Effectiveness and Efficiency Optimizations in Web Search Engines

B. Barla Cambazoglu

Barcelona, Spain

Previous Versions of the Tutorial

- Book chapter
  - The Information Retrieval Series, 2011
- Previous tutorials
  - Conferences
    - SIGIR, 2013
    - SIGIR, 2014
    - WWW, 2014
    - WSDM, 2015
  - Summer schools
    - COST 804 Training School, 2012
    - MUMIA, 2014
- Upcoming
  - CIKM, 2015

Outline of the Tutorial

- Background on web search
- Main sections
  - web crawling
  - indexing
  - query processing
- Concluding remarks
- Questions and open discussion

Background on Web Search

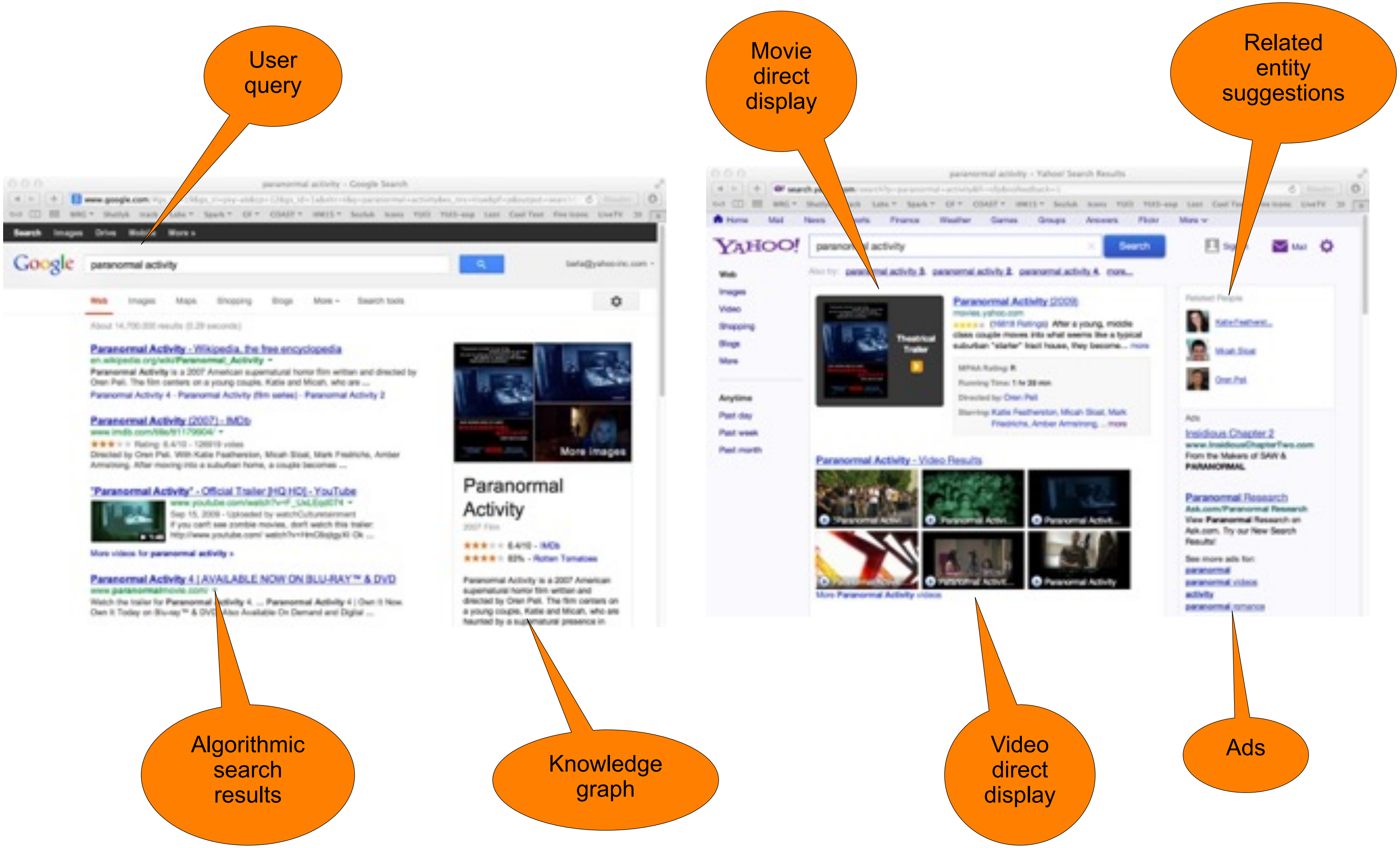
Brief History of Search Engines

- Past
  - Before browsers
    - Gopher
  - Before the bubble
    - Altavista
    - Lycos
    - Infoseek
    - Excite
    - HotBot
  - After the bubble
    - Google
    - Yahoo
    - Microsoft

- Current
  - Global
    - Google, Bing
  - Regional
    - Yandex, Baidu

- Future
  - Facebook ?
  - Apple ?
  - …

Anatomy of a Search Engine Result Page



Actors in Web Search

- User's perspective: accessing information
  - high-quality search results
  - fast response to queries

- Search engine's perspective: monetization
  - attract more users
  - increase the ad revenue
  - reduce the operational costs

- Advertiser's perspective: publicity
  - attract more users to their site
  - pay little

What Makes Web Search Difficult?

- Size

- Dynamicity

- Diversity

- All of these three features can be observed in
  - the Web
  - web users

What Makes Web Search Difficult?

- The Web
  - more than 190 million Web servers and 700 million host names
  - the largest data repository (estimated as 100 billion pages)
  - constantly changing
  - diverse in terms of content and data formats

- Users
  - too many
  - diverse in terms of their culture, education, and demographics
  - very short queries (hard to understand the intent)
  - changing information needs
  - little patience (few queries posed & few answers seen)

