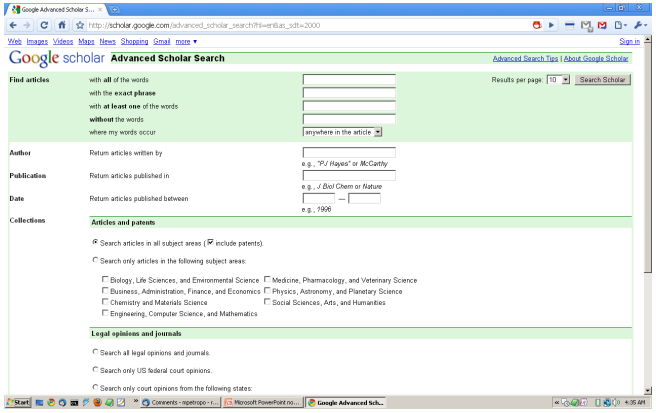
Approaches to Simplify Data Exploration

- Query Formulation
  - Simple Keyword based interface (a la Google)
    - Limited expressivity.
  - Advanced Search form
    - Difficult to Build and difficult to use.
  - Query Autocomplete
    - Works for a small number of keywords.

Some recent representative works:

- **DISCOVER: Keyword Search in Relational Databases.** Vagelis Hristidis, Yannis Papakonstantinou. VLDB 2002
- **Automated Creation of a Forms-based Database Query Interface** Magesh Jayapandian and H. V. Jagadish *VLDB* Auckland, New Zealand VLDB 2008
- **Combining Keyword Search and Forms for Ad Hoc Querying of Databases**. Eric Chu, Akanksha Baid, Xiaoyong Chai, AnHai Doan and Jeffrey Naughton. **SIGMOD 2009.**
- **Type Less, Find More: Fast Autocompletion Search With a Succinct Index.** H Bast, I Weber. SIGIR 06.

Advanced Search Google Scholar



Approaches to Simplify Data Exploration (cont'd)

- Results Navigation
  - Results Ranking
  - Results Categorization

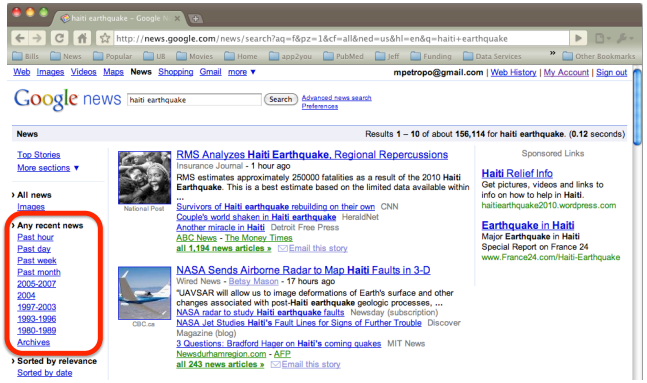
Results Ranking

- Results Ranking
  - Present an ordered list of results, ordered by a predefined Ranking function
    - PageRank (Brin et. al.)
    - ObjectRank (Hristidis et. al)
    - Many others (see works by V. Hristidis, S. Chaudhuri)
  - Problems:
    - Difficult to explain and customize
      - Ranking is not aligned with user preference
    - Problem becomes harder with structured data

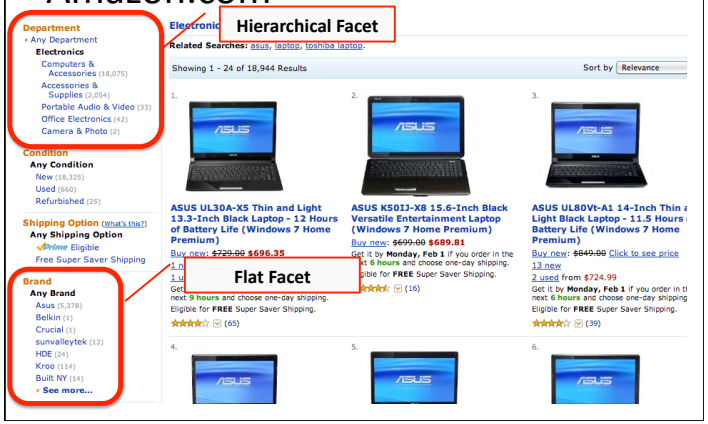
Results Categorization

- Organizes the results into categories
  - A category can be either:
    - A flat list of terms
    - Organized in an ontology or a concept hierarchy
  - Typically more than one
  - Used in conjunction with ranking
- Each categorization of the results is often referred to as a facet
  - Focus of this work (and presentation)

