Query logs

Ilaria Bordino

Department of Computer and Systems Science
Sapienza University of Rome
Rome, Italy
and
Department of Information and Communication Technologies
Universitat Pompeu Fabra
Barcelona, Catalunya, Spain

April 14th, 2010

Mining query logs

- A query log contains information about the interaction of users with search engines.
    - The queries that users make
    - The results returned by search engines
    - The documents that users click in search results

Query log analysis: Applications

- Analyzing the interests of users and their searching behavior;
- Finding semantic relations between queries: useful to build query taxonomies;
- Using the user feedback to improve the quality of the results returned by search engines;
- Error correction
- Query suggestion: recommending related queries
- Improving advertising algorithms and helping advertisers select bidding keywords;

Query logs: privacy issue

- Query logs contain sensitive information about the users;
- Security breaches may occur even after anonymization operations have been applied and the data appears to be secure.
- The data must be anonymized without destroying the wealth of knowledge embedded in query logs;
- The problem has great importance for the entire research community.

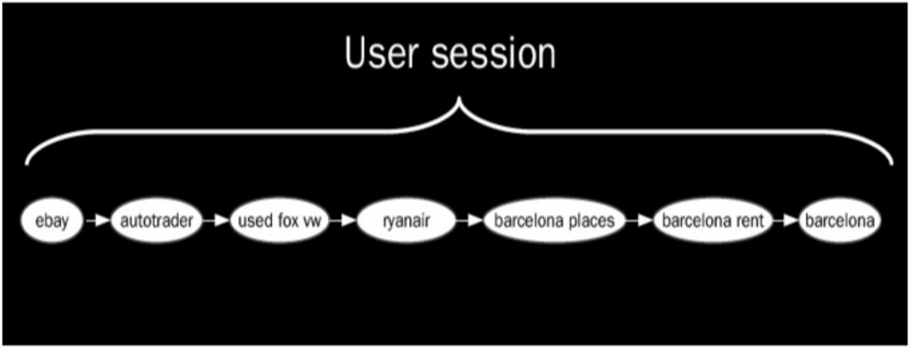
- A typical query log $\mathcal{L}$ is a set of records $\langle q_i, u_i, t_i, V_i, C_i \rangle$, where:
  - q_i is the query submitted by the user;
  - u_i is an anonymized identifier for the user who submitted the query;
  - t_i is a timestamp;
  - V_i the set of documents returned as results to the user;
  - C_i is the set of documents clicked by the user;
- Let $\mathcal{Q}$ be the set of queries, U the set of users and D the set of documents;
- Thus, $qi \in \mathcal{Q}, u_i \in \mathcal{U},$ and $C_i \subseteq V_i \subseteq \mathcal{D}$

- A **session** is defined as the sequence of the queries of one particular user within a specific time limit.
- More formally, a session $\mathcal{S}$ is a maximal ordered sequence

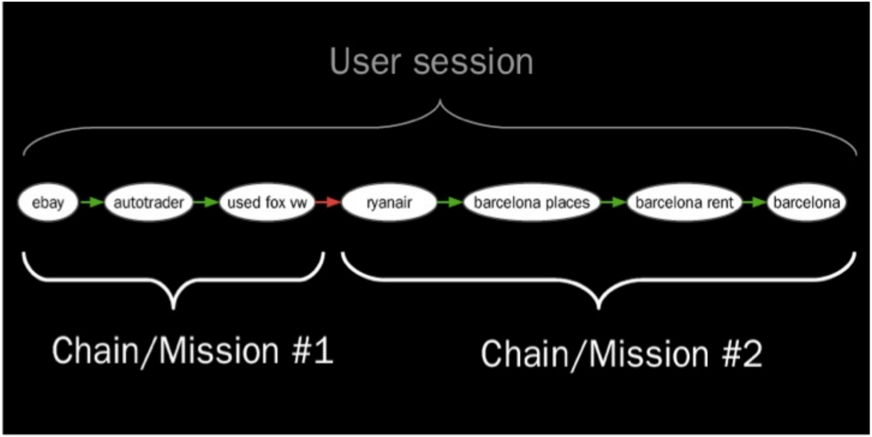
$$\mathcal{S} = \langle \langle q_{i_1}, u_{i_1}, t_{i_1} \rangle, \ldots, \langle q_{i_k}, u_{i_k}, t_{i_k} \rangle \rangle,$$

