Modern Information Retrieval

Chapter 11

Web Retrieval

with Yoelle Maarek

A Challenging Problem
The Web
Search Engine Architectures
Search Engine Ranking
Managing Web Data
Search Engine User Interaction
Browsing
Beyond Browsing
Related Problems

Introduction

- The Web
  - very large, public, unstructured but ubiquitous repository
  - need for efficient tools to manage, retrieve, and filter information
  - search engines have become a central tool in the Web
- Two characteristics make retrieval of relevant information from the Web a really hard task
  - the large and distributed volume of data available
  - the fast pace of change

A Challenging Problem

- Main challenges posed by Web are of two types
  - **data-centric:** related to the data itself
  - **interaction-centric:** related to the users and their interactions
- Data-centric challenges are varied and include
  - distributed data
  - high percentage of volatile data
  - large volume of data
  - unstructured and redundant data
  - quality of data
  - heterogeneous data

A Challenging Problem

- Second class of challenges — interaction-centric
  - expressing a query
  - interpreting results
- User key challenge
  - to conceive a good query
- System key challenge
  - to do a fast search and return relevant answers, even to poorly formulated queries

The Web

- Many studies investigated the Web in specific countries
  - many properties and characteristics of a subset of the Web are valid for the global Web
- Still, no full understanding of the Web and its dynamics

Characteristics

- Measuring the Internet and the Web is difficult
  - highly dynamic nature
  - more than 778 million computers in the Internet (Internet Domain Survey, October 2010)
  - estimated number of Web servers currently exceeds 285 million (Netcraft Web Survey, February 2011)
  - Hence, there is about one Web server per every three computers directly connected to the Internet

Characteristics

- How many institutions (not servers) maintain Web data?
  - number is smaller than the number of servers
  - many places have multiple servers
  - exact number is unknown
  - should be larger than 40% of the number of Web servers
- How many pages and how much traffic in the Web?
  - studies on the size of search engines, done in 2005, estimated over 20 billion pages
  - same studies estimated that size of static Web is roughly doubling every eight months

Characteristics

- Exact number of static Web pages important before wide use of dynamic pages
- Nowadays, the Web is infinite for practical purposes
  - can generate an infinite number of dynamic pages
  - Example: an on-line calendar
- Most popular formats on Web
  - HTML
  - followed by GIF and JPG, ASCII text, and PDF

Characteristics

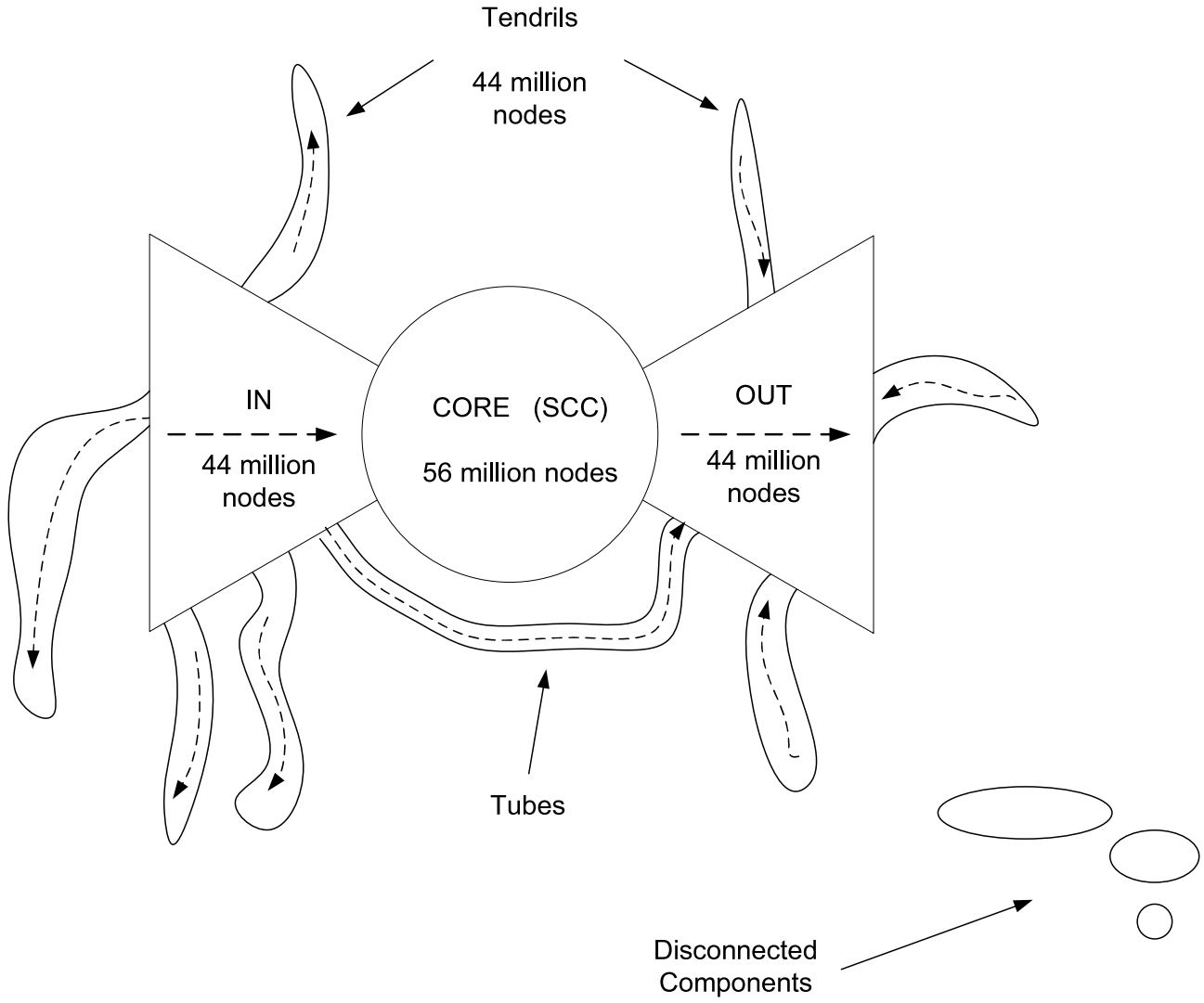
- Characteristics and statistics of HTML pages
  1. most HTML pages do not comply with HTML specifications
     - if browsers behave as strict HTML compilers, many pages not rendered
  2. HTML pages are small and contain few images
  3. Average number of external pages pointing to a page is close to zero
     - usually only pages from same domain point to a page
  4. Most referenced sites are the main Internet companies
  5. Sites with most links to outside sites
     - directories and Web 2.0 sites such as Wikipedia
     - without them, many more isolated portions or “islands”
  6. In 2000, around 70% of pages were in English
     - number of words in other languages growing faster than English
- Google Zeitgeist in January 2003
  - 50% of queries in English
  - down from 60% in 2001

Structure of the Web Graph

- The Web can be viewed as a graph, where
  - the nodes represent individual pages
  - the edges represent links between pages
- Broder *et al* compared the topology of the Web graph to a **bow-tie**

Structure of the Web Graph

- Original **bow-tie** structure of the Web

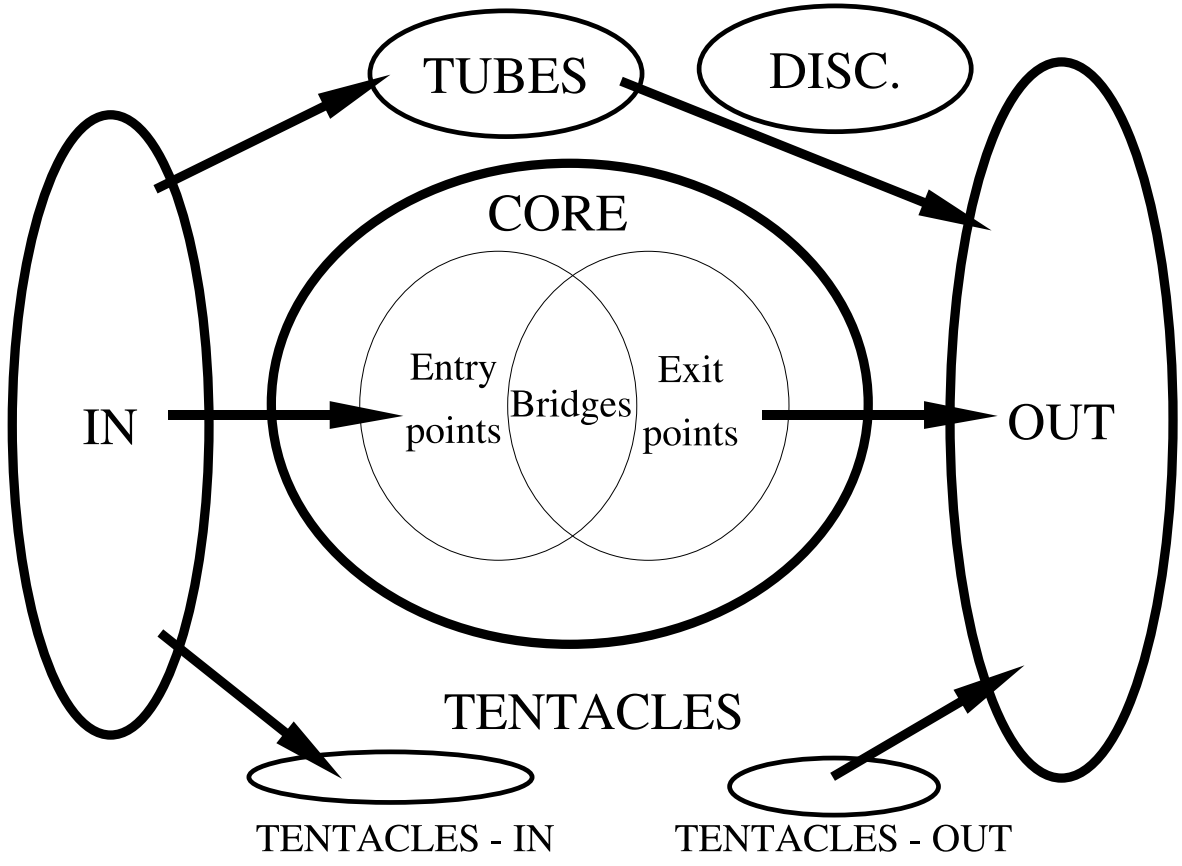


Structure of the Web Graph

- In Baeza-Yates *et al*, the graph notation was extended by dividing the CORE component into four parts:
  - **Bridges**: sites in CORE that can be reached directly from the IN component and that can reach directly the OUT component
  - **Entry points**: sites in CORE that can be reached directly from the IN component but are not in Bridges
  - **Exit points**: sites in CORE that reach the OUT component directly, but are not in Bridges
  - **Normal**: sites in CORE not belonging to the previously defined sub-components

Structure of the Web Graph

- More refined view of the bow-tie structure



Structure of the Web Graph

- In all Web studies
  - CORE component composed of a minority of the Web sites
  - on the other hand, it has a heavier density of Web pages
- **Link analysis**
  - correlation between structure and quality of the content
  - number of ISLANDS is much larger than we may think
  - most islands not easy to find unless registered with search engines

Modeling the Web

- CORE component follows a power law distribution
- **Power Law:** function that is invariant to scale changes

$$f(x) = \frac{a}{x^{\alpha}} \quad \text{with} \quad \alpha > 0$$

- Depending on value of α, moments of distribution will be finite or not
  - $\alpha \leq 2$**:** average and all higher-order moments are infinite
  - $2 < \alpha \leq 3$**:** mean exists, but variance and higher-order moments are infinite

Modeling the Web

- Web measures that follow a power law include
  - number of pages per Web site
  - number of Web sites per domain
  - incoming and outgoing link distributions
  - number of connected components of the Web graph
- Also the case for the **host-graph**
  - the connectivity graph at the level of Web sites

Modeling the Web

- Web power-law exponents: various countries and regions

Region	Page Size		Pages per site	In-degree	Out-degree	
	Small	Large			Small	Large
Brazil	0.3	3.4	1.6	1.89	0.67	2.71
Chile	0.4	3.2	1.6	2.01	0.72	2.56
Greece	0.4	3.2	1.6	1.88	0.61	1.92
Indochina	n/a	n/a	1.2	1.63	0.66	2.62
Italy	n/a	n/a	1.3	1.76	0.68	2.52
South Korea	0.4	3.7	3.2	1.90	0.29	1.97
Spain	n/a	2.25	1.1	2.07	0.86	4.15
United Kingdom	n/a	n/a	1.3	1.77	0.65	3.61
World	n/a	n/a	n/a	2.1	n/a	2.7

