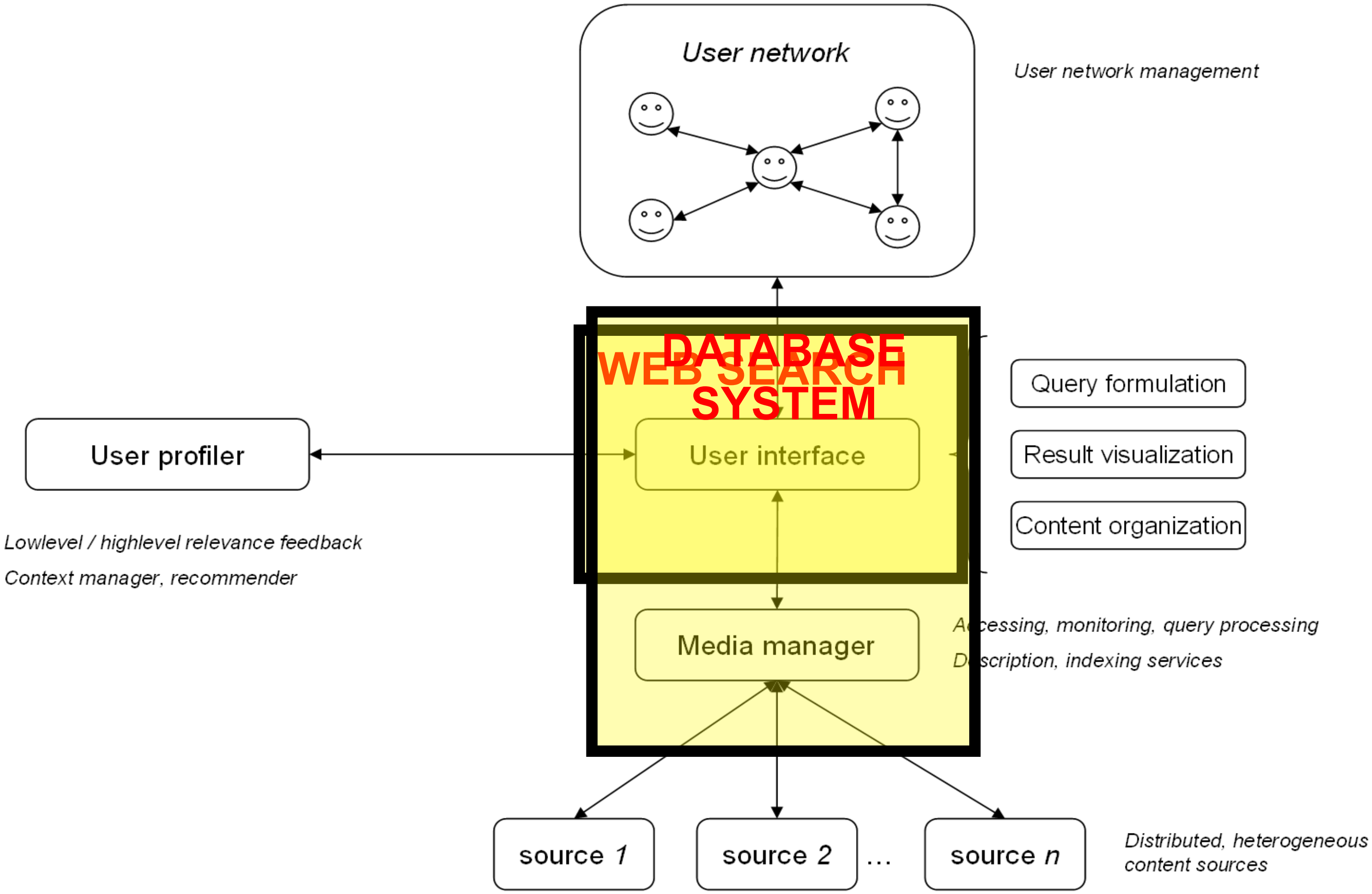
Personalization in web search and data management

Timos Sellis, Research Center "Athena" and National Technical Univ. of Athens

www.imis.athena-innovation.gr

(joint work with T. Dalamagas, G. Giannopoulos, G. Koutrika and A. Arvanitis)

Personalization – A general view

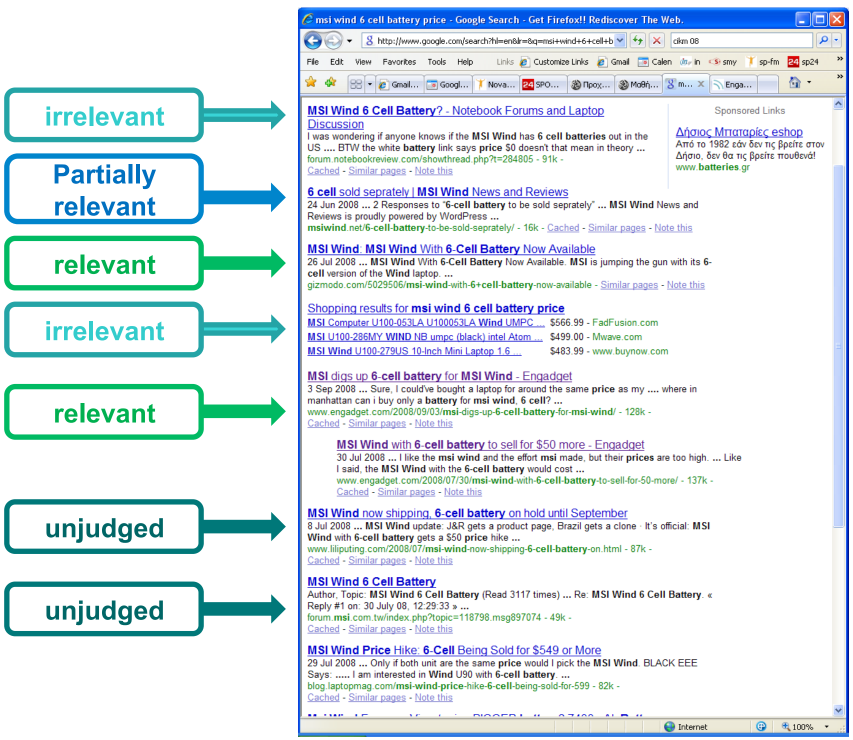


General methodology (1)

- How to personalize search results?