  where $u_{i_1} = \cdots = u_{i_k} = u \in \mathcal{U}$, $t_{i_1} \leq \cdots \leq t_{i_k}$, and $t_{i_{j+1}} - t_{i_j} \leq t_\theta$, for every $j = 1, 2, \ldots, k-1$;
- t_θ is a timeout threshold used for splitting query-log data into sessions: Typical value is 30 mins.
- A **supersession** is the sequence of sessions in which consecutive sessions are separated by time periods larger than t_θ.
- A **chain** is a topically coherent sequence of queries of one user. Also named as **mission** or **logical session**.

Example: Session

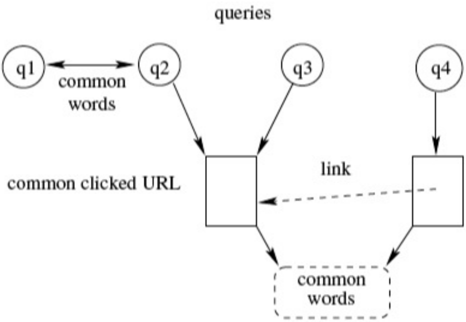


Example: Chains or Search Missions



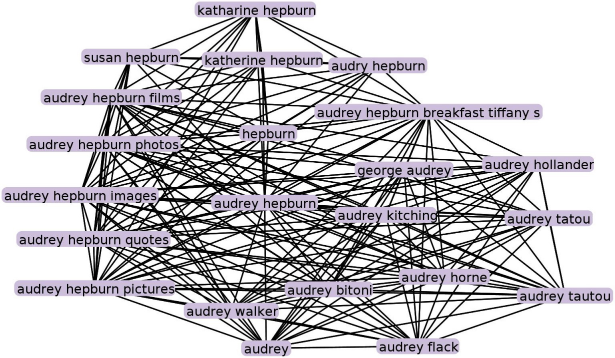
Graphs from query logs

- Graphs from search engine queries (Baeza-Yates, SOFSEM 2007)
- Idea: explore relations between queries based on different sources of information like words in the text of the query, clicked URLs in their answers, as well as their links or terms.



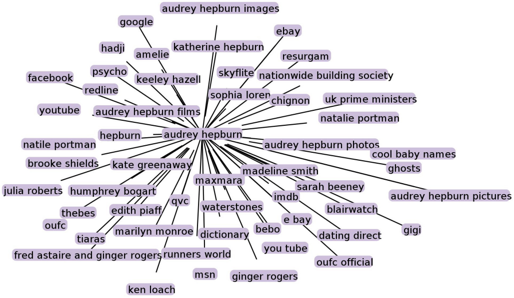
Word graph

- A node represent a query
- An undirected edge connects two nodes iff the text representations of the corresponding queries have non empty intersection;
- Node weight: number of occurrences of the query in the log; edge weight: number of common terms in the texts of the two queries.



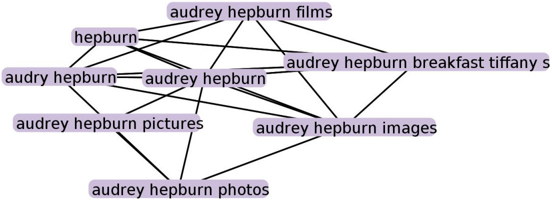
Session graph

- A node represent a query
- There is a directed edge between two queries iff they were submitted within the same session and the first happened before the second.
- Node weight: number of sessions which the query appears in; edge weight: number of sessions for which the edge condition is satisfied.



Click graph

- A node represent a query
- An undirected edge connects two nodes iff their sets of clicked answers have non empty intersection;
- Node weight: number of occurrences of the query in the log; edge weight: cosine similarity of the URL vectors associated with the two queries.