Modeling the Web

- For the page size, there are two exponents
  - one for the pages with less than 20KB
  - the rest
- The same for the out-degree
  - pages with roughly less than 20 out-links
  - pages with more out-links
- This is due to the *shame* counter effect to the *law of minimal effort* (Zipf)

Modeling the Web

- Distribution of document sizes: **self-similar model**
  - based on mixing two different distributions
- Main body of distribution follows a **Logarithmic Normal distribution**

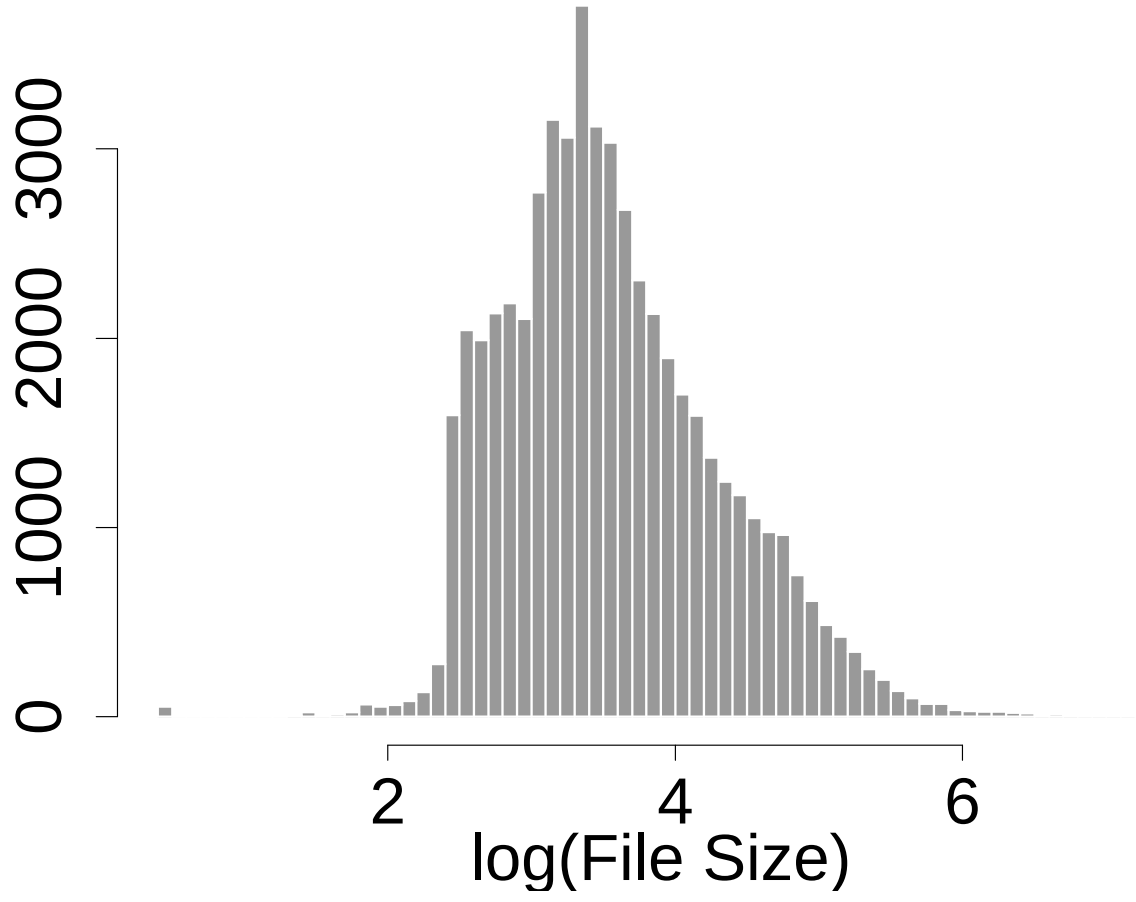
$$p(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-(\ln x - \mu)^2 / 2\sigma^2}$$

where

- x is the document size
- average size: $\mu = 9.357$ (in a sample)
- standard deviation: $\sigma = 1.318$ (in a sample)

Modeling the Web

- Example of file size distribution in a semi-log graph



Modeling the Web

- The right tail of the distribution is **heavy-tailed**
  - majority of documents are small
  - but there is a non trivial number of large documents, so the area under the curve is relevant
- Good fit is obtained with a Pareto distribution, which is similar to a power law

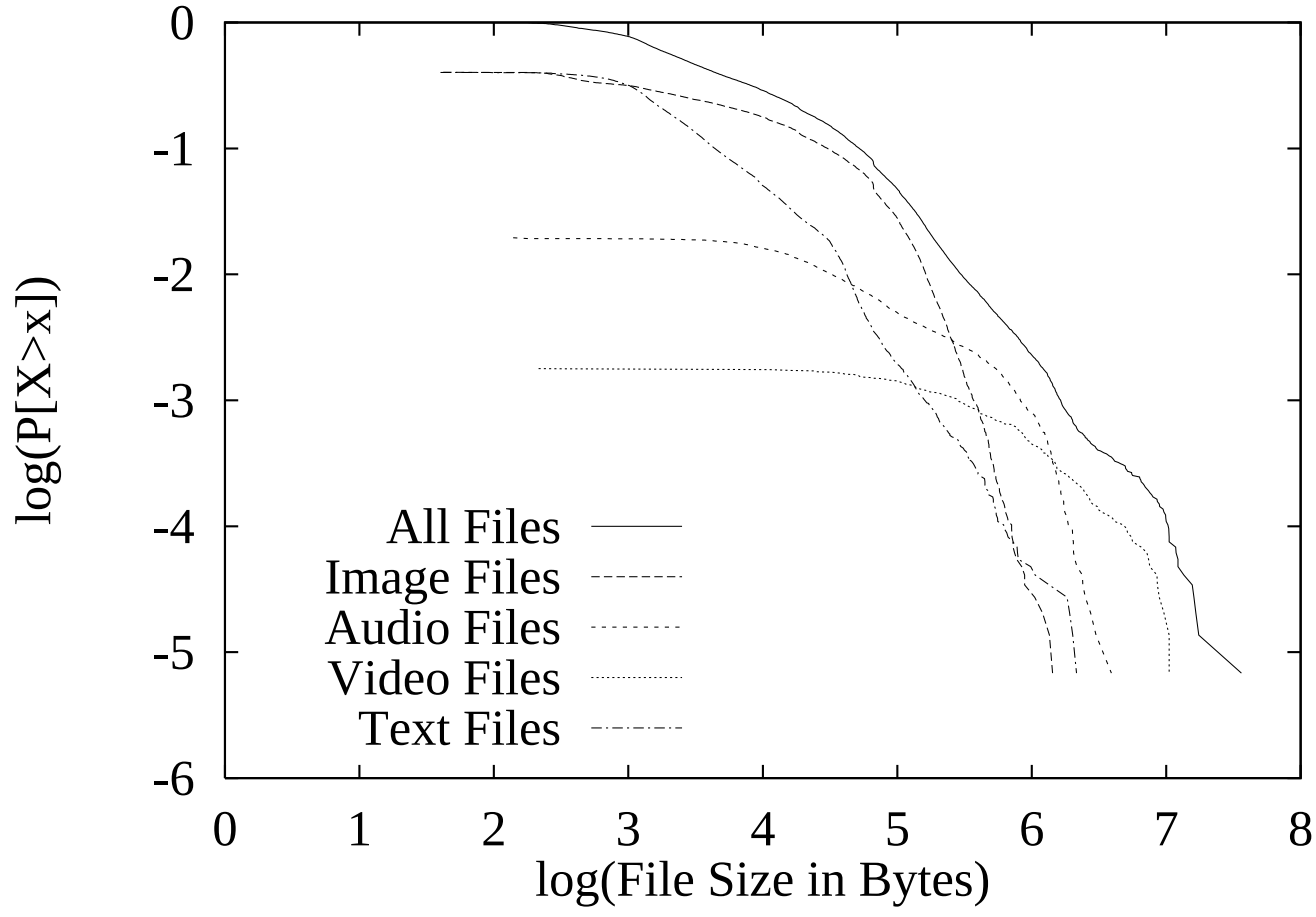
$$p(x) = \frac{\alpha k^{\alpha}}{x^{1+\alpha}}$$

where

- x is measured in bytes
- k and α are parameters of the distribution

Modeling the Web

- Right tail distribution for different file types (Web sample)



Link Analysis

- On the Web, we distinguish three levels of link analysis
  - **microscopic level:** related to the statistical properties of links and individual nodes
  - **mesoscopic level:** related to the properties of areas or regions of the Web
  - **macroscopic level:** related to the structure of the Web at large

Link Analysis

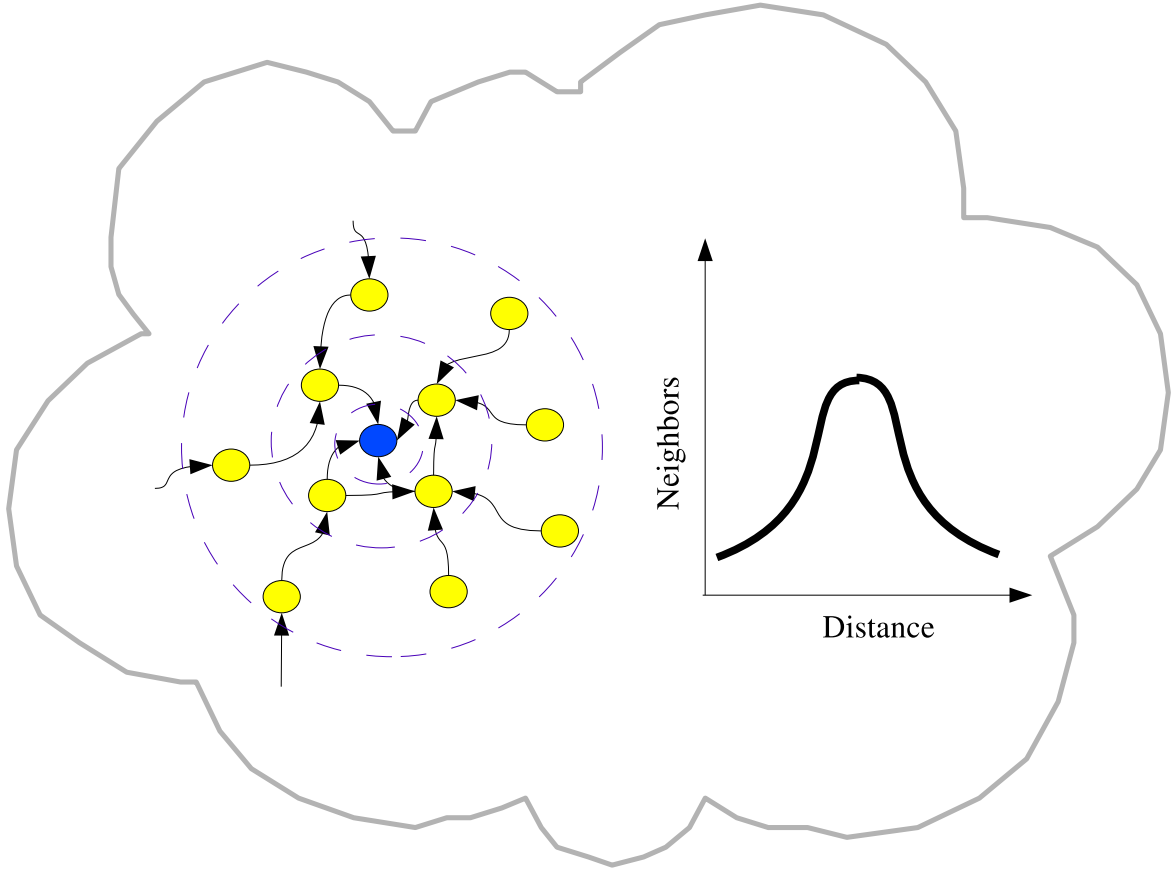
- The **macroscopic level** started with the seminal **bow-tie** work by Broder *et al* already explained
- A related macroscopic description is the **Jellyfish structure** proposed in Siganos *et al*
  - core portion surrounded by areas of decreasing link density
  - many nodes form long and loosely-connected chains or **tentacles**

Link Analysis

- **Mesoscopic link analysis**
  - related to the properties of the neighborhood of a node
  - context in which most link-based ranking functions work
- **Hop-plot:** way to describe neighborhood of a node

Link Analysis

- Schematic depiction **hop-plot**
  - plot of number of neighbors at different distances