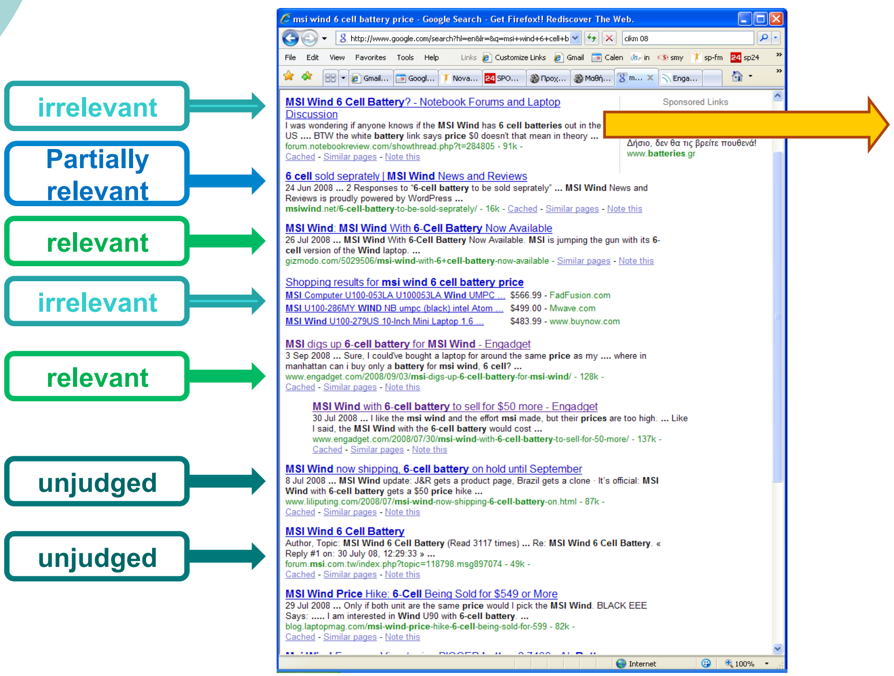
General methodology (2)

- How to personalize search results?
  - **Step 1**. Implicit (from user log clicks) or explicit feedback can record **relevance judgments**, i.e. irrelevant, partially relevant, relevant results



General methodology (3)

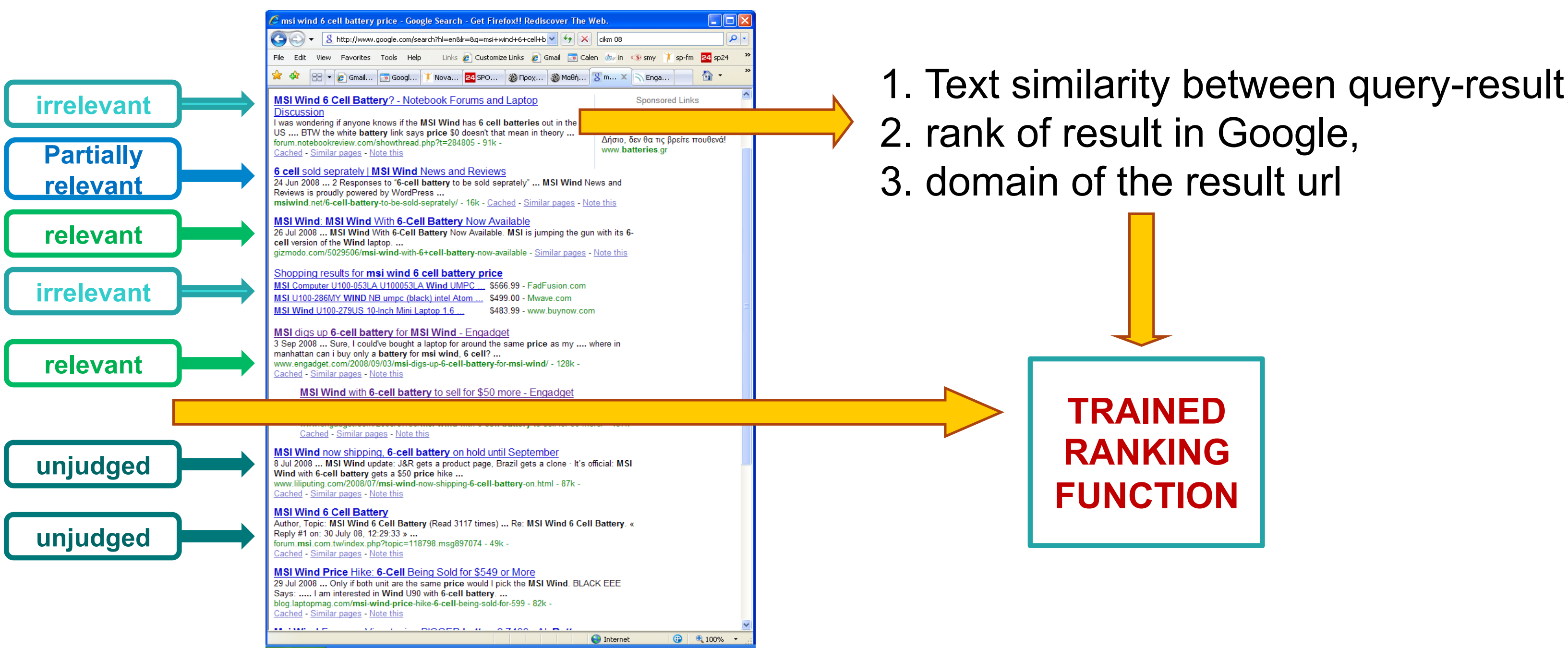
- How to personalize search results?
  - **Step 1**. Implicit (from user log clicks) or explicit feedback can record **relevance judgments**, i.e. irrelevant, partially relevant, relevant results
  - **Step 2**. Extract **features** from query-result pairs.



1. Text similarity between query-result,
2. rank of result in Google,
3. domain of the result url

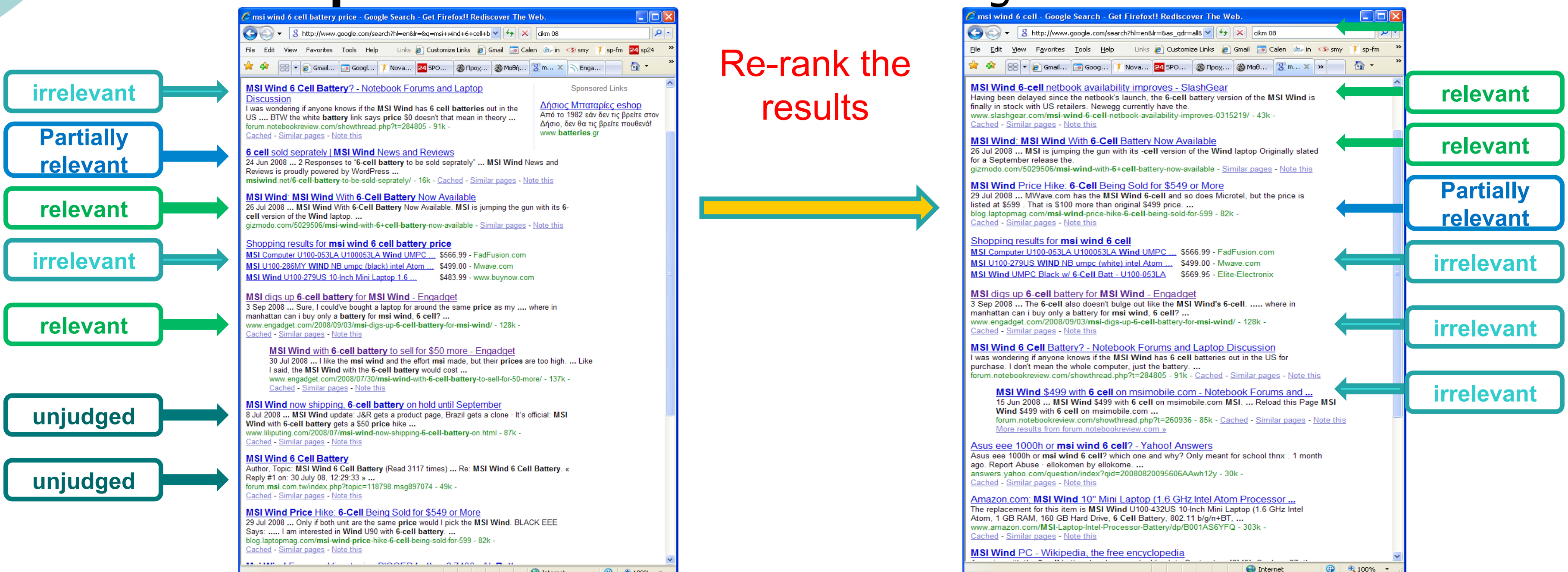
General methodology (4)