Link Graph

- **URL Link Graph:** Nodes represent queries; node weights: number of occurrences of the query in the log;

- There is a directed link from one query to another one iff there is at least one link from one page in the set of clicked results associated with the first query to a page that has been clicked for the second query. Edge weight: number of links of this kind;

URL Term Graph

- Nodes represent queries; node weights: number of occurrences of the query in the log;
- There is a directed edge connecting two queries iff there are at least a certain number of common terms in the text representations of one page belonging to the result set of the first query and a page clicked for the second query;
  Several possible choices for extracting a set of terms from every clicked URL:
  - Full text content of the page (after deleting HTML tags and stopwords);
  - Text snippet generated by the search engine for that page;
  - A subset of the text content of the URL (e.g., title, contents);
  - Anchor text in the links that point to the URL;
  - A combination of the above

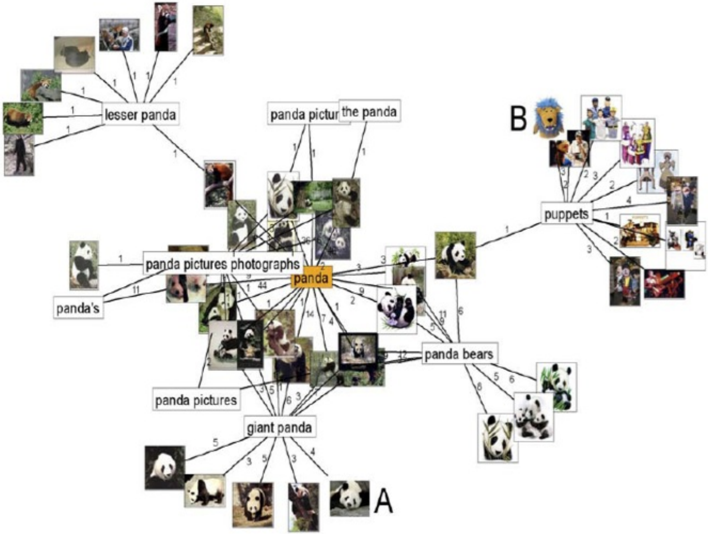
Graph from search engine queries: Applications

- Baeza-Yates and Tiberi: finding multi-topical documents
- Main idea: edges with low weight are most likely caused by multi-topic documents, like e-commerce sites;
- Low-weight edges are considered as voters for the documents shared by the two corresponding queries;
- The more votes a document receives, the more multi-topical it is.
- Huge number of potential applications: detecting polysemic or synonym words, clustering queries, query taxonomy.

Click graphs

- A click graph is a query-document bipartite graph;
- The node set is divided into two partitions: queries and documents;
- A query is connected to the documents that were clicked within the set of results returned by the search engine for that query;
- Applications: Suggesting related queries; finding high-quality results for a query; annotating documents with query-based descriptions; finding a set of diverse queries that cover different aspects of the original query.

Click graph: example



Random walks on the click graph

- Craswell and Szummer: random walk model on the click graph
- Application: ranking documents to queries
- Query-document pairs are soft relevance judgements
- Problems: noise, sparsity
- In the random-walk model, relevant documents may be ranked highly even if no previous user has clicked on them for a query;

Click graph: applications

- Query-to document search
- Query-to-query suggestion
- Document-to-query annotation
- Document-to-document relevance feedback

Random walk on the click graph

- The random walk on the click graph models a user who issues queries and clicks on documents according to the edge weights of the graph
- The weight of an edge in the click graph is the number of clicks for the query-document pair
- The weights are normalized to represent transition probabilities
- The random walk is performed by traversing the graph according to these probabilities
- Typical random walk models, e.g., PageRank, consider **forward** random walks
- Here the idea of **backward** walks is considered

