Random walking through the data: novel spectral methods for the analysis of networks

Fabrizio Silvestri
ISTI - CNR, Pisa, Italy

Random walking through the data: novel spectral methods for the analysis of networks

Fabrizio Silvestri
ISTI - CNR, Pisa, Italy

Random walking through the data: applications of a less known spectral method for the analysis of networks

Fabrizio Silvestri
ISTI - CNR, Pisa, Italy

Spectral Methods

- Deals with analyzing the spectrum of matrices...
  - ... we need to put our data in matrix form (or equivalently... graph!)
- In the context of Web data we are full of graphs, i.e. matrices

Applications

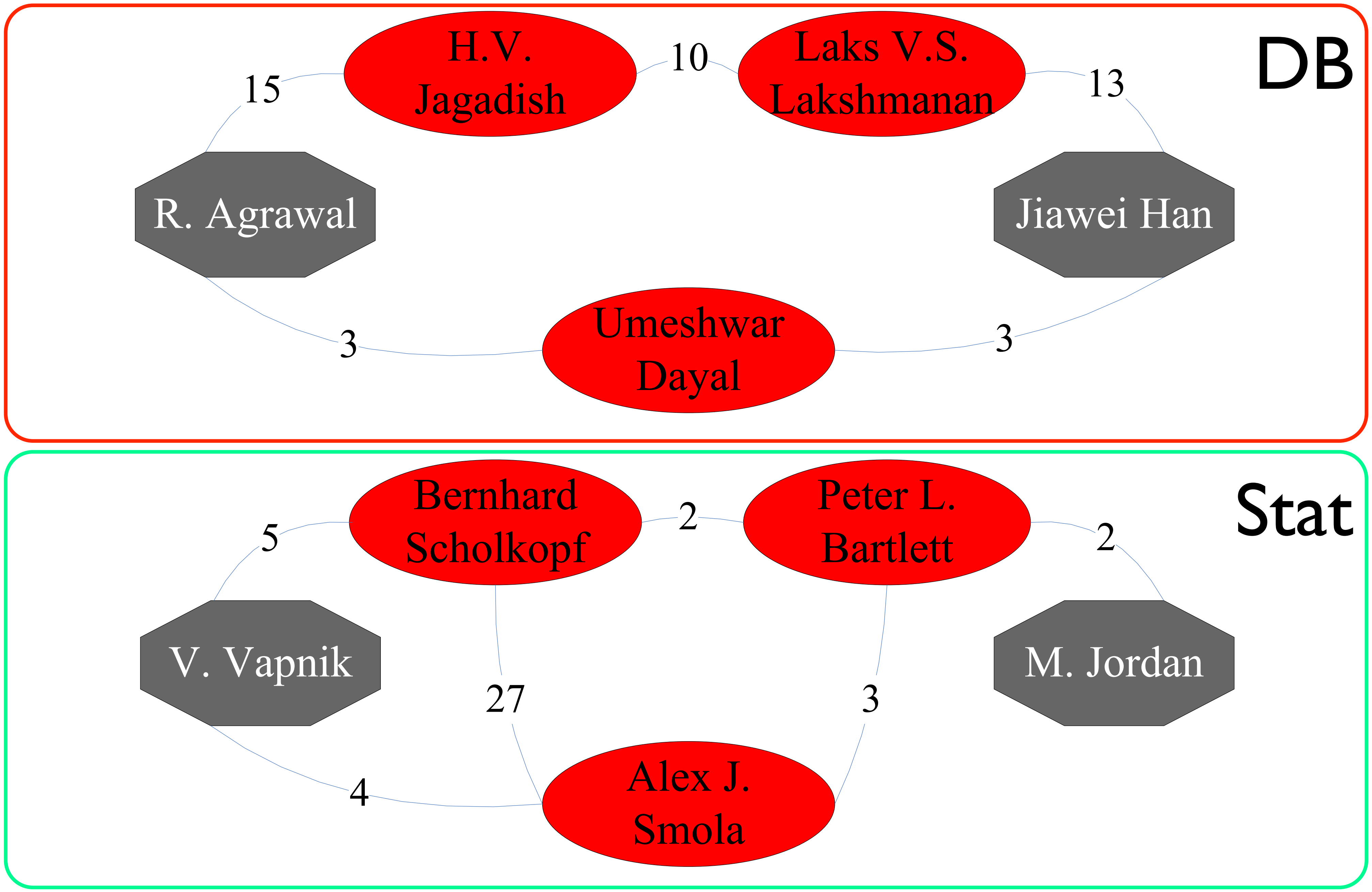
- Recommender systems:
  - Tourist recommender system
  - Query recommender system
- How do they mix?
  - Stay tuned!

Preliminary
(Center-piece Subgraph)

- Hanghang Tong and Christos Faloutsos. **Center-piece subgraphs: problem definition and fast solutions**. In Proceedings of KDD'06.
- It is a generalization of the connection-subgraph problem:
  - ***Given***: an edge-weighted undirected graph *G*, set vertices Q from *G*, and an integer budget *b* ***Find***: a connected subgraph *H* containing vertices in *Q* and at most *b* other vertices that maximizes a "goodness" function *g(H)*.

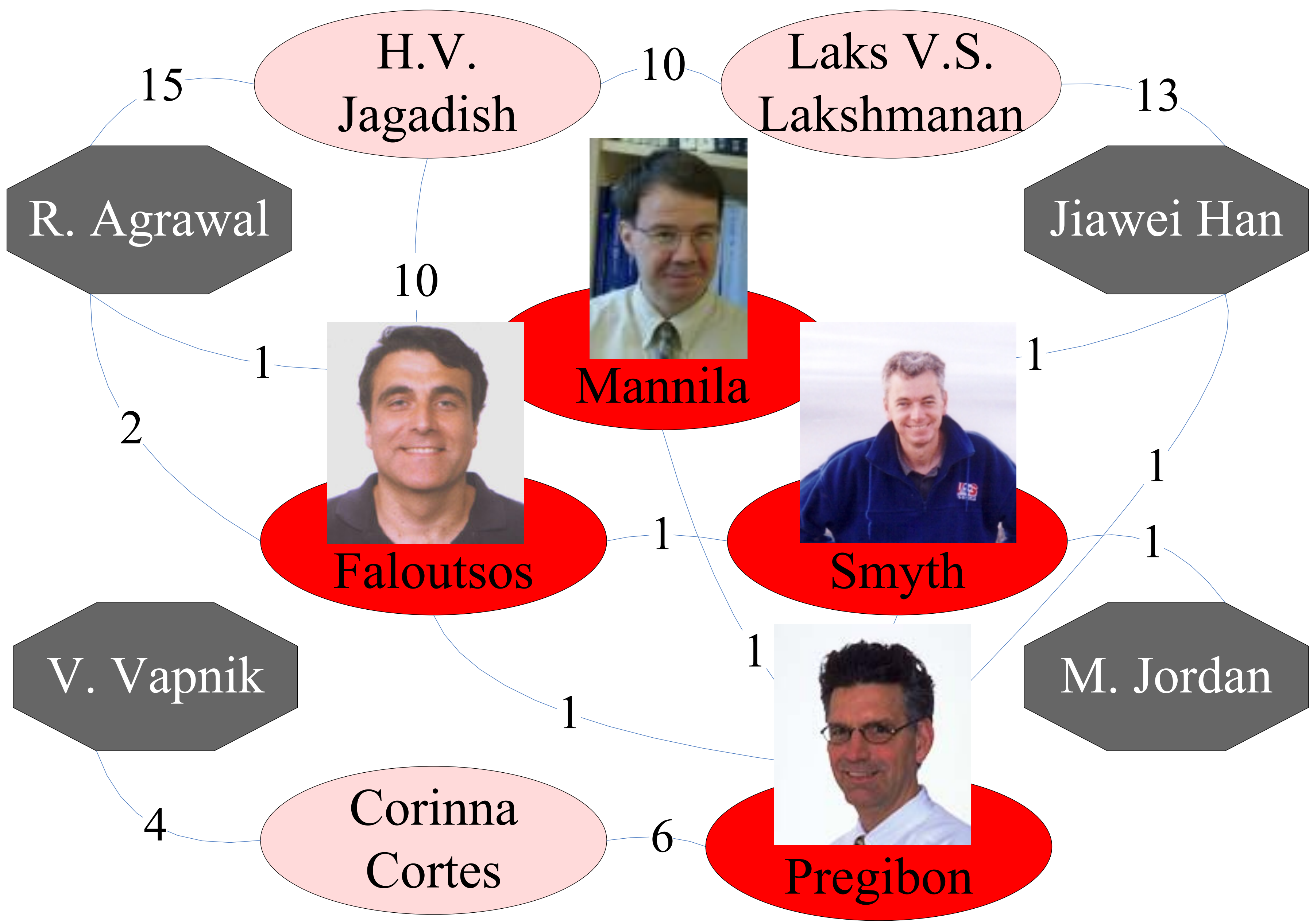
Example

(from H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In KDD'06.)



Example

(from H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In KDD'06.)



softAND

- Indeed, Center-Piece Subgraph problem has been defined in terms of a *softAND coefficient*:
  - ***Given***: *n* edge-weighted undirected graph *W*, *Q* nodes as source queries $Q = \{q_i\}$ ($i = 1,...,|Q|$), the softAND coefficient *k* and an integer budget *b*
  - ***Find***: a suitably connected subgraph *H* that
    - contains all query nodes q_i, at most *b* other vertices,
    - it maximizes a "goodness" function *g(H)*, and
    - intermediate nodes must have good connections to "at least" *k* of the query nodes.

softAND

- Indeed, Center-Piece Subgraph p
  terms of a *softAND coefficient*:
  - ***Given***: *n* edge-weighted undir
    source queries $Q = \{q_i\}$ ($i = 1$,
    coefficient *k* and an integer bu
  - ***Find***: a suitably connected subgraph *H* that
    - contains all query nodes q_i, at most *b* other vertices,
    - it maximizes a "goodness" function *g(H)*, and
    - intermediate nodes must have good connections to "at least" *k* of the query nodes.

In our applications we don't use the softAND coefficient.

How to Compute it

- Let us first define the goodness score for nodes. For a given node *j*, we have two types of goodness score for it:
  - Let $r(i, j)$ be the goodness score of a given node *j* w.r.t. the query q_i;
  - Let $r(Q, j)$ be the goodness score of a given node *j* w.r.t. the query set *Q*.

How to Compute it

- The goodness criterion of *H* can be defined as:

$$g(\mathcal{H}) = \sum_{j \in \mathcal{H}} r(\mathcal{Q}, j)$$

$$r(\mathcal{Q}, j) \triangleq r(\mathcal{Q}, j, Q) = \prod_{i=1}^{Q} r(i, j)$$

where $r(i,j)$ is the steady-state probability of a single node j w.r.t. query node q_i.

FAST CePS

(from H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In KDD'06.)

Input: the weighted graph $\mathbf{W}$, the query set $\mathcal{Q}$, $K\_softAND$ coefficient k, the budget b, and the number of partitions p

Output: the resulting subgraph $\mathcal{H}$

Step 0: pre-partition $\mathbf{W}$ into p pieces (one-time cost)

Step 1: pick up partitions of $\mathbf{W}$ that contain all the query nodes to construct the new weighted graph $\mathbf{nW}$

Step 2:. run $CEPS$ as in table 1 on $\mathbf{nW}$

CEPS

(from H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In KDD'06.)

Input: the weighted graph $\mathbf{W}$, the query set $\mathcal{Q}$, $K\_softAND$ coefficient k and the budget b

Output: the resulting subgraph $\mathcal{H}$

Step 1: Individual Score Calculation. Calculate the goodness score $r(i,j)$ for a single node j wrt a single query node q_i

Step 2: Combining Individual Scores. Combine the individual score $r(i,j)$ to get the goodness score $r(\mathcal{Q},j)$ for a single node j wrt the query set $\mathcal{Q}$

Step 3: "*EXTRACT*". Extract quickly a connection subgraph $\mathcal{H}$ with budget b maximizing the goodness criteria $g(\mathcal{H})$

EXTRACT

(from H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In KDD'06.)

1. Initialize output graph $\mathcal{H}$ null
2. Let len be the maximum allowable path length
3. While $\mathcal{H}$ is not big enough
   3.1. Pick up destination node $pd = argmax_{j \notin \mathcal{H}} r(\mathcal{Q}, j)$
   3.2. For each active source node q_i wrt node pd
      3.2.1. use table 3 to discover a key path $P(q_i, pd)$
      3.2.2. add $P(q_i, pd)$ to $\mathcal{H}$
4. Output the final $\mathcal{H}$

Single Key Path Discovery

(from H. Tong and C. Faloutsos. Center-piece subgraphs: problem definition and fast solutions. In KDD'06.)