Example: Unstructured Data Google news



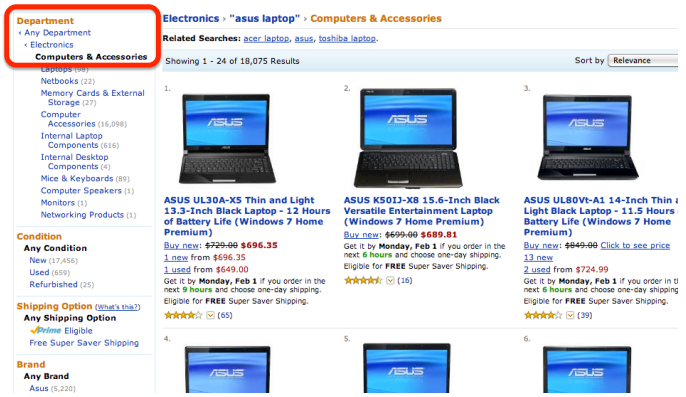
Example: Structured Data Amazon.com



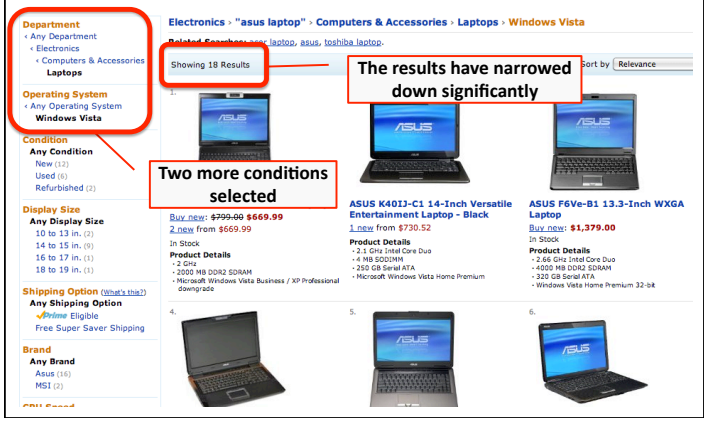
Results Categorization (cont'd)

- Categorization reduces the user effort during Results Navigation.
- Users navigate the results by selecting conditions from one or more facets.
- Each selection narrows down the results. (or *refines* the query)
- The user refines the results until she narrows down to the subset of results that satisfies her information need.

Example: Amazon.com (cont'd)



Example: Amazon.com (cont'd)



Faceted Navigation

- The user navigates the facet classification instead of the results.
- This classification is typically smaller and better organized than the resultset
- Problems:
  - The facet classification is not *small* enough
    - The set of all available choices can easily overwhelm the user.
    - Amazon was a very simple example
      - Try navigating DBLP or Genome databases.

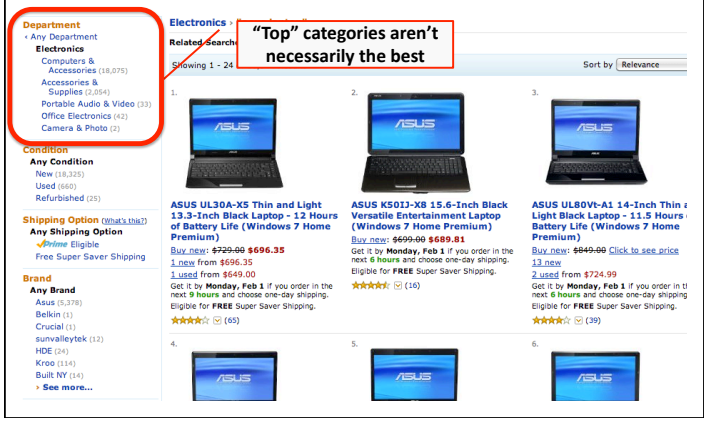
Hidden Slide

- You might need a more tedious example than Amazon here

Managing Faceted Navigation

- How should the facets and facet conditions be presented to the user?
- Solution: Show only a small subset of facets and facet conditions
  - *Almost all* interfaces select facets and conditions based on cardinality (number of results).
  - Can result in *sub-optimal* navigation!
  - Remember: The objective is to decrease user effort.