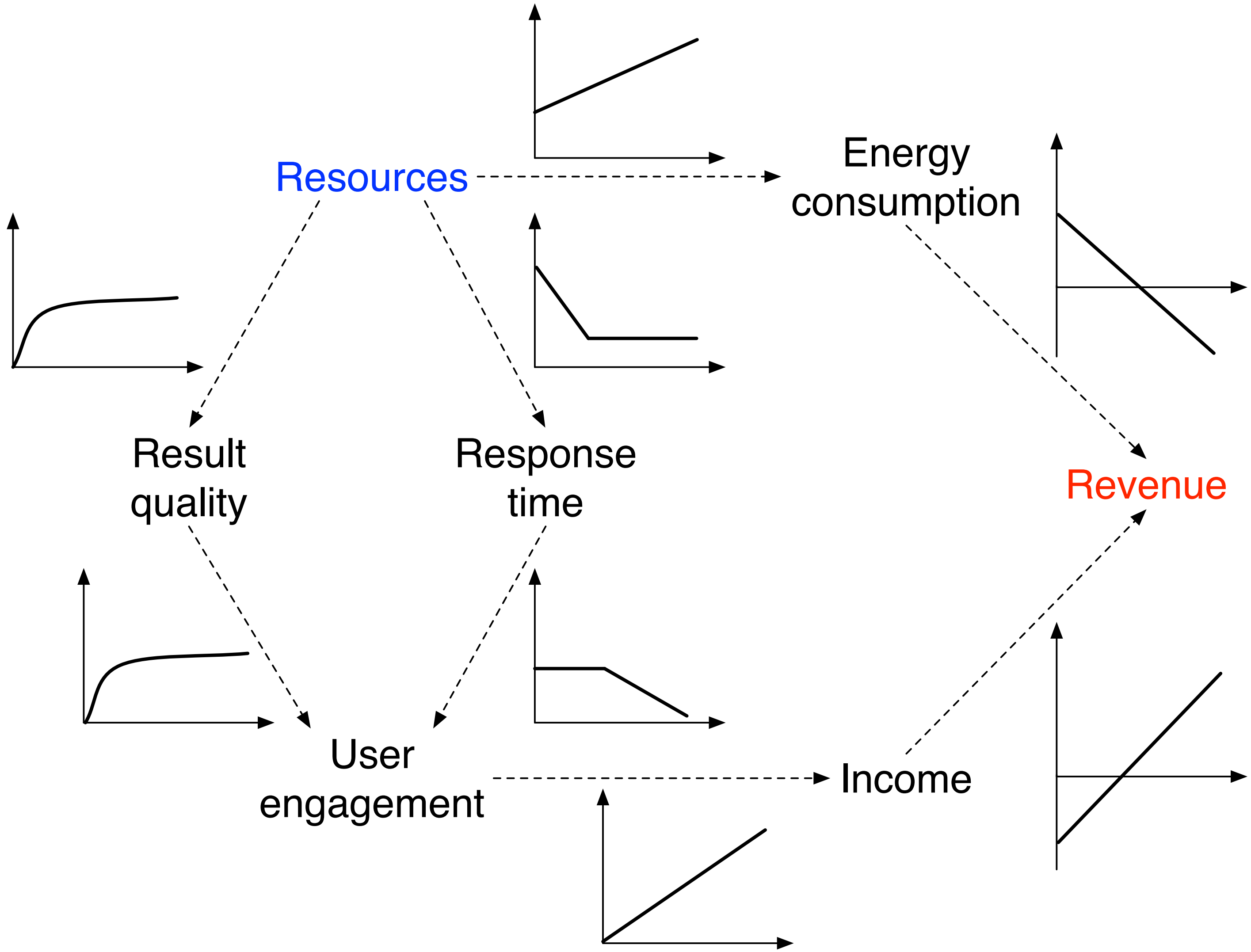
Expectations from a Search Engine

- Crawl and index a large fraction of the Web
- Maintain most recent copies of the content in the Web
- Scale to serve hundreds of millions of queries every day
- Evaluate a typical query under several hundred milliseconds
- Serve high-quality results for a given user query

Search Infrastructure

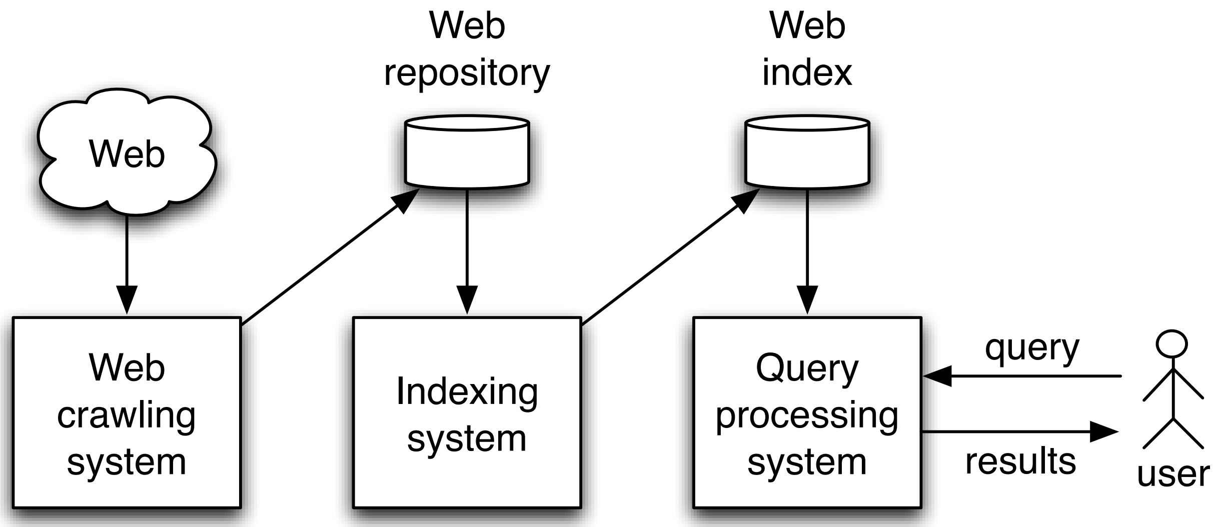
- Quality and performance requirements imply large amounts of compute resources, i.e., very large data centers
- High variation in data center sizes
  - hundreds of thousands of computers
  - a few computers

Role of Result Quality and Efficiency



Architectural Components

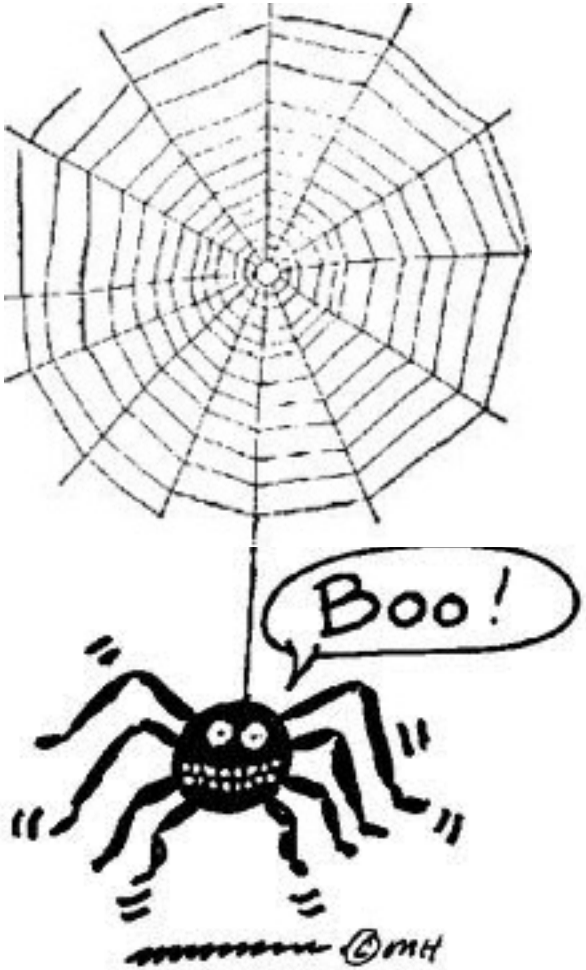
- Web crawling
- Indexing
- Query processing



Structure of the Main Sections

- Definitions
- Success measures
- Issues and techniques
  - single node
  - multiple node (cluster of computers)
  - multiple data centers (multi-site search engine)