1. Let len be the maximum allowable path length
2. For $j \leftarrow [1, ..., n]$
   2.1. Let $v = u_j$
   2.2. For $s \leftarrow [2, ..., len]$
   If v is already in the output subgraph
   $s' = s$
   Else
   $s' = s - 1$
   Let $C_s(i, v) = max_{u|u \rightarrow d_i, v}(C_{s'}(i, u) + r(\mathcal{Q}, v))$
3. Output the path maximizing $C_s(i, pd)/s$, where $s \neq 0$

Overall Cost

- Cost of Partitioning +
- for each “query” *Q*:
  - *CEPS(Q)* = *RWR(i,j)* (for each node *j* in *W*) + *EXTRACT(Q)*
  - *EXTRACT(Q)* = *b*(key path discovery)*

Overall Cost

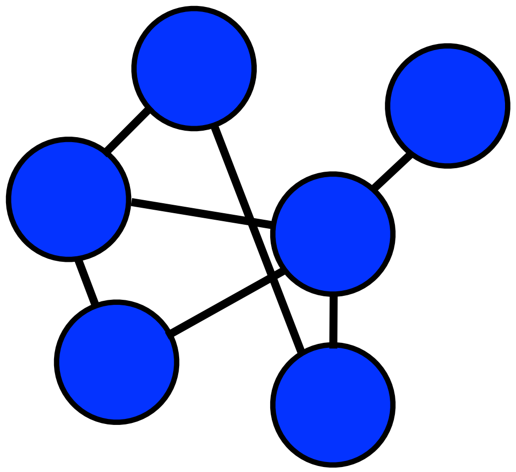
- Cost of Partitioning +
- for each "query" *Q*:
  - *CEPS(Q)* = *RWR(i,j)* (for each node *j* in *W*) + *EXTRACT(Q)*
  - *EXTRACT(Q)* = *b**(key path discovery)*

- Prohibitively high to compute it for several *Q* arriving online

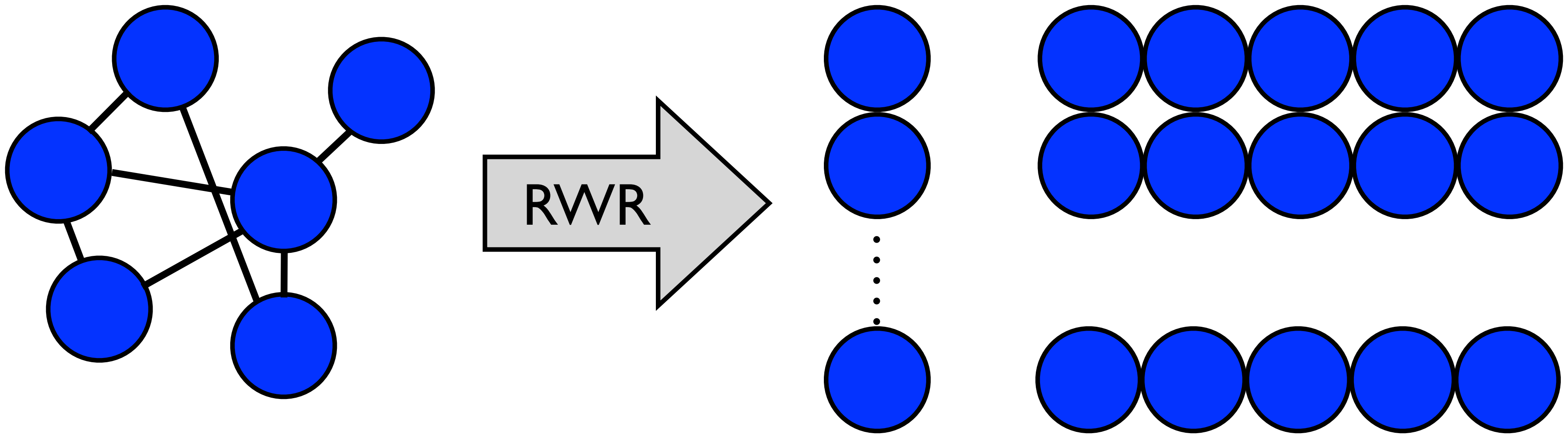
Our Take on Center-Piece Subgraph

- *Goal*:
  - to find a representation for the graph allowing online computation of CePS for multiple query sets *Q*
- *Motivations*:
  - In the context of recommender systems queries arrive online and need to be answered in a fraction of a second.