Example: Amazon.com (cont'd)



Managing Faceted Navigation: Our Approach

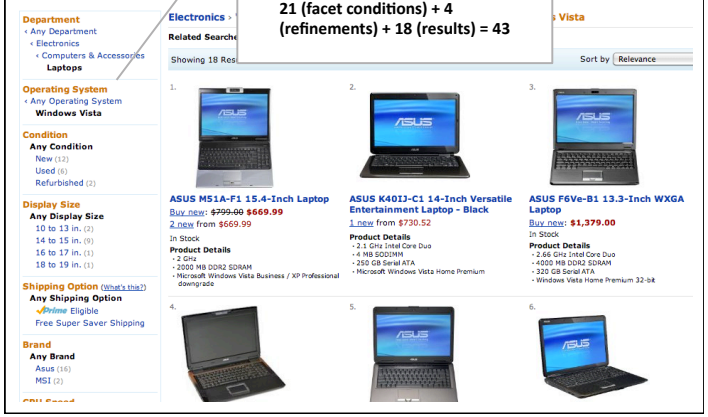
- Idea:
  - The objective is to decrease user effort.
  - So, select the set of facet conditions that minimize the user effort and show them to the user.
- Problems:
  - How to measure user effort?
  - Even if we could, how do me measure it even before the user begins the navigation?

Measuring User Effort

- A user navigating the results spends time and effort in:
  - Reading the labels of facet conditions
  - Deciding and clicking on the selecting the facet condition
  - Reading the resultset.
- Each of the above action contributes to navigation effort or navigation cost

Example: A ... d)

Example:
The total cost of navigation in the previous example of "asus laptop" is:
21 (facet conditions) + 4 (refinements) + 18 (results) = 43



Decreasing Navigation Effort

- In the above navigation, the user went through:
  Electronics >> Computers… >> Laptops >> Windows Vista
- **Instead**, if the navigation had
  - landed directly in laptops, the cost would be:
    6 (operating systems) + 1 (refine) + 18 (results) = 25!
  - Could have been even less if fewer choices for operating systems were shown.
  - Gets better with more facets and more complex datasets.

Cost-Based Approach Decreasing Navigation Effort

- We claim:
  A decreased navigation cost translates to:
  - Fewer navigation actions
    - User reach the results they are interested in quickly
  - Decreased navigation time
  - Better user experience
- And, experiments support the claim.

Example
Result Sets and Facets

R_Q

	Make	Year	State	Color
t_1	Honda	2001	NY	Red
t_2	Honda	2005	NY	Green
t_3	Honda	2001	NY	Gold
t_4	Honda	2005	NY	Green
t_5	Toyota	2005	NY	White
t_6	Toyota	2005	NY	Black

$C(R_Q)$

Facet Condition
Make=Honda
Make=Toyota
Color=Red
Color=Gold
Color=Green
Color=White
Color=Black
State=NY
Year=2001
Year=2005

Cost-Based Navigation

- Problem Statement: FACET-SELECTION
- Given
  - R_Q: result of a query Q
  - $C(R_Q)$: the facet classification of R_Q
  - Select a subset $C_S(R_Q)$ of $C(R_Q)$, such that the *overall* navigation cost is minimized.
  - Constraint: $C(R_Q)$ should *cover* R_Q i.e. the set of conditions should not hide away any result.