Link Analysis

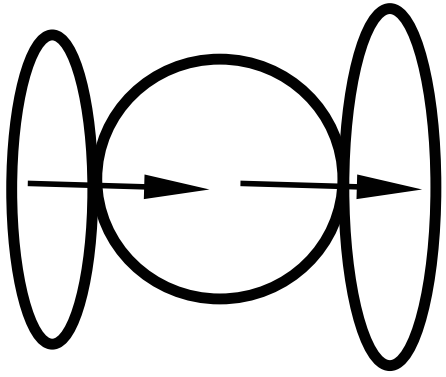
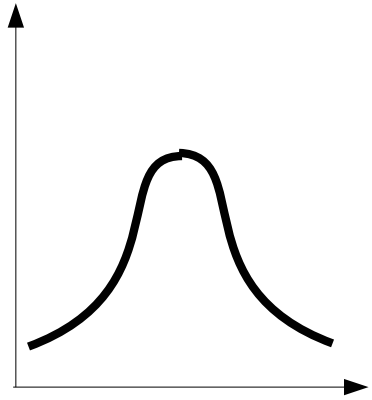
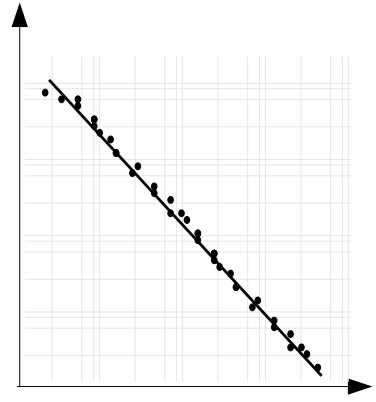
- **Microscopic level**
  - distribution of number of links of a page p is very skewed
  - in scale-free networks, as the Web, it follows a power-law

$$Pr(\text{page } p \text{ has } k \text{ links}) \propto k^{-\alpha}$$

where usually $2 < \alpha < 3$

Link Analysis

- Visual summary of levels of link-based analysis

Macroscopic	Mesoscopic	Microscopic
		
Connected components Jellyfish structure Bow-tie structure ...	Hop-plots Link-based ranking Clusters, communities ...	Zipf's law Degree distributions ...

Link Analysis

- Link analysis can be used to
  - infer relevance
  - prioritize crawling
  - identify sub-structures such as communities on the Web graph

Search Engine Architectures

- Web query processing and ranking
  - done without accessing the source of the documents
  - no remote access to pages through the network at query time
  - for snippet generation, source of documents is used
    - restricted to top 10 results
    - based on local copies of pages

Basic Architecture

- **Centralized crawler-indexer architecture**
  - used by most engines
  - crawlers are software agents that traverse the Web copying pages
  - pages crawled are stored in a central repository and then indexed
  - index is used in a centralized fashion to answer queries
  - most search engines use indices based on the **inverted index**
  - only a logical, rather than a literal, view of the text needs to be indexed
- Details in Chapter 12

Basic Architecture

- Normalization operations
  - removal of punctuation
  - substitution of multiple consecutive spaces by a single space
  - conversion of uppercase to lowercase letters
- Some engines eliminate stopwords to reduce index size
- Index is complemented with metadata associated with pages
  - creation date, size, title, etc.

Basic Architecture

- Given a query
  - 10 results shown are subset of complete result set
  - if user requests more results, search engine can
    - recompute the query to generate the next 10 results
    - obtain them from a partial result set maintained in main memory
- In any case, a search engine never computes the full answer set for the whole Web

Basic Architecture

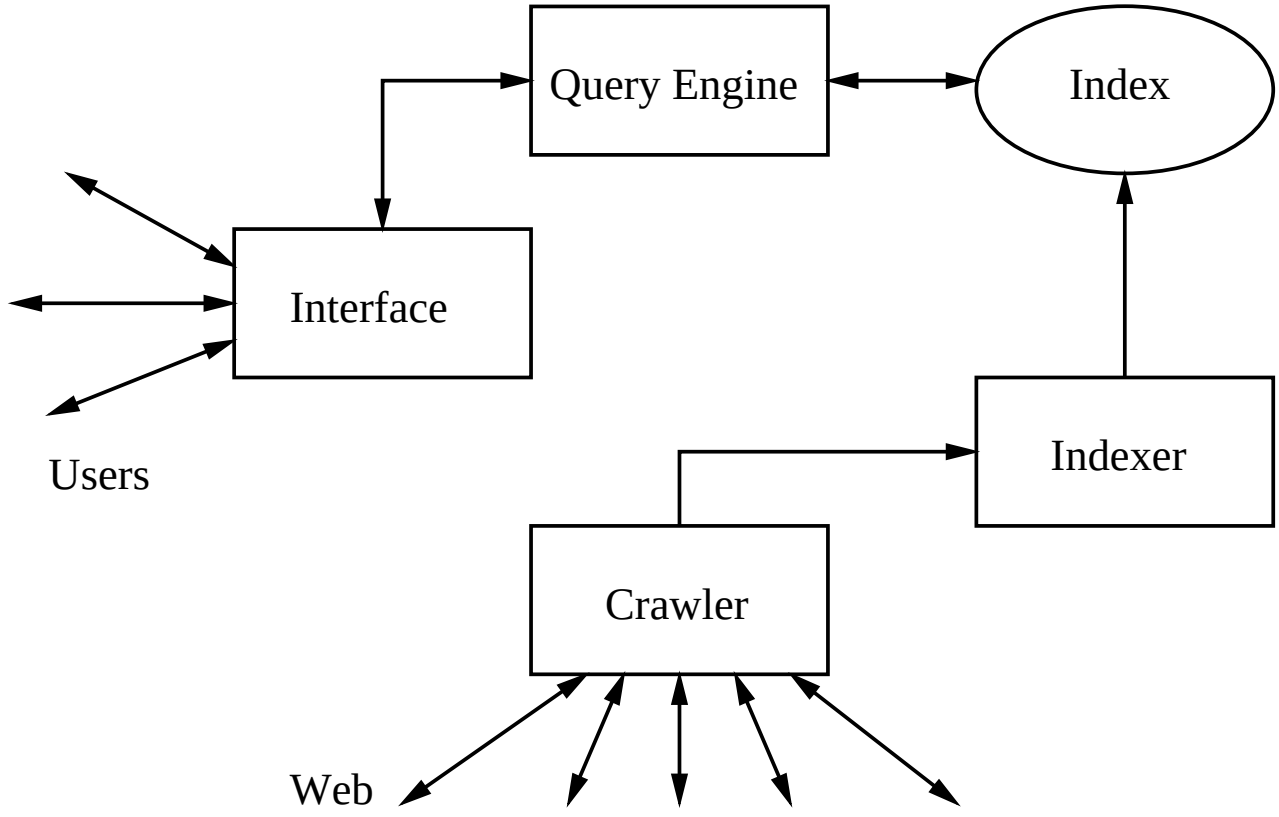
- State of the art indexing techniques
  - can reduce index to about 30% of original size
  - index can be used to answer queries composed of multiple words
    - combine list of documents for individual words
  - many engines support exact phrase and/or proximity search, which requires
    - additional information on the position of the terms in the documents
    - indexing frequent phrases as single indexing units

Basic Architecture

- Search can be conducted efficiently if each word is not too frequent
  - seldom the case on the Web
- For this reason, all engines use lazy evaluation query processing
  - only the first results are computed
  - further results are computed on demand

Basic Architecture

- Schematic software architecture of early search engine (AltaVista)



Basic Architecture

- Main problem faced by this architecture
  - gathering of the data, and
  - sheer volume
- Crawler-indexer architecture could not cope with Web growth (end of 1990s)
  - solution: distribute and parallelize computation

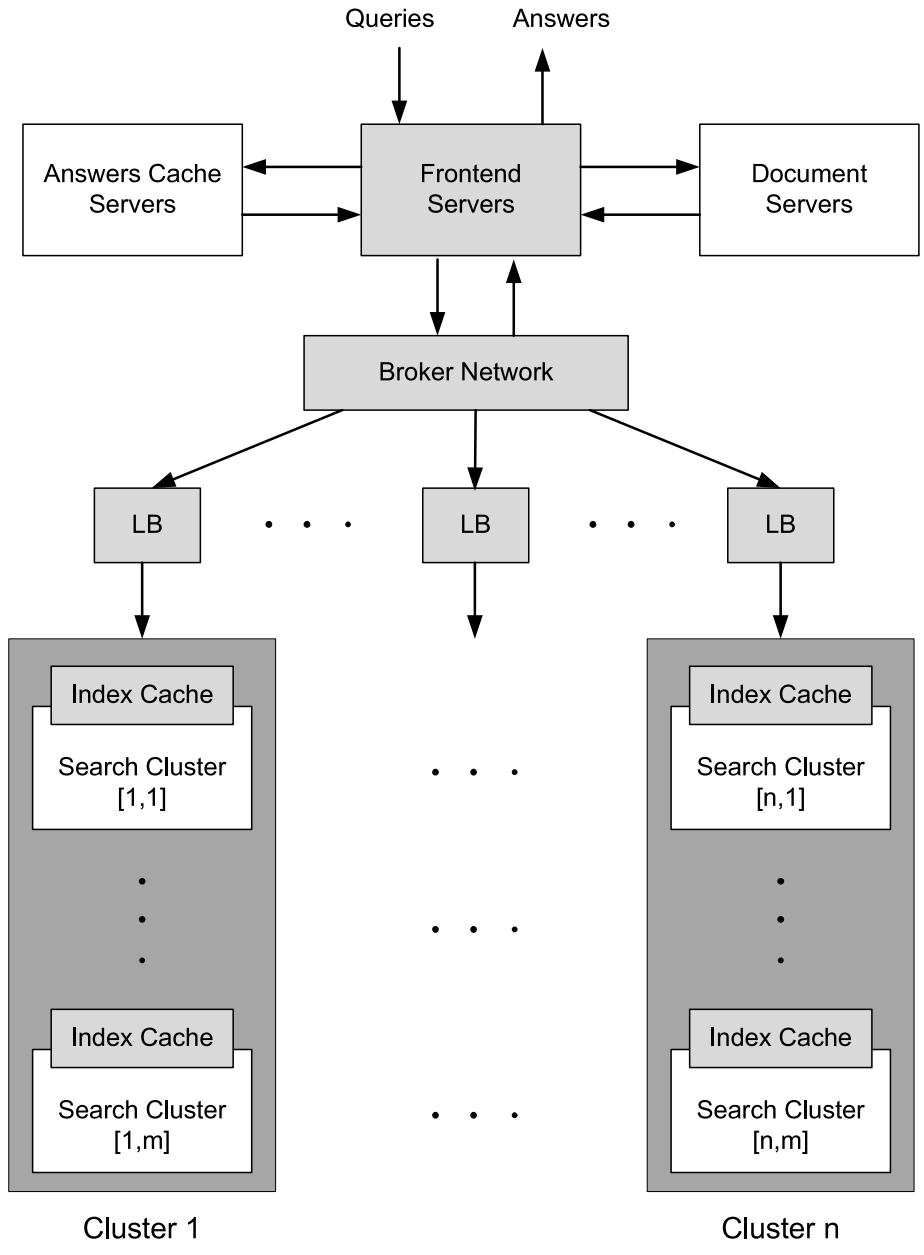
Cluster-based Architecture

- Current engines adopt a massively parallel **cluster-based architecture**
  - document partitioning is used
  - replicated to handle the overall query load
  - cluster replicas maintained in various geographical locations to decrease latency time (for nearby users)

Cluster-based Architecture

- Many crucial details need to be addressed
  - good balance between the internal and external activities
  - good load balancing among different clusters
  - fault tolerance at software level to protect against hardware failures

Cluster-based Architecture



Cluster-based Architecture

- Orlando *et al* present a parallel and distributed search engine architecture based on two strategies
  - **a task parallel strategy**: a query is executed independently by a set of homogeneous index servers
  - **a data parallel strategy**: a query is processed in parallel by index servers accessing distinct partitions of the database
- Chowdhury and Pass introduce a queuing theory model of a search architecture for document partitioning
  - architecture is then used to analyze inherent operational requirements: throughput, response time, and utilization

