

Query Suggestions

Debapriyo Majumdar
Information Retrieval
Indian Statistical Institute Kolkata

Search engines



User needs some
information



A search engine tries
to bridge this gap



Assumption: the required
information is present
somewhere

How:

- User *expresses* the *information need* – in the form of a *query*
- Engine returns – list of documents, or by some better means

Search queries

Navigational queries

- We know the answer (which document we want), just using a search engine to navigate
 - tendulkar date of birth → Wikipedia / Bio page
 - serendipity meaning → dictionary page
 - air india → simply the URL: www.airindia.in
 - In a people database, typing the name → the record of the person we are looking for
- Straightforward to formulate such queries
- Query suggestion is primarily for saving time and typing

Simple information / calculation related queries

- 100 USD in INR → Currency conversion requested
- kolkata weather → weather information requested

Complex informational queries

- We do not know the answers
- Hence, we may not express the question perfectly

Why query suggestion?



User needs some information

User: information need → a query in words (language)



A search engine tries to bridge this gap

We may not know what information is available, or in what form the it is present, or we cannot express it well

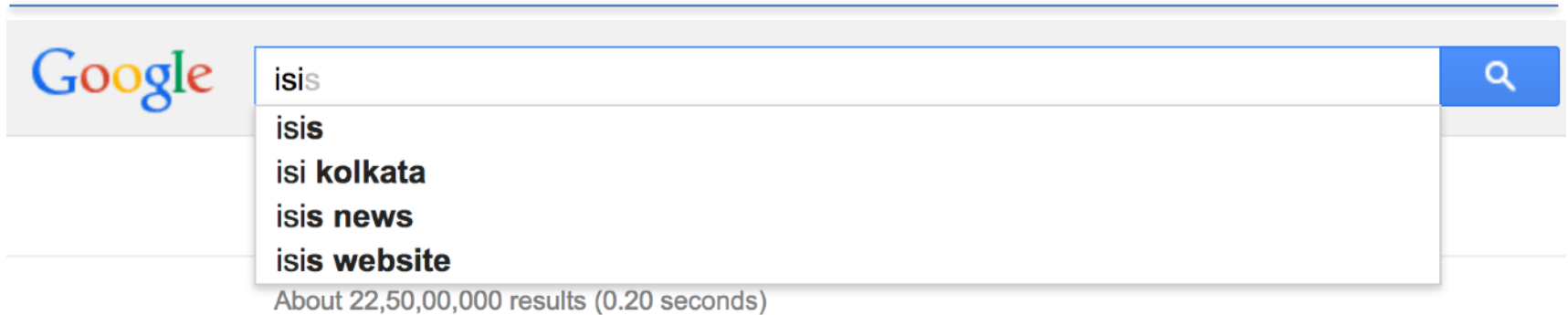
If you know what exactly you want, it's easier to get it