Web Crawling

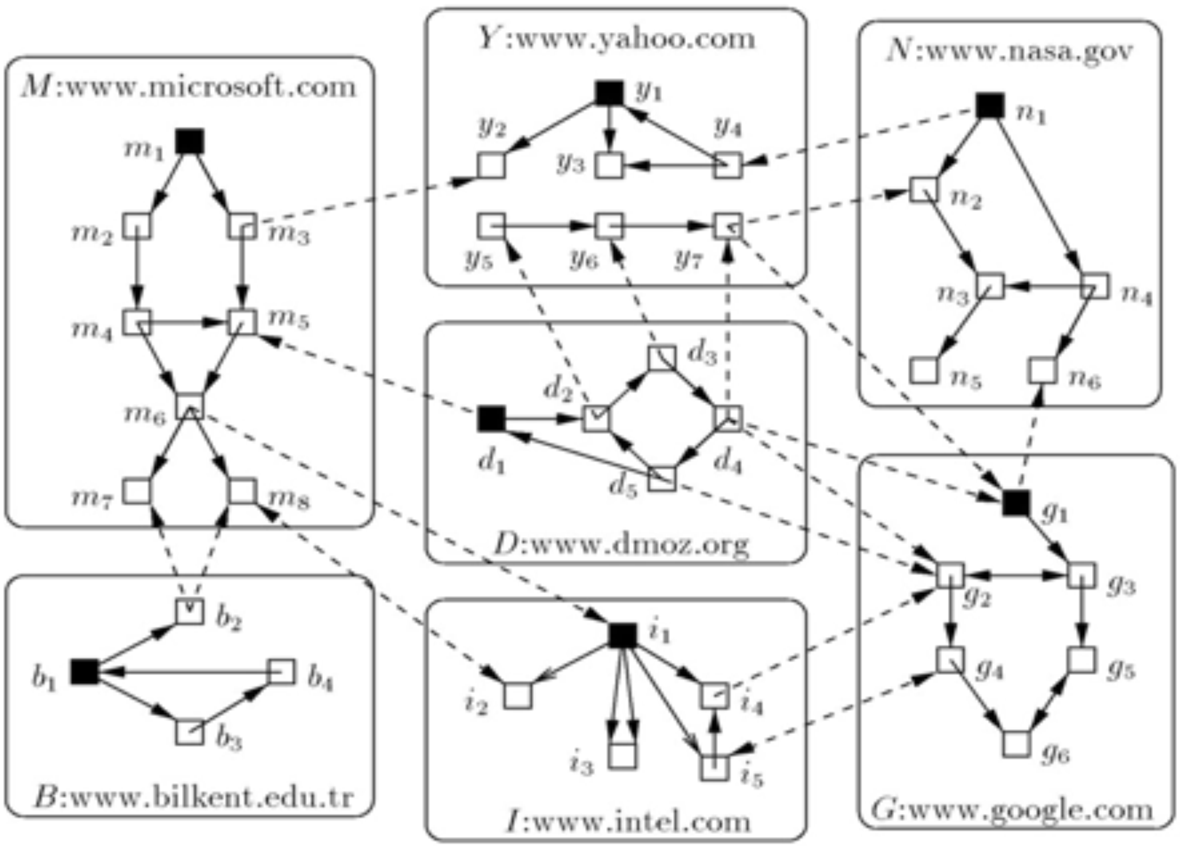


Web Crawling

- Web crawling is the process of locating, fetching, storing, and maintaining the pages available in the Web.
- Computer programs that perform this task are referred to as
  - crawlers
  - spider
  - harvesters
- Web crawler repositories
  - cache the online content in the Web
  - provide quick access to the physical copies of pages in the Web
  - help to speed up the indexing process

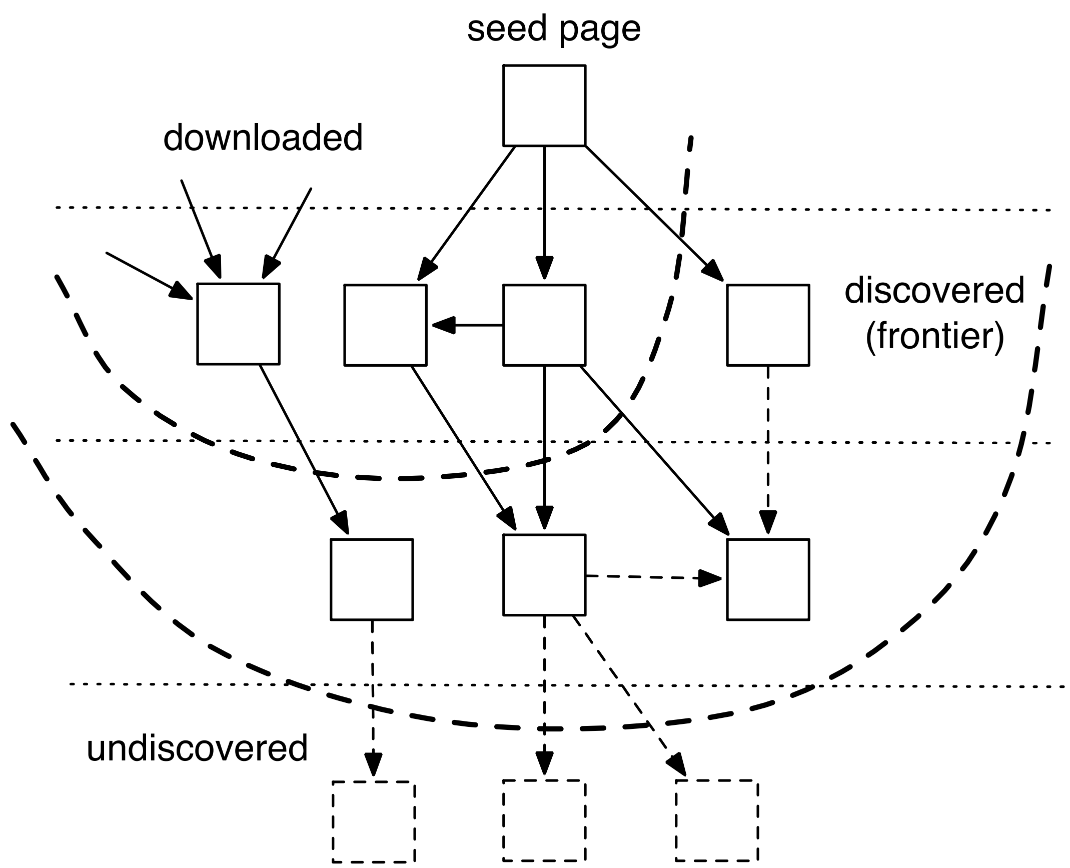
Web Graph

- Web crawlers exploit the hyperlink structure of the Web to discover new web pages



Incremental Web Crawling

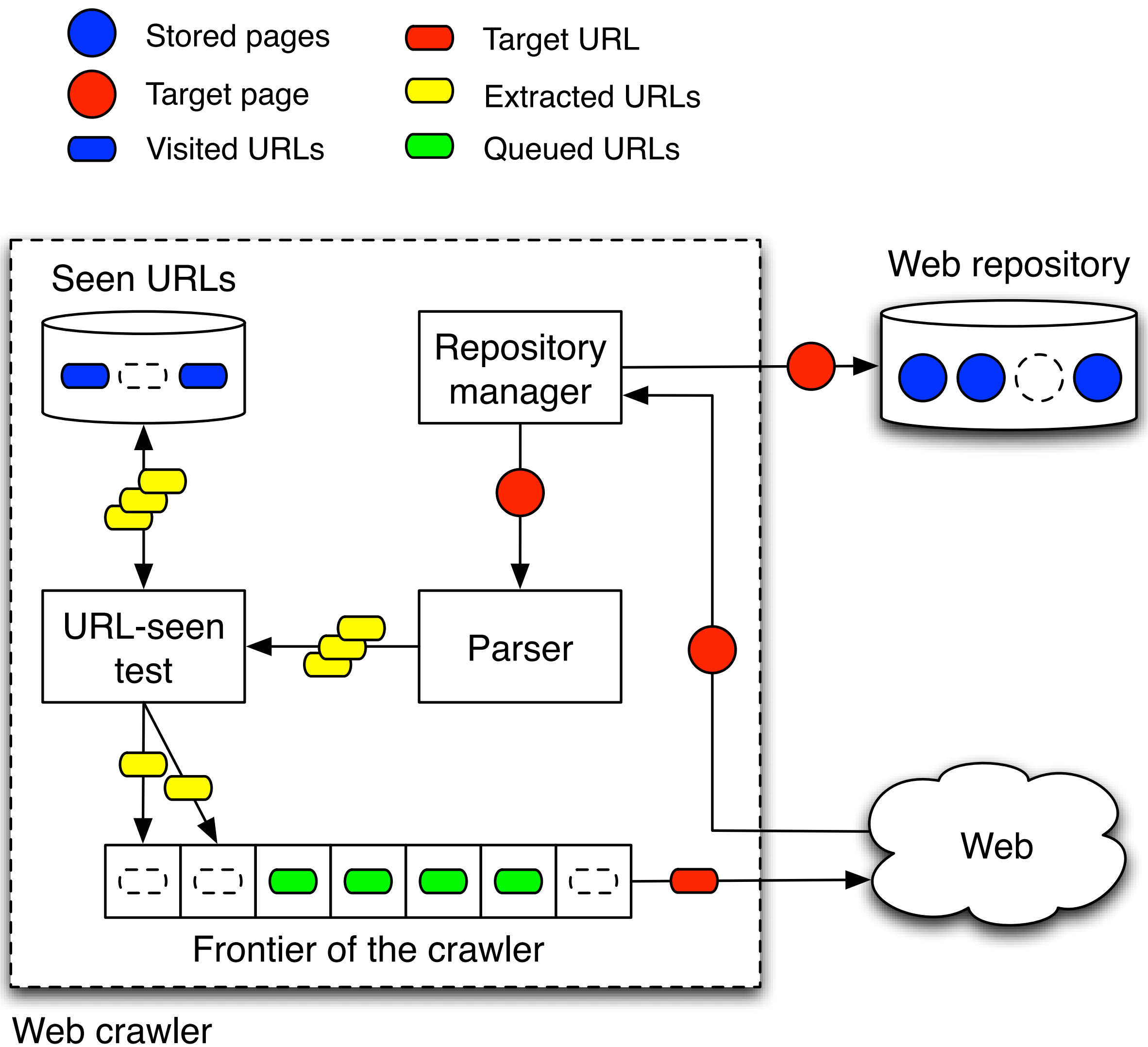
- Crawling process divides the Web into three subsets
  - downloaded
  - discovered
  - undiscovered