Example

R_Q

	Make	Year	State	Color
t_1	Honda	2001	NY	Red
t_2	Honda	2005	NY	Green
t_3	Honda	2001	NY	Gold
t_4	Honda	2005	NY	Green
t_5	Toyota	2005	NY	White
t_6	Toyota	2005	NY	Black

$C(R_Q)$

Facet Condition
Make=Honda
Make=Toyota
Color=Red
Color=Gold
Color=Green
Color=White
Color=Black
State=NY
Year=2001
Year=2005

$C_S(R_Q)$

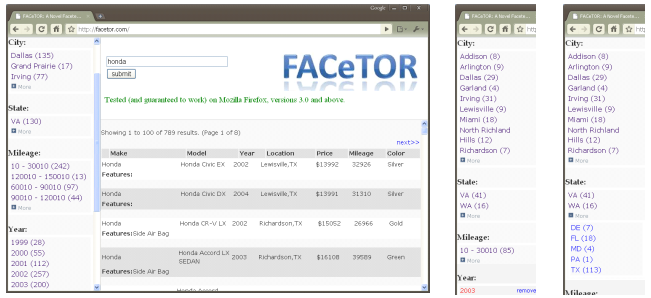
(a)
- **Color**
  - Red (1)
  - White (1)
  - Green (2)
  - Gold (1)
  - Black (1)

(b)
- **Make**
  - Honda (4)
- **Year**
  - 2005 (4)

(c)
- **Make**
  - Toyota (2)
- **Year**
  - 2001 (2)
- **Color**
  - Green (2)

Computing Navigation Cost

- Remember: Navigation cost is the function of user actions navigating the result set R_Q.
  - i.e., cost depends on sequence of user interaction with the interface.
- Therefore, to compute navigation cost, we need a model of user navigation that:
  - Follows the actions of user in the interface.
  - Captures the navigation cost



(a) Initial Query Results and Suggested Facet Conditions (b) REFINE Action (c) EXPAND Action

Navigation Actions

- **SHOWRESULT (R_Q) :-** The user reads through all the results in R_Q
- **REFINE(Q, c) :-** The user chooses a suggested condition $c \in C_S(R_Q)$ and refines query Q, that is, Q becomes $Q \wedge c$.
- **EXPAND (A_i, R_Q) :-** The user is dissatisfied with (rejects) *all* suggested conditions in $C_S(R_Q)$.
  - Instead, he selects an attribute A_i and selects one of the non-suggested condition $c' \in C_S(R_Q) \backslash C(A_i)$

Navigation Model

Formally:

NAVIGATE(Q)
1 Choose one of the following:
2 SHOWRESULT(R_Q)
3 Examine all suggested conditions $C_S(R_Q)$
4 Choose one of the following:
5 REFINE(Q, c)
6 $Q = Q \wedge c$
7 EXPAND(A_i, R_Q)
8 Examine all remaining conditions in $C(A_i) \backslash C_S(R_Q)$
9 Choose a condition $c' \in (C(A_i) \backslash C_S(R_Q))$
10 $Q \leftarrow Q \wedge c'$
11 NAVIGATE(Q)

Cost Model

- Given R_Q , $C_S(R_Q)$ and the set of user actions performed by the user, the total navigation cost can be (approximately) computed.

Cost Model Example

R_Q

	Make	Year	State	Color
t_1	Honda	2001	NY	Red
t_2	Honda	2005	NY	Green
t_3	Honda	2001	NY	Gold
t_4	Honda	2005	NY	Green
t_5	Toyota	2005	NY	White
t_6	Toyota	2005	NY	Black

$C_S(R_Q)$

Color
- Red (1)
- White (1)
- Green (2)
- Gold (1)
- Black (1)

- If the user clicks on Green and then sees the two results, the total cost is:

$|C_S(R_Q)|=5$
+ 1 (REFINE(Q,color=Green))
+ 2 (SHOWRESULTS)
= 5 + 1 +2 = 8

Cost Model (cont'd)

- **But,** we have to suggest facet conditions **before** the navigation begins!
  - These conditions should ***potentially*** minimize the navigation cost.
- We (have to) do the next best thing:
  - Estimate it!