Assumption: the required information is present somewhere

Engine processes the documents

Interactive query suggestion

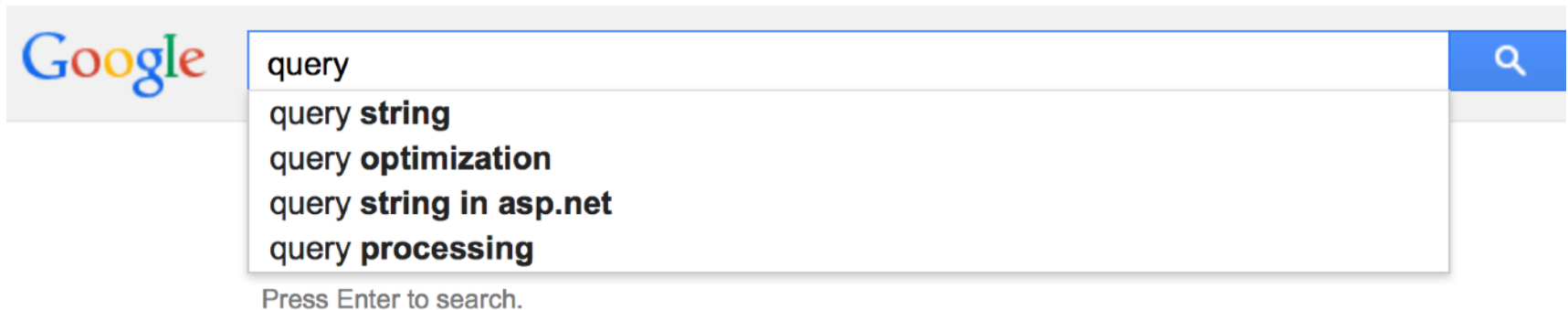


Google

isis

- isis
- isi **kolkata**
- isis **news**
- isis **website**

About 22,50,00,000 results (0.20 seconds)

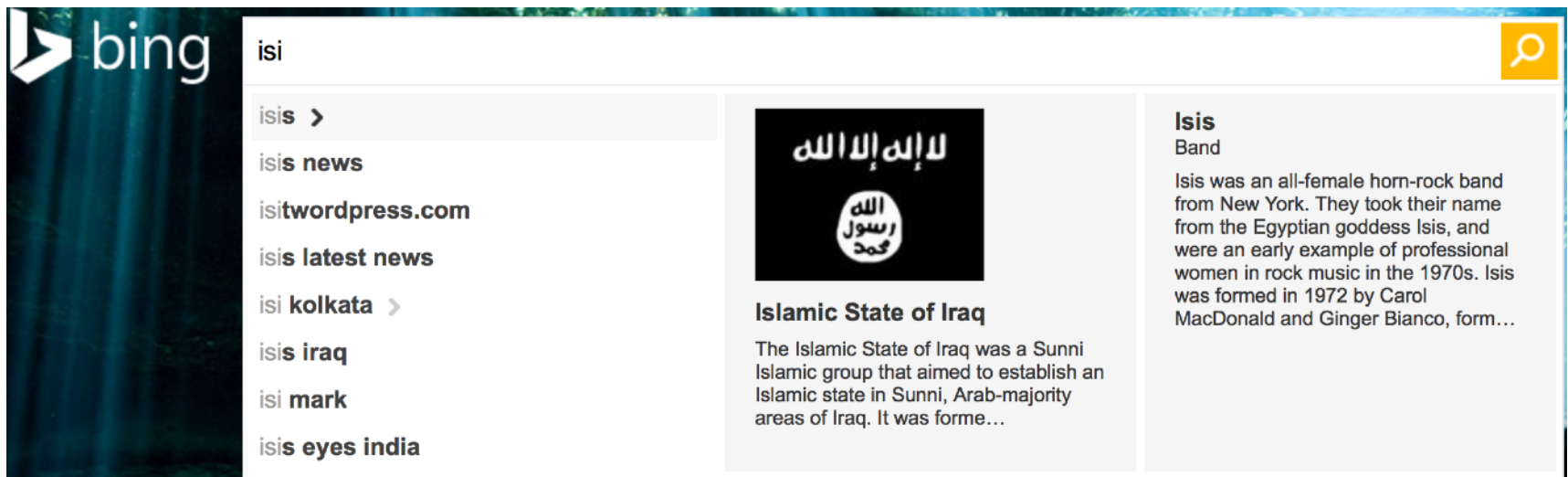


Google

query

- query **string**
- query **optimization**
- query **string in asp.net**
- query **processing**


Press Enter to search.



bing

isi

- isis >
- isis **news**
- isitwordpress.com
- isis **latest news**
- isi **kolkata** >
- isis **iraq**
- isi **mark**
- isis **eyes india**



Islamic State of Iraq

The Islamic State of Iraq was a Sunni Islamic group that aimed to establish an Islamic state in Sunni, Arab-majority areas of Iraq. It was forme...

Isis
Band

Isis was an all-female horn-rock band from New York. They took their name from the Egyptian goddess Isis, and were an early example of professional women in rock music in the 1970s. Isis was formed in 1972 by Carol MacDonald and Ginger Bianco, form...