- How to personalize search results?
  - **Step 1**. Implicit (from user log clicks) or explicit feedback can record **relevance judgments**, i.e. irrelevant, partially relevant, relevant results
  - **Step 2**. Extract **features** from query-result pairs.
  - **Step 3**. Train a **ranking function** (i.e. Ranking SVM) using judgments and features.



General methodology (5)

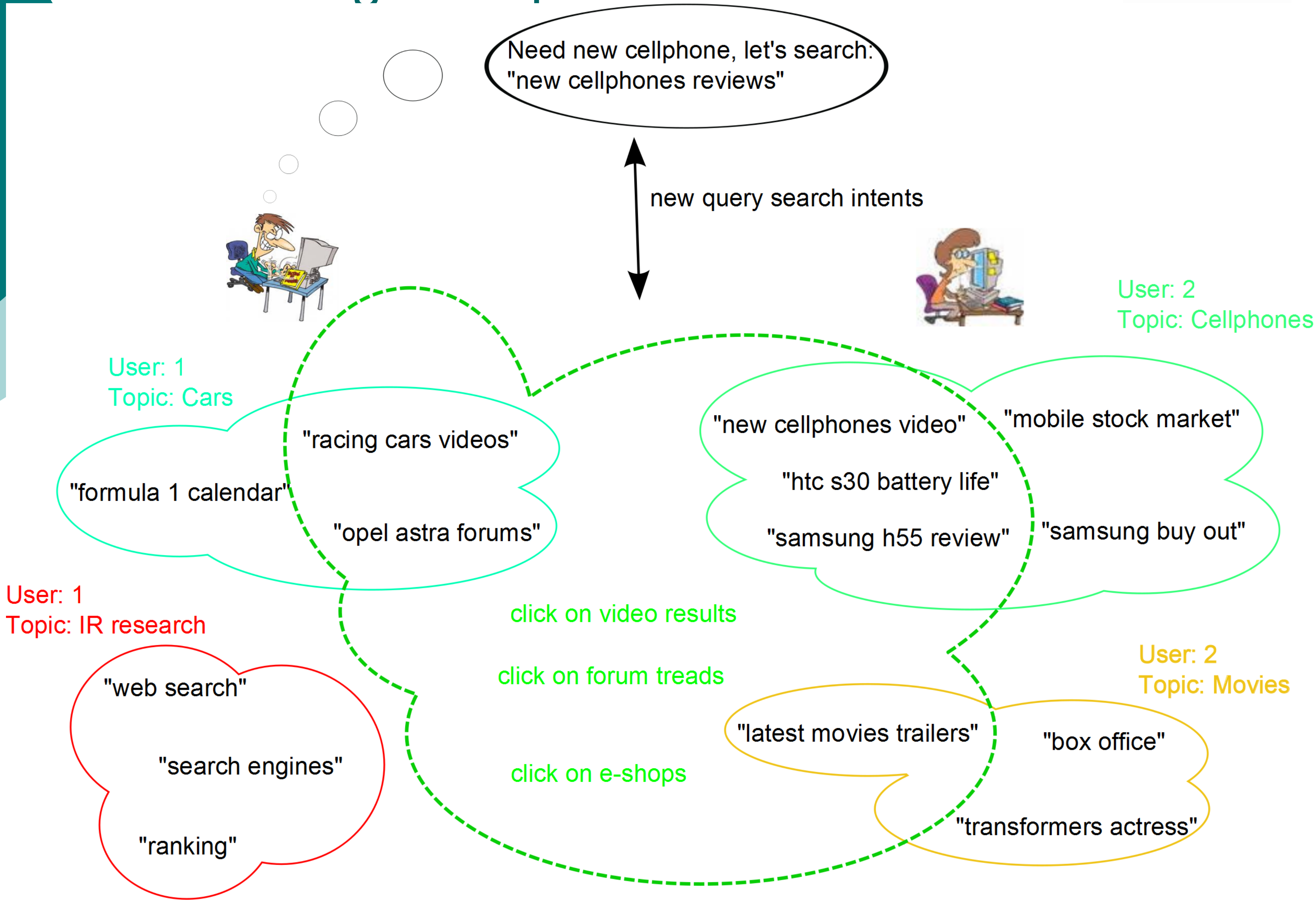
- How to personalize search results?
  - **Step 1**. Implicit (from user log clicks) or explicit feedback can record **relevance judgments**, i.e. irrelevant, partially relevant, relevant results
  - **Step 2**. Extract **features** from query-result pairs.
  - **Step 3**. Train a **ranking function** (i.e. Ranking SVM) using judgments and features.
  - **Step 4**. **Re-rank** the results using trained function