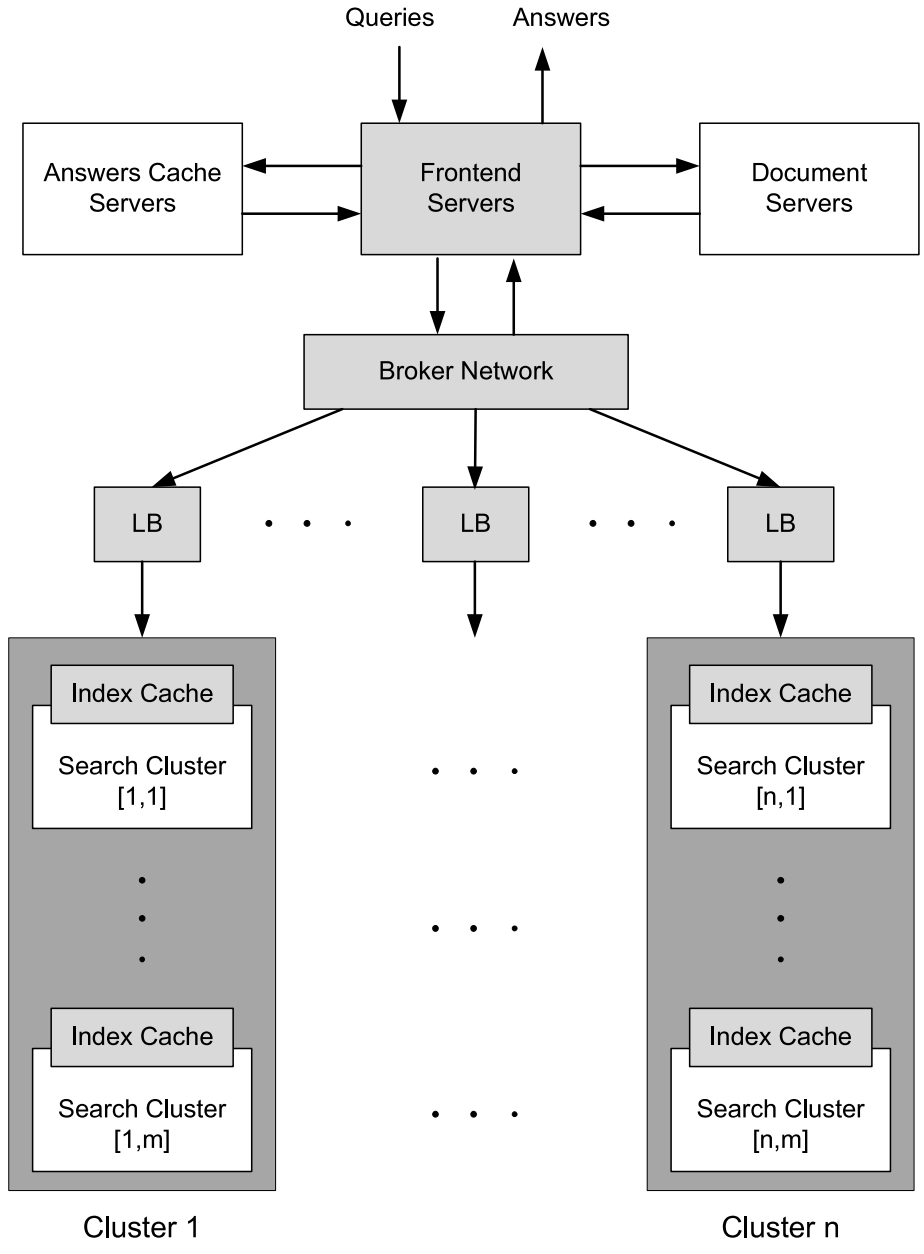
Caching

- Search engines need to be fast
  - whenever possible, execute tasks in main memory
  - caching is highly recommended and extensively used
    - provides for shorter average response time
    - significantly reduces workload on back-end servers
    - decreases the overall amount of bandwidth utilized
- In the Web, caching can be done both at the client or the server side

Caching

- **Caching of answers**
  - the most effective caching technique in search engines
  - query distribution follows a power law
    - small cache can answer a large percentage of queries
  - with a 30% hit-rate, capacity of search engine increases by almost 43%
- Still, in any time window a large fraction of queries will be unique
  - hence, those queries will not be in the cache
  - 50% in Baeza-Yates *et al*
  - this can be improved by also **caching inverted lists** at the search cluster level

Caching: Cluster-based



Caching

- Raghavan and Sever proposed to use query logs to improve retrieval effectiveness for future queries
- Markatos proved existence of query temporal locality, showing that static caching is useful for small caches
- Cao proposed caching policies that take into account parameters other than locality, such as:
  - the size of the object to be cached
  - the time needed to fetch objects from disk
  - an index of precomputed results and inverted lists of most frequent query-terms are kept in main memory
  - remaining part of the index is kept in secondary storage

Caching

- As systems are often hierarchical, there are proposals for multiple level caching architectures
- Saraiva *et al* proposed a new architecture using a two-level dynamic caching system
  - second-level cache can effectively reduce disk traffic, thus increasing the overall throughput
- Baeza-Yates and Saint Jean proposed a three-level index with frequency based static cache of the inverted lists

Caching

- Lempel and Moran proposed a new caching policy called **Probabilistic Driven Caching (PDC)**
  - attempts to estimate probability distribution of all follow-up queries
  - first policy to adopt prefetching in anticipation of a user request
- Fagni *et al* combined static and dynamic caching policies together with an adaptive prefetching policy
  - devoting a large fraction of entries to static caching along with prefetching produces the best hit rate

Caching

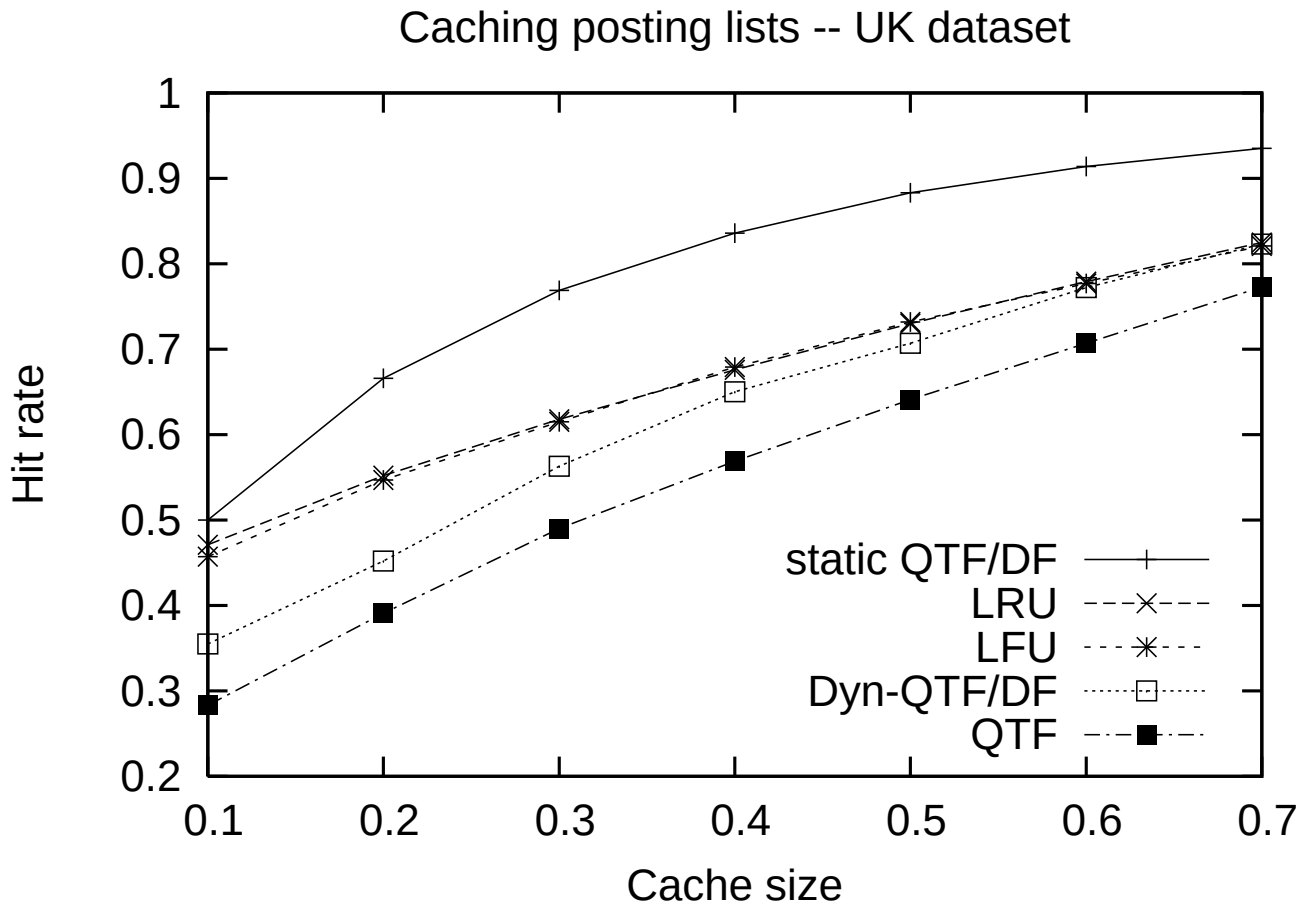
- Zhang *et al* studied caching of blocks of compressed inverted lists using several dynamic caching algorithms
  - evicting from memory the least frequently used blocks of inverted lists performs very well in terms of hit rate
- Baeza-Yates *et al* studied impact of static and dynamic caching
  - focus on inverted list caching and memory allocation for results
  - optimal results were achieved when dedicating around 30% of memory to caching results and the rest to inverted lists

Caching

- Baeza-Yates *et al* also proposed a new algorithm for static caching of inverted lists
  - based on a well-known heuristic for the **Knapsack problem**
  - uses ratio of the query frequency to the inverted list length to decide what to cache
  - changes on query distribution are small and have only small effect on static solution (which can be recomputed periodically)

Caching

- Results of Baeza-Yates *et al* algorithm as compared to LRU, LFU, and previous solutions



Multiple Indexes

- Hierarchical indexes represent another type of improvement
- To illustrate, consider a two-level or two-tier index
  - first tier is a small and fast index for the most frequent queries
  - second tier is a larger and slower index for the rest of the queries
- Risvik proposed to use a multi-tier system for scalability purposes
  - tiers act as filters to allow query to **fall through** to next tier based on
    - the number of hits in a given tier
    - the relevance score of the query results from that tier

Multiple Indexes

- One disadvantage of the previous technique is that
  - some queries will have slower answers
  - in particular, if tiers are searched sequentially, one after the other
- A solution to this problem is to predict which queries need to go to next tier
- For this, Baeza-Yates *et al* proposed a machine learning based predictor to decide whether search should be conducted in parallel or not

Multiple Indexes

- Liu *et al* showed how to reduce corpus size by 95%
  - to produce a **cleansed corpus**
  - still retain retrieval performance
  - more than 90% of queries could be answered from the cleansed corpus
  - exploited query-independent features to classify each page into
    - potential retrieval target page (cleansed corpus), or
    - ordinary page (removed)

Distributed Architectures

- There exist several variants of the crawler-indexer architecture
- We describe here the most important ones
  - most significant early example is Harvest
  - among newest proposals, we distinguish the multi-site architecture proposed by Baeza-Yates *et al*