Web Crawling

- A commercial web crawler maintains two separate download queues
  - discovery queue
    - downloads pages pointed by already discovered links
    - tries to increase the coverage of the crawler
  - refreshing queue
    - redownloads already downloaded pages
    - tries to increase the freshness of the repository

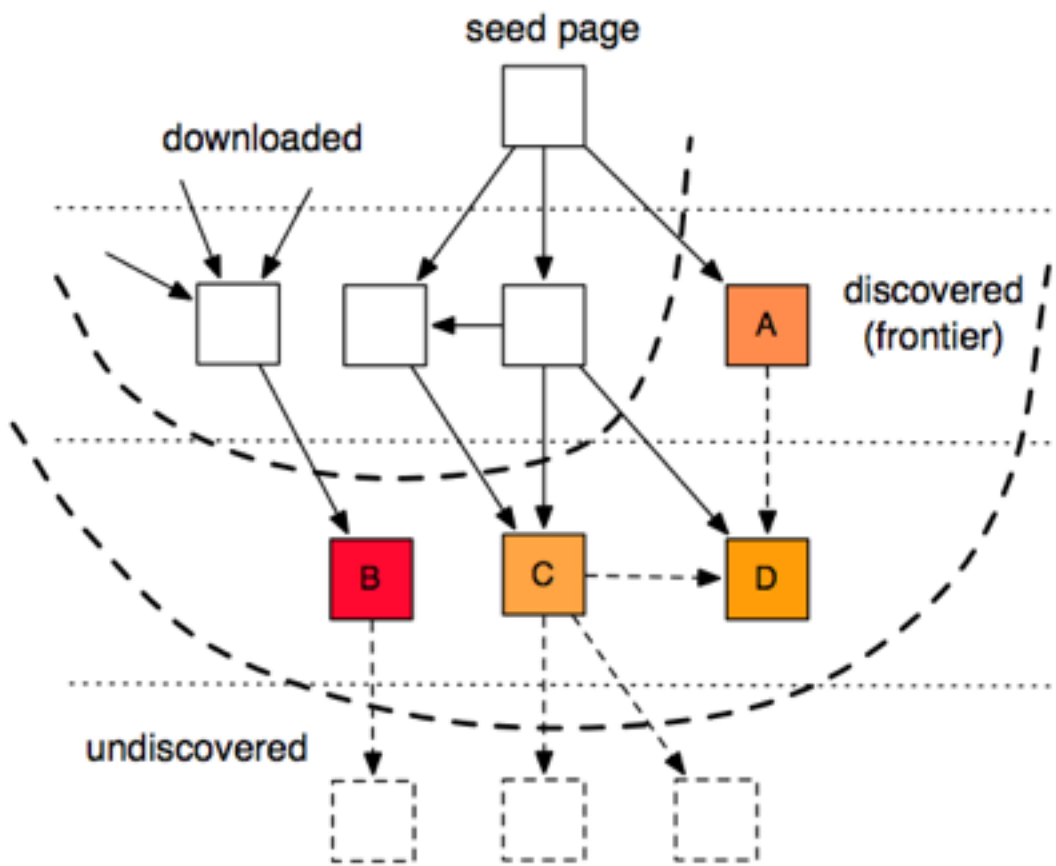
Discovery



Stored pages
Target page
Visited URLs
Target URL
Extracted URLs
Queued URLs
Seen URLs
Repository manager
Web repository
URL-seen test
Parser
Web
Frontier of the crawler
Web crawler

URL Prioritization

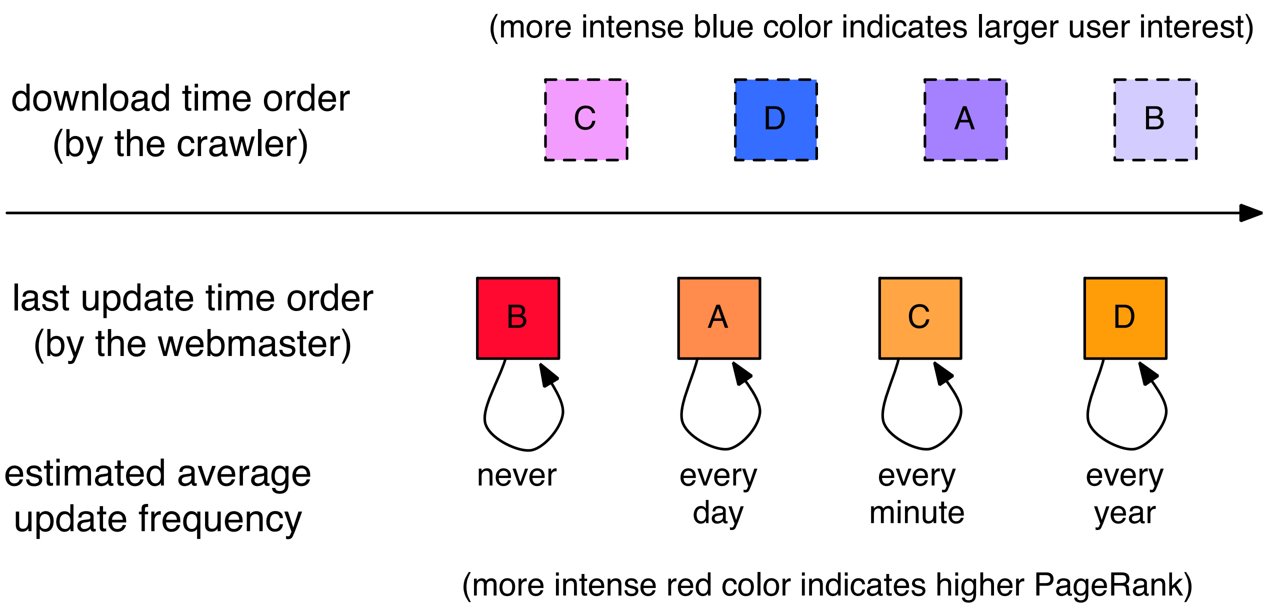
- Random (A, B, C, D)
- Breadth-first (A)
- In-degree (C)
- PageRank (B)