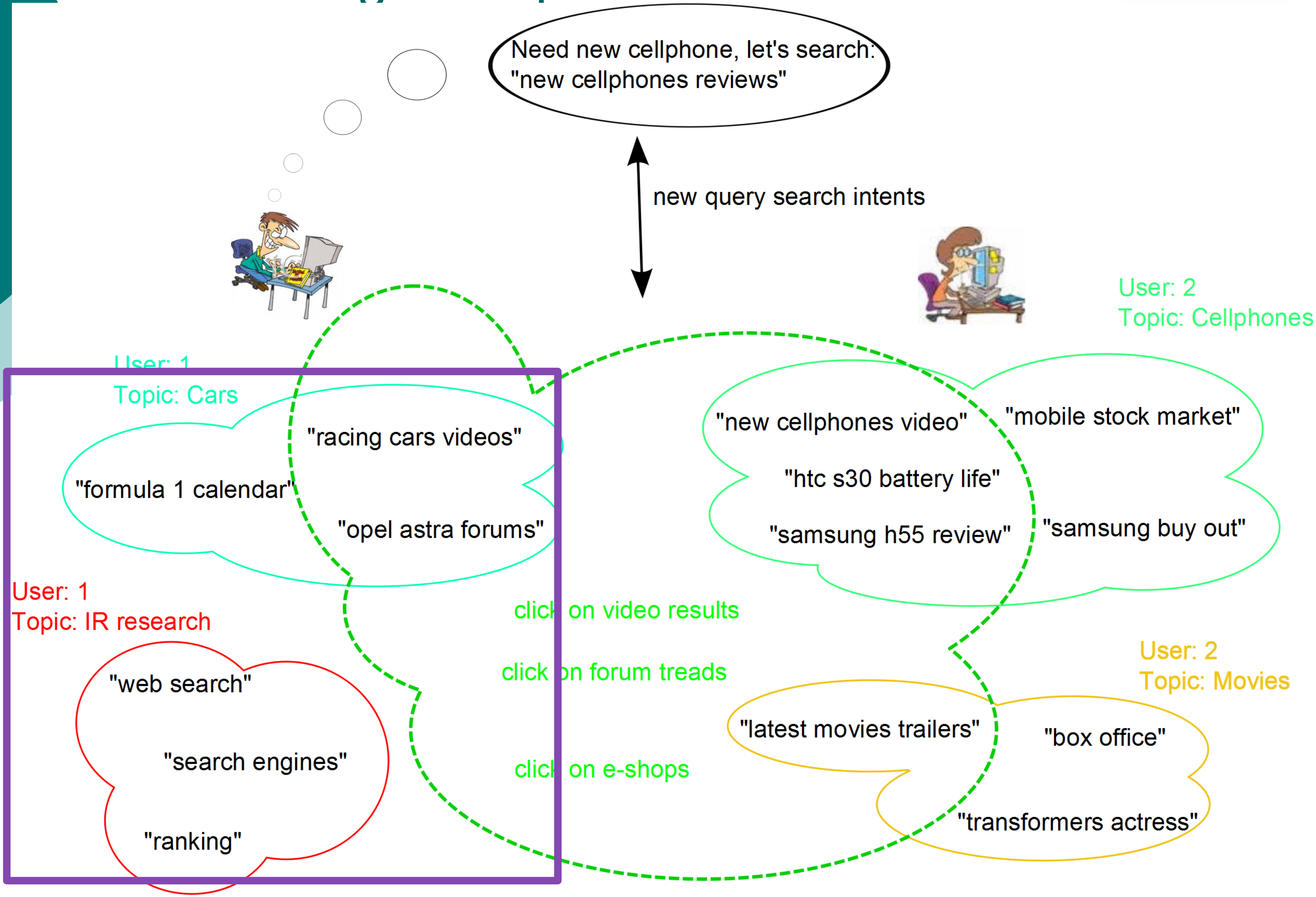
Personalization scenarios

- A lot of work on personalization techniques, however:
  - Main focus on algorithms, and models
    - Not on the items to be searched
  - Personalization is mainly **user-centric** or **content-centric**
    - What about **query/behavior-centric** approaches?
- Re-rank results of users' queries based on:
  - User search history (implicit feedback)
  - User profile (explicit feedback)
  - User/query search intent/behavior
- Examples:
  - In the past, when searching for "java" I clicked on programming related results and not on coffee related results
  - I have stated that I prefer news articles results or results from greek domain
  - My current need, searching for "java", is to read forum discussions and not tutorials