Harvest

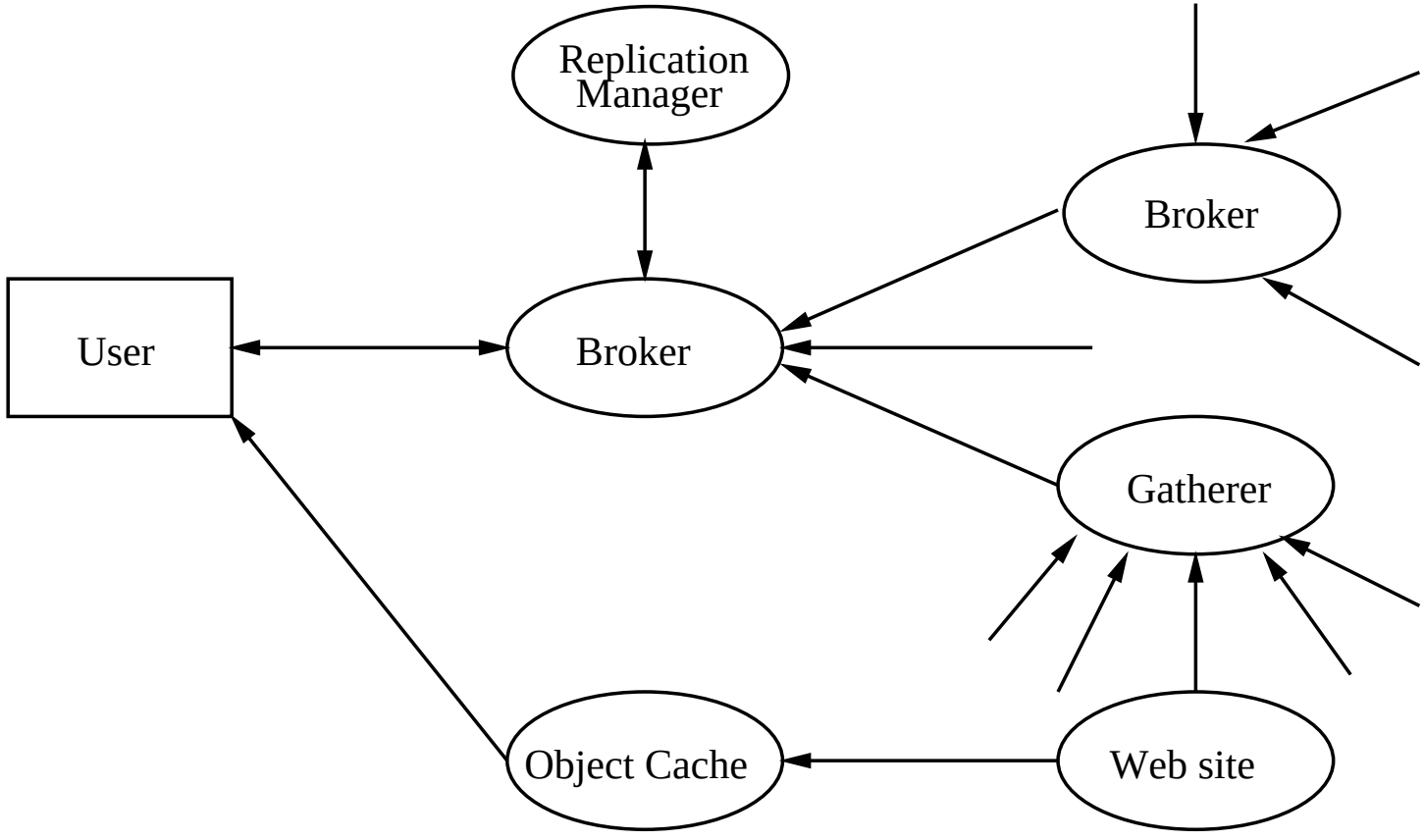
- Harvest uses a distributed architecture to gather and distribute data
  - interestingly, it does not suffer from some of common problems of the crawler-indexer architectures, such as
    - increased servers load caused by reception of simultaneous requests from different crawlers
    - increased Web traffic, due to crawlers retrieving entire objects, while most content is not retained eventually
    - lack of coordination between engines, as information is gathered independently by each crawler

Harvest

- To avoid these issues, two components are introduced: gatherers and brokers
  - **gatherer:** collects and extracts indexing information from one or more Web servers
  - **broker:** provides indexing mechanism and query interface to the data gathered

Harvest

- Example of the Harvest architecture



Multi-site Architecture

- As the document collection grows
  - capacity of query processors has to grow as well
  - unlikely that growth in size of single processors can match growth of very large collections
    - even if a large number of servers is used
    - main reasons are physical constraints such as size of single data center and power and cooling requirements

Multi-site Architecture

- Distributed resolution of queries using different query processors is a viable approach
  - enables a more scalable solution
  - but also imposes new challenges
  - one such challenge is the routing of queries to appropriate query processors
    - to utilize more efficiently available resources and provide more precise results
    - factors affecting query routing include geographical proximity, query topic, or language of the query

Multi-site Architecture

- Geographical proximity: reduce network latency by using resources close to the user posing the query
- Possible implementation is DNS redirection
  - according to IP address of client, the DNS service routes query to appropriate Web server
    - usually the closest in terms of network distance
- As another example, DNS service can use the geographical location to determine where to route queries to
- There is a fluctuation in submitted queries from a particular geographic region during a day
  - possible to offload a server from a busy region by rerouting some queries to query servers in a less busy region

Multi-site Architecture

- Baeza-Yates *et al* recently proposed a cost model for this kind of search engines
  - simple distributed architecture that has comparable cost to a centralized search architecture
  - architecture based on several sites that are logically connected through a star topology network
    - central site is the one with the highest load of local queries
  - main idea: answer local queries locally and forward to other sites only queries that need external pages in their answers
  - to increase percentage of local queries, use caching of results and replicate small set of popular documents in all sites
    - increase in number of local results from 5% to 30% or more

Multi-site Architecture

- In complementary paper, Cambazoglu *et al* show that
  - resources saved by answering queries locally can be used to execute a more complex ranking function
  - this can improve the results
- In a more recent paper, Cambazoglu *et al* show that query processing can be improved by using linear programming to know when to re-route queries

Search Engine Ranking

Search Engine Ranking

- Ranking
  - hardest and most important function of a search engine
- A key first challenge
  - devise an adequate process of **evaluating the ranking**, in terms of relevance of results to the user
  - without such evaluation, it is close to impossible to fine tune the ranking function
  - without fine tuning the ranking, there is no state-of-the-art engine—this is an empirical field of science

Search Engine Ranking

- A second critical challenge
  - identification of **quality content** in the Web
- Evidence of quality can be indicated by several signals such as:
  - domain names
  - text content
  - links (like PageRank)
  - Web page access patterns as monitored by the search engine
- Additional useful signals are provided by the layout of the Web page, its title, metadata, font sizes, etc.

Search Engine Ranking

- A third critical challenge
  - avoiding, preventing, managing **Web spam**
  - spammers are malicious users who try to trick search engines by artificially inflating signals used for ranking
  - a consequence of the economic incentives of the current advertising model adopted by search engines
- A fourth major challenge
  - defining the **ranking function** and computing it

Ranking Signals

- Distinct types of signals used for ranking: content, structure, or usage
- **Content signals**
  - related to the text itself
  - can vary from simple word counts to a full IR score such as BM25
  - can be provided by the layout, that is, the HTML source
    - simple format indicators (more weight given to titles/headings)
    - sophisticated indicators as the proximity of certain tags in the page

Ranking Signals

- **Structural signals**
  - intrinsic to the linked structure of the Web
  - some of them are textual in nature, such as **anchor text**
  - others pertain to the links themselves, such as **in-links** and **out-links** from a page
  - link-based signals find broad usage beyond classic search engine ranking

Ranking Signals

- **Web usage signals**
  - main one is the implicit feedback provided by the **user clicks (click-through)**
  - other usage signals include
    - information on the user's geographical context (IP address, language)
    - technological context (operating system, browser)
    - temporal context (query history by the use of cookies)

Link-based Ranking

- **Number of hyperlinks** that point to a page provides a measure of its popularity and quality
- Many links in common among pages are indicative of page relations with potential value for ranking purposes
- Examples of ranking techniques that exploit links are discussed next

Early Algorithms

- Use incoming links for ranking Web pages
- But, just counting links was not a very reliable measure of authoritativeness
  - easy to externally influence this count by creating new links to a page

Early Algorithms

- Yuwono and Lee proposed three early ranking algorithms (in addition to the classic TF-IDF vector ranking)
  - **Boolean spread**
    - given page p in result set
    - extend result set with pages that point to and are pointed by page p
  - **Vector spread**
    - given page p in result set
    - extend result set with pages that point to and are pointed by page p
  - **most-cited**
    - a page p is assigned a score given by the sum of number of query words contained in other pages that point to page p

Early Algorithms

- WebQuery is an early algorithm that allows visual browsing of Web pages
  - takes a set of Web pages
  - ranks them based on how connected each Web page is
- A related approach is presented by Li

PageRank

- The basic idea is that good pages point to good pages
- Let p, r be two variables for pages and L a set of links
- **PageRank Algorithm**

1. $p :=$ initial page the user is at;
2. while (stop-criterion is not met) {
3. $\quad L := links\_inside\_page(p)$;
4. $\quad r := random(L)$;
5. $\quad$ move to page pointed by r;
6. $\quad p := r$;
7. }

PageRank

- Notice that PageRank simulates a user navigating randomly on the Web
- At infinity, the probability of finding the user at any given page becomes stationary
- Process can be modeled by a Markov chain
  - stationary probability of being at each page can be computed
- This probability is a property of the graph
  - referred to as **PageRank** in the context of the Web
- PageRank is the best known link-based weighting scheme
- It is also part of the ranking strategy adopted by Google

PageRank

- Let
  - Let $L(p)$ be the number of outgoing links of page p
  - Let $p_1 \ldots p_n$ be the pages that point to page a
  - User jumps to a random page with probability q
  - User follows one of the links in current page with probability $1 - q$
  - **PageRank** of page a is given by the probability $PR(a)$ of finding our user in that page

$$PR(a) = \frac{q}{T} + (1 - q) \sum_{i=1}^{n} \frac{PR(p_i)}{L(p_i)}$$

where

- T: total number of pages on the Web graph
- q: parameter set by the system (typical value is 0.15)

PageRank

- Problem: handling of pages that do not have out-going links
  - solution: use $q = 1$ for these pages
  - simpler solution: remove them and only compute their PageRanks at the end, using the PageRank of their parents

PageRank

- Baeza-Yates *et al* defines family of link-based ranking algorithms that propagate page weights through links
  - based on damping function that decreases with distance
  - authors study three damping functions:
    - linear decay on length of path
    - exponential decay
    - hyperbolic decay
  - Exponential decay corresponds to PageRank
    - other functions are new
  - They give an explanation for the typical value of q

HITS

- **HITS (Hypertext Induced Topic Search)**
  - better idea due to Kleinberg
  - considers set of pages S that point to or are pointed by pages in answer set
  - pages that have many links pointing to it are called **authorities**
  - pages that have many outgoing links are called **hubs**
  - positive two-way feedback
    - better authority pages come from incoming edges from good hubs
    - better hub pages come from outgoing edges to good authorities

HITS

- Let
  - $H(p)$: hub value of page p
  - $A(p)$: authority value of page p
- $H(p)$ and $A(p)$ are defined such that

$$H(p) = \sum_{u \in S \mid p \rightarrow u} A(u) \ , \qquad A(p) = \sum_{v \in S \mid v \rightarrow p} H(v)$$

  where $H(p)$ and $A(p)$ are normalized
- Does not work with non-existent, repeated, or automatically generated links
  - solution: weigh each link based on surrounding content

HITS

- A second problem is that of **topic diffusion**
  - due to link weights, the result set might include pages that are not directly related to the query
  - example: a query might be expanded to a more general topic that properly contains the original answer
- One solution: associate a score with content of each page
  - this score is then combined with the link weight
  - experiments show that recall and precision for first ten results increase significantly

Link Ranking

- Summary of power-law exponents for link-based measures of various countries

Country	PageRank	HITS	
		Hubs	Auth
Brazil	1.83	2.9	1.83
Chile	1.85	2.7	1.85
Greece	1.83	2.6	1.83
South Korea	1.83	3.7	1.83
Spain	1.96	n/a	n/a

Simple Ranking Functions

- Simplest ranking scheme
  - use a global ranking function such as PageRank
  - in this case, quality of a Web page in the result set is independent of the query
  - the query only selects pages to be ranked
- More elaborate ranking scheme
  - use a linear combination of different ranking signals
  - for instance, combine BM25 (text-based ranking) with PageRank (link-based ranking)

Simple Ranking Functions

- To illustrate, consider the pages p that satisfy query Q
- Rank score $R(p, Q)$ of page p with regard to query Q can be computed as

$$R(p, Q) = \alpha \; BM25(p, Q) \; + \; (1 - \alpha) PR(p)$$

  - $\alpha = 1$: text-based ranking, early search engines
  - $\alpha = 0$: link-based ranking, independent of the query

Simple Ranking Functions

- Current engines combine a text-based ranking with a link-based ranking, most of them a lot more complex than BM25 and PageRank
  - value of α tuned experimentally using
    - labeled data as ground truth, or
    - clickthrough data
- α might even be query dependent
  - for navigational queries α could be made smaller than for informational queries

Learning to Rank

- Distinct approach for computing a Web ranking
  - apply machine learning techniques to **learn the ranking of the results**
  - use a learning algorithm fed with training data that contains ranking information
  - loss function to minimize: number of mistakes done by learned algorithm
- Given query Q, three types of training data can be used:
  - **pointwise**: a set of relevant pages for Q
  - **pairwise**: a set of pairs of relevant pages indicating the ranking relation between the two pages
  - **listwise**: a set of ordered relevant pages: $p_1 \succ p_2 \cdots \succ p_m$