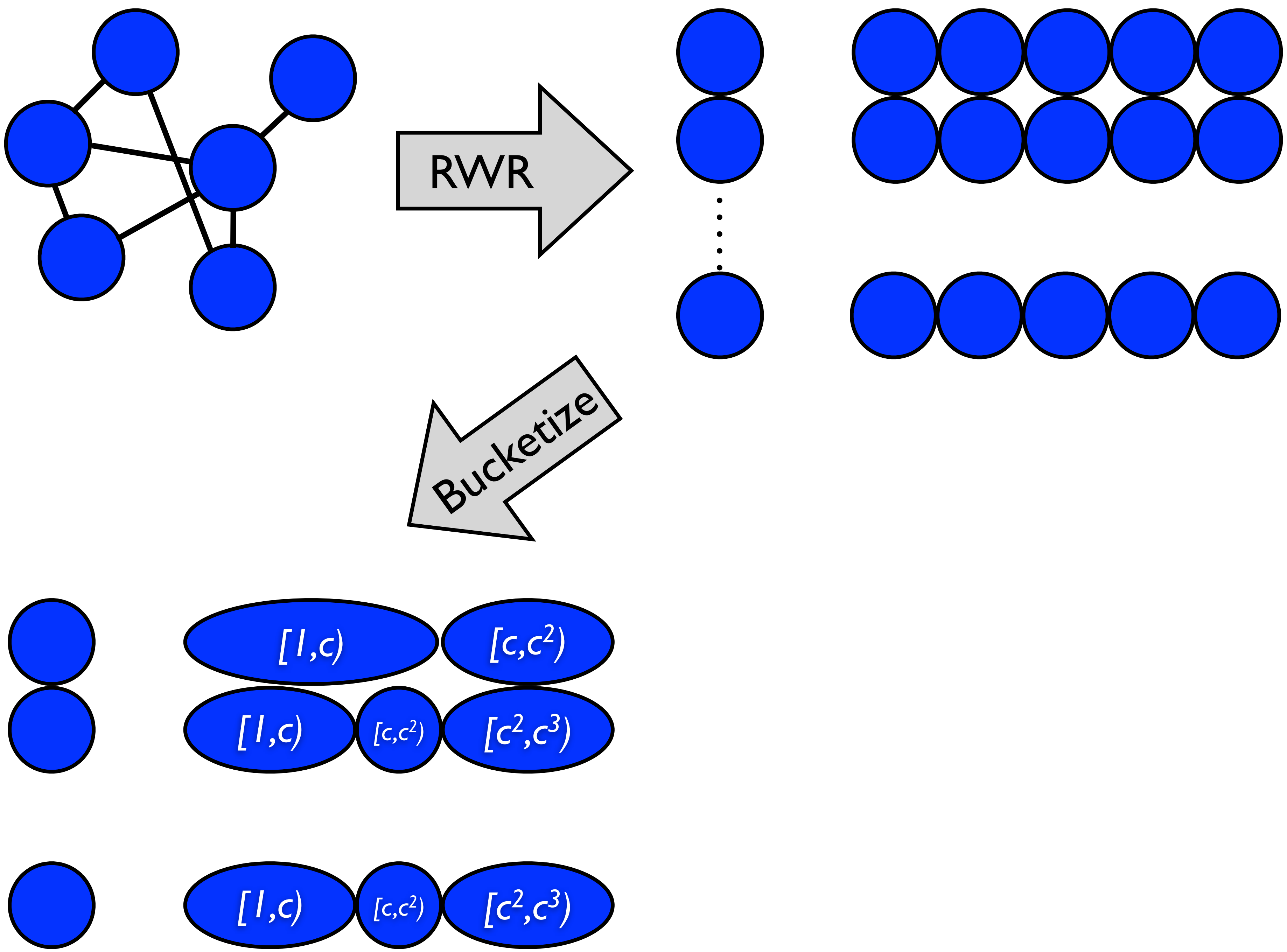
The Idea



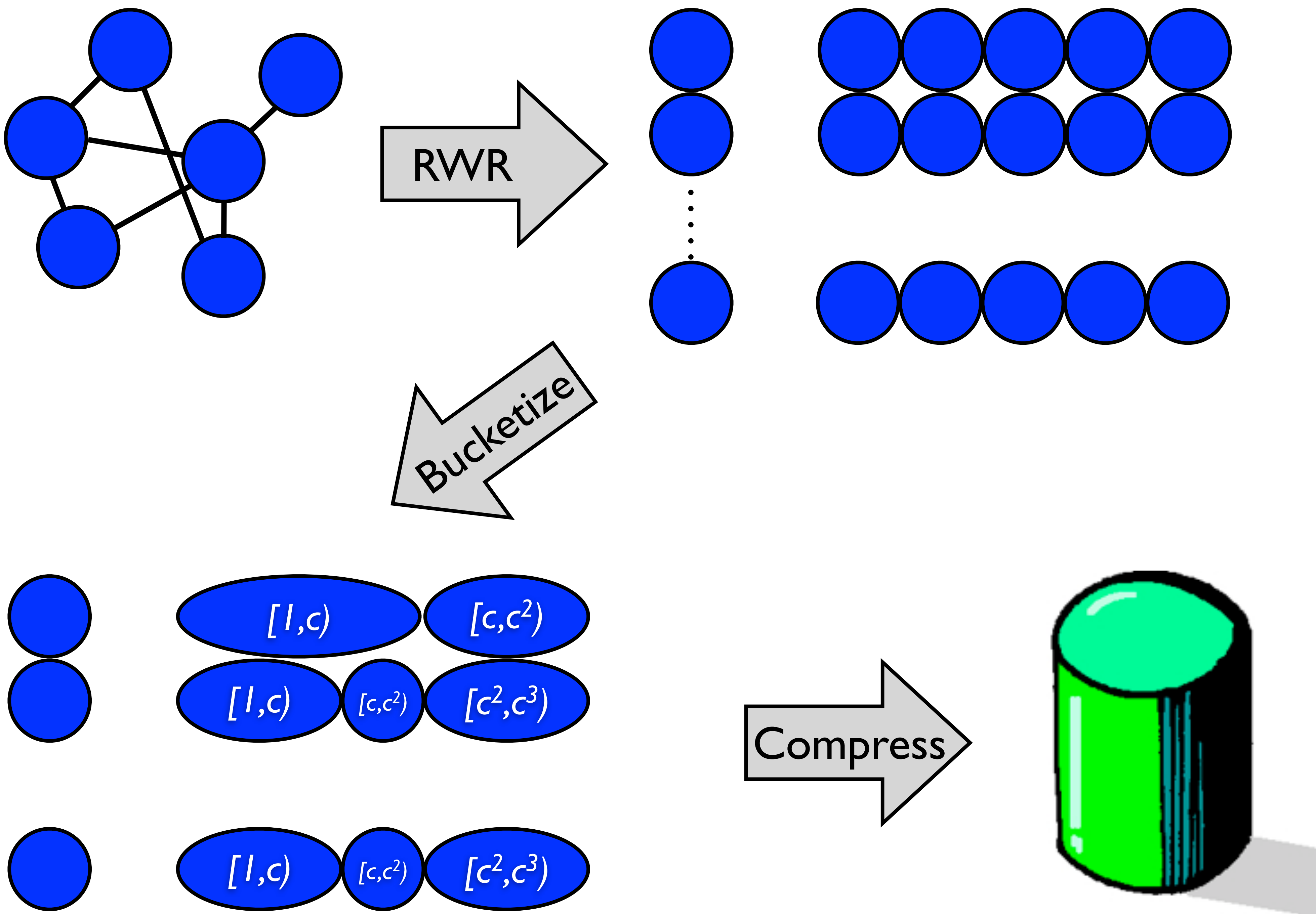
The Idea



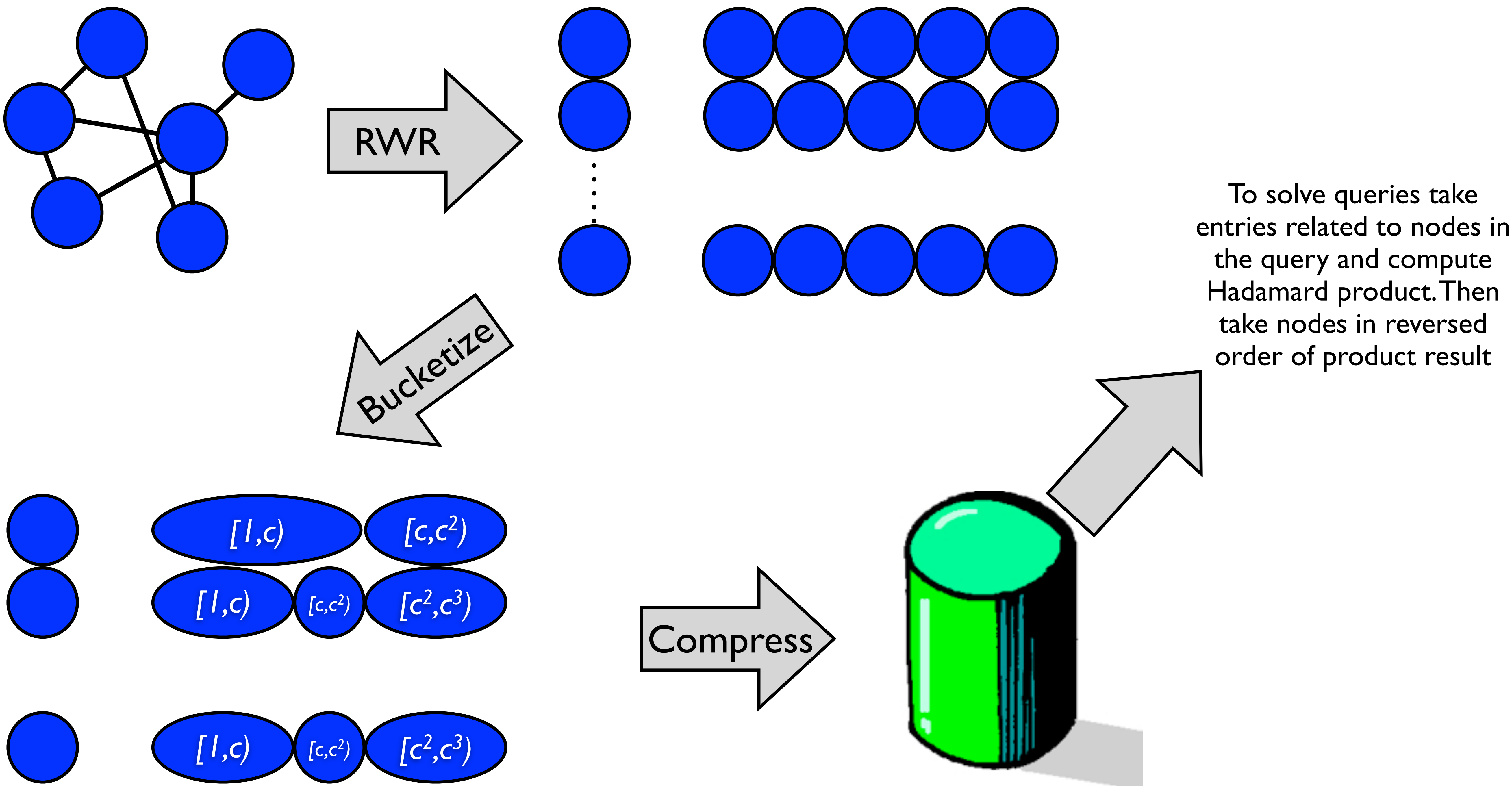
The Idea



The Idea



The Idea

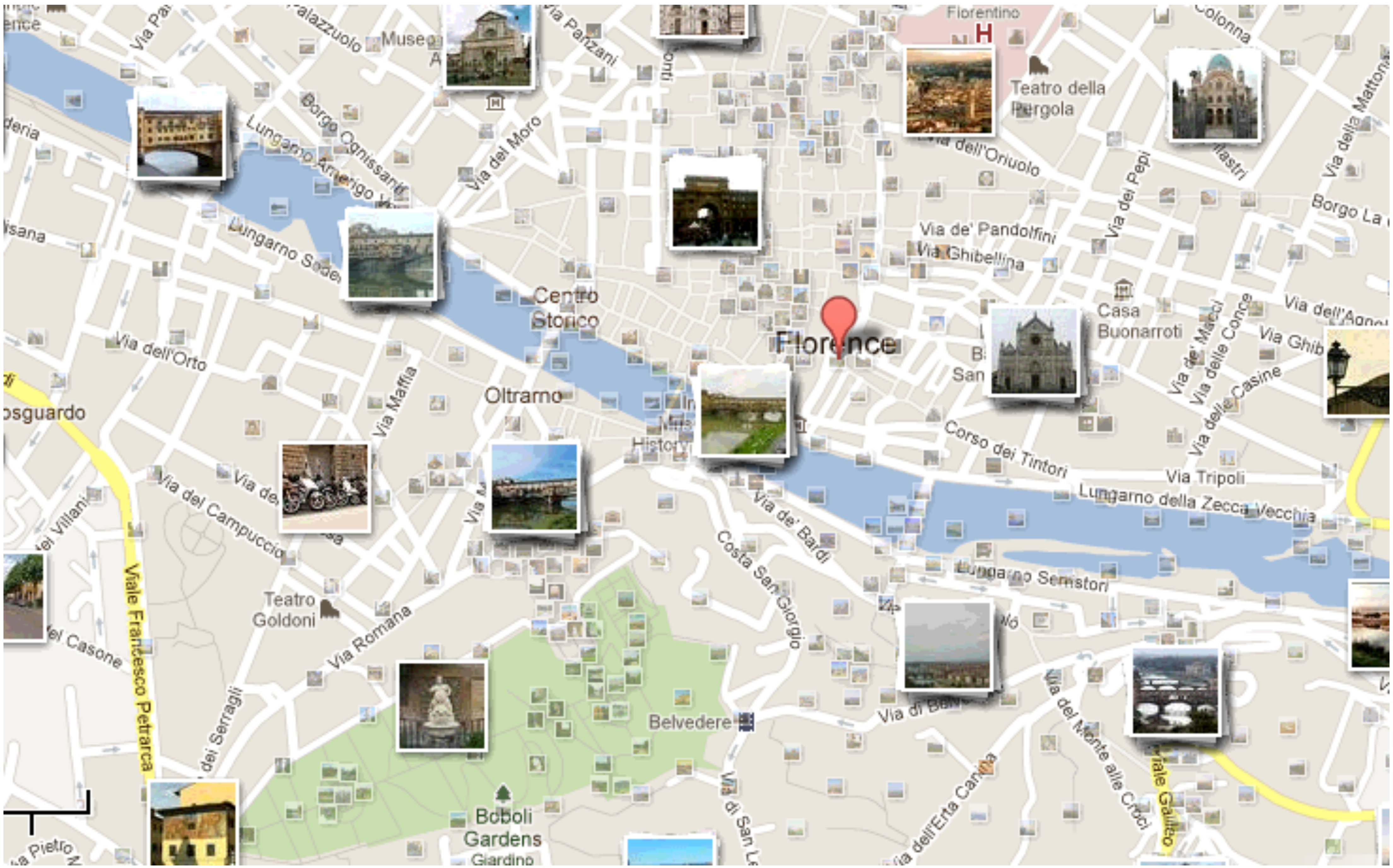


RWR
Bucketize
[1,c)
[c,c^2)
[1,c)
[c,c^2)
[c^2,c^3)
[1,c)
[c,c^2)
[c^2,c^3)
Compress
To solve queries take entries related to nodes in the query and compute Hadamard product. Then take nodes in reversed order of product result

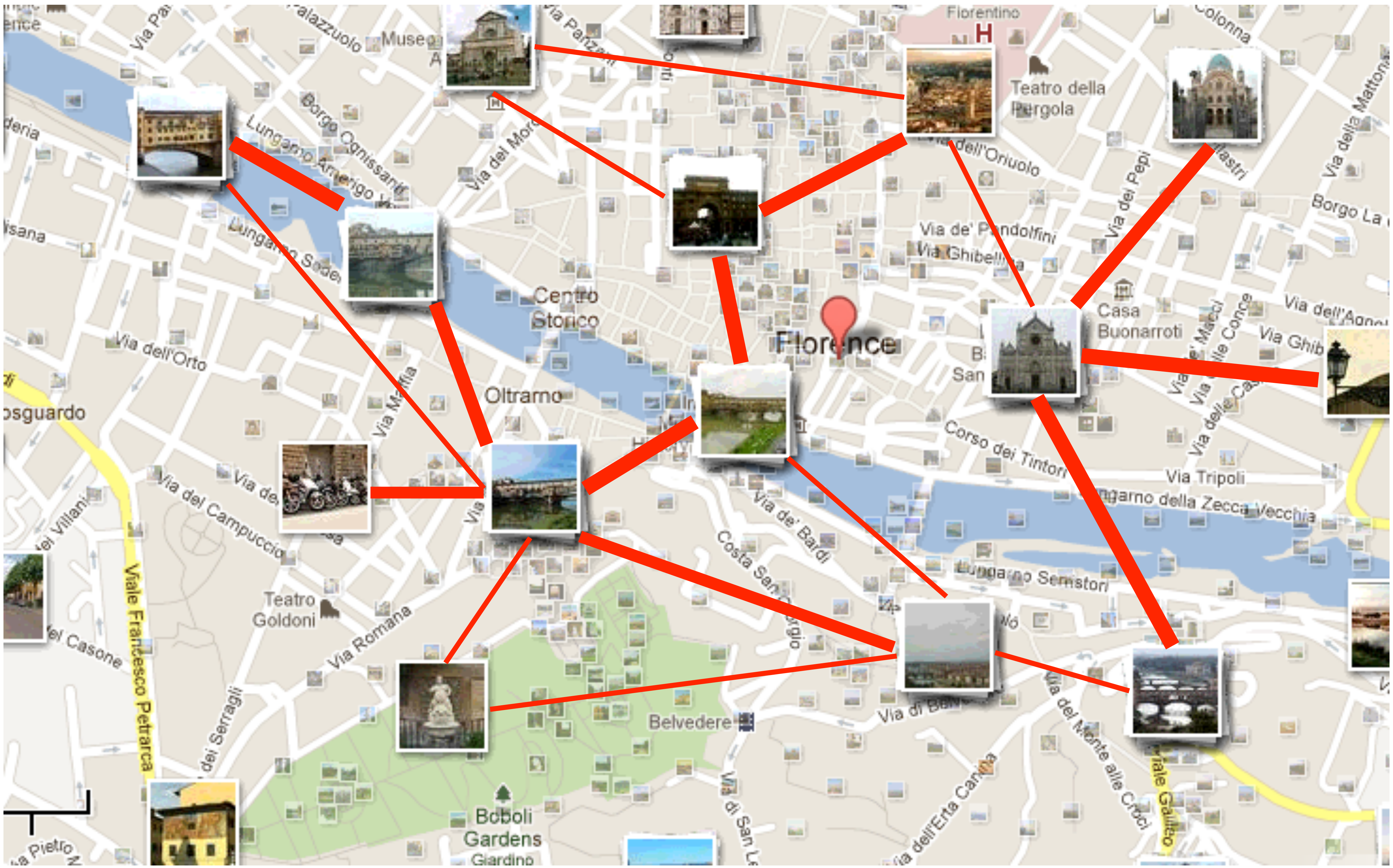
A Tale of Two Applications

- Tourist Recommender System:
  - C. Lucchese, R. Perego, F. Silvestri, H. Vahabi, R. Venturini. **How random walks can help tourism**. 34th European Conference on Information Retrieval (ECIR), 2012.
- Query Recommender System:
  - F. Bonchi, R. Perego, F. Silvestri, H. Vahabi, and R. Venturini. **Efficient Query Recommendations in the Long Tail via Center-Piece Subgraphs**. SIGIR 2012: To Appear.

Tourist Recommenders



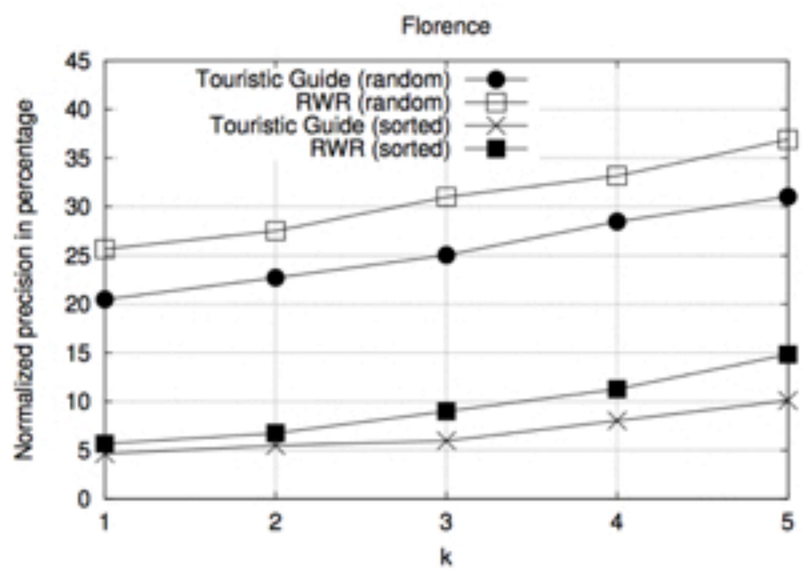
Tourist Recommenders

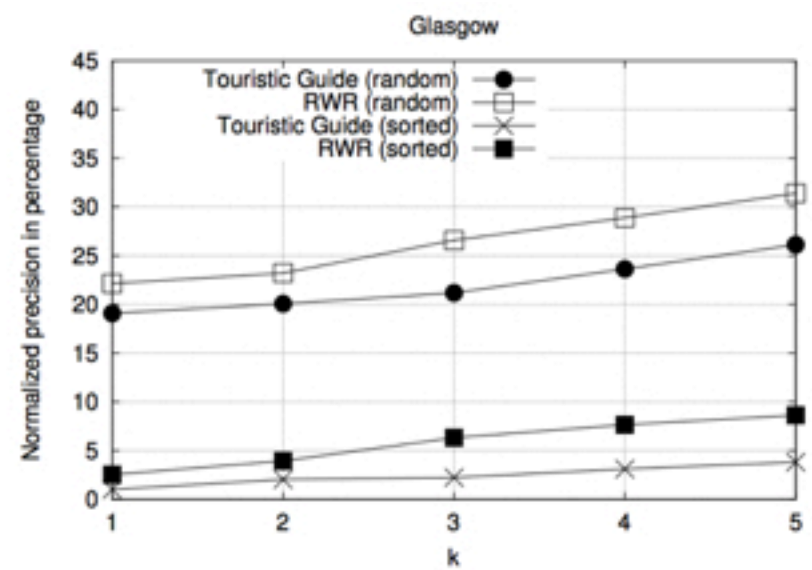


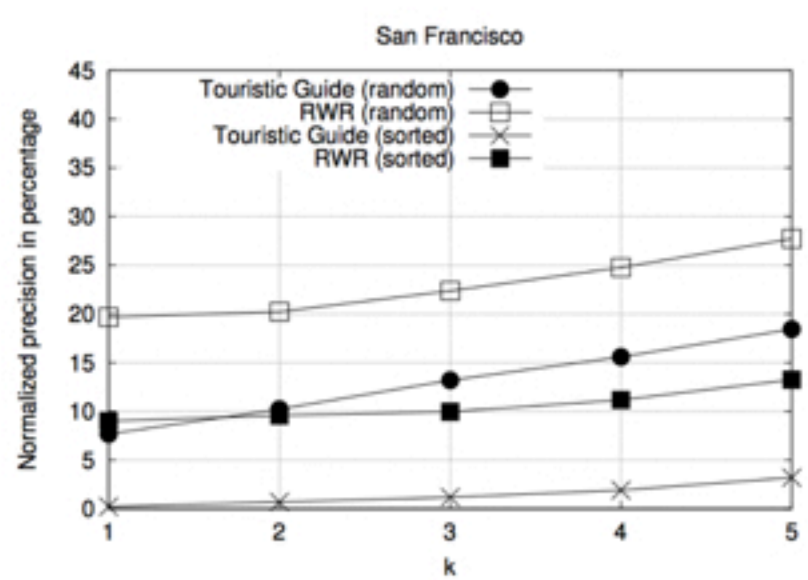
Tourist Recommenders

Some Results

- Baseline: suggest always the top-*k* visited PoIs in a city
- We used three datasets: Florence, Glasgow, and San Francisco.