Cost Model
Navigation Cost Estimation

- At each step, the user has several choices:
  - REFINE, EXPAND or SHOWRESULTS.
- Since the actions of the user are not known in advance…
- …we associate uncertainty measures with each of these actions.
  - These uncertainty measures also capture user preference

Cost Model
Modeling Uncertainty

- **SHOWRESULT Probability $P_{SR}(R_Q)$:** This is the probability that the user examines all tuples in the result set and thus terminates the navigation.
- **REFINE Probability P(c):** This is the probability that the user refines the query Q by a suggested condition $c \in C_S(R_Q)$.
- **EXPAND Probability $P_E(R_Q)$:** The probability that the user does not choose a suggested condition and instead performs an EXPAND action

Cost Model
Computing Estimated Cost

The total *expected* cost of navigation is given by:

$$cost(Q) = P_{SR}(R_Q) \cdot |R_Q| + \left(1 - P_{SR}(R_Q)\right) \cdot \left[B + |C_S(R_Q)| + (1 - P_E(R_Q)) \cdot refine\left(Q, C_S(R_Q)\right) + P_E(R_Q) \cdot \sum_{A_i \in S_R} P_A(A_i) \cdot \left(|C(A_i) \setminus C_S(R_Q)| + refine\left(Q, C(A_i) \setminus C_S(R_Q)\right) \right) \right] \quad (1)$$

where

$$refine(Q, C) = \sum_{c \in C} \left(P_{norm}(c) \cdot cost(Q \wedge c) \right) \quad (2)$$

Cost Model
Cost Formula Explanations

NAVIGATE(Q)
1 Choose one of the following:
2 SHOWRESULT(R_Q)
3 Examine all suggested conditions $C_S(R_Q)$

- If the user selects SHOWRESULT, the *expected navigation cost is* $P_{SR}(R_Q)|R_Q|$.
- Else, the user chooses to continue faceted navigation with *expected* cost
  - $(1\text{-}P_{SR}(R_Q)) \times$ (amortized cost of Facet Navigation)

Cost Model
Cost Formula Explained

3 Examine all suggested conditions $C_S(R_Q)$
4 Choose one of the following:
5 REFINE(Q, c)
6 $Q = Q \wedge c$
7 EXPAND(A_i, R_Q)

- The user first examines all suggested conditions $C_S(R_Q)$ with cost $|C_S(R_Q)|$
- Then the user either:
  - Refines with est. cost: $(1\text{-}P_E(R_Q)) \times$ refine(Q, $C_S(R_Q)$)
  - Expand with est. cost: $P_E(R_Q) \times$ (amortized cost of Expand)

Computing Suggested Conditions Algorithms

- Naïve Algorithm
  - At each navigation step, for each set of candidate facet conditions
  - Compute the cost formula in Equation 1 seen previously
  - Return the set of conditions that have the minimum navigation cost
- Problem: Exponential! $O(2^{2n})$.

Complexity Results

- FACET-SELECTION problem is NP-Hard
- We prove hardness for a simpler version of the problem:
  - **NAVIGATE-SINGLE:** Given R_Q and $C(R_Q)$, where all attributes of R_Q are boolean (0,1). WLOG assume that all conditions in $C_S(R_Q)$ are positive.
  - The problem is to then select the set of conditions that cover the result R_Q and is smallest in size.
  - **Theorem:** NAVIGATE-SINGLE is NP-COMPLETE
  - Reduction from HITTING-SET Problem

Computing Suggested Conditions Heuristics

- To efficiently select the best set of suggested conditions, we propose 2 heuristics.
  1. **ApproximateSetCover** Heuristic
  2. **UniformSuggestions** Heuristic

Computing Suggested Conditions ApproximateSetCover Heuristic

- Given a resultset R_Q and its facet classification $C(R_Q)$, The objective is to compute the set of suggested conditions $C_S(R_Q)$, such that:
  - $C_S(R_Q)$ covers R_Q
  - Expected navigation cost is minimal.
- This problem closely resembles the well-known NP-hard weighted set cover problem!

Computing Suggested Conditions ApproximateSetCover Heuristic

- **Weighted set cover problem** – given a set system (U, S), such that , $\bigcup_{s \in S} s = U$ and weights w:S->R^+, find a subfamily F such that $\bigcup_{s \in F} s = U$ and $\sum_{s \in F} w(s)$ is minimal.
- There is a linear time approximation algorithm for the **weighted set cover problem**
  - V. Chvatal: **A Greedy Heuristic for the Set Cover Problem**. Mathematics of Operations Research 4(3): 233-235 (1979)

Computing Suggested Conditions

ApproximateSetCover Heuristic

- To the approximation algorithm, we need to define the weight function:

$$w: c \in C(R_Q) \rightarrow 1/P(c)$$

- Because, based on the cost formula,
  - The condition chosen should have a high P(c).
  - Otherwise, the probability of the user choosing EXPAND $P_E(R_Q) = \prod_{c \in C_S(R_Q)} (1 - P(c))$, increases significantly.