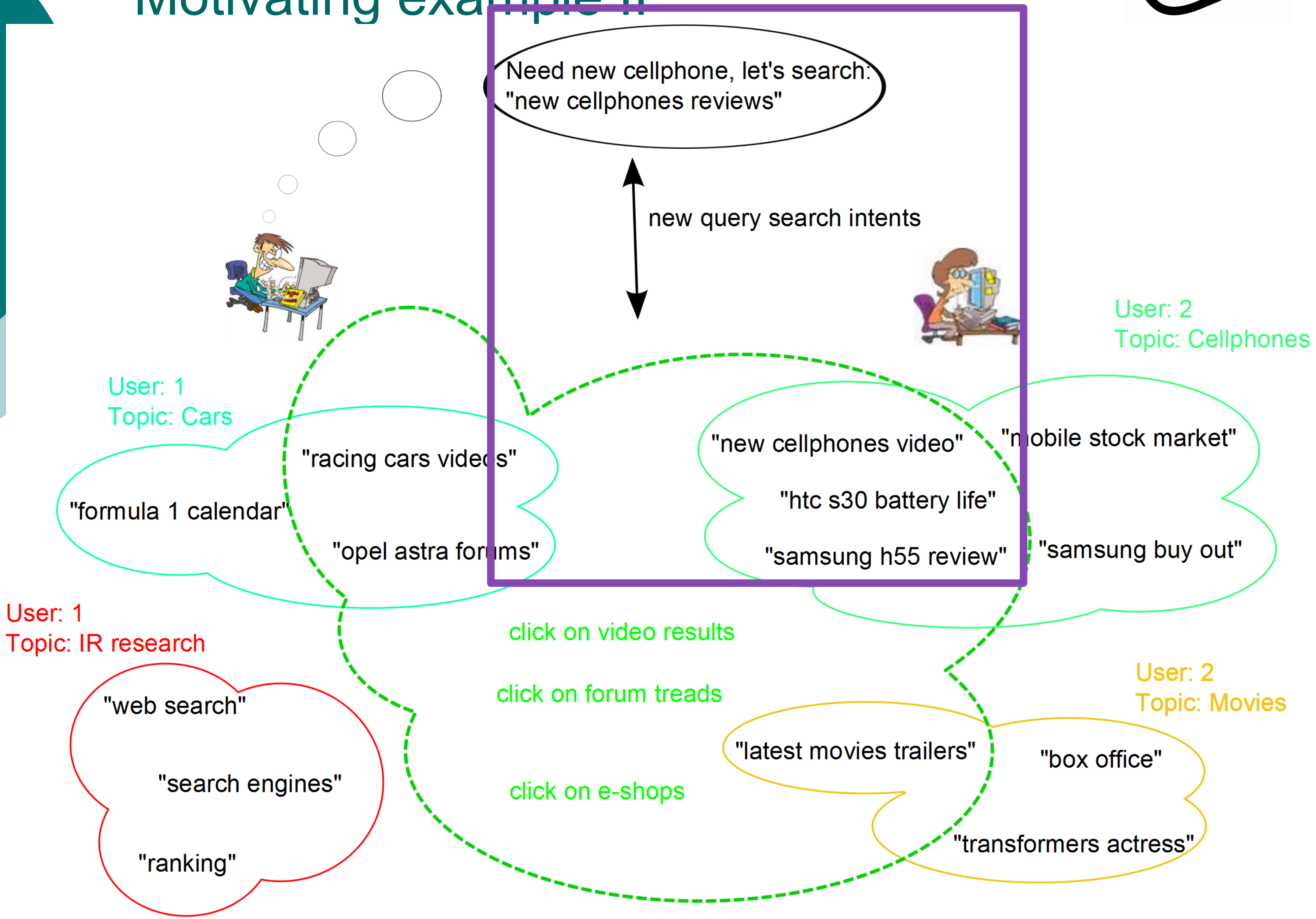
Motivating example



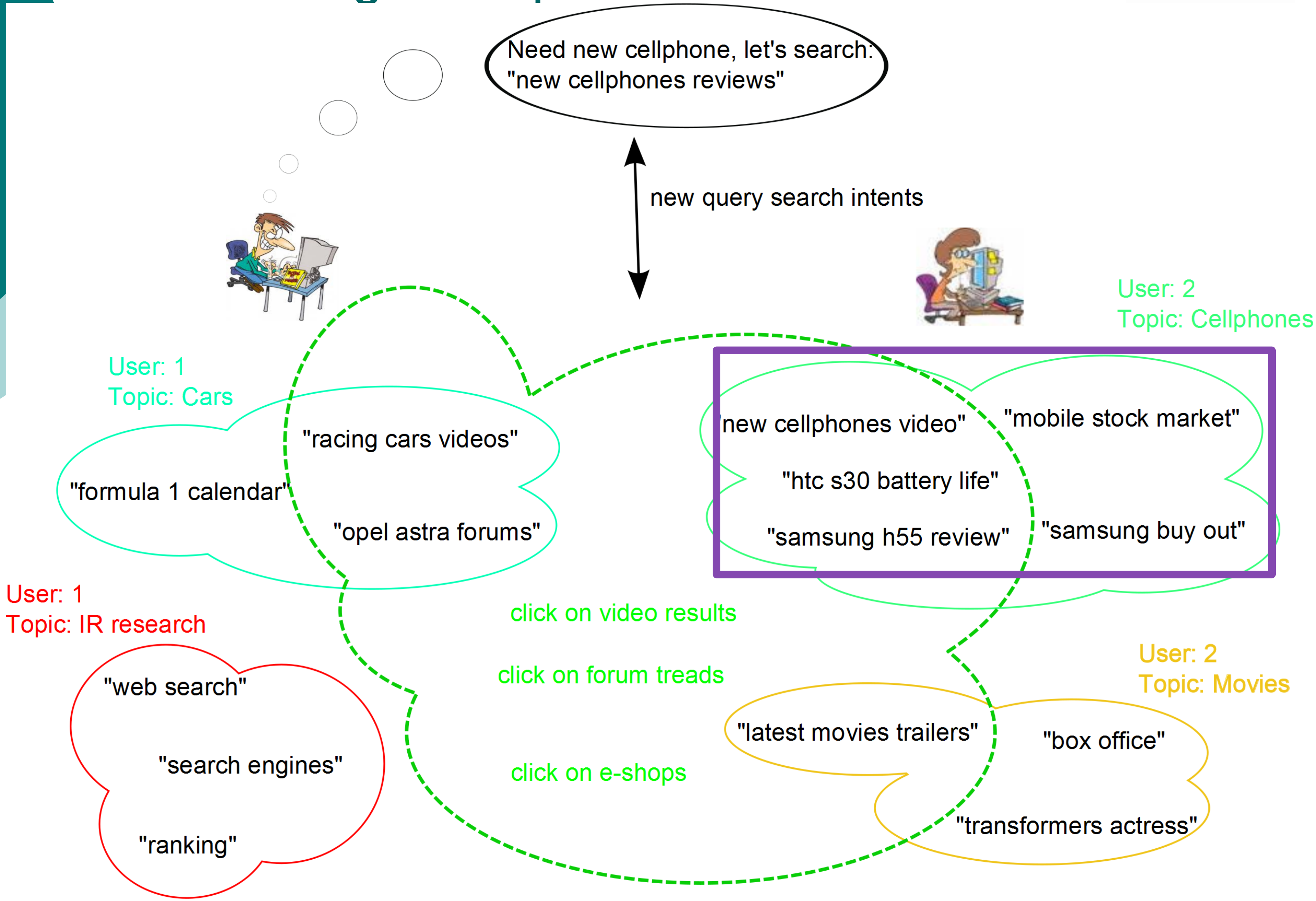
Motivating example I



Motivating example II



Motivating example III



Motivating example III

Collaborative training (Solution)

- Train **multiple ranking functions**
- Each ranking function corresponds:
  - **Not** to a single user
  - **Not** to a group of users
  - **Not** to a topic area
  - But to a **search behavior**:
    - Group of search results with similar characteristics w.r.t. the specific queries posed, i.e. **similar feature vectors**, collected from all users
- When re-ranking search results:
  - Check which search behaviors match with each new query
  - Re-rank the query's results according to the ranking functions trained for those search behaviors

Collaborative training (Search Behavior Capturing)

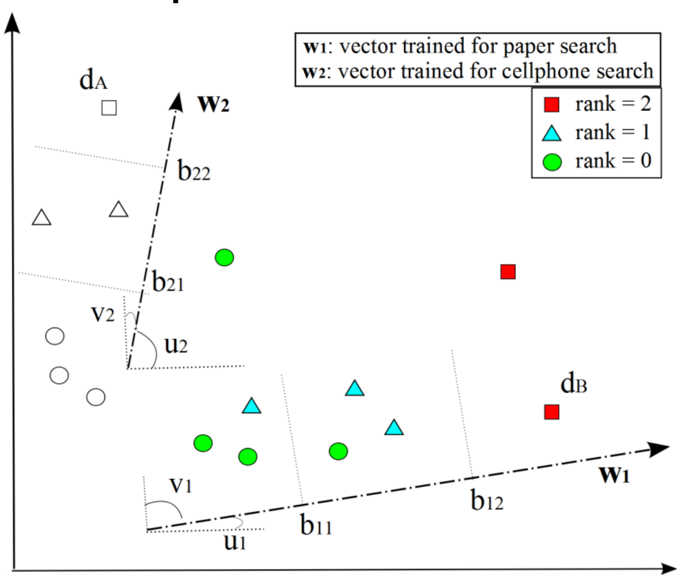
- Analyze search results into feature vectors
  - Represent each query result in the feature space
  - Mark each result's ranking class (relevance judgment)

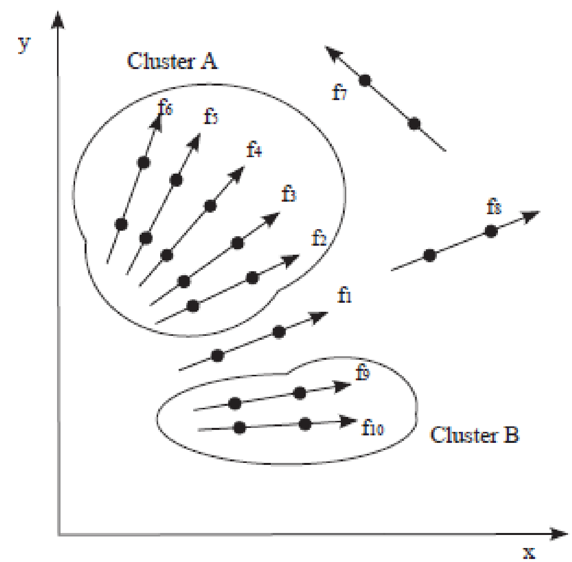
Feature space:

- Textual similarity between query-result
- Rank of result in Google
- Domain of the result url
- Frequent words/urls in the result
- Existence of video, images, etc, in the result
- Category of result site (social, media, market…)
- Result popularity

Collaborative training (Search Behavior Capturing)

- Take into account the geometric characteristics of the baseline ranking model (Ranking SVM)
- Define search behavior in terms of those geometric characteristics
  - **Feature vectors** correspond to specific **user search behaviors**
- **Cluster** feature vectors to find groups of queries that correspond to **similar search behaviors**





Collaborative training (Training and re-ranking)

- Train one ranking function (ranking model) <u>per search behavior cluster</u>
- For each new query:
  - Calculate its textual similarity with each search behavior cluster
  - Re-rank its results using the ranking models trained on the most similar clusters to the query