Learning to Rank

- Training data may be originated from
  - **editorial judgements** made by humans
  - **click-through data**, which is available in large volume
    - we can learn the ranking from click-based preferences (see Chapter 5)
    - for query Q, if p_1 has more clicks than p_2, then $[p_1 \succ p_2]$

Learning the Ranking Function

- A different scheme consists of learning the ranking function, rather than the ranking order
- The idea is to use a genetic algorithm
  - members of the population are function instances over a given set of ranking features
  - at every step of the genetic algorithm, different functions are mutated or mixed
  - the goodness of each learning function is evaluated through a set of ground truth or training data
  - after many iterations, the fittest function is selected

Learning the Ranking Function

- Approach has clear advantage of offering insight about the important features for ranking
- Idea was discovered in parallel and independently by
  - Trotmann for document ranking
  - Lacerda *et al* for advertisement ranking
- As this technique is quite new, further research is needed to
  - improve quality of results
  - improve efficiency of the technique

Quality Evaluation

- To evaluate quality, Web search engines typically use
  - human judgements of which results are relevant for a given query
  - some approximation of a **ground truth** inferred from user's clicks
  - a combination of both

Precision at 5, 10, 20

- To evaluate search results use precision-recall metrics
  - but, on the Web, it is almost impossible to measure recall
  - thus, standard precision-recall figures cannot be applied directly
  - most Web users inspect only the top 10 results
  - it is uncommon that a user inspects beyond the top 20 results
  - since queries tend to be short and vague, human evaluation of results should be based on distinct relevance assessments for each query-result pair
- The compounding effect of these observations is that
  - precision of Web results should be measured only at the top positions in the ranking, say P@5, P@10, and P@20
  - each query-result pair should be subjected to 3-5 independent relevant assessments

Click-through Data

- An advantage of using click-through data to evaluate the quality of answers derives from its scalability
- On the other hand works less well in smaller corpora
- Note that users' clicks are not used as a binary signal but in significantly more complex ways such as:
  - considering whether the user remained a long time on the page it clicked (a good signal),
  - jumped from one result to the other (a signal that nothing satisfying was found), or
  - the user clicked and came back right away (possibly implies Web spam)
- These measures and their usage are complex and kept secret by leading search engines

Click-through Data

- An important problem when using clicks is to take into account that the click rate is biased by
  - the ranking of the answer
  - the user interface
- Hence we have to unbias the click data

Evaluating the Quality of Snippets

- A related problem is to measure the quality of the snippets in the results
  - **search snippets** are the small text excerpts associated with each result generated by a search engine
- Increased research activity has been observed in this area lately
  - In Kaisser *et al* the authors study how variations in snippet length affect search results quality
  - In Kanungo the authors study how to predict the readability of search snippets
  - In Alonso the authors proposed to associate time information with search snippets and evaluated how it improves results

Web Spam

- There is an economic incentive from Web site owners to rank high in the result lists of search engines
- **Web spam** or **spamdexing**: all deceptive actions that try to increase the ranking of a page in search engines
- Any evaluation strategy that counts replicable features of Web pages is prone to manipulation
- In practice, such manipulation is widespread, and in many cases, successful

Web Spam

- A **spam page** is
  - a page that is used for direct spamming
  - a page whose score is artificially inflated because of other spam pages
- Multiple **spamdexing** techniques exist and new ones continue to be invented in a continuous fight between spammers and search engine companies
- A spam page may contain an abnormally high number of keywords, or have other text features

Web Spam

- Link spamming includes **link farms** that either
  - create a complex structure linking Web sites of the same owner
  - collude to deceive the search engine
- Click spam is done by **robots**
  - which specify queries and click on preselected pages or ads
- A third approach is **programmatic spamming**
  - Web spammers inject piece of code in a Web page, say in Javascript
  - code, when executed on client side, displays information to the user that is different from the one crawled by the search engine, e.g., a fake login page of the user's bank
  - this is a particular form of what is called **cloaking**

Web Spam

- Some people often confuse Web spam with Search Engine Optimization (SEO)
  - SEO are techniques to improve the description of the contents of a Web page
  - proper and better descriptions improve the odds that the page will be ranked higher
  - these are legitimate techniques, particularly when they follow the guidelines published by most search engines
  - in contrast, malicious SEO is used by Web spammers who want to deceive users and search engines alike

Managing Web Data

Assigning Identifiers to Documents

- **Document identifiers**
  - numerical identifiers used to represent URLs in several data structures
  - usually assigned randomly or according to the ordering with which URLs are crawled
  - used to number nodes in Web graphs
  - also used to identify documents in search engines repositories
- Careful ordering of documents leads to assignment of identifiers from which both index and Web graph storing methods can benefit
- Assignment based on a global ranking scheme may simplify ranking of answers

Assigning Identifiers to Documents

- Assigning IDs in ascending order of lexicographically sorted URLs improves the compression rate
  - Hypothesis is that documents sharing correlated and discriminant terms are very likely to be hosted by the same site and will therefore also share a large prefix of their URLs

Metadata

- 20 billion URLs require at least 1TB of storage
  - to hold all metadata on the corresponding Web pages, using a compressed format
- Managing all this information efficiently implies a very fast and space efficient database
  - which in turn implies the availability of a very efficient file system
- **Google's BigTable**
  - perhaps the best example of a database incarnation at Web scale
  - used to store data in a distributed system
  - data input and querying done using the Map-Reduce paradigm

Metadata

- **Google's BigTable**
  - not a traditional database:
    - a sparse, distributed multi-dimensional sorted map
    - designed to scale up to petabytes across hundreds or thousands of machines
    - machines can be added to the system and used without any reconfiguration
  - as a database, BigTable shares characteristics of both row-oriented and column-oriented databases
  - each table has multiple dimensions, with values kept in a compressed form
  - optimized to the underlying file system, which is the Google File System or GFS

Metadata

- **HBase**
  - open source database inspired by BigTable
  - also a distributed database written in Java
  - runs on top of the Hadoop Distributed File System (HDFS)
  - provides BigTable-like capabilities to Hadoop, the open source version of map-reduce
  - column oriented and features compression, in-memory operations, and Bloom filters
- Other options: Hypertable and Cassandra
  - **Cassandra** runs in **Amazon Dynamo**
    - Dynamo is an Amazon proprietary key-value storage system
    - high availability
    - combines properties of a database with those of a distributed hash table

Compressing the Web Graph

- Web graphs may be represented with adjacency lists
  - lists that contain, for each vertex v of the graph, the list of vertices directly reachable from v
- Almost 80% of all links are local, that is, they point to pages of the same site
  - if we assign closer identifiers to URLs referring to the same site, the adjacency lists that will contain very close ids
- Using a d-gapped representation will lead to d-gapped adjacency lists having long runs of 1's
- Exploiting these redundancies of the Web graph make it possible to reach high compression rates

Compressing the Web Graph

- Purpose of Web graph compression schemes is
  - to provide empirical succinct data structures
  - to allow fast access, as the graph will be needed for link analysis and other applications
- **WebGraph**
  - compresses typical Web graphs at about 3 bits per link
  - provides access to a link in few hundreds of nanoseconds

Handling Duplicated Data

- Problem has two flavors:
  1. detect multiple URLs that represent exactly the same page (for example, mirrors)
  2. detect multiple URLs that point to partially duplicated content
- Identifying duplicates also reduces the size of the collection that needs to be indexed and searched

Handling Duplicated Data

- Defining what is a duplicate page is not obvious
  - to illustrate, two pages that contain the same text but differ on their HTML formatting (or CSS) have distinct layouts
- In mirroring systems, duplicates can be detected by using a **hash key** (computed over the whole document)
  - should be easy to compute
  - should have very low probability of collision
  - standard hashing functions normally used for this purpose
    - MD (Message Digest) hashing family
    - SHA (Secure Hash Algorithms) hashing family

Handling Duplicated Data

- **Near duplicates** are more complex to handle
  - mirror page which differs only by a date change
- To identify near duplicates use the cosine distance as similarity measure
  - Kolcz proposed to ignore very popular terms
- Another option is to use the **resemblance measure**
  - choose the function W (see Section 6.5.3)
  - pick the best threshold t so as to ensure an efficient computation
  - two documents are considered duplicate if the similarity between them is above a threshold value t

Handling Duplicated Data

- Optimizations that approximate the distance and differ in their efficiency and probability of error
  - COPS, KOALA, and DSC
- First idea is to use a hash value associated to each shingle
  - hashing values can be computed incrementally in linear time
- Second idea is to just consider some of the shingles, forming super shingles
- Chowdhury proposed computing a hashing code of every document without considering too infrequent and too frequent terms
  - in the worst case, this algorithm is $O(d \log d)$ for a collection of d documents, but $O(d)$ in practice

Search Engine User Interaction

Search Engine User Interaction

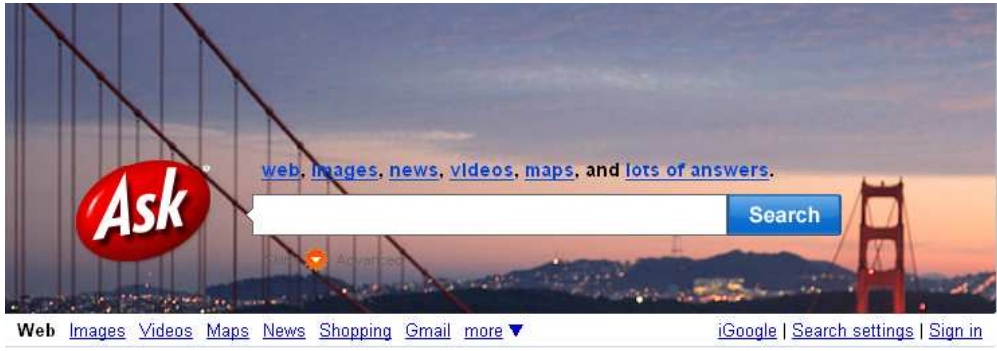
- Most search engine users have very little technical background
  - design of the interface has been heavily influenced by **extreme simplicity rules**
- Typical user interaction models for the most popular Search engines of today are discussed

The Search Rectangle Paradigm

- Users are now accustomed with specifying their queries in a **search rectangle**
  - commonly referred to as the search box
- Some portals embed the search rectangle in a privileged area of the site
  - `yahoo.com`
  - `aol.com`

The Search Rectangle Paradigm

- Search rectangle of four major search engines



The Search Rectangle Paradigm

- While search rectangle is the favored layout style, there are alternatives
  - many sites include an **Advanced Search page** (rarely used)
  - **search toolbars** provided by most search engines as a browser plug-in can be seen as a version of the search rectangle
  - **ultimate rectangle**, introduced by Google's Chrome **omnibox**, merges the functionality of the address bar with that of the search box

Query Languages

- Users typically express their queries as a sequence of words
- Some search engines declare that the underlying semantic of a query is an **AND** of all terms
- The query language typically consists of:
  - unary operators such as "**+**", "**-**", and "**site:**" to qualify the immediately following word
  - binary operators like OR to operate on the preceding and succeeding words
  - delimiters such as double quotes to indicate exact phrase match

Query Languages

- Common query operators

Operator Syntax	Google	Yahoo Search	Bing	Ask
“..”	yes	yes	yes	yes
+	yes	yes	yes	yes
–	yes	yes	yes	yes
OR	yes	yes	yes	yes
site:	no	yes	no	yes
url:	no	yes	yes	no
inurl:	no	yes	no	yes
intitle:	no	yes	yes	yes
inlink:/inanchor:	yes	no	yes	yes

Query Languages

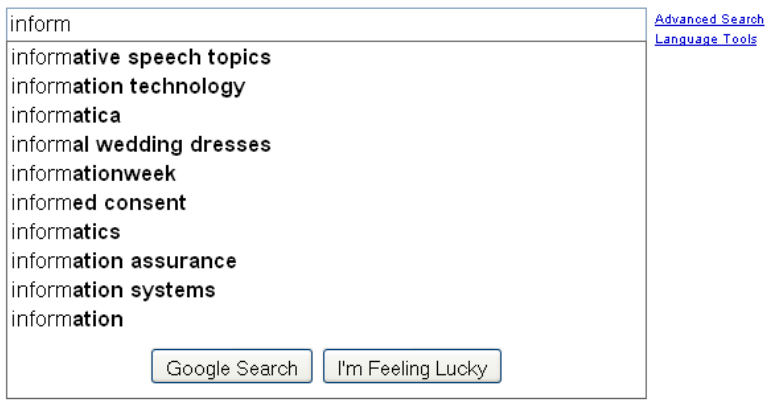
- In addition to these common operators, we list below some unique ones supported by a single search engine
- Ask's temporal operators
  - **afterdate:**, **beforedate:**
  - **betweendate:**
  - **last**:
- Bing
  - **AND/&**
  - **( )**
  - Bing has a long list of unique operators such as **filetype:, contains:, ip:, feed:, prefer:**, etc.