Anecdotes

Starting PoIs in U
Palazzo Vecchio
Piazza della Signoria

Top-10 ranked PoIs

PoI	Probability
Ponte Vecchio	$5.9 \cdot e^{-4}$
Piazzale Michelangelo	$2.1 \cdot e^{-4}$
Palazzo Pitti	$1.9 \cdot e^{-4}$
Giotto's Campanile	$6.8 \cdot e^{-5}$
Boboli Gardens	$4.9 \cdot e^{-5}$
Loggia dei Lanzi	$4.6 \cdot e^{-5}$
Piazza Santa Croce	$4.2 \cdot e^{-5}$
Uffizi	$4.1 \cdot e^{-5}$
Basilica of Santa Croce	$3.9 \cdot e^{-5}$
Ponte alle Grazie	$3.4 \cdot e^{-5}$

a)

Starting PoIs in U
La Specola
Museo Fiorentino di Preistoria
Museo Horne
Bargello

Top-10 ranked PoIs

PoI	Probability
Uffizi	$1.4 \cdot e^{-10}$
Giotto's Campanile	$1.2 \cdot e^{-10}$
Palazzo Medici Riccardi	$9.8 \cdot e^{-11}$
Vasari Corridor	$7.4 \cdot e^{-11}$
Medici Chapel	$6.5 \cdot e^{-11}$
Basilica of Santa Croce	$5.3 \cdot e^{-11}$
San Marco's National Museum	$1.3 \cdot e^{-11}$
Dante Alighieri's House	$9.6 \cdot e^{-12}$
Modern Art Gallery	$9.3 \cdot e^{-12}$
Museo Stibbert	$8.0 \cdot e^{-12}$

b)

Starting PoIs in U
Clyde Tunnel
Govan Subway Station
Hillhead Subway Station
Renfrew Airport

Top-10 ranked PoIs

PoI	Probability
Glasgow International Airport	$1.2 \cdot e^{-8}$
Buchanan Street Subway Station	$4.2 \cdot e^{-9}$
Kelvinbridge	$6.8 \cdot e^{-10}$
Glasgow Seaplane Terminal	$2.4 \cdot e^{-10}$
St Enoch Subway Station	$2.0 \cdot e^{-10}$
Glasgow City Heliport	$2.0 \cdot e^{-10}$
Buchanan Bus Station	$9.5 \cdot e^{-11}$
Ibrox Subway Station	$9.5 \cdot e^{-11}$
Kelvinhall Subway Station	$8.3 \cdot e^{-11}$
Cowcaddens Subway Station	$9.5 \cdot e^{-12}$

c)

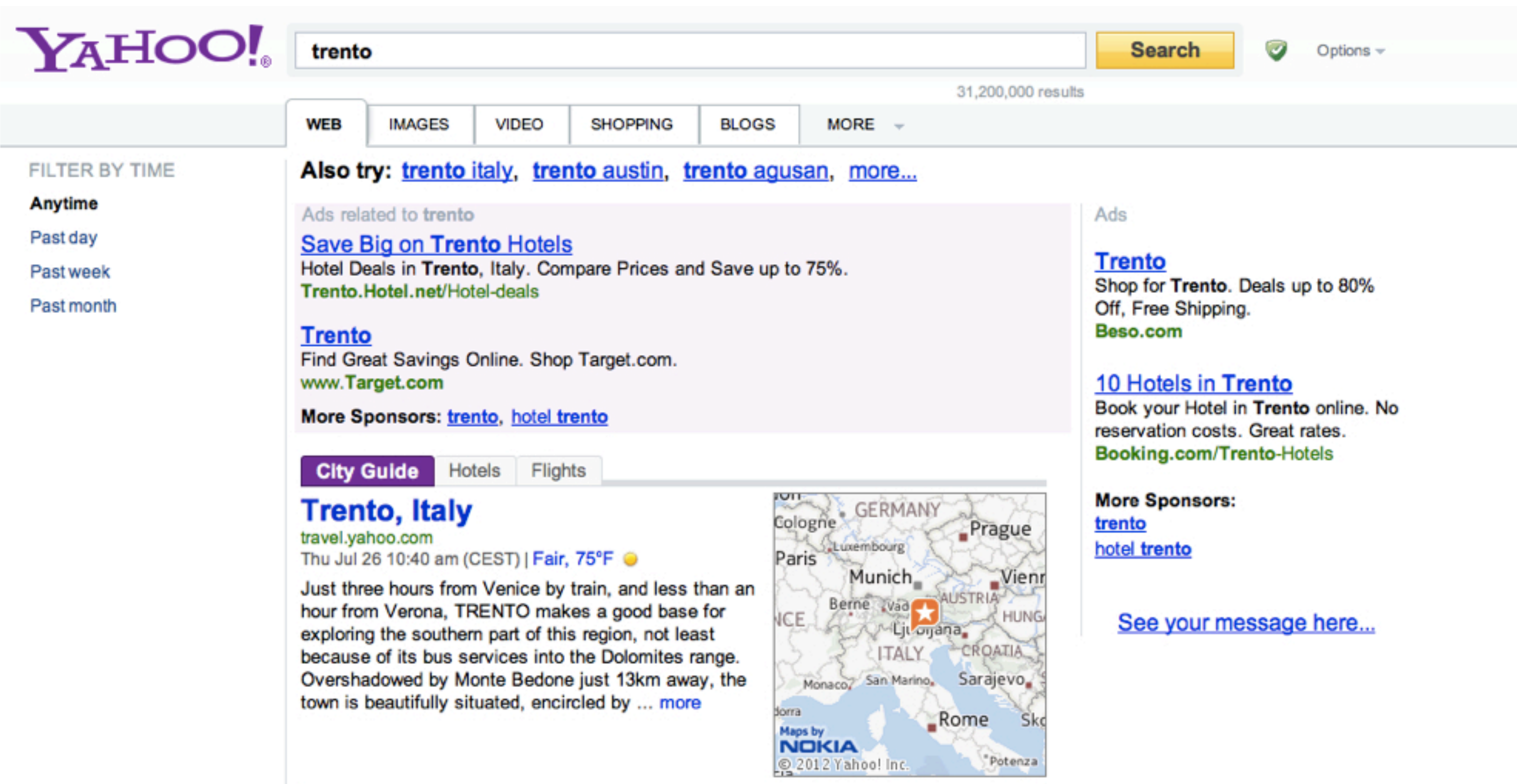
Starting PoIs in U
Golden Gate Theatre
San Francisco Conservatory of Music

Top-10 ranked PoIs

PoI	Probability
War Memorial Opera House	$1.1 \cdot e^{-5}$
Dolores Park	$1.0 \cdot e^{-5}$
Castro Theatre	$8.1 \cdot e^{-6}$
Yerba Buena Gardens	$7.8 \cdot e^{-6}$
Embarcadero Center	$7.3 \cdot e^{-6}$
Metreon	$6.3 \cdot e^{-6}$
Golden Gate Bridge	$5.5 \cdot e^{-6}$
Pacific-union Club	$4.2 \cdot e^{-6}$
Lake Merritt	$4.1 \cdot e^{-6}$
American Conservatory Theater	$3.9 \cdot e^{-6}$

d)

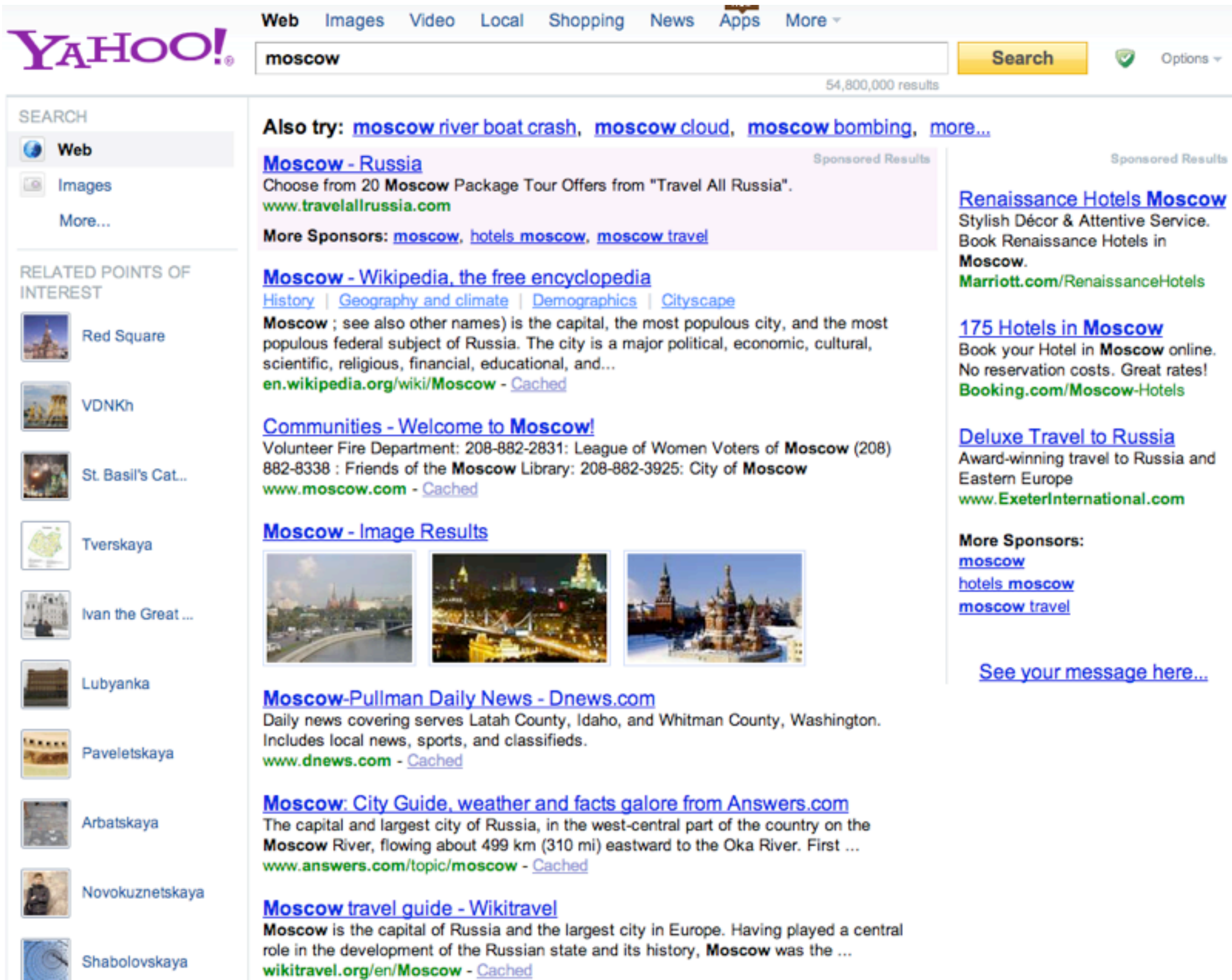
Query Recommender



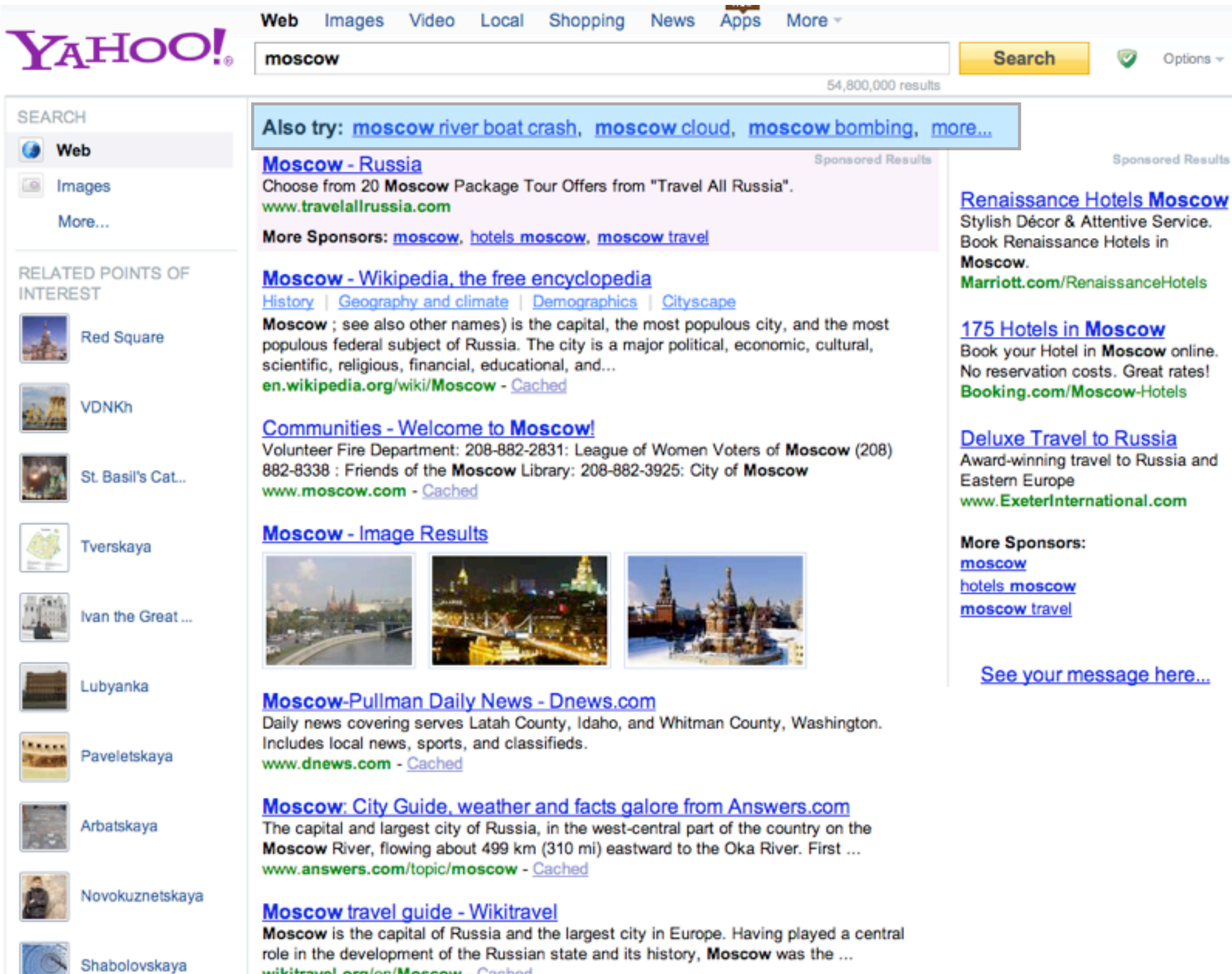
Query suggestion practices

- Use of the Wisdom of the Crowd mined from Query Logs to recommend related queries that are likely to better specify the information need of the user
  - shorten length of user sessions
  - enhance perceived QoE