(more intense red color indicates higher PageRank)

Refreshing

- Random (A, B, C, D)
- PageRank (B)
- Age (C)
- User interest (D)
- Longevity (A)



Success Measures

- Quality measures
  - web coverage: percentage of the Web discovered or downloaded by the crawler
  - repository quality: percentage of useful pages in the repository
  - repository freshness: outdatedness of the local copies of pages relative to the pages' original copies on the Web
- Performance measures
  - download rate: number of bytes downloaded per unit of time

Issues in Web Crawling

- Dynamic nature of the Web
  - web growth
  - content change
- Malicious intent
  - hostile sites (e.g., spider traps, infinite domain name generators)
  - spam sites (e.g., link farms)
  - delay attacks
- Web site properties
  - unstable sites (e.g., variable host performance, unreliable networks)
  - sites with restricted content (e.g., robot exclusion),
  - soft 404 pages

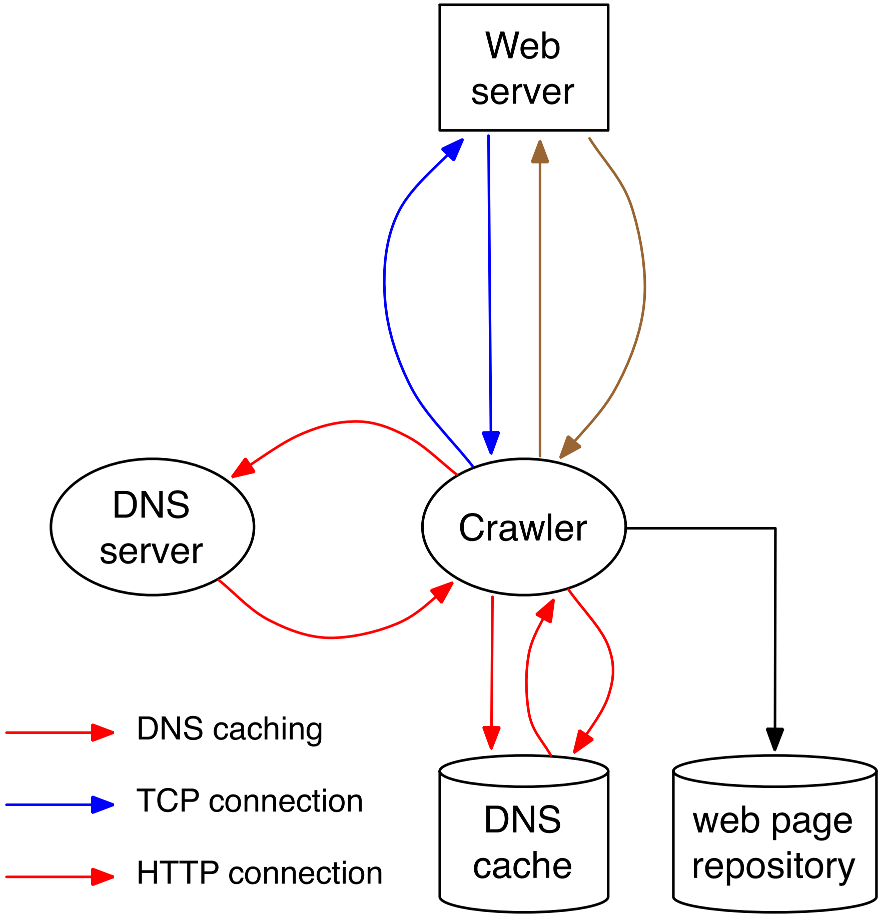
Implementation Issues

- DNS caching
- Multi-threading
- Politeness
- Robot exclusion protocol
- Mirror sites
- Data structures

DNS Caching

- Before a web page is crawled, the host name needs to be resolved to an IP address

- Since the same host name appears many times, DNS entries are locally cached by the crawler



Multi-threading

- Multi-threaded crawling
  - crawling is a network-bound task
  - crawlers employ multiple threads to crawl different web pages simultaneously, increasing their download rate significantly
  - in practice, a single node can run around up to a hundred crawling threads
  - multi-threading becomes infeasible when the number of threads is very large due to the overhead of context switching

Politeness

- Multi-threading leads to politeness issues
- If not well-coordinated, the crawler may issue too many download requests at the same time, overloading
  - a web server
  - an entire sub-network
- A polite crawler
  - closes the established TCP-IP connection after the web page is downloaded from the server
  - puts a delay between two consecutive downloads from the same server (a commonly used delay is 20 seconds)

Robot Exclusion Protocol

- A standard from the early days of the Web
- A file (called robots.txt) in a web site advising web crawlers about which parts of the site should not be crawled
- Crawlers often cache robots.txt files for efficiency purposes

```
User-agent: googlebot          # all services
Disallow: /private/            # disallow this directory

User-agent: googlebot-news     # only the news service
Disallow: /                    # on everything

User-agent: *                  # all robots
Disallow: /something/          # on this directory

User-agent: *                  # all robots
Crawl-delay: 10                # wait at least 10 seconds

Disallow: /directory1/              # disallow this directory
Allow: /directory1/myfile.html      # allow a subdirectory

Host: www.example.com          # use this mirror
```

Mirror Sites

- A mirror site is a replica of an existing site, used to reduce the network traffic or improve the availability of the original site
- Mirror sites lead to redundant crawling and, in turn, reduced discovery rate and coverage for the crawler
- Mirror sites can be detected by analyzing
  - URL similarity
  - link structure
  - content similarity

Data Structures

- Good implementation of data structures is crucial for the efficiency of a web crawler

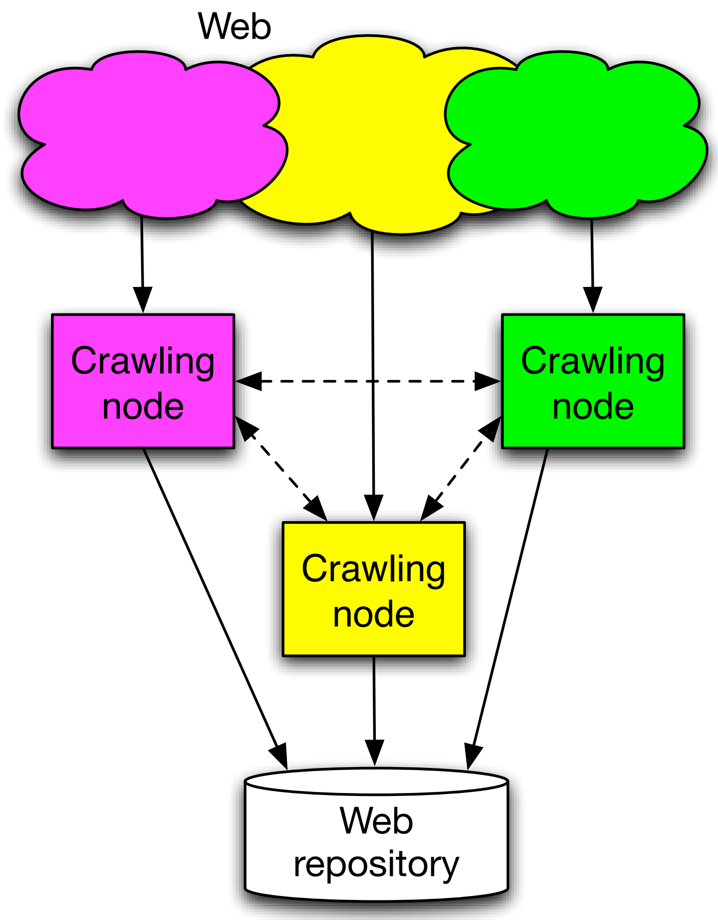
- The most critical data structure is the “seen URL” table
  - stores all URLs discovered so far and continuously grows as new URLs are discovered
  - consulted before each URL is added to the discovery queue
  - has high space requirements (mostly stored on the disk)
    - URLs are stored as MD5 hashes
    - frequent/recent URLs are cached in memory