Variations of the problem

- Scenario 1: suggesting alternative queries for a query
 - Input: one query (possibly a complete query)
- Scenario 2: helping the user to quickly complete a query
 - Input: a few characters typed by the user
 - Can be a complete query, but also may be extended
- Scenario 3: suggesting queries as possible correction
 - Input: one possibly complete query
- For all the variations, the output is a list of suggested queries

- We will see: these scenarios are not technically exclusive
- Methods applicable for one of the scenarios can also fit another

The different variants are not technically exclusive

- Primarily two kinds of algorithms in literature
- A: given a query, suggest a ranked list of alternatives
- B: given an incomplete query, fast prefix search
- (A) can be applied even if the input query is incomplete
 - Incomplete query = a few complete words + incomplete word
 - Approach 1: Find suggestions for the complete part and filter only those which match the incomplete part as well
 - Approach 2: Use some prefix search to expand the incomplete word into a few possible completions, and then apply (A) on all of these
- Even in interactive query suggestion, whether the user has completed typing a query is not clear often

With or without using query logs?

- Without query logs
 - Custom search engines in the enterprise world
 - Small scale search, not so much of log, cold start (Google scholar did not have a query suggestion until recently)
 - Site search (search within a particular website)
 - Desktop search – only one user
 - Legally restricted environment – if you cannot expose other users' queries even anonymously
 - Method: have to suggest collection of words that are likely to form meaningful queries matching the partial query typed by the user so far
- With query logs
 - The log has “actual” queries submitted by users (better formed)
 - Advantage of having click data

Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. "Query recommendation using query logs in search engines." In *EDBT 2004*, pp. 588-596. Springer, Berlin, Heidelberg.

Query suggestion using query similarity

Outline

Problem: given a query, output a ranked list of suggestions

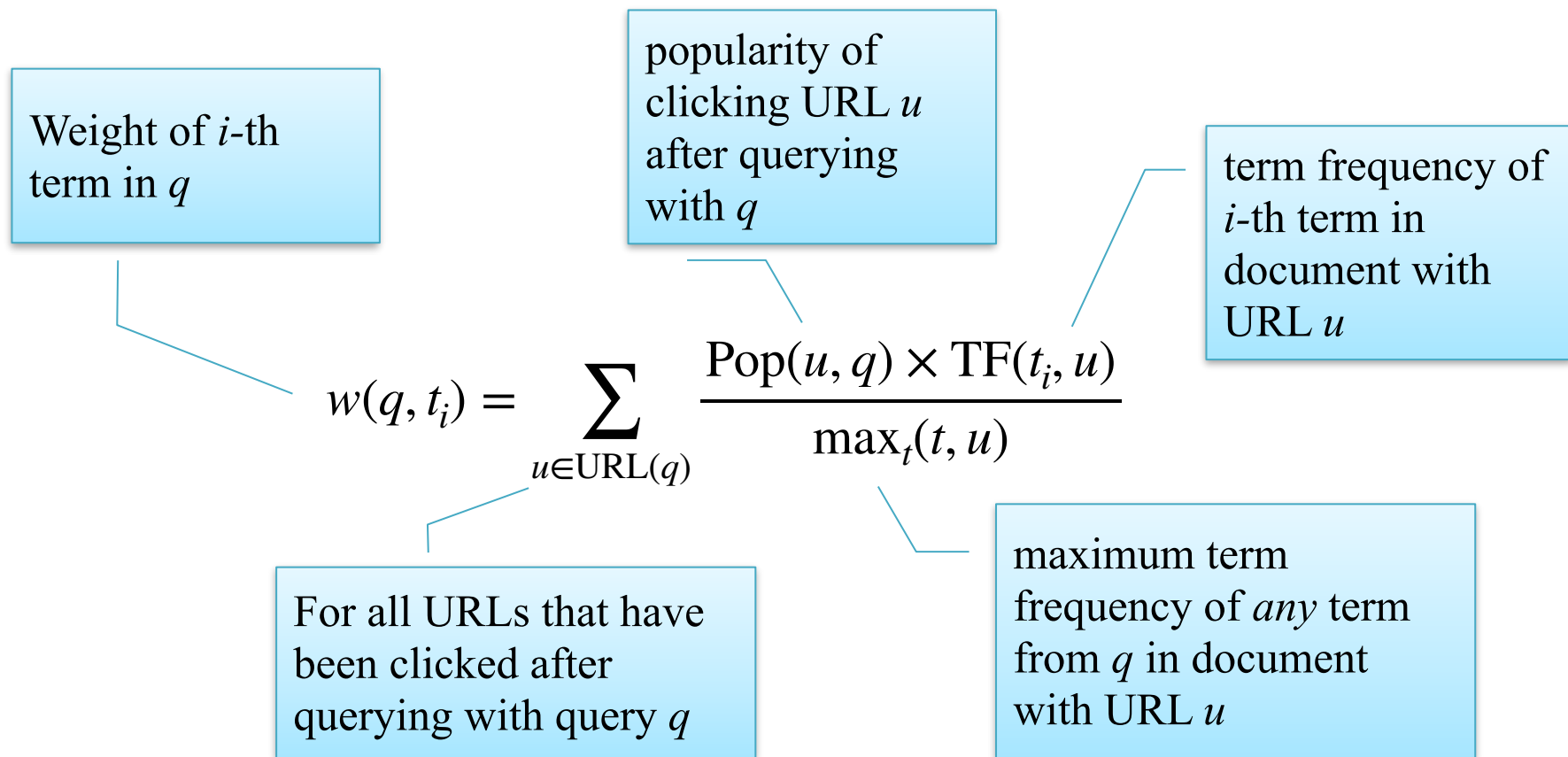
Preprocessing (offline)