Evaluation

- Dataset:
  - Yahoo! query log
  - 76037 queries
    - More that 5 results
  - 453 distinct users
    - More than 100 queries
  - Training set
    - 30053 queries (40%)
  - Test set
    - 45984 queries (60%)
  - Clicks -> Relevance judgments
    - 3.2 relevance judgments per query

Evaluation

- Comparison of our method (**Intent**) with baselines:
  - Naïve: training one ranking function for all data (**single**)
  - Ideal: training one ranking function per user (**user**)
  - Competitive: training multiple ranking functions based on content
    - Terms (words) as clustering dimensions (**content-1**)
    - Standard IR features as clustering dimensions (**content-2**)
- Results
  - Mean Average Precision:

Method	MAP	Increase
Single	0.709	-
User	0.806	13.7%
Content-1	0.748	5.5%
Content-2	0.734	3.5%
Intent	0.754	6.3%

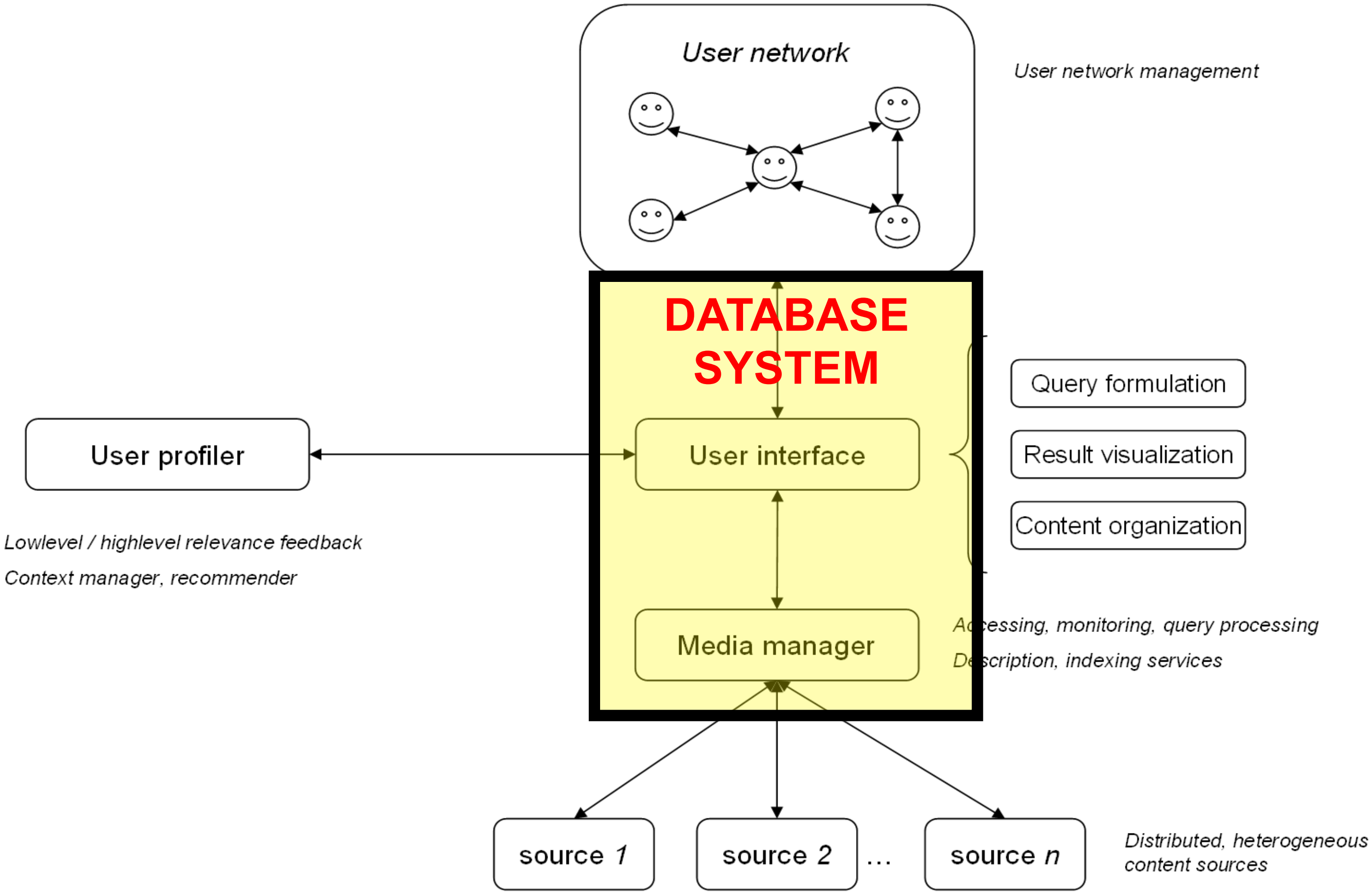
Extensions (1)

- Collaborative training
  - More sophisticated clustering process
  - Enhance cluster-query matching process
  - Combine content-based and search behavior based and user based approaches methods
  - More exteensive experiments
- Apply collaborative training on semantic web data
  - Training, re-ranking, personalization on:
    - RDF
    - Linked data
  - Introduction of structured data-specific:
    - Feature construction
    - Relevance judgments expansion methods

Extensions (2)

- Open issues on search for specific scientific areas
  - How can personalization techniques be adapted when:
    - The searched entities **frequently change names**
    - There are several categories of searched entities (e.g., biological publications, biological entities)

Personalization – DB View



Motivation – Information Overload

- searching for a used car
  - price < $5000 AND model_year > 2007 AND mileage < 50000km

  NO RESULTS!

- what to do next?
  - adjust query constraints
  - iterate until finding satisfying results

  frustrating process!

Motivation – Personalized Needs

- Preferences
  - Alice likes ‘Toyota’ cars
  - Bob would prefer transmission = ‘automatic’
- not a strict requirement, wishes
- if not present in the query both users would get the same results!

Solutions

- use preferences to
  - relax an empty query
    - return cars with mileage<70000 as well
  - filter available choices
    - transmission = ‘automatic’
  - rank results
    - Toyota cars should appear first
- However, no standard solution to manipulate preferences in a db or integrate them in SQL queries

Previous approaches

- Broadly classified into:
  - Plug-in methods
    1. filter query results
    2. evaluate preferences on qualified results
       - top-k, skylines
  - Native methods
    - special operators inside the database core
      - RankSQL, winnow operator
    - FlexPref
      - easier definition of preference strategies by implementing a set of interface functions

Limitations of previous approaches

- Plug-in methods
  - Performance and scalability
    - database used as ‘black-box’
    - only coarse-grained optimizations possible
  - Flexibility
    - how to use preferences to filter, rank etc. is hard-wired in application logic

Limitations of previous approaches

- Native methods
  - only applicable to specific query types
    - RankSQL -> top-k, winnow -> skylines
    - FlexPref -> only pref. strategies that can be defined based on the specified API
    - filtering logic cannot be extended to other preference/query types, such as:
      - conditional preferences
      - at least *m* preferences must be satisfied
      - keep the maximum pref. score for each tuple
  - they require modifications of the database source code

PrefDB: A different approach (ICDE' 12)