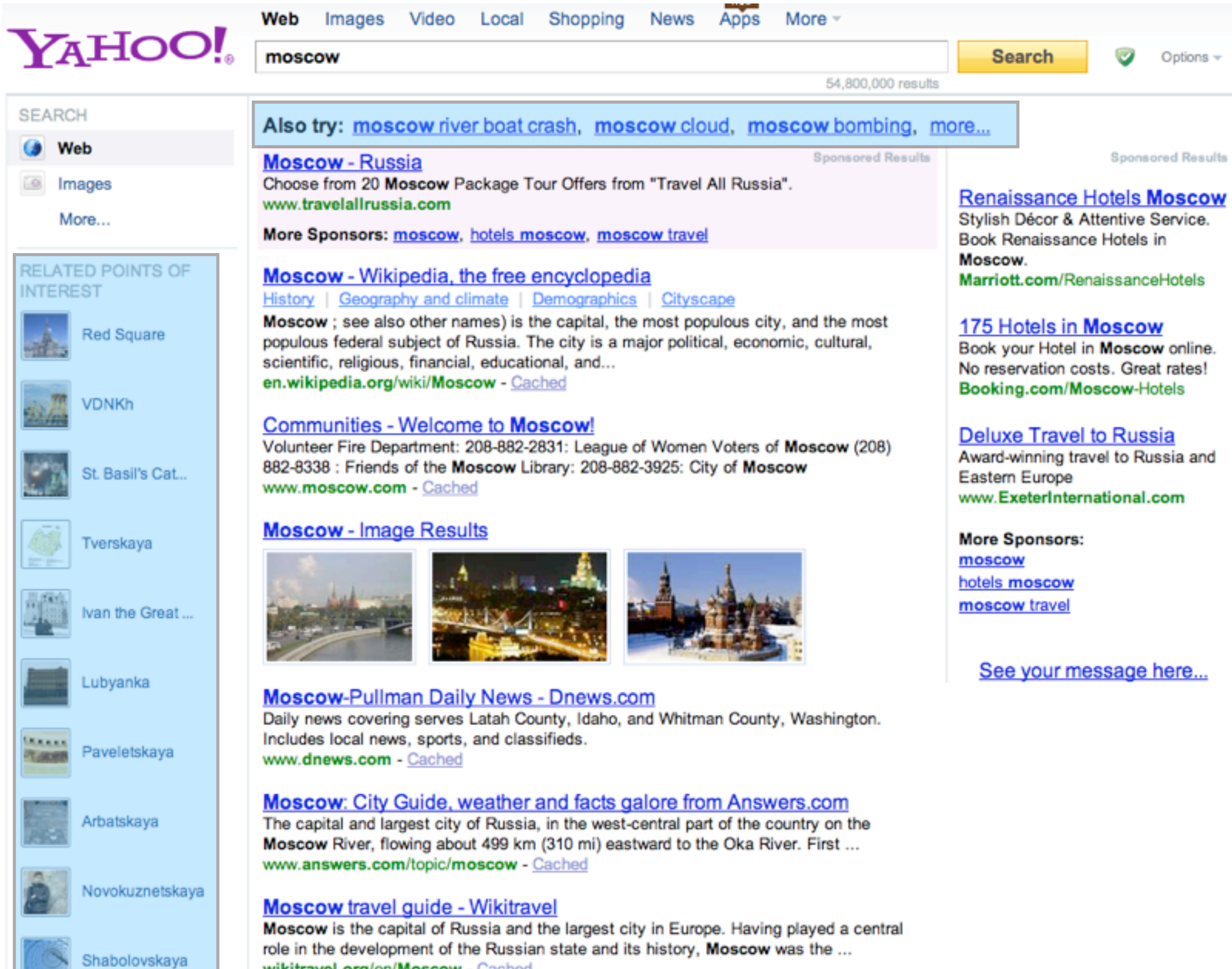
Queries in the Head



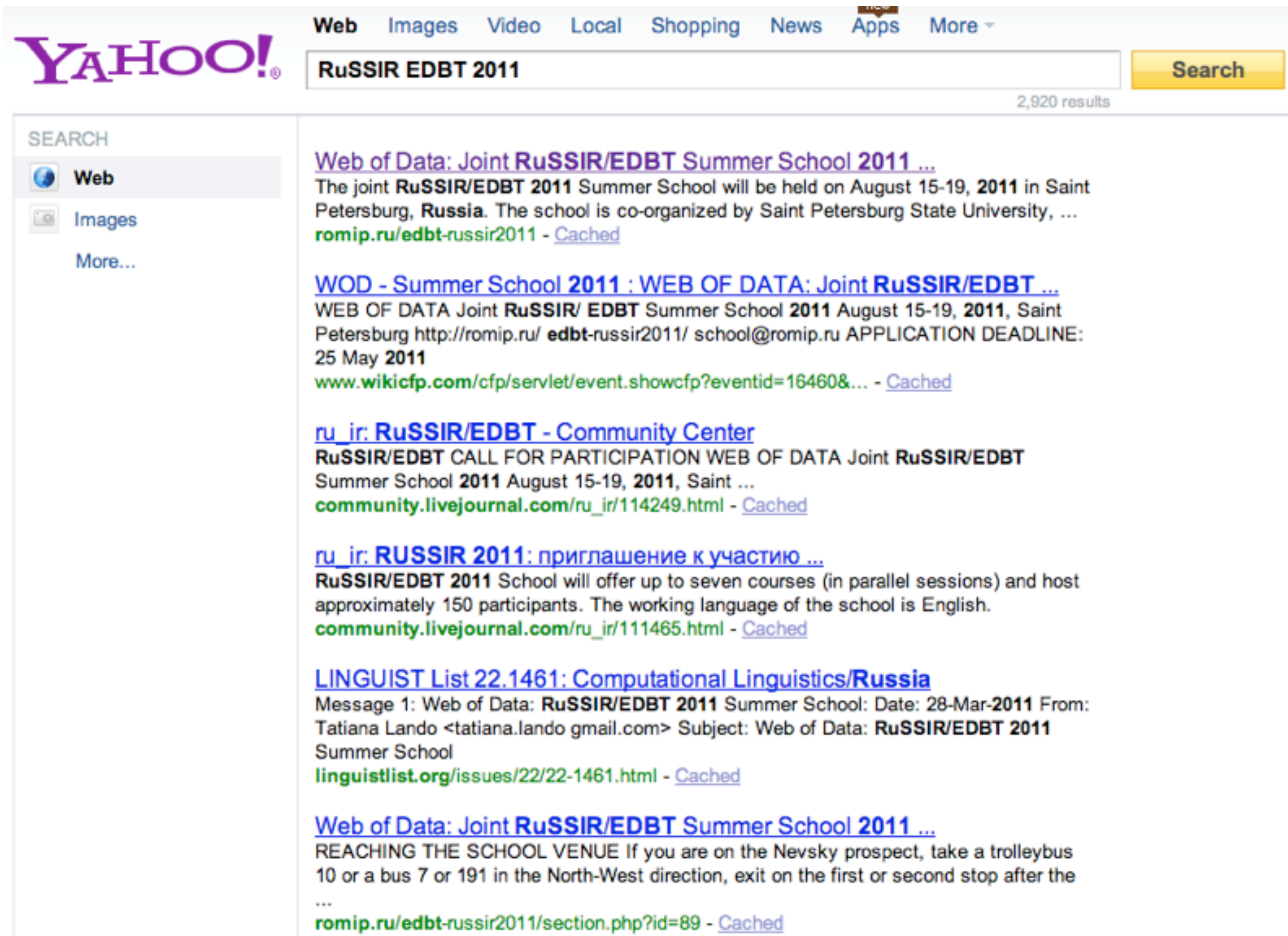
Queries in the Head



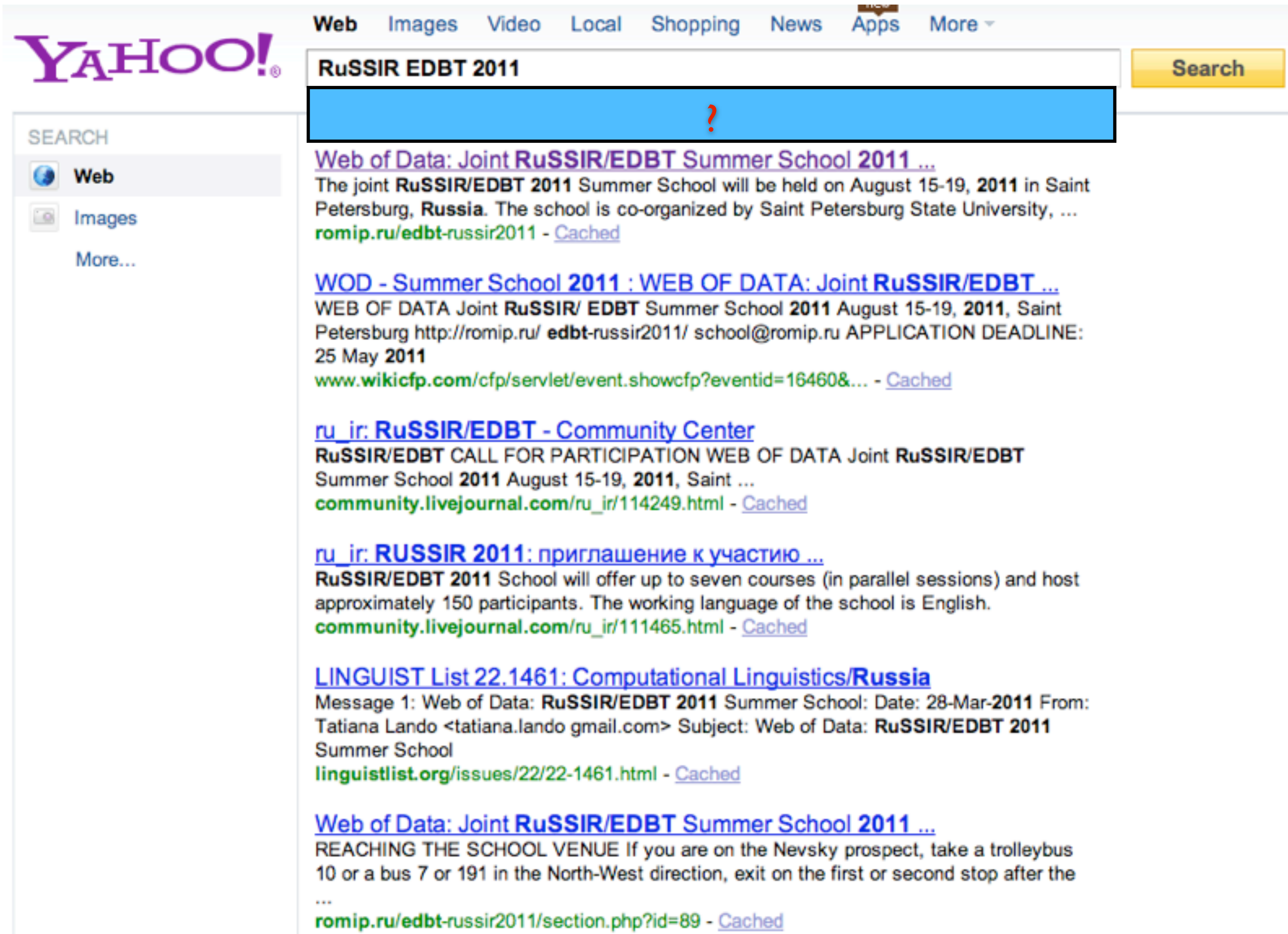
Queries in the Head



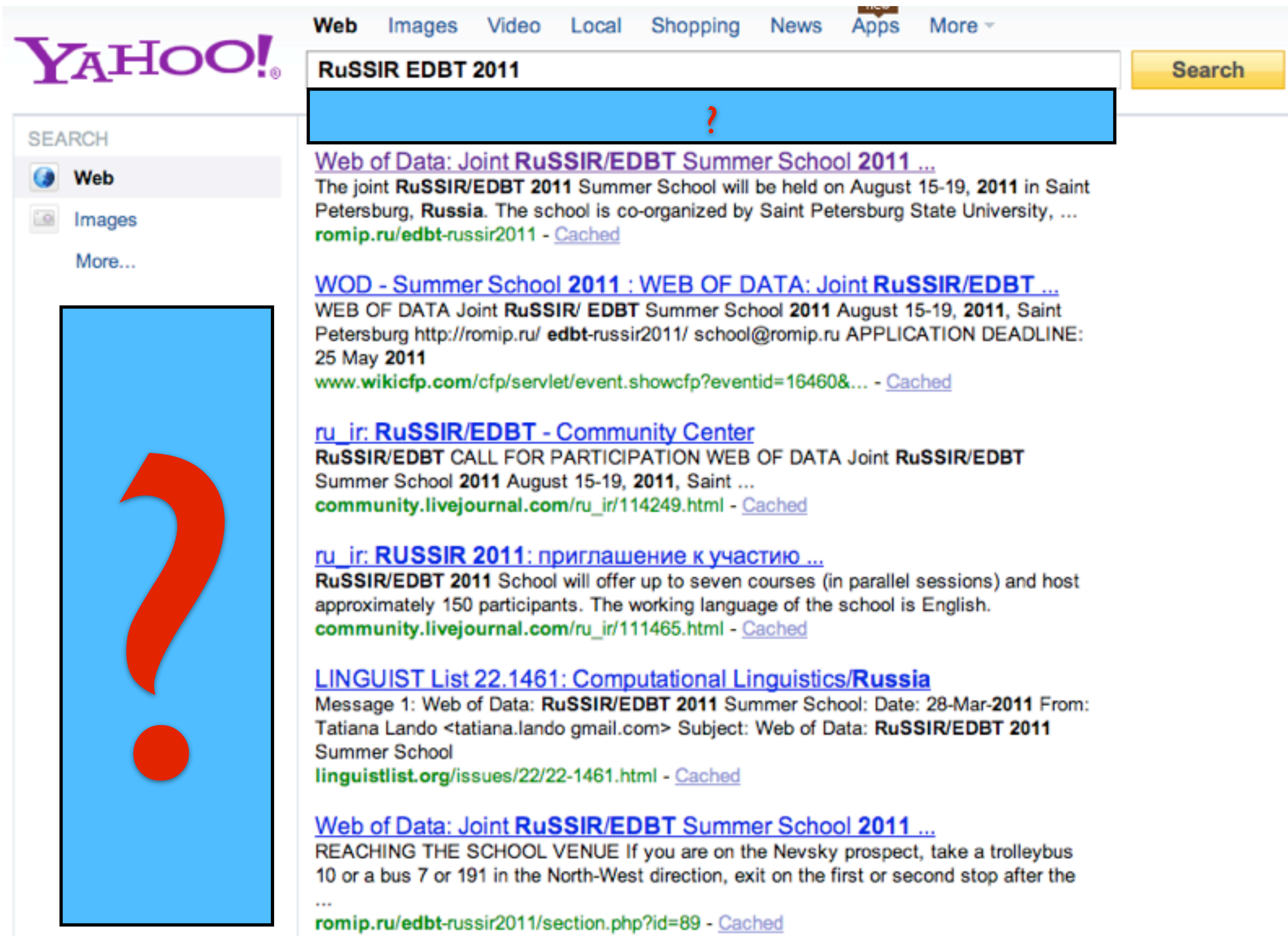
Queries in the Long Tail



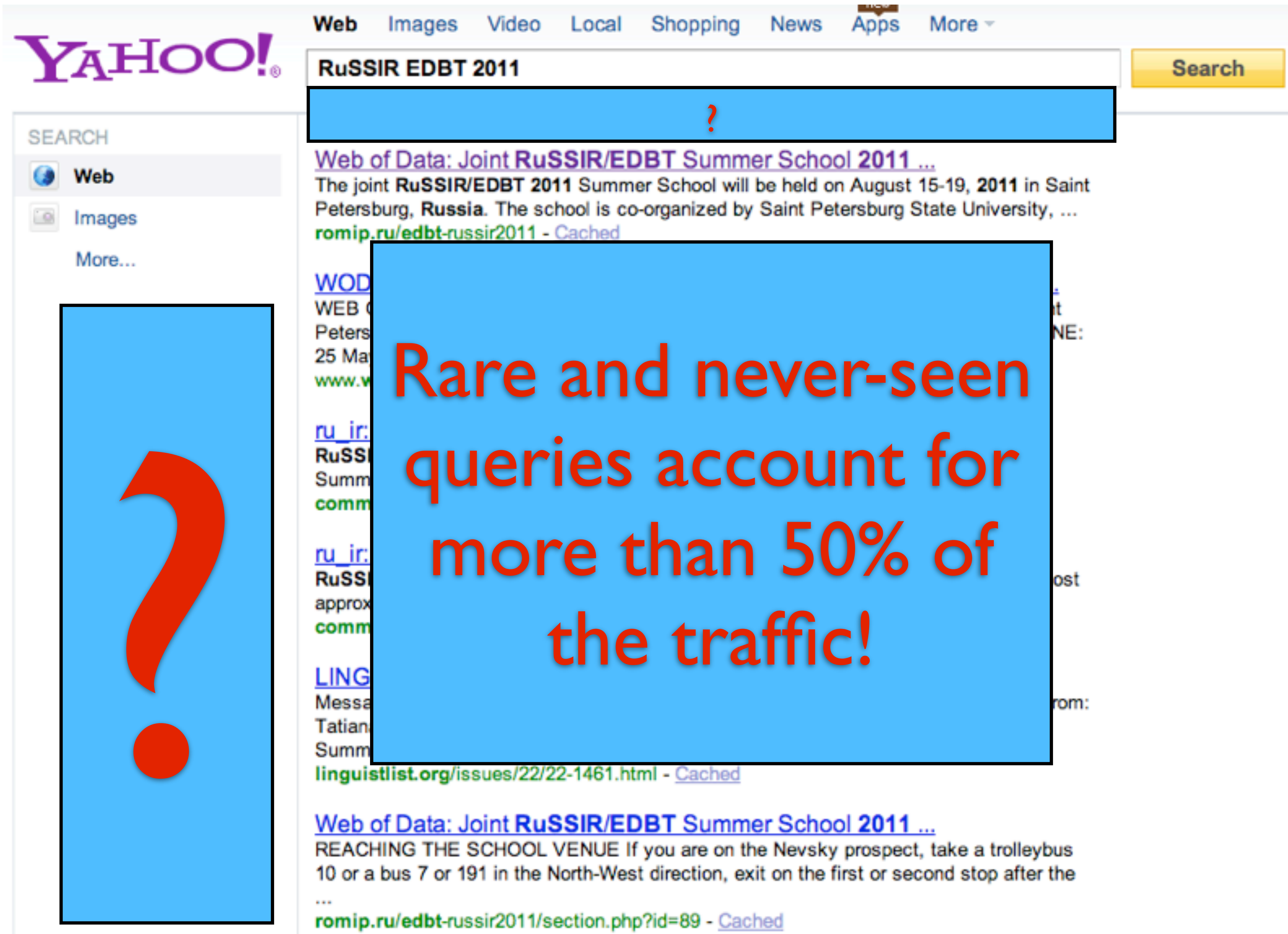
Queries in the Long Tail



Queries in the Long Tail



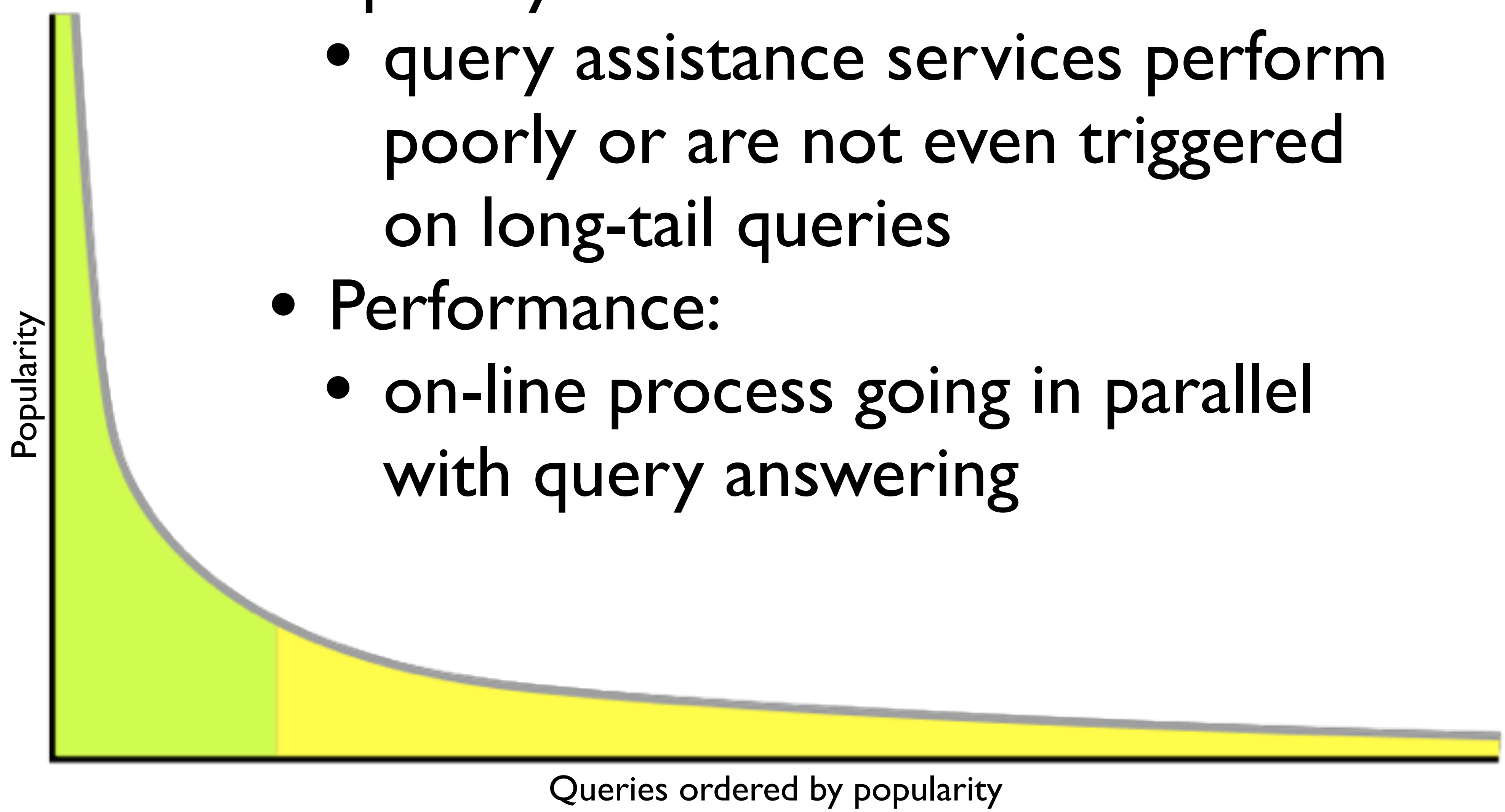
Queries in the Long Tail



Rare and never-seen queries account for more than 50% of the traffic!

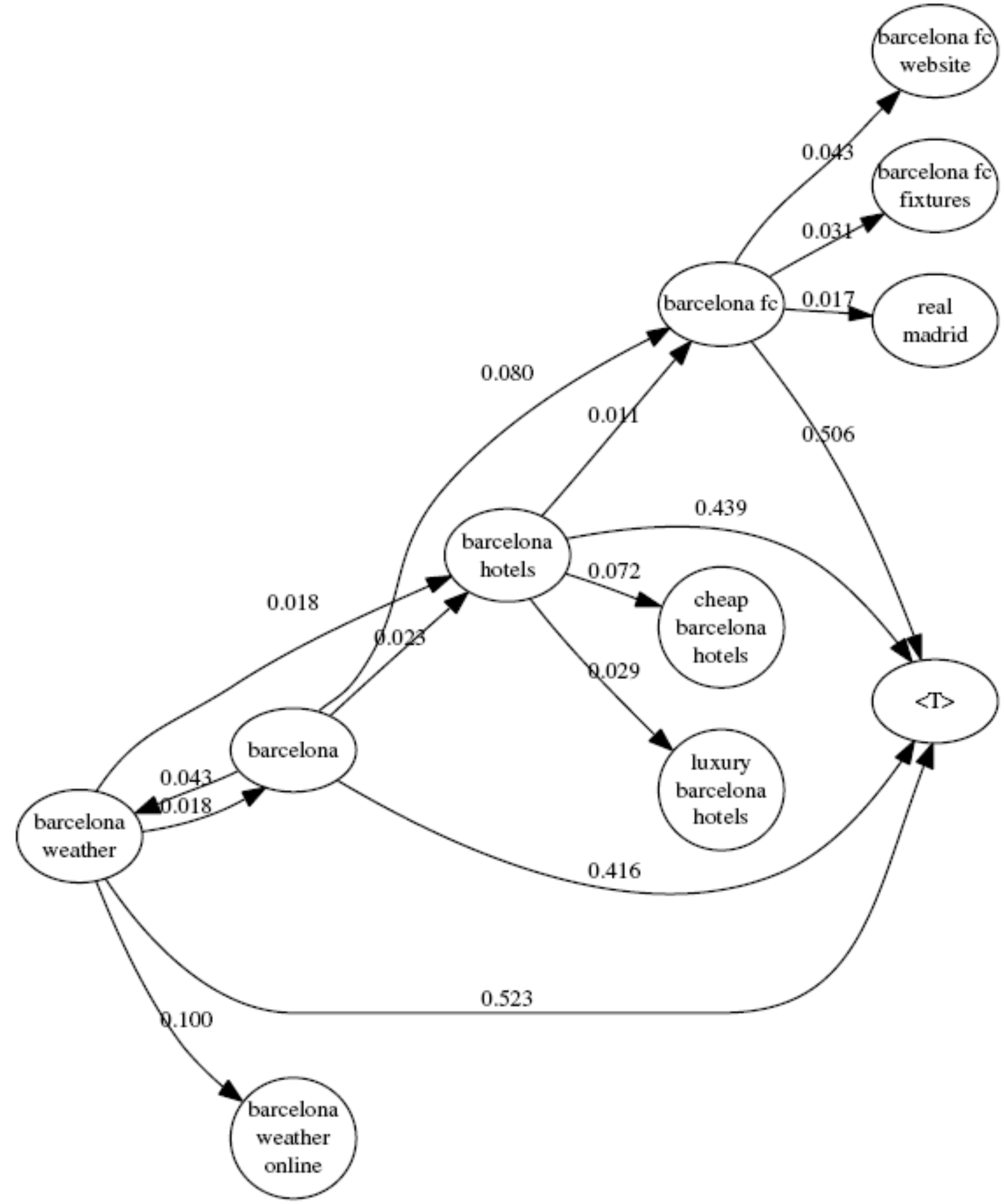
Open issues

- Sparsity of models:
  - query assistance services perform poorly or are not even triggered on long-tail queries
- Performance:
  - on-line process going in parallel with query answering



SoA: Query Flow Graph

- Query-centric approach