Computing Suggested Conditions

ApproximateSetCover Heuristic

Algorithm: ApproximateSetCover(Q, R_Q)
Input: A query Q, a result set R_Q
Output: The suggested conditions $C_S(R_Q) \subseteq C(R_Q)$

1 $C_S(R_Q) \leftarrow \emptyset$
2 $V \leftarrow \emptyset$ // V are results covered so far
3 while $V \neq R_Q$ // while not all results covered
4 $c \leftarrow argmax_{c \in C(R_Q)}(P(c) \cdot |R_{Q \wedge c} \backslash V|)$
5 $C_S(R_Q) \leftarrow C_S(R_Q) \cup \{c\}$
6 $V \leftarrow V \cup R_{Q \wedge c}$
7 $Q \leftarrow Q \wedge c$
8 return $C_S(R_Q)$

Computing Suggested Conditions

Example

R_Q

	Make	Year	State	Color
t_1	Honda	2001	NY	Red
t_2	Honda	2005	NY	Green
t_3	Honda	2001	NY	Gold
t_4	Honda	2005	NY	Green
t_5	Toyota	2005	NY	White
t_6	Toyota	2005	NY	Black

$C(R_Q)$

Facet Condition	P(c)
Make=Honda	0.8
Make=Toyota	0.7
Color=Red	0.1
Color=Gold	0.1
Color=Green	0.4
Color=White	0.1
Color=Black	0.1
State=NY	0.2
Year=2001	0.5
Year=2005	0.7

$C_S(R_Q)$

(a) Color • Red (1) • White (1) • Green (2) • Gold (1) • Black (1)

(b) Make • Honda (4) Year • 2005 (4)

(c) Make • Toyota (2) Year • 2001 (2) Color • Green (2)

- In the first iteration, the algorithm selects **Make=Honda**, since this facet condition covers 4 results and has the maximum value of $P(c)|R_{Q \wedge c}|$ amongst all the conditions in $C(R_Q)$
- In the next iteration, two results (t1 & t3) remain uncovered and are covered by facet condition **Year=2005**

Computing Suggested Conditions

UniformSuggestions Heuristic

- In this heuristic, we follow the cost formula more closely.
- Evaluating suggestions using the cost formula is very expensive.
  - Because it involves recursively evaluating the cost equation for each combination of candidate facet conditions ($O(2^{2n})$ in total).

Computing Suggested Conditions

UniformSuggestions Heuristic

- Idea,
  - The combinatorial recursion of the cost formula creates a recursion tree that is both wide and deep!
  - Instead of evaluating this large tree, which considers a set of suggestions,
    - Evaluate a bunch of small trees, one for each candidate.
    - i.e. evaluate each condition independently.

Computing Suggested Conditions

UniformSuggestions Heuristic

- Based on a (long winded) analysis of the cost model [see sections 3.3 & 6.2],
- We came up with a simplified version of the cost formula :-

$$cost(c,|R_Q|) = \begin{cases} |R_Q| & ,|R_Q| < 1 \\ P_{SR}(R_Q)\cdot|R_Q| + \left(1-P_{SR}(R_Q)\right)\cdot \\ \left[B + n + \left(1-P_E(c)\right)\cdot cost\left(c,|R_Q|/n\right) + \right. & ,|R_Q| > 1 \\ \left. P_E(c)\cdot\left(|\mathbb{C}(A_C)| + cost\left(c,|R_Q|/|\mathbb{C}(A_c)|\right)\right)\right] & \end{cases}$$

- That can evaluate each condition independently.