Crawling Architectures

- Single node
  - CPU, RAM, and disk becomes a bottleneck
  - not scalable
- Multiple nodes
  - parallel crawler in a single data center
  - scalable
- Geographically distributed
  - parallel crawlers in multiple data centers
  - scalable
  - reduces the network latency

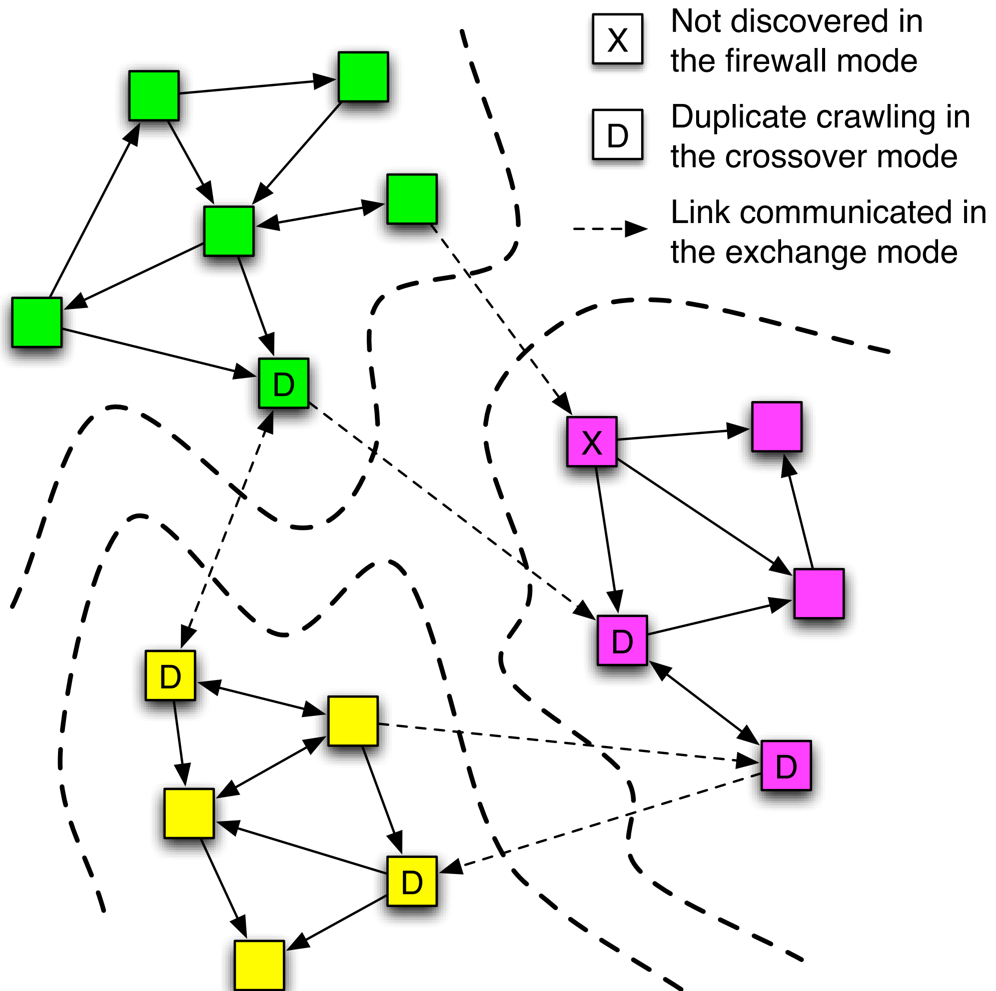
Parallel Web Crawling

- Web partitioning
  - typically based on the MD5 hashes
    - URLs
    - host names
  - host-based partitioning is preferable because URL-based partitioning may lead to politeness issues if the crawling decisions given by individual nodes are not coordinated



Coordination Between Nodes

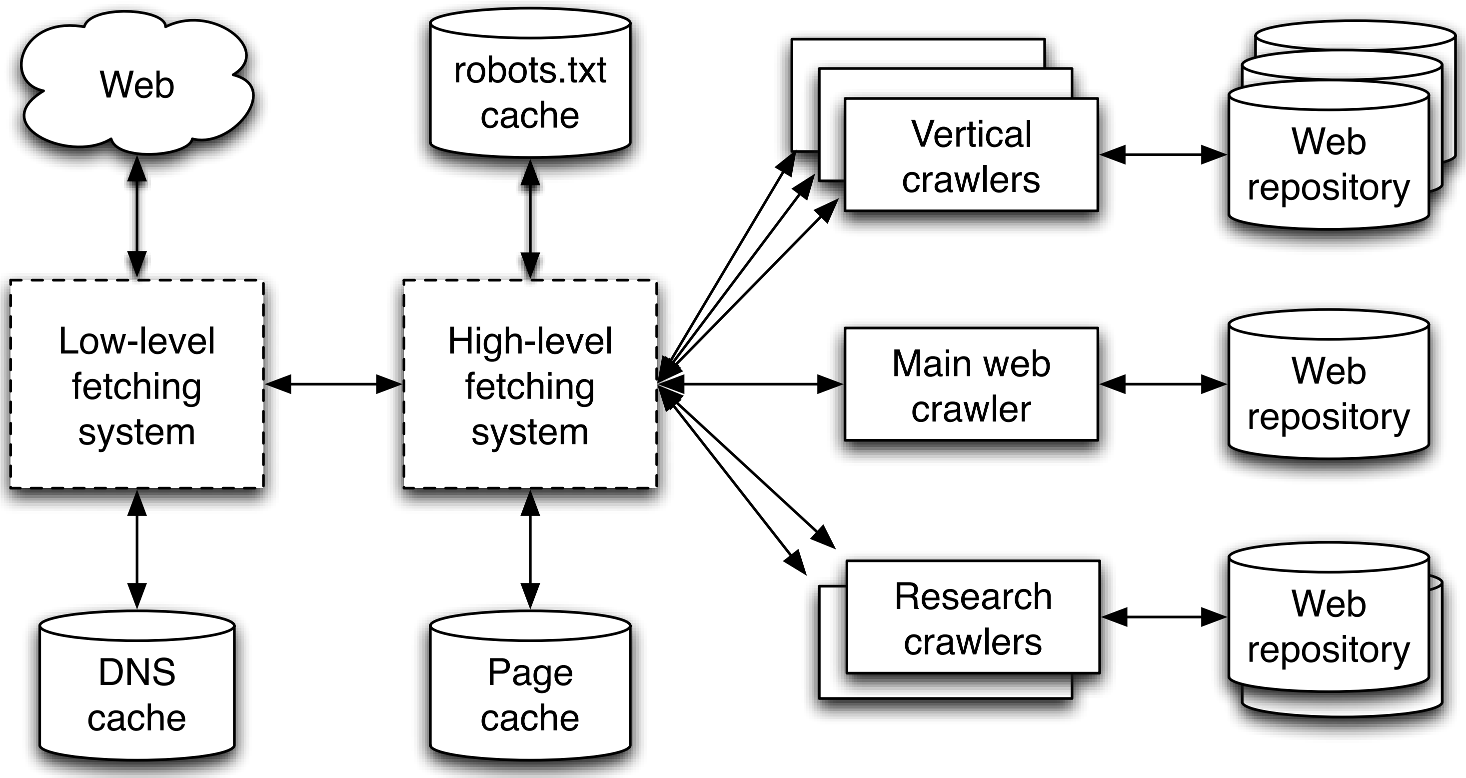
- Firewall mode
  - lower coverage
- Crossover mode
  - duplicate pages
- Exchange mode
  - communication overhead