- addressing preferences as first-class citizens
  - preference model over relational data
- extend relational data model and algebra with preference processing
- revisit the traditional query paradigm
  - preferences do not disqualify results
  - different query types supported
    - top-k scores, most preferences satisfied…

Models

- Preference model
  - **conditional** part, which tuples are affected
  - **scoring** part, how tuples are scored
  - **confidence** part, indicates preference credibility, importance or relevance
- Extended relational data model
  - p-relations
    - additional score and confidence attributes
    - values assigned after evaluating preferences on database tuples or by joining/aggregating scored tuples

Prefer operator

- $\lambda_p(R)$ evaluates a preference p on R
  - for all tuples satisfying the preference selection condition, λ applies the ranking function

m_id	rating	year	director
m_1	7.8	1996	Allen
m_2	8.3	2004	Eastwood
m_3	8.5	2000	Allen
m_4	6.4	2010	Stone
m_5	7.8	1992	Eastwood

(a) R

- $p_a[R] = (\sigma_{year>2000}, 0.1, 1)$

m_id	score	conf
m_1	$\perp$	0
m_2	0.83	1
m_3	$\perp$	0
m_4	0.64	1
m_5	$\perp$	0

(b) R after evaluating preference p_a

- $p_b[R] = (\sigma_{director=\text{'Eastwood'}}, 0.8, 1)$

m_id	score	conf
m_1	$\perp$	0
m_2	0.815	2
m_3	$\perp$	0
m_4	0.64	1
m_5	0.8	1

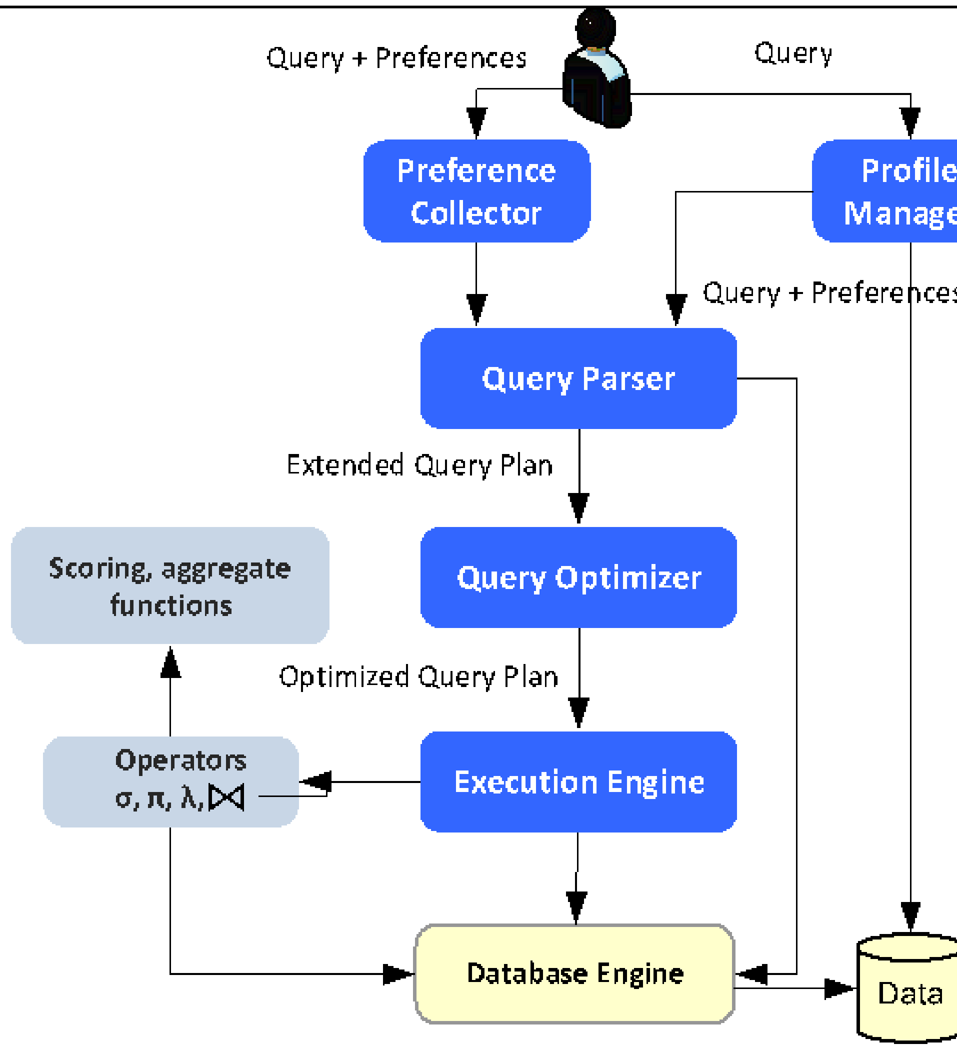
(d) R after evaluating $p_a \vee p_b$

Preferential Queries

- Consider a video-on-demand service application
  - Alice is searching for a recent movie
  - p_1: She loves comedies
  - p_2: She trusts user ratings
  - p_3: She is a fan of ‘Ben Stiller’

- $Q: \pi_{title,rating,genre} \{\sigma_{year>=2010}(MOVIES) \bowtie \lambda_{p1}(GENRES) \bowtie \lambda_{p2}(RATINGS) \bowtie CAST \bowtie \lambda_{p3}(ACTORS)\}$

Prototype System

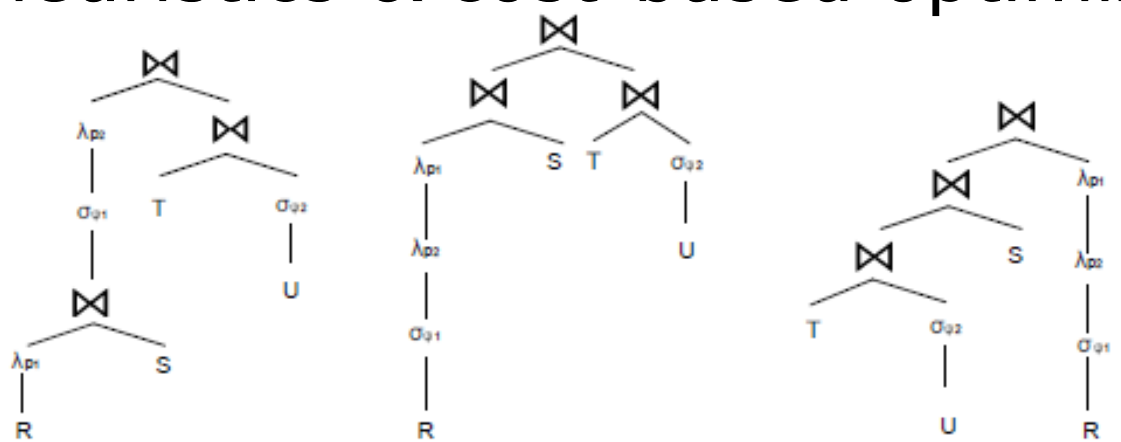


Overview

- hybrid architecture:
  - tighter coupled to the DBMS
  - operator-level query optimizations
  - easily deployable to any standard RDBMS-no source code modifications
- users input queries and preferences declaratively
- queries are transparently executed by the system