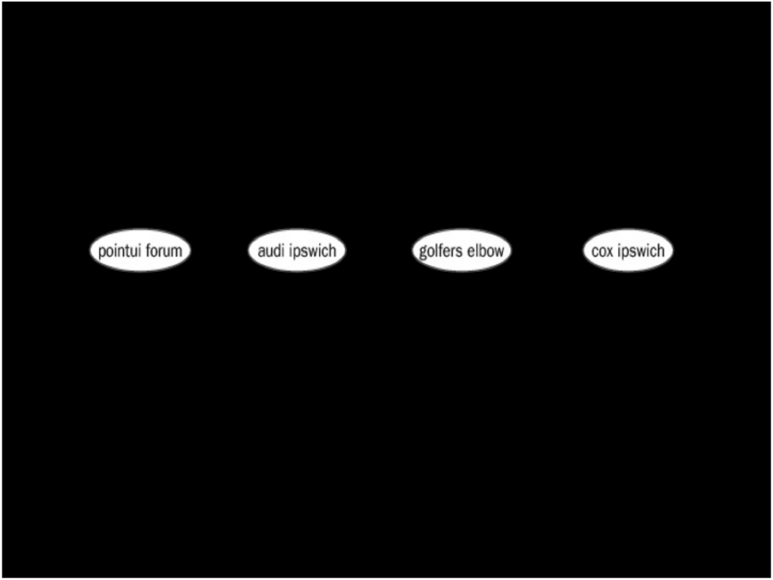
Backward random walk on the click graph

- Use Bayes to compute the probability that the walk started at node k after t steps is at node j.
- Difference between forward and backward random walk:
- The stationary distribution of the forward walk is independent from the initial distribution
- The limiting distribution of the backward walk is uniform.
- Running the backward walk for a small number of steps allows meaningful differentiation between the nodes in the graph
- Application: ad-hoc search in image databases

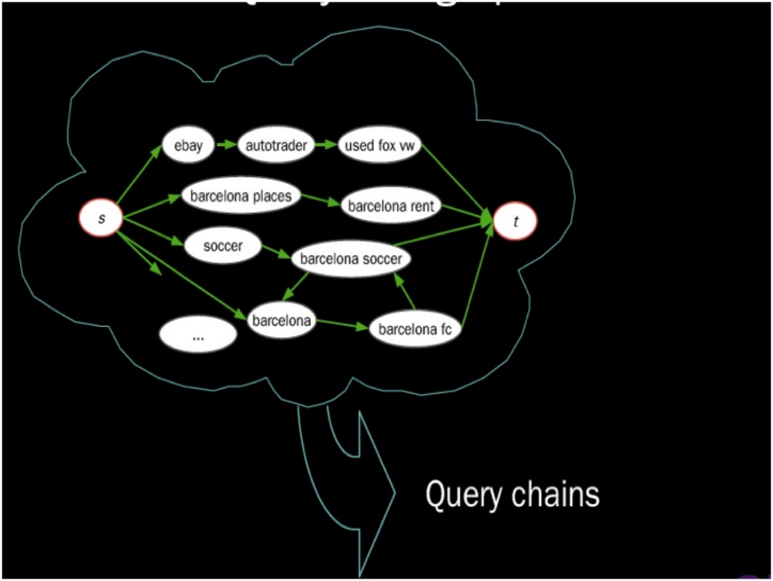
View graph and anti-click graph

- Alternatives to the click graph are the **view graph** and the **anti-click graph**:
- The view graph generalizes the click graph: queries are connected to documents whose URLs have been viewed in the result list returned to the user;
- The view graph is more noisy than the click graph because it does not contain any user feedback;
- The anti-click graph: a query is connected to documents that appear within its list of top-ranked results, but were not clicked by the users who submitted the query;
- The idea is to capture the negative feedback that users give to the top-ranked results when they ignore them and click on results ranked below.
- The graphs can also be defined on *hosts*;
- Application: Spam detection (Spam sites try to be in the result lists of different queries);

The query-flow graph



The query-flow graph



The query-flow graph

- The query-flow graph models user behavioral patterns and query dependencies;
- The focus is on the sequentiality of similar queries;
- The fundamental two dimensions that are taken into consideration are the temporal order of queries and their similarity;

The query-flow graph

- The nodes are all the queries in the log
- A directed edge between two queries q_i and q_j has a weight $w(q_i, q_j)$ representing the probability that the two queries appear in a given order and are part of the same search mission;
- When $w(q_i, q_j)$ is high, we may think of q_j as a typical reformulation of q_i, thus a step ahead towards the successful formulation of a specific information need.

The query-flow graph: definition

- The query-flow graph is a directed graph $G_{qf} = (V, E, w)$ where:
  - $V = Q \cup \{s, t\}$: the set of nodes includes all the distinct queries submitted to the search engine and two special nodes, s and t, which representing the starting state and the terminal state of any chain;
  - $E \in V \times V$ is a set of directed edges;
  - $w \rightarrow (0, 1]$ is a weighting function that associates every edge with a weight representing the probability that the two queries are part of the same chain;
  - Even if a query has been submitted multiple times to the search engine, possibly by multiple users, it is represented by a single node in the query-flow graph;
  - The existence of an edge (s, q_i) represents that q_i may be potentially a starting query in a chain;
  - The existence of an edge (q_i, t) represents that q_j may be potentially the terminal query in a chain;
  - The edge weights are obtained by applying a machine-learning algorithm.