- Suggest queries by computing Random Walks with Restarts (RWRs) on the **query-flow graph (QFG)** by starting from the current user query



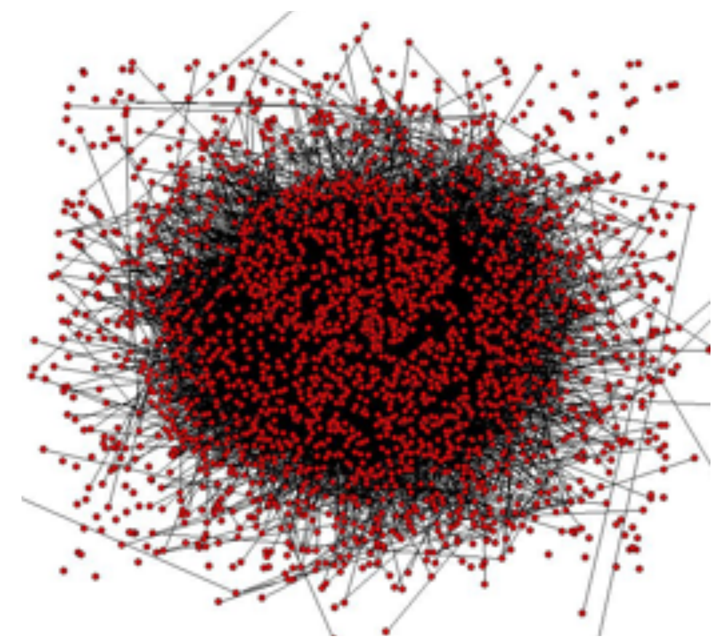
P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, S. Vigna: **The query-flow graph: model and applications.** CIKM 2008: 609-618

P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, S. Vigna: **Query suggestions using query-flow graphs.** WSCD, 2009

Query-centric suggestions

Computing RWRs on a huge graph, e.g., built from a QL recording 580,797,850 queries (from Y! us):

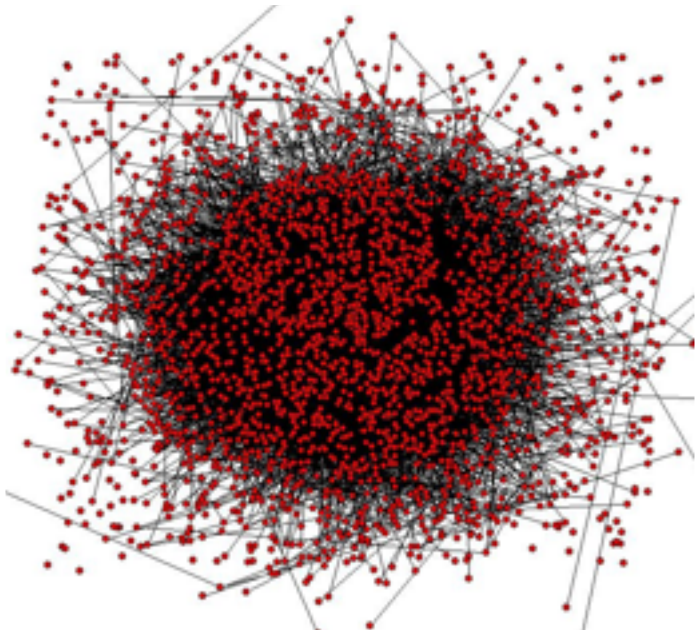
- |*V*| *28,763,637*
- |*E*| *56,250,874*



Query-centric suggestions

Computing RWRs on a huge graph, e.g., built from a QL recording 580,797,850 queries (from Y! us):