- Represent queries as **term weighted** vectors
- Cluster queries using **similarity** between queries
- **Rank** queries in each cluster

Query time (online)

- Given the user's query q
- **Find cluster** C in which q should belong
- Suggest top k queries in cluster C
 - Based on their **rank** and **similarity** with q

Query term weight model



Query similarity is computed using cosine similarity
Cluster queries using this similarity (k -means, CLUTO)

Query support and ranking

- What is a good query?
 - Several users are submitting the same query
 - For some queries, more returned documents are clicked by some user
 - For some other queries, less returned documents are clicked
 - If no result is ever clicked \implies Not a good query at all
 - Query goodness \sim fraction of returned documents clicked by some user
 - A global score \implies rank within cluster

Final ranking at query time

- Rank using a combination of query support and similarity with the given query
- Exact parameters for this combination is left as flexible

Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. "The query-flow graph: model and applications." In *CIKM 2008*, pp. 609-618.

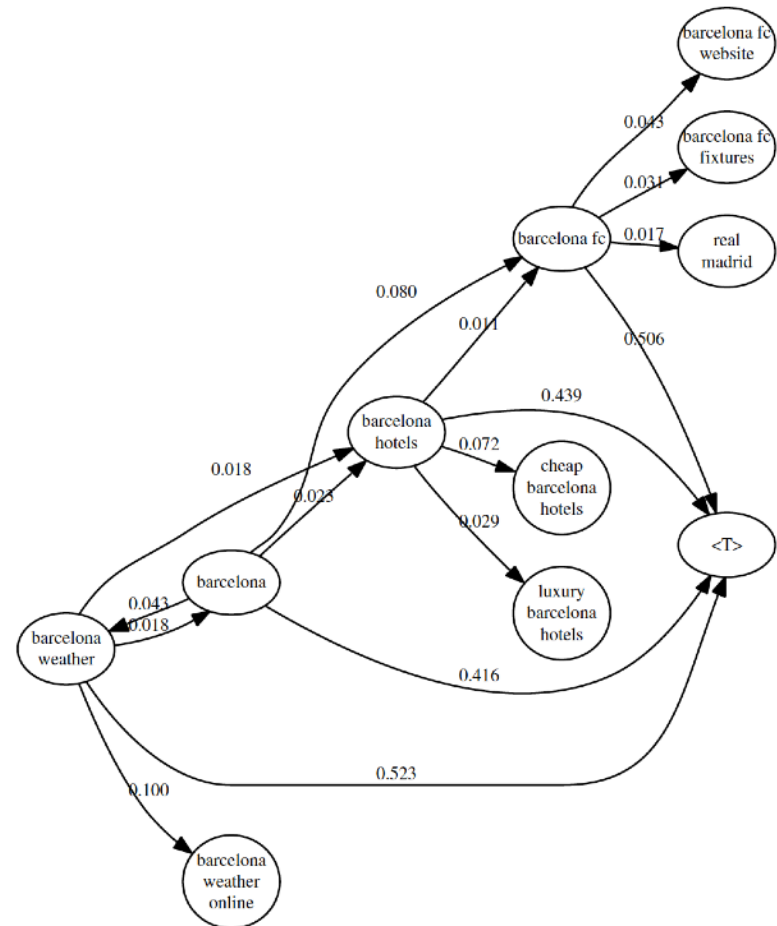
Query suggestion using query-flow graph

Key idea

- Assumptions:
 - User is happy when the information need is fulfilled
 - User keeps reformulating the query until satisfied
- Goal of the system: suggest the query that the user is likely to ultimately submit after reformulation
 - Learn from the query log
- How to know if one query in the query log was a reformulation of another?
 - A session: given a time threshold t , a sequence of queries by one user such that gap between any two consecutive queries is $< t$
 - Chain: a sequence of queries representing the same information need
"brake pads"; "auto repair"; "auto body shop";
"batteries"; "car batteries"; "buy car battery online"
 - A session may not ensure the same information need
- Need to *identify* query chains within the *same session*

The query-flow graph

- Query-flow graph:
 $G_{\text{qf}} = (V, E, w)$
- Vertices are queries in the query log
- An edge from query q_1 to query q_2 if q_2 appear in some *user session* after q_1
- Edge weights (w): weights should be high for two queries in the same *chain*, low otherwise