Building the query-flow graph

- Input: a set of sessions extracted from the log of a search engine;
- The set of sessions can be easily constructed by sorting the queries by user-id and by timestamp, and then splitting them by using a time threshold.
- Key issue: choose the weighting function to be used to assign edge probabilities;
- First step: we *tentatively* connect two queries by an edge if there is at least one session in which they are consecutive
- Next, we use a set of features to associate each edge with a probability $w(q_i, q_j)$

Computing the edge weights

- For each edge we compute a set of features that aggregate various kinds of information:
- time difference in which the queries are submitted
- textual similarity of the two queries
- number of sessions in which they appear
- Training data: random set of edges manually labeled by human assessors ;

Computing edge weights: the features

Textual features

- Cosine similarity
- Jaccard coefficient
- Set of intersection

Computing edge weights: the features

Session features

- Number of sessions
- Average session length
- Average number of clicks in the sessions
- Average position of the queries in the sessions

Computing edge weights: the features

Time-related features

- Average time difference between the queries in the sessions in which the transition appears
- Sum of reciprocals of time difference over all appearances of the query pair

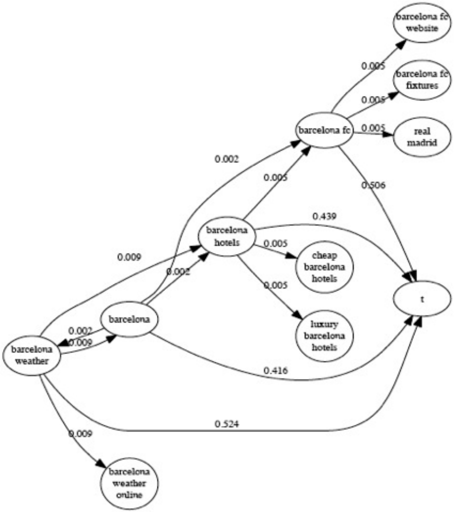
Computing the edge weights: learning the function

- A machine-learning algorithm is used to predict edge labels
- Two distinct classification subproblems:
  - Pairs of queries appearing together only once: logistic regression
  - Pairs of queries appearing together more than once: rule-based model
  - The model is used to assign a weight to each edge;

Computing the edge weights

- Weights are normalized (Query-flow graph as a stochastic matrix)
- Add starting and terminal state
- Add an edge from node s to the first query of each session
- Add an edge from the last query of each session to node t

Sample query-flow graph



References

- R. Baeza-Yates.
  Graphs from search engine queries.
  In *Theory and Practice of Computer Science (SOFSEM)*, volume 4362 of *LNCS*, pages 1–8, Harrachov, Czech Republic, January 2007. Springer.
- R. Baeza-Yates and A. Tiberi.
  Extracting semantic relations from query logs.
  In *KDD'07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 76–85, New York, NY, USA, 2007. ACM.
- D. Beeferman and A. Berger.
  Agglomerative clustering of a search engine query log.
  In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416, New York, NY, USA, 2000. ACM.
- N. Craswell and M. Szummer.
  Random walks on the click graph.
  In *SIGIR*, 2007.

References (II)

- C. Castillo, C. Corsi, D. Donato, P. Ferragina, and A. Gionis.
  Query-log mining for detecting spam. In *AIRWeb'08: Proceedings of the 4th Workshop on Adversarial Information Retrieval on the Web*, ACM International Conference Proceeding Series, 2008.
- P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna.
  The query-flow graph: model and applications.
  In *CIKM'08: Proceeding of the Information and Knowledge Management Conference*, pages 10 pp.+, October 2008.
- P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna.
  Query suggestions using query-flow graphs.
  In *WSCD*, 2009.
- J.-R. Wen, J.-Y. Nie, and H.-J. Zhang.
  Clustering user queries of a search engine.
  In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 162–168, New York, NY, USA, 2001. ACM.