- $|V|$ *28,763,637*
- $|E|$ *56,250,874*



- *|{q: f(q)=1}| 162,221,967 (28%)*

Term-centric opportunities

But, in the same Y! QL:

- queries *580,797,850*
- Term occurrences *1,343,988,549*

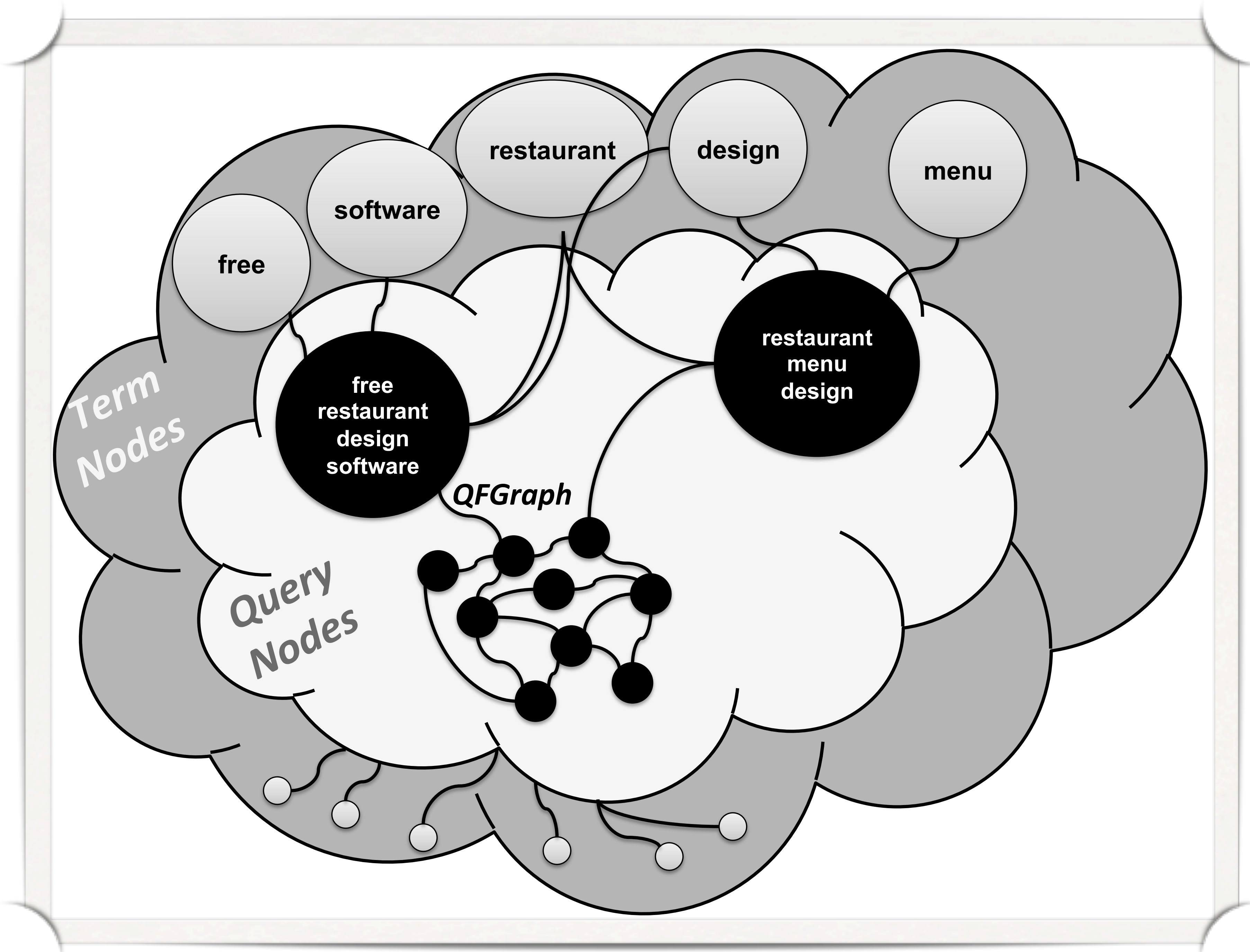
Term-centric opportunities

But, in the same Y! QL:

- queries *580,797,850*
- Term occurrences *1,343,988,549*

- $|\{t: f(t)=1\}|$ *5,099,145 (0.04%)*

The TQ-Graph



restaurant
design
menu
software
free
Term Nodes
free restaurant design software
restaurant menu design
QFGraph
Query Nodes

TQG effectiveness

- User study results comparing TQG and QFG effectiveness for two different testbeds (Y! US and MSN QLs).

TREC on MSN	useful	somewhat	not useful
TQGraph $\alpha = 0.9$	57%	16%	27%
QFG	50%	9%	42%

100 queries on Yahoo!	useful	somewhat	not useful
TQGraph $\alpha = 0.9$	48%	11%	41%
QFG	23%	10%	67%

Effectiveness on rare queries

- Anecdotal evidence

Query: lower heart rate

Suggested Query	Score
things to lower heart rate	$2.9\ e^{-14}$
lower heart rate through exercise	$2.6\ e^{-14}$
accelerated heart rate and pregnant	$2.9\ e^{-15}$
web md	$2.0\ e^{-16}$
heart problems	$8.0\ e^{-17}$

Query not occurring in the training log

Query: dog heat

Suggested Query	Score
heat cycle dog pads	$4.3\ e^{-10}$
what happens when female dog is in heat & a male dog is around	$4.0\ e^{-10}$
boxer dog in heat	$3.99\ e^{-10}$
dog in heat symptoms	$3.98\ e^{-10}$
behavior of a male dog around a female dog in heat	$3.95\ e^{-10}$

Query occurring twice in the training log

TQG pros

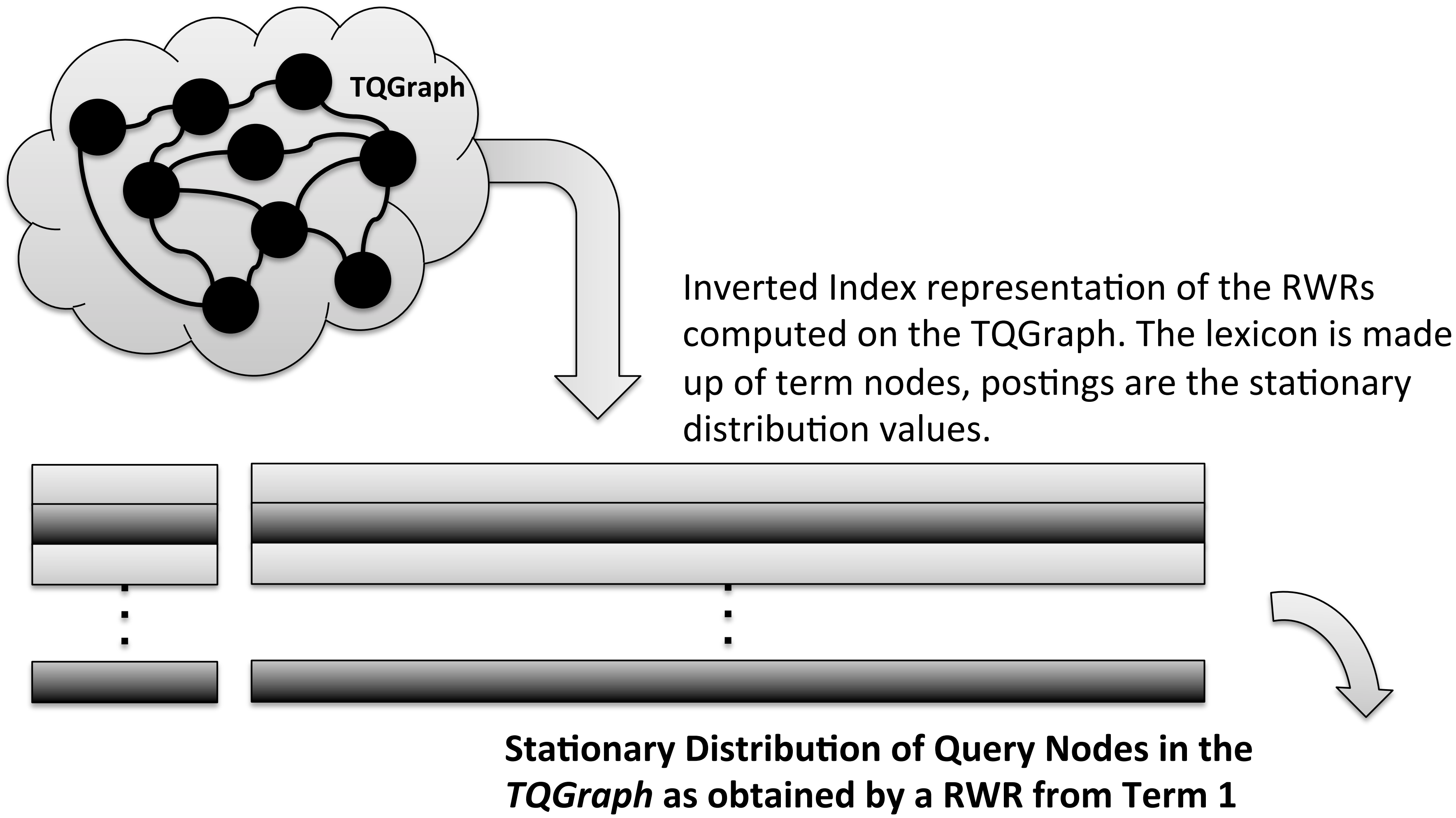
- provide query suggestions of quality comparable/better than QFG even for rare and unique queries
- several possible optimizations for achieving

TQG pros

- provide query suggestions of quality comparable/better than QFG even for rare and unique queries
- several possible optimizations for achieving

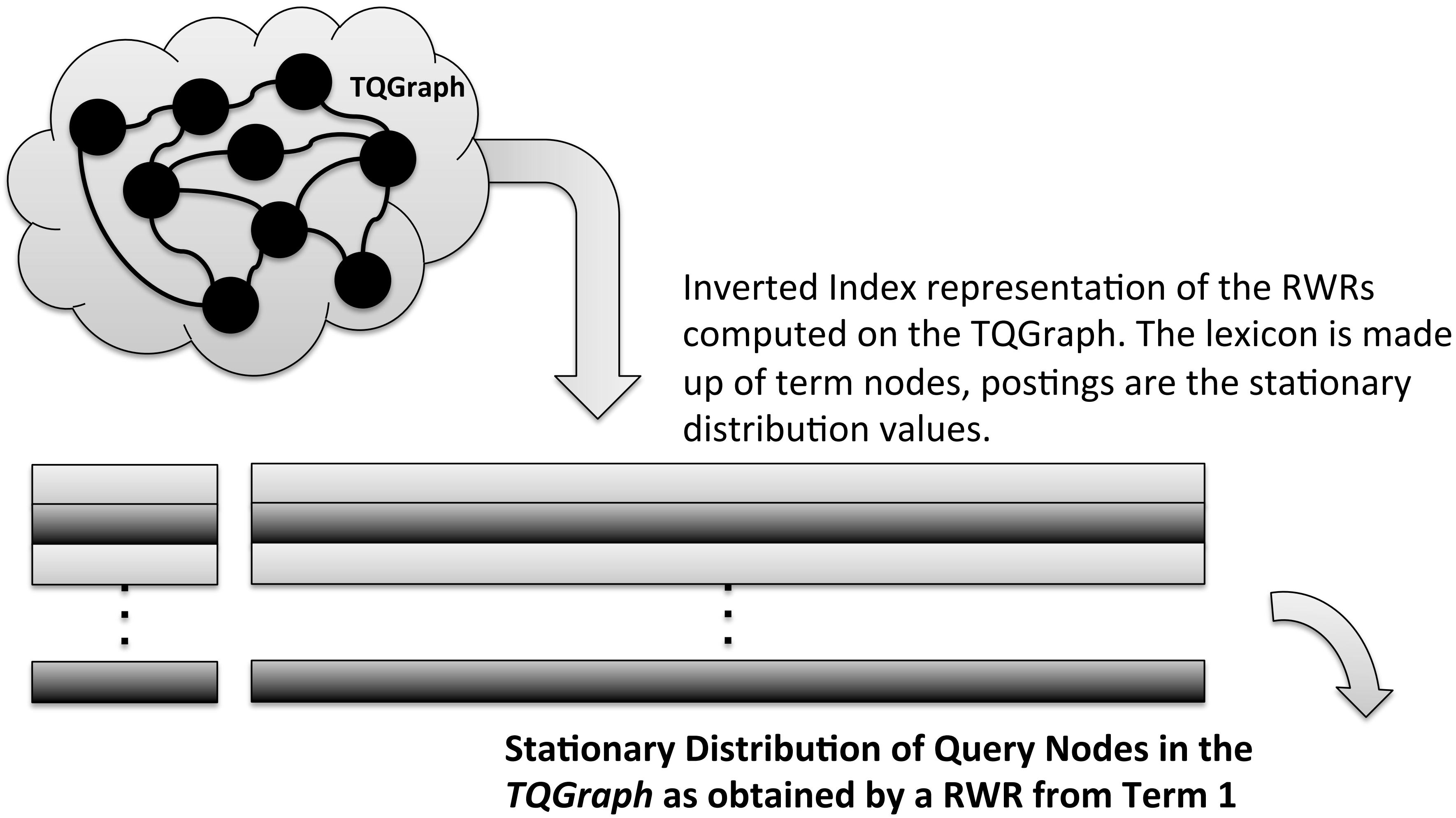
an efficient on-line query recommendation service

Indexing precomputed suggestions



- recommendations for an incoming query are computed by processing the posting lists associated with the terms in the query