 - Two approaches proposed



Example of query-flow graph

Picture source: the paper

Approach 1: Learn chaining probability

- Extract several features for each edge
 - Textual features: cosine similarity, Jaccard coefficient, size of intersection, etc.
 - Session features: number of sessions in which the two queries co-appear, and some others.
 - Time related: average time difference between the queries, etc.
 - Total 18 features, exact details are not provided!
- Training data: took a sample of pairs of queries in the same session, manually labeled whether *same chain* (same information need)
- A *logistic regression* model (after testing many others) to learn the feature weights

Approach 2: query reformulation as a Markov chain

- The weight of the edge $q_1 \rightarrow q_2$ is given by the within – session reformulation probability of q_2 from q_1

$$w(q_1, q_2) = P(q_1 \rightarrow q_2) = P_{\text{session}}(q_2 | q_1) = \frac{f(q_2, q_1)}{f(q_1)}$$

Probability of q_2
appearing after q_1 in a
session

Number of
occurrences of q_2
appearing followed
by q_1 in all sessions

Number of occurrences
of the query q_1 in all
sessions

Query recommendation for a query q

- Simple approach: for input query q , suggest the ranked list $\{q'\}$ with highest $w(q, q')$
 - Issue: tends to drift towards *popular* queries, outside the topic
- Approach 2: similar to personalized PageRank
- Random walk score for other queries q' *relative* to a query q
 - Start at query q
 - Random walk with probability $\beta = 0.85$
 - Teleport back to q with probability $1 - \beta = 0.15$
 - Compute the stationary distribution using power iteration
 - For each query q' , denote the score relative to q by $s_q(q')$
- Similarly, also compute the standard PageRank score $r(q')$
 - Start at any random query, and teleport randomly to any query

Query recommendation for a query q

- Precomputation:
 - For all queries q' , standard random walk score $r(q')$
 - For all queries q' with an edge from q , the relative random walk score $s_q(q')$
- Online: when the query q comes
 - Rank other queries q' by

$$\frac{s_q(q')}{\sqrt{r(q')}}$$

- Experimental trial and error
- Just using $s_q(q')$ prefers queries q' which are just popular
- Normalizing (dividing) by $r(q')$ penalizes those queries too much