Geographically Distributed Web Crawling

- Benefits
  - higher crawling throughput
    - geographical proximity
    - lower crawling latency
  - improved network politeness
    - less overhead on routers
  - resilience to network partitions
    - better coverage
  - increased availability
    - continuity of business

Common Fetching Infrastructure



Published Web Crawler Architectures

- Bingbot: Microsoft's Bing web crawler
- FAST Crawler: Used by Fast Search & Transfer
- Googlebot: Web crawler of Google
- PolyBot: A distributed web crawler
- RBSE: The first published web crawler
- WebFountain: A distributed web crawler
- WebRACE: A crawling and caching module
- Yahoo Slurp: Web crawler used by Yahoo Search

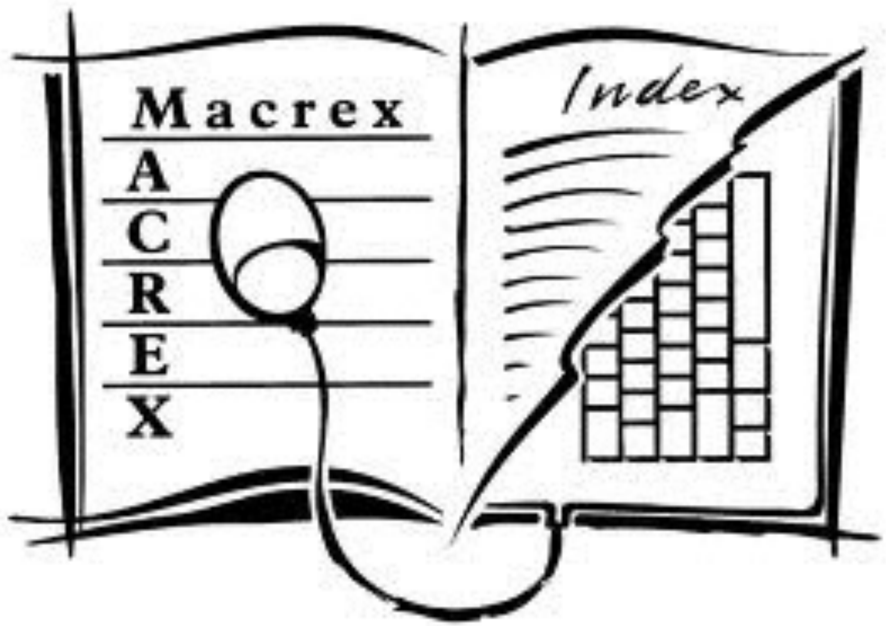
Open Source Web Crawlers

- DataparkSearch: GNU General Public License.
- GRUB: open source distributed crawler of Wikia Search
- Heritrix: Internet Archive's crawler
- ICDL Crawler: cross-platform web crawler
- Norconex HTTP Collector: licensed under GPL
- Nutch: Apache License
- Open Search Server: GPL license
- PHP-Crawler: BSD license
- Scrapy: BSD license
- Seeks: Affero general public license

Key Papers

- Cho, Garcia-Molina, and Page, "Efficient crawling through URL ordering", WWW, 1998.
- Heydon and Najork, "Mercator: a scalable, extensible web crawler", World Wide Web, 1999.
- Chakrabarti, van den Berg, and Dom, "Focused crawling: a new approach to topic-specific web resource discovery", Computer Networks, 1999.
- Najork and Wiener, "Breadth-first crawling yields high-quality pages", WWW, 2001.
- Cho and Garcia-Molina, "Parallel crawlers", WWW, 2002.
- Cho and Garcia-Molina, "Effective page refresh policies for web crawlers", ACM Transactions on Database Systems, 2003.
- Lee, Leonard, Wang, and Loguinov, "IRLbot: Scaling to 6 billion pages and beyond", ACM TWEB, 2009.

Indexing



Indexing

- The indexing system performs several tasks
  - performs information extraction, filtering, and classification on downloaded web pages
  - provides meta-data, metrics, and other kinds of feedback to the crawling and query processing systems
  - converts the pages in the web repository into appropriate index structures that facilitate searching the textual content of pages.

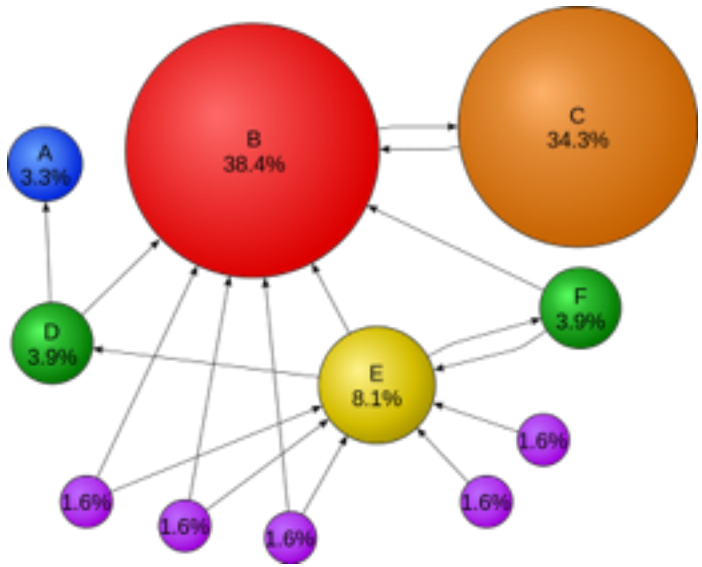
Document Processing Pipelines

- An indexing system involves various document processing pipelines, each performing different tasks on web pages

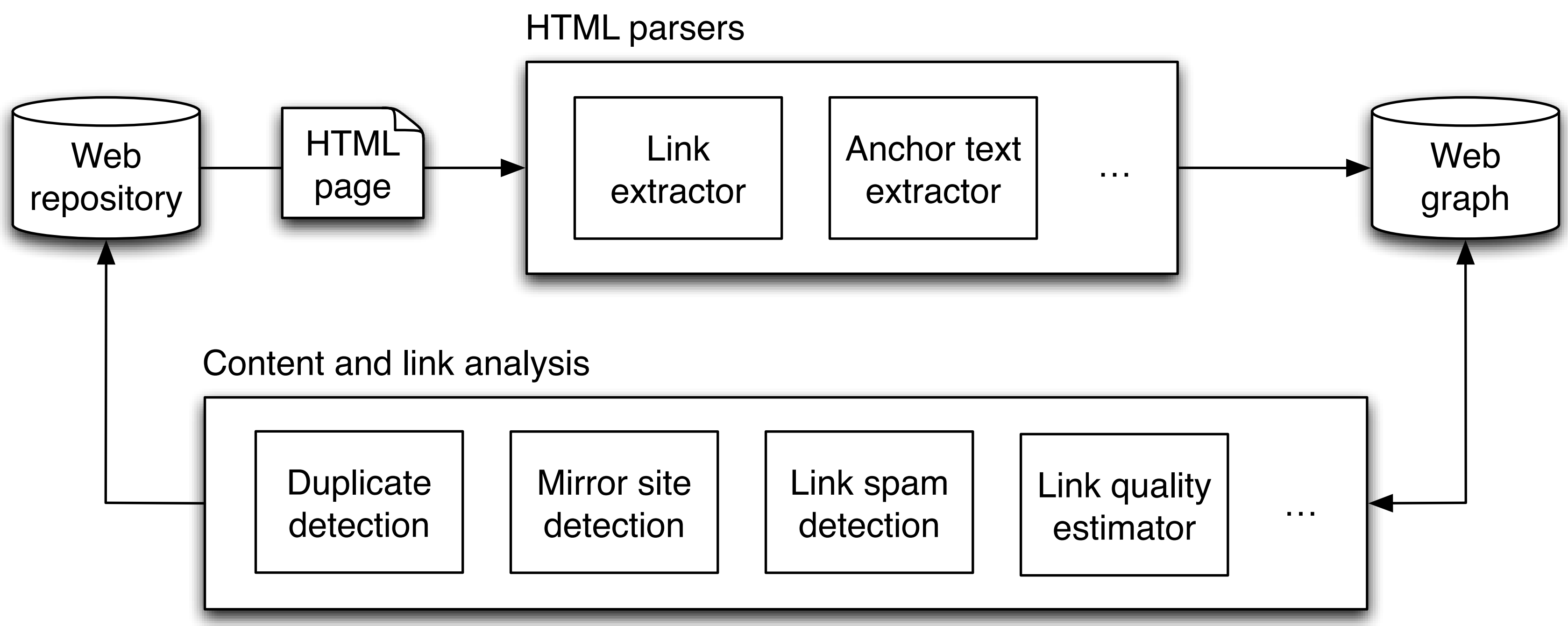
- Common data structures generated by these pipelines
  - web graph
  - page attributes
  - inverted index