Computing Suggested Conditions

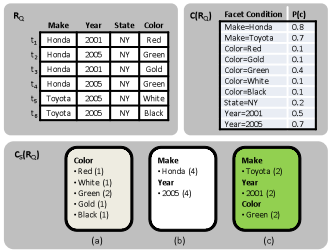
UniformSuggestions Heuristic

Algorithm: UniformSuggestions(Q, R_Q)
Input: A query Q, a result set R_Q
Output: $C_S(R_Q) \subseteq C(R_Q)$, the suggested conditions.

```
1  Q' ← Q; C_S(R_Q) ← ∅; Y ← R_Q          // Y: uncovered results
2  P_SR ← P_SR(R_Q, C(R_Q))
3  while Y ≠ ∅ do
4    foreach c ∈ C(R_Q)
5      n ← |Y|/|Y ∩ R_{Q'∧c}|
6      P_SR ← P_SR(R_Q)
7      u ← |Y|
8      compute cost(c, u) using Equation 5
9    endFor
10   Let cmin be the suggestion with min estCost(c, |Y|)
11   C_S(R_Q) ← C_S(R_Q) ∪ cmin
12   Q' ← Q ∧ cmin
13   Y ← Y \ R_{Q'∧cmin}
14   C(R_Q) ← C(R_Q) \ {cmin}
15 endWhile
16 return C_S(R_Q)
```

UniformSuggestions Heuristic

Example



Filling in the Gaps

Estimating Probabilities

- Estimating prob. of SHOWRESULT: $P_{SR}(R_Q)$
- Intuition,
  - If the results are widely distributed, then the user would probably REFINE the results
  - Else, the user would be better off reading the results
- Therefore, we use the information theoretic measure of entropy to measure $P_{SR}(R_Q)$:

$$Entropy\left(R_Q, C(R_Q)\right) = -\sum_{c \in C(R_Q)} \left(|R_{Q \wedge c}|/N\right) \ln\left(|R_{Q \wedge c}|/N\right)$$

Filling in the Gaps

Estimating Probabilities

- Estimating $P_A(A_i)$ and P(c):
  - These probabilities are subjective measures of user preference
  - Can be estimated in multiple ways
    - Data Frequency, Explicit User Preference, Mining Query Logs, etc.
- In this work, we estimate:
  - $P_A(A_i)$ by query logs
  - P(c): by data frequency

Experiments

- We perform experiments to:
  - Validate the cost model
    - To see if it actually reduces navigation cost.
  - Compare it with existing state-of-the art[1]
  - See what users think about it: **User Study**

[1] S. B. Roy, H. Wang, G. Das, U. Nambiar, M. K. Mohania: **Minimum-Effort Driven Dynamic Faceted Search in Structured Databases**. CIKM 2008

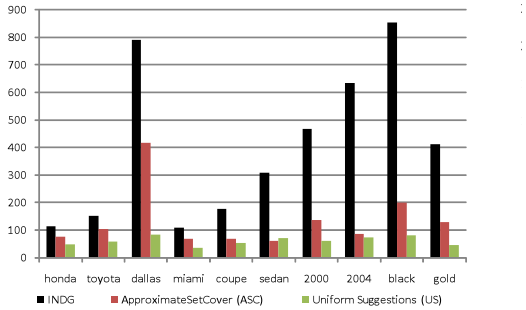
Datasets

- Yahoo! UsedCars
  - 15,191 cars
  - 41 Facets – 7 categorical, 3 numeric and rest Boolean
- IMDB Movies database
  - ~40K movies
  - All Facets categorical
  - Some facets are set valued; each movie has multiple actors etc.

Experiment Methodology

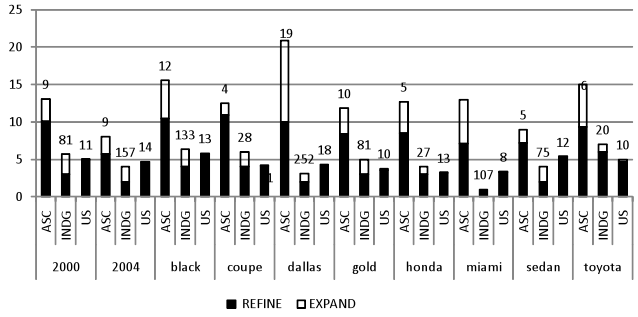
- For each dataset,
  - we select a number of queries – 10 each
  - for each query, we designate a random result as the target of navigation
- We count the number of navigation actions (cost) required to reach the target result
  - In a *guided random simulation*