Query Optimization and Processing

- heuristics & cost-based optimizations



(a) Input Query Plan

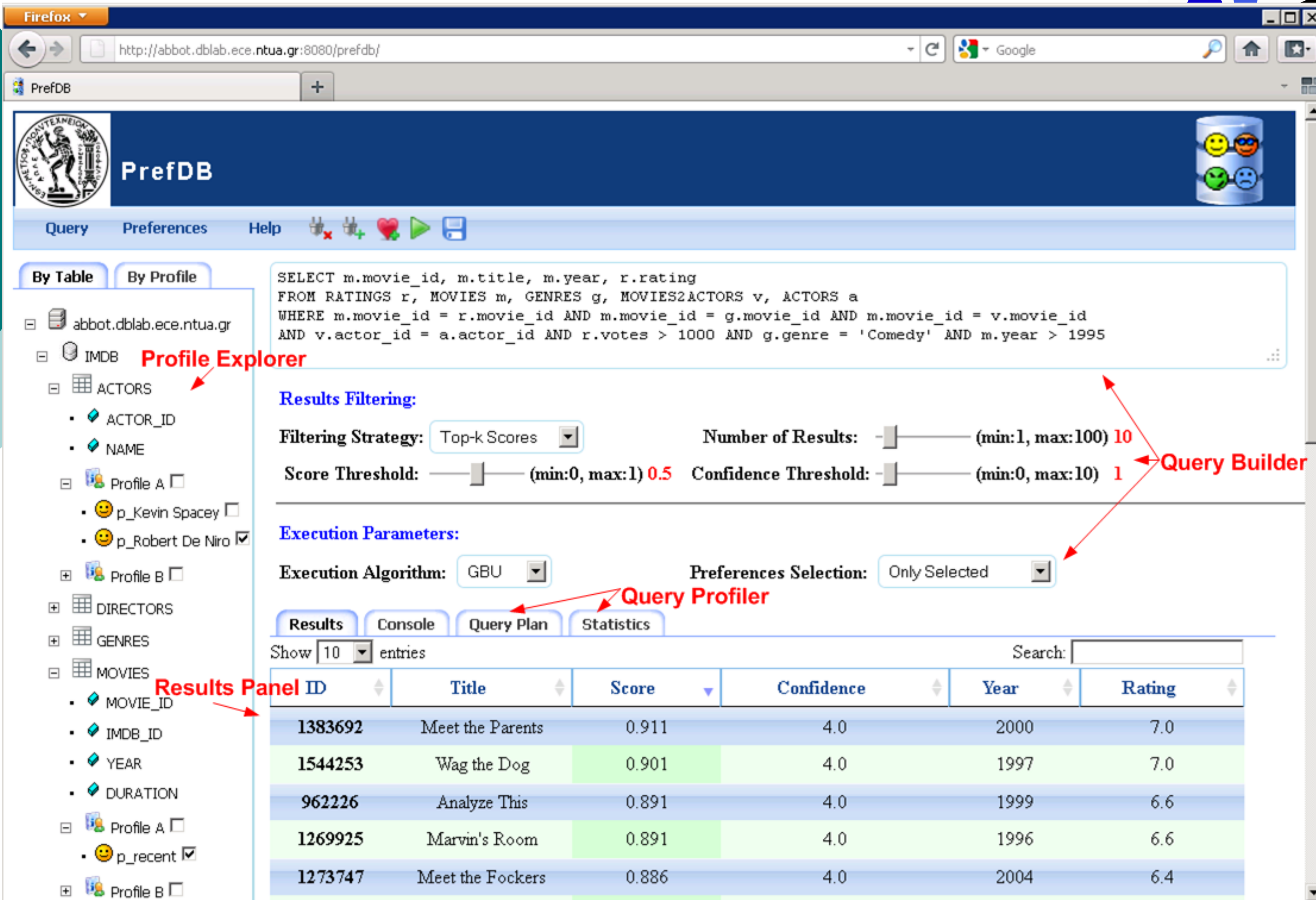
(b) Optimized Query Plan

(c) Output Query Plan

- blended processing strategies:
  - minimize intermediate materializations
  - defer operator execution wherever possible

PrefDBAdmin (SIGMOD' 12 demo)

- graphical tool for DBAs and application designers:
  - build and execute different types of preferential queries
  - experiment with different processing strategies (both plug-in or blended ones)
  - inspect query execution, preference-aware query plans, statistics, profiling info


Firefox
http://abbot.dblab.ece.ntua.gr:8080/prefdb/
Google
PrefDB
PrefDB
Query
Preferences
Help
By Table
By Profile
abbot.dblab.ece.ntua.gr
IMDB
Profile Explorer
ACTORS
ACTOR_ID
NAME
Profile A
p_Kevin Spacey
p_Robert De Niro
Profile B
DIRECTORS
GENRES
MOVIES
MOVIE_ID
IMDB_ID
YEAR
DURATION
Profile A
p_recent
Profile B
SELECT m.movie_id, m.title, m.year, r.rating
FROM RATINGS r, MOVIES m, GENRES g, MOVIES2ACTORS v, ACTORS a
WHERE m.movie_id = r.movie_id AND m.movie_id = g.movie_id AND m.movie_id = v.movie_id
AND v.actor_id = a.actor_id AND r.votes > 1000 AND g.genre = 'Comedy' AND m.year > 1995
Results Filtering:
Filtering Strategy: Top-k Scores
Number of Results: (min:1, max:100) 10
Score Threshold: (min:0, max:1) 0.5
Confidence Threshold: (min:0, max:10) 1
Query Builder
Execution Parameters:
Execution Algorithm: GBU
Preferences Selection: Only Selected
Query Profiler
Results
Console
Query Plan
Statistics
Show 10 entries
Search:
Results Panel
ID
Title
Score
Confidence
Year
Rating
1383692
Meet the Parents
0.911
4.0
2000
7.0
1544253
Wag the Dog
0.901
4.0
1997
7.0
962226
Analyze This
0.891
4.0
1999
6.6
1269925
Marvin's Room
0.891
4.0
1996
6.6
1273747
Meet the Fockers
0.886
4.0
2004
6.4

Epilogue

- Personalization is a key issue in information systems of various kinds
- Can be crucial in user satisfaction especially on Web, Database and Cloud environments
- We definitely need more work on models, processing techniques, efficient algorithms, and optimization techniques in order to seamlessly integrate personalization in existing paradigms

References

- Giorgos Giannopoulos, Theodore Dalamagas and Timos Sellis, **Collaborative Ranking Function Training for Web Search Personalization**, PersDB'09.
- Giorgos Giannopoulos, Theodore Dalamagas and Timos Sellis, **Search Behavior-Driven Training of Multiple Ranking Models to Enhance Result Re-ranking**, TPDL' 11.
- Giorgos Giannopoulos, Ulf Brefeld, Theodore Dalamagas and Timos Sellis, **Ranking Models for User Intent**, CIKM'11.
- A. Arvanitis, G. Koutrika, **Towards Preference-aware Relational Databases** (ICDE' 12)
- A. Arvanitis, G. Koutrika, **PrefDB: Bringing Preferences closer to the DBMS** (SIGMOD' 12)