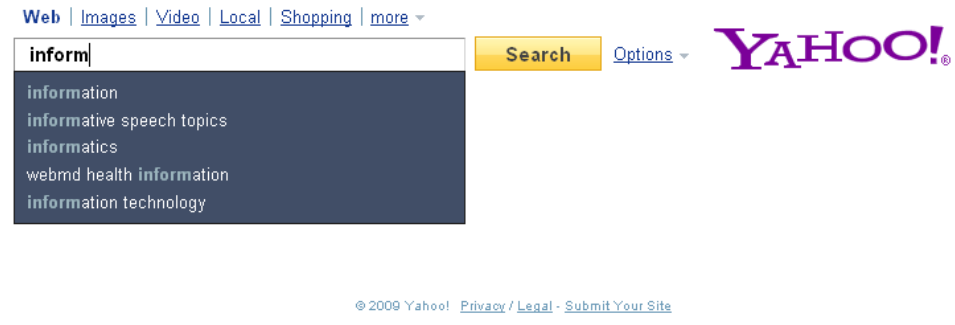
Query Languages

- Google
  - The wildcard stands for a missing full term and indicates to the search engine that it should be treated “as a placeholder for any unknown term”
- Yahoo! Search
  - **link:**
  - Yahoo! offers direct access via a set of reserved tokens which can be retrieved by typing **!list** in the search rectangle
  - These include, for instance, **!news, !flickr, !wiki, !map**

Dynamic Query Suggestions

- **Dynamic query suggestions** services enrich the search rectangle with interactive capabilities
- As users enter characters in the search box, one at a time, query suggestions are offered to them





Dynamic Query Suggestions

- **Dynamic query suggestions** systems should be distinguished from **query recommendations** systems
  - dynamic suggestions systems work with very little information
  - their main input being a prefix rather than a well formed query
- Dynamic suggestions systems are not entirely new as a similar functionality was offered by
  - early editors like Emacs, which supported command completion
  - shell scripts like Korn shell, which would complete commands when a user would enter a tab or space character

Dynamic Query Suggestions

- Key differences between these early features and modern dynamic query suggestion
  - the source and scale of modern suggestions corpora
  - the performance demands as modern suggestions services need to serve a huge number of users at the same time
  - user-experience wise, the fact that modern suggestions are triggered automatically as the user types rather than upon request
- Addressing these challenges was made possible by two major changes in the Web environment
  - growth in search traffic
  - performance enhancements

Dynamic Query Suggestions

- By using query logs rather than the corpus at hand, a closer language model could be used
- Dynamic query suggestions are, on average, five times heavier (in terms of queries per second) than regular search
  - by default, a request is sent to the server each time a new character is entered
- Interestingly, dynamic suggestion services had to overcome the two classic IR technical challenges in order to bring value to users
  - **efficiency**, so as to return suggestions fast enough to be usable
  - **effectiveness**, so as to present the most relevant suggestions

Dynamic Query Suggestions

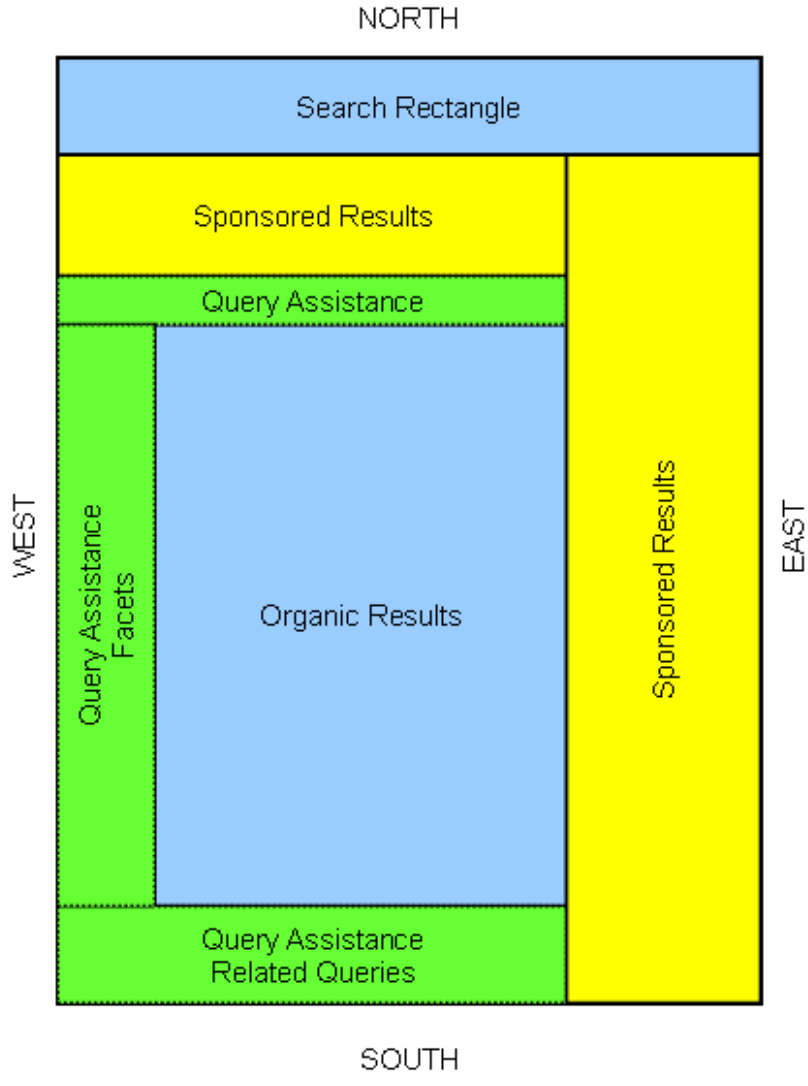
- Technical issues considered to ensure relevance
  - automatic spelling correction
  - inappropriate suggestions filtering
  - de-duplication of queries suggestions
  - diversity of suggestions
  - freshness
  - personalization
- Overall, the effectiveness of dynamic suggestions can be measured by coverage and quality
  - achieving good coverage becomes especially tricky when prefixes get longer
  - quality is obviously a must as well, as users will expect the suggestion service to “read their mind”

The Basic Layout

- The classic presentation style of the Search Engine Result Page, often referred to as SERP consists of
  - a list of “organic” or “algorithmic” results, which appear on the left hand side of the results page
  - a list of paid/sponsored results (ads), which appear on the right hand side
- Additionally, the most relevant paid results might appear on top of the organic results in the North area
- By default, most search engines show ten results in the first page
  - some engines such as Bing allow users to customize the number of results to show on a page

The Basic Layout

- Typical SERP layout



The Basic Layout

- These engines might differ on small details
  - the “query assistance” features, which might appear in the North, South or West region of the page
  - the position of the navigational tools, which might or might not be displayed on the West region
  - the position of spelling correction recommendations, which might appear before of after the sponsored results in the North region
  - the position of ads, typically in the East region but sometimes also in the North and/or the South
- Search engines constantly experiment with small variations of layout

The Title/Snippet/URL Entity

- Major search engines use a very similar format to display individual results composed basically of
  - a title shown in blue and underlined
  - a short snippet consisting of two or three sentences extracted from the result page
  - a URL, that points to the page that contains the full text
- When a page does not have a title, anchor texts pointing to it can be used to generate a title

The Title/Snippet/URL Entity

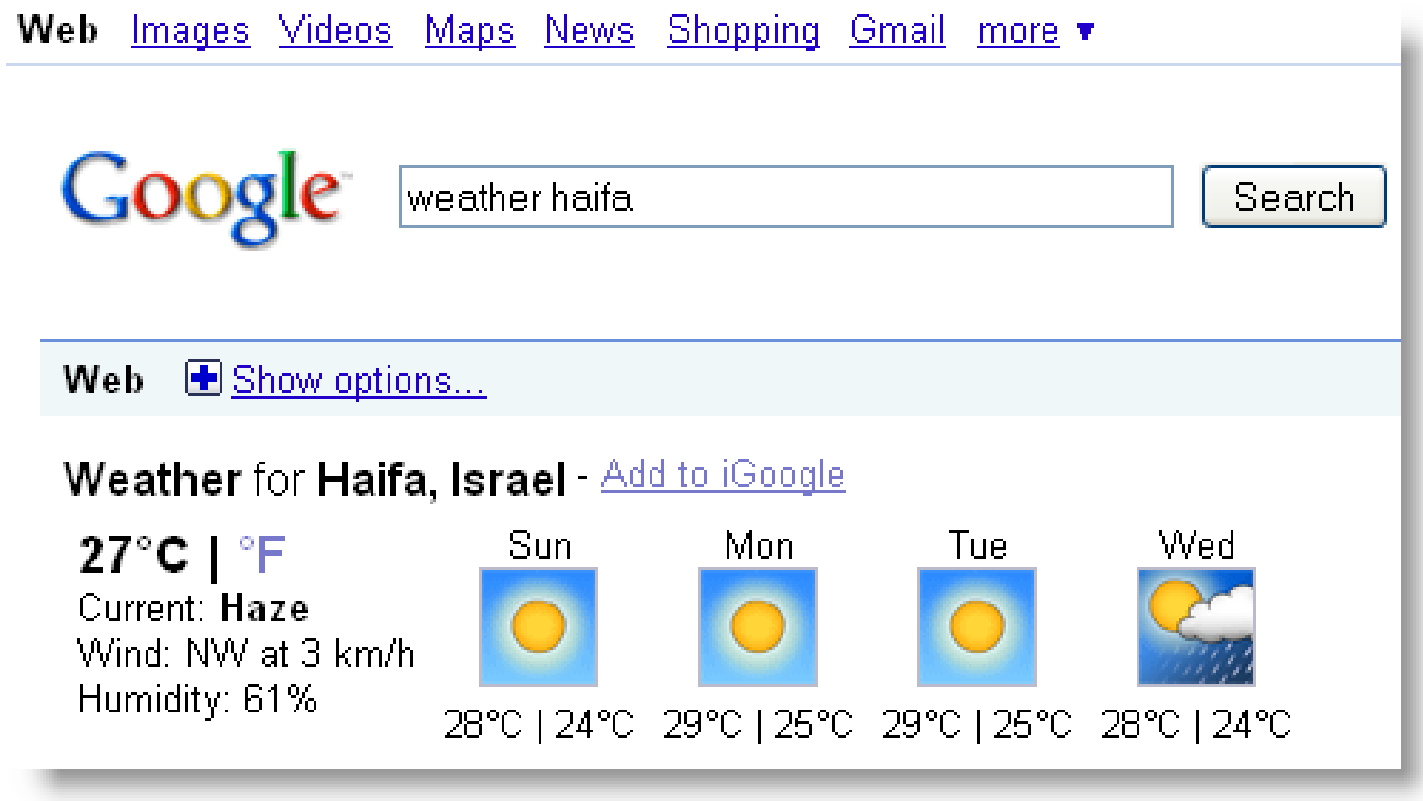
- **Snippets**
  - automatically generated excerpts aimed at highlighting topics in the page associated with the user's query
  - aim at facilitating the decision of whether to click on a link or not
  - key challenge: need to be generated at run time
  - important: query words are usually highlighted via bold fonts
- When several results originate from a same site or domain, search engines group them by indenting less relevant representatives of the same site
- A superior and more recent approach is the **Sitelink** or **Quicklink** format
  - navigational shortcuts are displayed below the Web site homepage on a search results page

More Structured Results

- In addition to results fed by the main Web corpus, search engines include additional types of results
  - **onebox results**
    - very specific results, produced in response to very precise queries, that are susceptible of having one unique answer
    - example: query **who is the governor of california** on Google
      - first result will show **California – Governor: Arnold Schwarzenegger**
  - **Universal search results:** other properties
    - images
    - videos
    - products
    - maps
    - all of which come with their own vertical search

More Structured Results

- Google Weather onebox: no need to click to get full answer



More Structured Results

- Web results can also appear in a different format
- Example from Ask.com
  - answers originating from Q&A sites like Yahoo! Answers or WikiAnswers are displayed directly without a snippet
- Common trend: show in slightly different format results originating from specific sources
  - part of the main Web corpus, or
  - serviced by other properties
  - example: SearchMonkey
    - Yahoo! Search open platform that allows publishers to *"share structured data with Yahoo! Search to display a standard enhanced result"*