Results: UsedCars



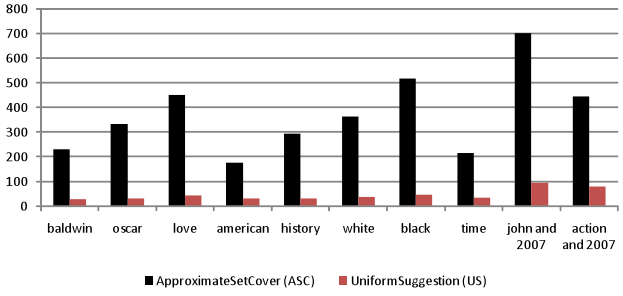
(a)

Results: UsedCars



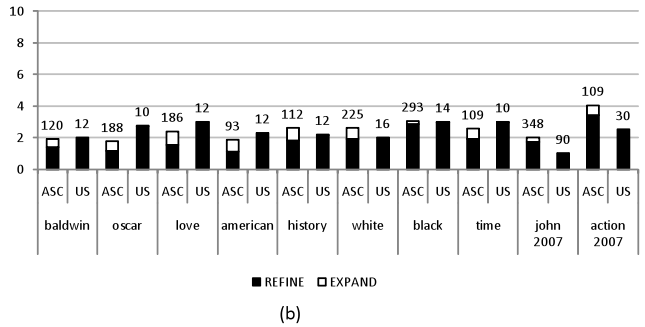
(b)

Results: IMDB dataset



(a)

Results: IMDB dataset



(b)

User Study

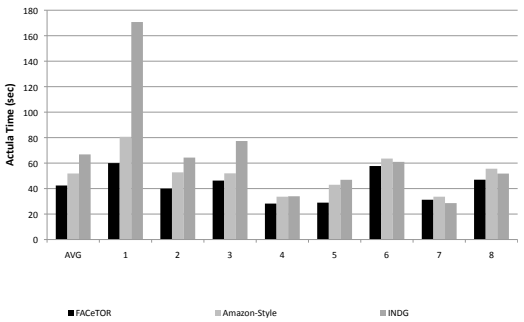
We measure the following:

1. The *actual time* it took users to navigate to designated target tuples using different interfaces
2. How realistic is our cost model, by studying the relationship of the actual time (actual cost)
3. The *users perception* of the faceted interfaces through a questionnaire

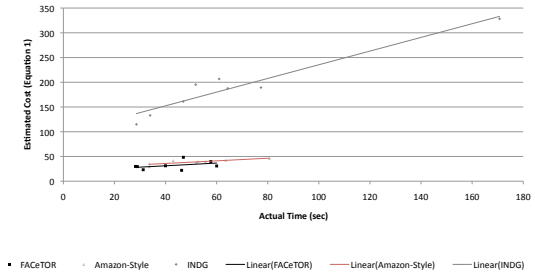
User Study: Comparisons

1. FACeTOR,
2. Amazon-Style, which suggests at most 5 facet conditions with the highest cardinality for each attribute, and
3. One-attribute-at-a-time INDG, where an attribute is selected at each step and all its conditions are displayed

User Study: Results



User Study: Results

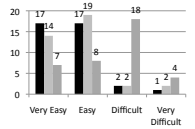


User Study: Survey

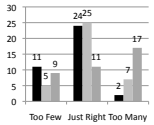
We asked the following questions:

1. **Quality Of Suggestions:** Did the suggestions presented at each navigation step make the task of finding the target car(s) easier?
2. **Quantity of Suggestions:** Were the number of suggestions presented at each step satisfactory?
3. **Difficulty in Selection**: At each navigation step, how difficult or easy was to decide which facet condition to choose?

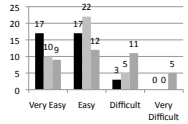
User Study: Survey



(a) Quality of Suggestions



(c) Quantity of Suggestions



(b) Difficulty in Selection

FACeTOR Amazon-Style INDG