Indexing precomputed suggestions



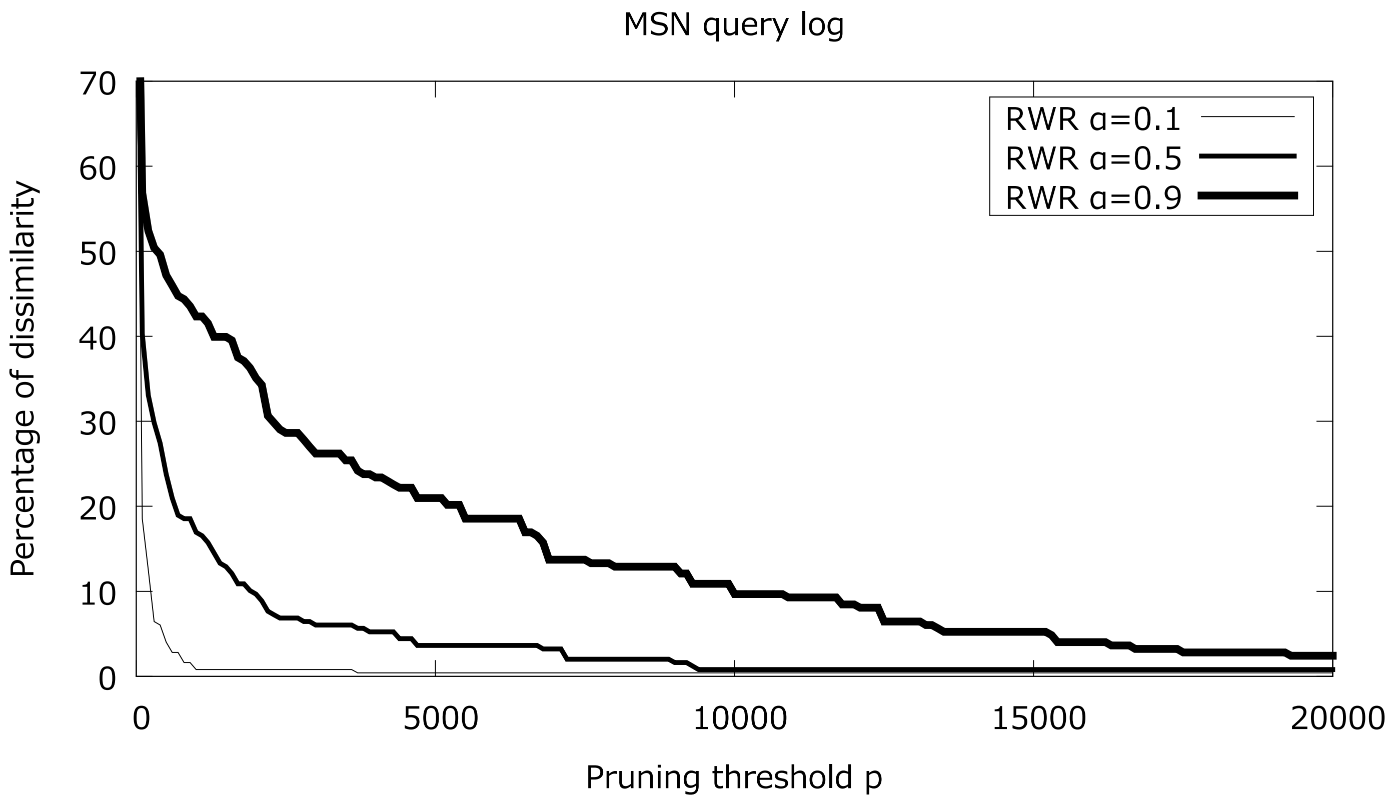
- recommendations for an incoming query are computed by processing the posting lists associated with the terms in the query

:) *O(|T|)* posting lists

:(*O(|Q|)* length of each posting list

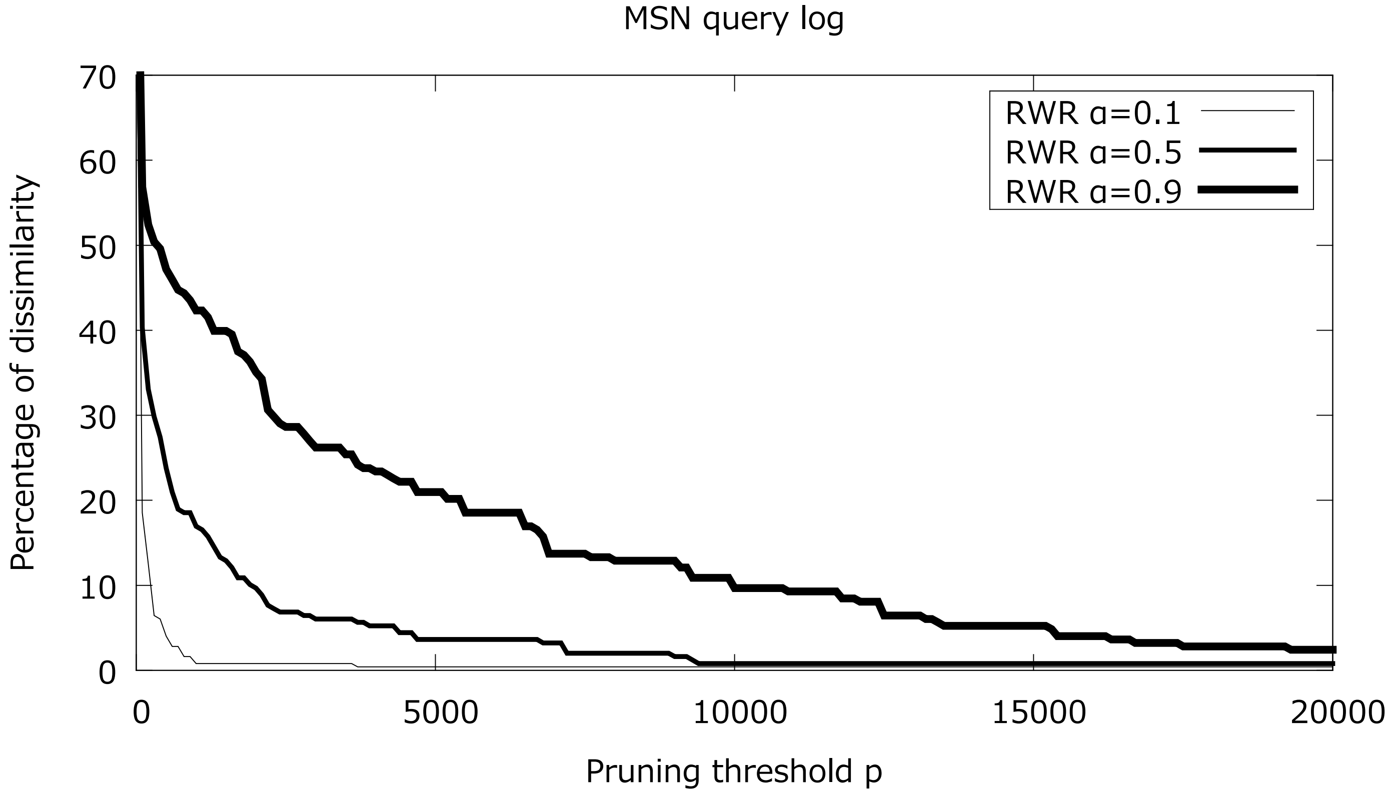
Pruning posting lists

- sort postings by probability and prune them at a reasonable threshold *p*, e.g. 20,000



Pruning posting lists

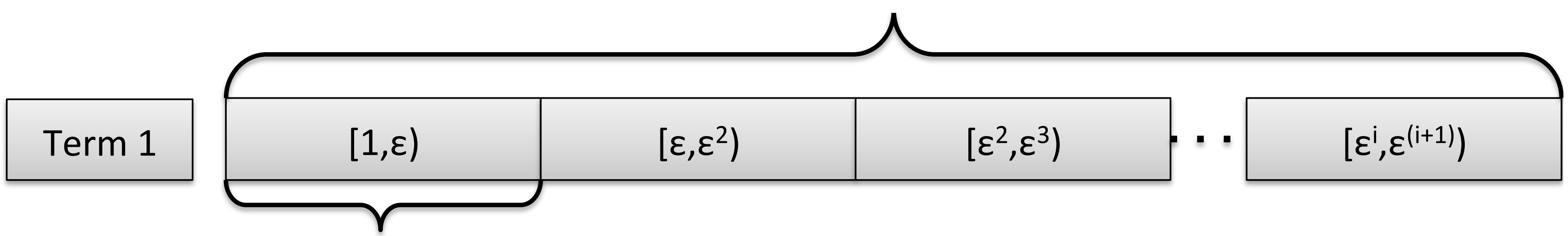
- sort postings by probability and prune them at a reasonable threshold *p*, e.g. 20,000



O(|T|) lists, each of size *O(p)* and no loss in quality!

Bucketing probabilities

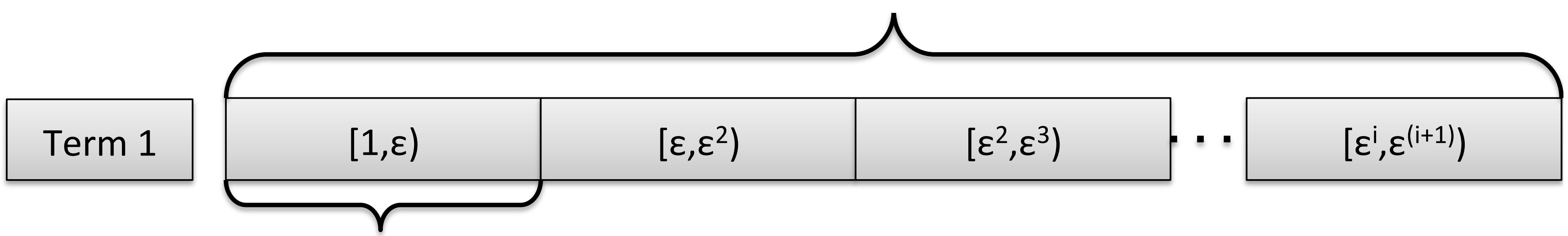
- Most space used for storing probabilities
- Given ε < 1, we can arrange postings in buckets implicitly coding the approximate probabilities



Within buckets queries are sorted by their IDs. Scores are approximated by the greatest bound, i.e. ε^i for all $i \geq 0$.

Bucketing probabilities

- Most space used for storing probabilities
- Given $\varepsilon < 1$, we can arrange postings in buckets implicitly coding the approximate probabilities

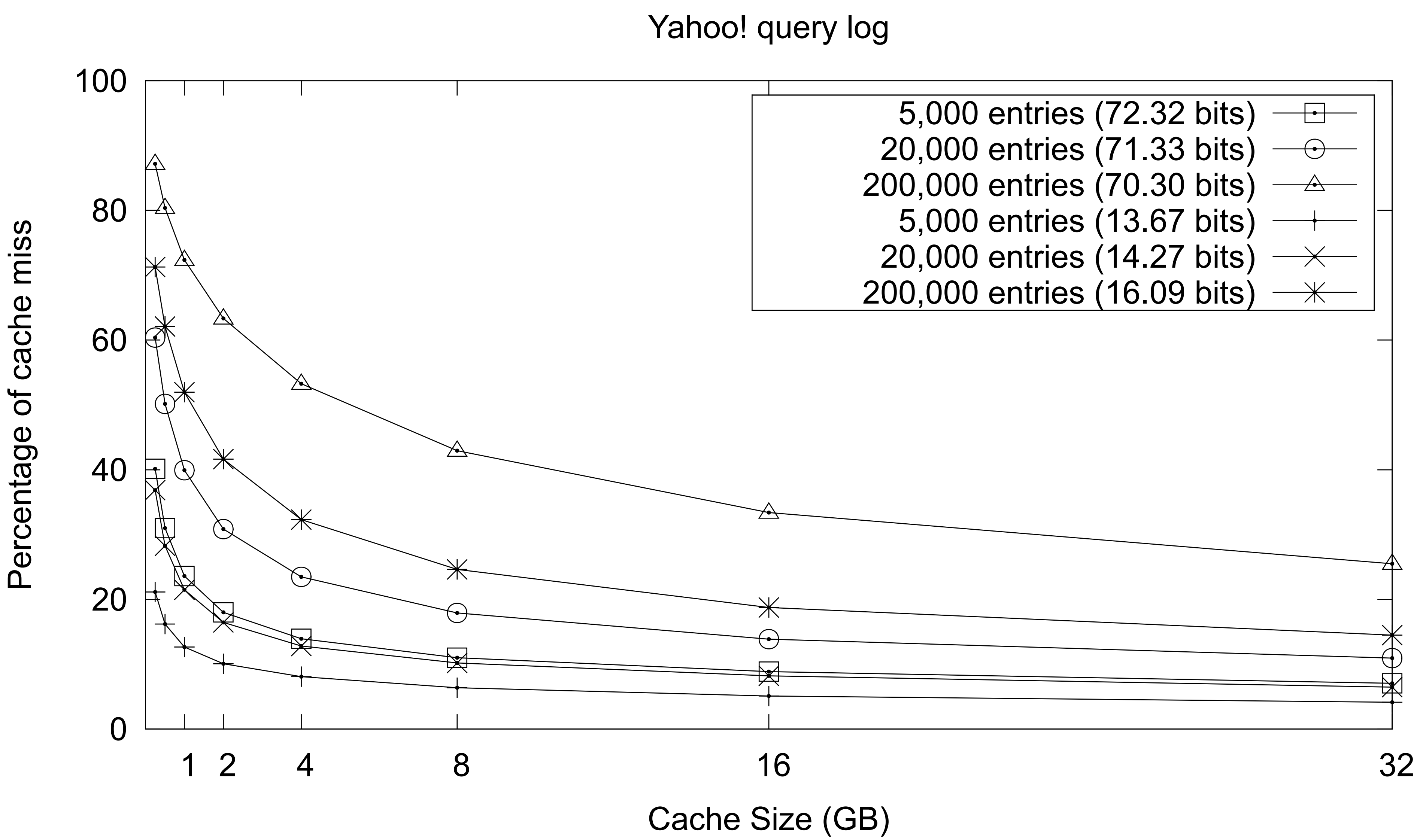


Within buckets queries are sorted by their IDs. Scores are approximated by the greatest bound, i.e. ε^i for all $i \geq 0$.

- Each entry coded with a few bits, e.g., 11-19 bits
- ~5x reduction!
- no loss in quality!

Caching posting lists

- achieving in-memory query suggestion



Conclusions

- TQG model to overcome limitations of current query recommenders
  - based on a principled, term-centric approach supporting rare and never-seen queries
- deployment with a efficient inverted index resulting in effectiveness comparable/better to SoA approaches
- the pruning, bucketing, caching techniques proposed constitute a independent contribution in the area of efficiency in large scale RWR computations
  - reduction of about 80% in the space occupancy w.r.t. uncompressed data structures
  - in-memory RWRs on huge graphs with 90+ % hit-ratio cache

Open Questions

- Is it possible to speed up computation of RWR from a “single” node?
- Is it possible to combine multiple RWRs in single iteration of the process?
- Other applications?
- Is there any benefit in using the softAND coefficient?
- Are there any other spectral method one could use for the problems I presented?

Questions

- **Fabrizio Silvestri**
ISTI - CNR, Pisa, Italy
fabrizio.silvestri@isti.cnr.it
http://hpc.isti.cnr.it/~fabriziosilvestri
http://google.it/search?q=fabrizio+silvestri