More Structured Results

- **SearchMonkey**
  - launched in 2007
  - partly inspired on Peter Mika's earlier research on microformats
  - example: all Wikipedia results in Yahoo! are displayed in a SearchMonkey format
  - Google recently followed Yahoo!'s example
    - explored rich result format when it launched its rich snippets in 2009

More Structured Results

- Bing is investing a great deal of efforts in structured results, offering digest-type results for specific domains
- Some Bing's examples
  - **travel results** feature a fare trend indicator
  - **shopping results** include some convenient shortcuts to
    - users' and experts' reviews
    - product details
    - price comparison and the best price among results
    - ratings, ease of use, affordability, visual cues
    - the novel "Bing cashback"
  - **health results** indicate authoritative sources, such as the Mayo clinic
  - **local results** comprise review-based scored cards with visual cues for topics such as "overall", "atmosphere", etc

Query Assistance on the SERP

- Once users have looked at the results, their informational, navigational, and transactional needs may be
  - satisfied
  - partially satisfied
  - not satisfied
- The user need is satisfied, when
  - answer is produced directly from a onebox result, such as calculator, weather, sports results, or
  - user clicks on one or a few of the top results

Query Assistance on the SERP

- The user is partially satisfied, when
  - has undertaken a research task and there is no page that holds all the information
  - Yahoo!'s Search Pad has been designed precisely to gather, annotate, and organize partial answers into one coherent unit, which can then
    - be stored for later usage, or
    - published/shared with others
  - some needs are more susceptible to trigger research tasks
    - user is looking for hotels, restaurants, and entertainment opportunities
    - user has homework and is working on an assignment
    - user seeks health information on an illness, its symptoms and treatment options

Query Assistance on the SERP

- The user is not satisfied at all when
  - the query was not well formulated
  - relevant content simply does not exist
- it is still almost impossible for search engines to decide when relevant content does not exist
- by default, most engines assume the query was not well formulated and try to help users with reformulating the query

Spelling Assistance

- **Did you mean**
  - offered by Google, it is now famous
  - most successful example of query assistance
  - revolutionized spelling correction by departing from the usual dictionary-based model
- Classic approach was to use edit distances to identify typing mistakes such as letter inversions
- Instead, Did you mean learns the spelling corrections
  - simply from usage, a great deal of usage
  - it extensively uses query logs analysis for spelling
  - one example: query "Britney Spears"
    - query logs show name misspelled in hundreds of ways
    - yet, the most frequent spelling by far is simply the correct one

Spelling Assistance

- The example of "Britney Spears" illustrates the **wisdom of crowds** at its best
  - sheer frequency signal is less effective for long tail queries, or
  - in domains for which logs are not large enough (suffers from “small corpus challenge”)
  - in such cases, other signals can be used that require less evidence
  - example: Douglas Merrill, former Google CIO, in one of his Search 101 talks, explained that by simply observing users rephrase their queries in two successive queries, the engine can learn the correct spelling of a query
  - Cucerzan and Brill investigated this approach and showed how to learn query correction models from query reformulations in the query logs

Query Recommendations

- **Query recommendations** provide other means of query assistance on the SERP
- Typically consist of queries related in some sense to the original query
  - most useful when users struggle with expressing their needs
  - in this case, they tend to turn to related, hopefully better formulated, queries
  - differ from dynamic query suggestions provided in the search rectangle because can take advantage of richer types of information
    - full-formed queries (as opposed to partially specified one)
    - rich set of results, their snippets and relevance signals

Query Recommendations

- Research works on mining query logs to generate query recommendations are of three main categories
  - content-ignorant approaches
  - content-aware approaches
  - query-flow approaches

Content-ignorant Approaches

- **Content-ignorant approaches** for query recommendation
  - Well represented by Befferman and Berger work
    - infer similarity between queries from common clicked URLs
  - Impact of such methods is somehow limited
    - because number of clicks in results pages is relatively small
    - the associated query-to-query distance matrices remain sparse
    - this sparsity could be diminished though by using larger query logs if allowed by legislation

Content-aware Approaches

- **Content-aware approaches** for query recommendation rely on search results or target pages

1. Work done by Raghavan and Sever
   - attempted to measure query similarity by determining differences in the ordering of docs in results set
   - approach has the advantage of richer information as provided by the document collection
   - poses challenges in terms of scalability
2. Work done by Fitzpatrick and Dent
   - measured query similarity using the normalized set intersection of the top 200 results
   - technique suffered from scalability issues as the intersection of semantically similar queries that use different synonyms is typically very small

Content-aware Approaches

3. Work done by Sahami
   - used query similarity based on snippets of results
   - each snippet treated as a query
     - submitted to search engine to find docs that contain terms in the original snippets
     - returned documents used to create a context vector for the original snippet
   - works badly if the snippet comes from a Web spam page

Query-flow Approaches

- **Query-flow approaches** for query recommendation
  - consider the users' sequential search behavior to better understand query intent
  - Fonseca *et al* and Zhang *et al* are good examples of this school
- Work done by Fonseca *et al*
  - view query logs as a set of transactions
  - each transaction represents a **session** in which a single user submits a sequence of related queries in a given time interval
  - method shows good results, however two problems arise
    - first, it is difficult to determine sessions of successive queries that belong to the same search process
    - the most interesting related queries, those submitted by different users, cannot be discovered

Query-flow Approaches

- **Query-flow approaches** for query recommendation seem really promising
  - based on mining relations from the query flow
  - sessions are usually physical sessions and not logical sessions
  - four subsequent queries in a short time interval might be related to two drastically different tasks
- Recent attempts at formalizing the query flow graph should lead to better mining techniques

In Practice

- Most modern approaches use hybrid approaches for higher precision
- Work done by Baeza-Yates *et al*
  - use content of clicked Web pages to define a term-weight vector model for a query
  - consider terms in the URLs clicked after a query
  - each term is weighted according to number of occurrences of query and number of clicks of docs in which term appears
- Search engines do not communicate their methods
  - they use the “best of breed” and multiple signals
  - note the lack of unanimity on placement of these suggestions
  - this has direct impact on usage and indicates how much search engines trust their recommendation tools

Query Recommendations

- Google displays query recommendations under the label **Search related to:**
  - at the bottom of the SERP
  - arranged in four columns of two candidates
  - consequently, it can be expected that the click-through rate of this feature is relatively small
- **Tool belt**
  - recently launched search options feature of Google
  - provides access to “related searches” and to the original **wonder wheel**
    - gives a graphical representation of related search terms
    - clicking on any node of the wheel leads to related topics in the interactive animated wheel
    - during animation, results keep being updated on the right side

Query Recommendations

- Yahoo! Search also displays **related results** at several locations, such as
  - right below the search rectangle, under the label “Also try”
  - on the left navigation pane
  - within the search rectangle, side by side with regular dynamic query suggestions
    - which has a different scope than regular query suggestions
    - appear on the SERP search rectangle, only when user continues entering a query or voluntarily expands it
- Bing displays related results on the West region navigation pane and labels them as **related searches**

Query Refinement via Facets

- Queries can also be refined by restricting results along certain **facets**
- **Faceted search**
  - navigation mechanism: "enables users to navigate a multi-dimensional information space by combining text search with a progressive narrowing of choices in each dimension"
  - viewed here as a query refinement mechanism since, in practice, user has to select a facet
  - user-provided input augments query with additional information
    - better specify the user's needs
    - narrow the results set
  - Example research systems: Flamenco, Aquabrowser
  - Example vertical search services: Yelp.com, see Chapter 2

Query Refinement via Facets

- Faceted navigation on the Web, one approach
  - map attributes of the results, their type (video, audio) or source (Wikipedia, YouTube, Yahoo! answers), into navigational facets

West navigation pane shows relevant sources for narrowing query

Query Refinement via Facets

- Faceted navigation on the Web
  - Bing also uses a similar approach as Yahoo!

Query Refinement via Facets

- Faceted navigation on the Web
  - Google offers a similar functionality through its tool belt feature
  - User can “slice and dice” the results via various types of facets
    - type or source: “Video, Forums, Reviews” facets
    - time-based: “Past 24 hours”, “Past week” and “Past year” facets
  - While implementation details for this mechanism are not public, one can envision a simple implementation
    - index stores these static attributes
    - engine fetches and processes them at run time

Query Refinement via Facets

- Faceted navigation on the Web
  - A more complex case consists of displaying number of results in each facet
    - no Web search engine offers this feature yet
    - it has been offered in the past by
      - enterprise faceted search engines such as Endeca
      - multiple shopping sites
    - to estimate these counts within decent response time at the scale of the Web is not trivial
  - Even more complex cases have been investigated in research: **hierarchical, correlated, dynamic facets**
    - they require computing at run-time
    - also require associated visual interfaces to be determined at run-time

Actionable Results

- Variety of promising tools that allow to do more with search results than simply interpret or navigate them
- Some features simply operate on the result itself for various purposes
  - Yahoo! Search
  - Google “Cached” link and “Similar” link
- A more advanced feature is Google **“Translate this page”**
  - displayed next to the results title
  - provides a translation of the target page into the user default language
  - statistical translation techniques are used

Actionable Results

- Another tool is a **built-in search rectangle**
  - typically displayed below sitelinks/quicklinks
  - allows users to search within the site that the result belongs to
  - now is part of of several Web search engines
  - example: a search for New York Times on Google
    - leads to a link to the newspaper homepage and to one such associated search-within-site rectangle
    - issuing a query from this rectangle will augment the query with a “site:nytimes.com” qualifier

Actionable Results

- More intriguing tools include Google Stars in Search and Yahoo! Search Pad
- Before "Stars in Search", Google launched the more complex, and less successful, **Searchwiki**
  - allowed user to provide feedback on any result via three small icons displayed next to the result URL
    - a bubble for “comment”
    - an arrow up for “promote”
    - an “x” for “remove”
  - user could thus annotate, promote, and get rid of any results at will
  - user would see this personalization of results persist if they re-issued the same query in the future
  - user could visit their own Searchwiki notes at any time

Actionable Results

- Searchwiki was recently replaced by a leaner version called **Stars in Search**
  - launched in March 2010
  - three small icons were replaced by a single star that when selected turns yellow and acts as a sort of marker of favorites results
  - for subsequent similar searches, user will see previously starred results appear at the top of the results list in a special section
- Interesting lesson to remember: all search engines carefully monitor adoption and might modify or entirely retire features that have not gained enough traction

Actionable Results

- **Yahoo! Search Pad**
  - interesting feature that belongs to the same family as Google notebook, yet uses a different approach
  - allows users to easily keep trace of results they have consulted, and arrange and annotate them for later usage or for sharing with others
  - concept is not new, it was pioneered by Bharat
  - novelty is that Search Pad is triggered only when the search engine decides that the user is investigating a topic rather than looking for quick, “disposable” results
  - visited pages are automatically added to the appropriate search pad, without requiring the user to specifically “mark” them like in early research work, Ask “My Stuff” or the now discontinued Google Notebook

Educating the User

- We should expect users, especially youngsters, to
  - become more and more Internet savvy
  - take more control of the search process
- Advanced search interfaces allow better control
  - for more control, sophisticated users can specify as many terms as possible
  - they can also indicate which terms should be included in the results (via the “+” operator) and which ones should not (via the “-” operator)
  - the user can reduce the size of the result set by
    - restricting the search to a field (for example, the page title)
    - limiting some attributes (date, country)
    - using operators within the query

Educating the User

- Even if we are able to issue good queries, the result set can still be quite large
- To fix this the user must learn from experience
- There are many strategies to quickly find relevant answers
  - if users are looking for an institution, they can always try to guess the corresponding URL by using the `www` prefix, followed by a guessed institution acronym or brief name, and finished by a top level domain (country code or `com`, `edu`, `org`, `gov` for the US)
  - if this does not work, the user can search the institution name in a Web directory

Educating the User

- Another somewhat frequent task is for a user to search for published work on a specific topic
  - select an article related to the topic, if possible with non-common author surnames or title keywords
  - use a search engine to find all Web pages that have all those surnames and keywords
  - many of the results are likely to be relevant because they include references to
    - newer papers that reference the initial reference
    - personal Web pages of the authors
    - pages about the topic that point to many relevant references
- This strategy can be iterated by changing the reference used initially as better references appear during the search

Educating the User

- Key lessons here are
  - search engines still return too much hay together with the needle
  - Web directories do not have enough depth to find the needle
  - we recommend to use the following rules of thumb, when issuing queries
    - specific queries: look at an Encyclopedia, that is the reason that they exist, so do not forget libraries
    - broad queries: use Web directories to find good starting points
    - vague or exploratory queries and iterative refinements: use Web search engines and improve query formulation based on relevant answers