Web Graph

- Web graph
  - node: attributes about the page
    - URL
    - inbound/outbound links
    - geographical region
    - language
  - edges: attributes about the links
    - anchor text
- Built at different granularities
  - page-level: duplicate detection
  - host-level: host quality estimation
  - site-level: mirror site detection



Web Graph


HTML parsers
Web repository
HTML page
Link extractor
Anchor text extractor
…
Web graph
Content and link analysis
Duplicate detection
Mirror site detection
Link spam detection
Link quality estimator
…

Link Analysis

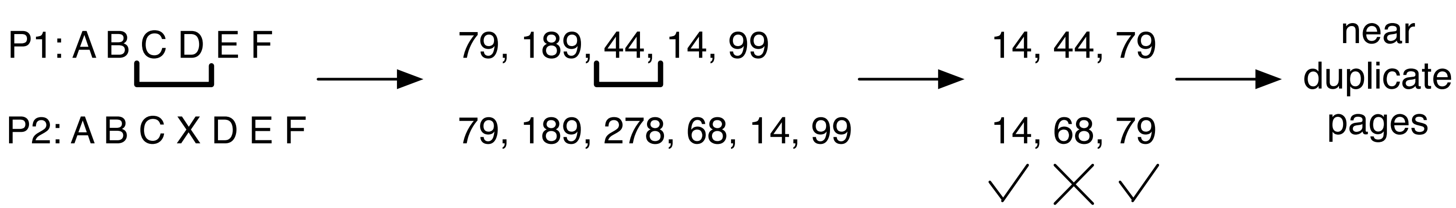
- PageRank: A link analysis algorithm that assigns a weight to each web page indicating its importance
- Iterative process that converges to a unique solution
- Weight of a page is proportional to
  - number of inbound links of the page
  - weight of linking pages
- Other algorithms
  - HITS
  - TrustRank

Spam Detection

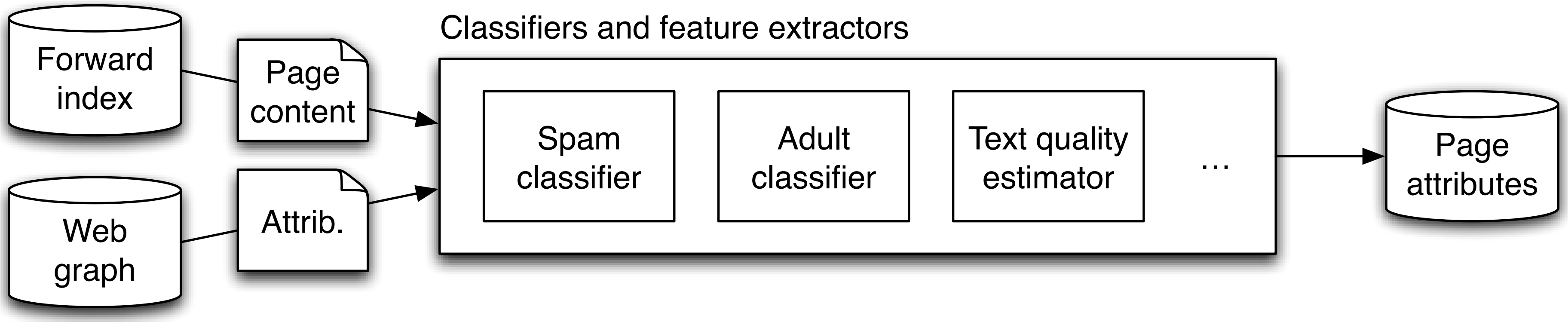
- Types of spam: link spam, content spam, cloaking/redirection spam, click spam

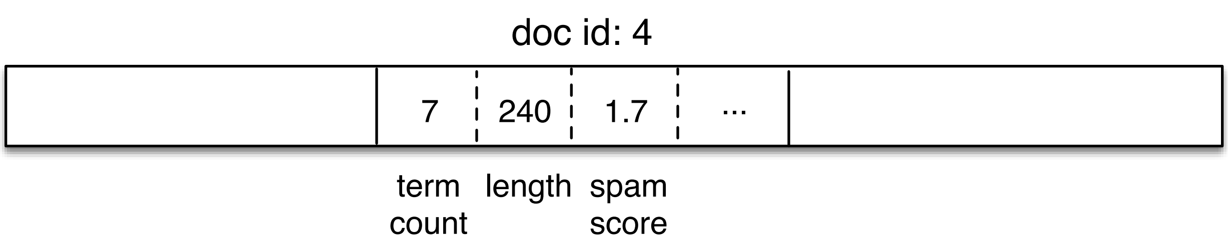
Duplicate Page Detection

- Detecting pages that have duplicate content
  - exact duplicates (solution: computing/comparing hash values)
  - near duplicates
    - locality sensitive hashing
    - shingles



Page Attributes

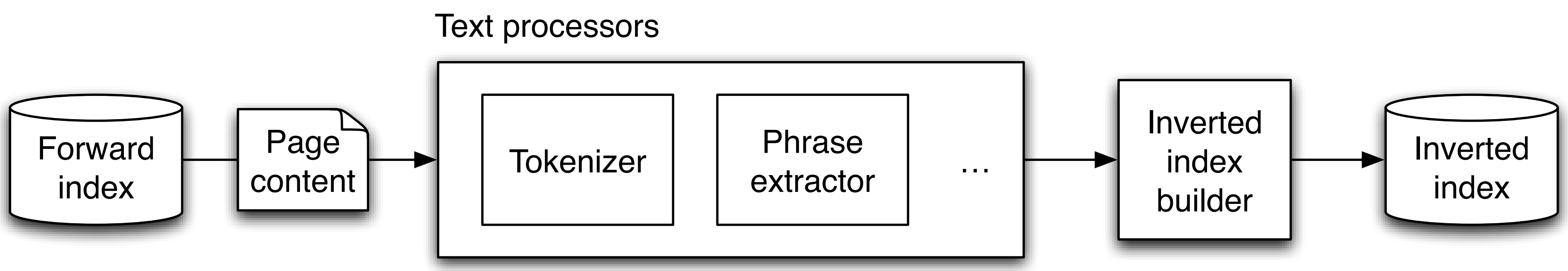




Query-Independent Features

Feature	Source	Description
Content spam	Page content	Score indicating the likelihood that the page content is spam
Text quality	Page content	Score combining various text quality features (e.g., readability)
Link quality	Web graph	Page importance estimated based on page’s link structure
CTR	Query logs	Observed click-through rate of the page in search results (if available)
Dwell time	Query logs	Average time spent by the users on the page
Page load time	Web server	Average time it takes to receive the page from the server
URL depth	URL string	Number of slashes in the absolute path of the URL

Inverted Index



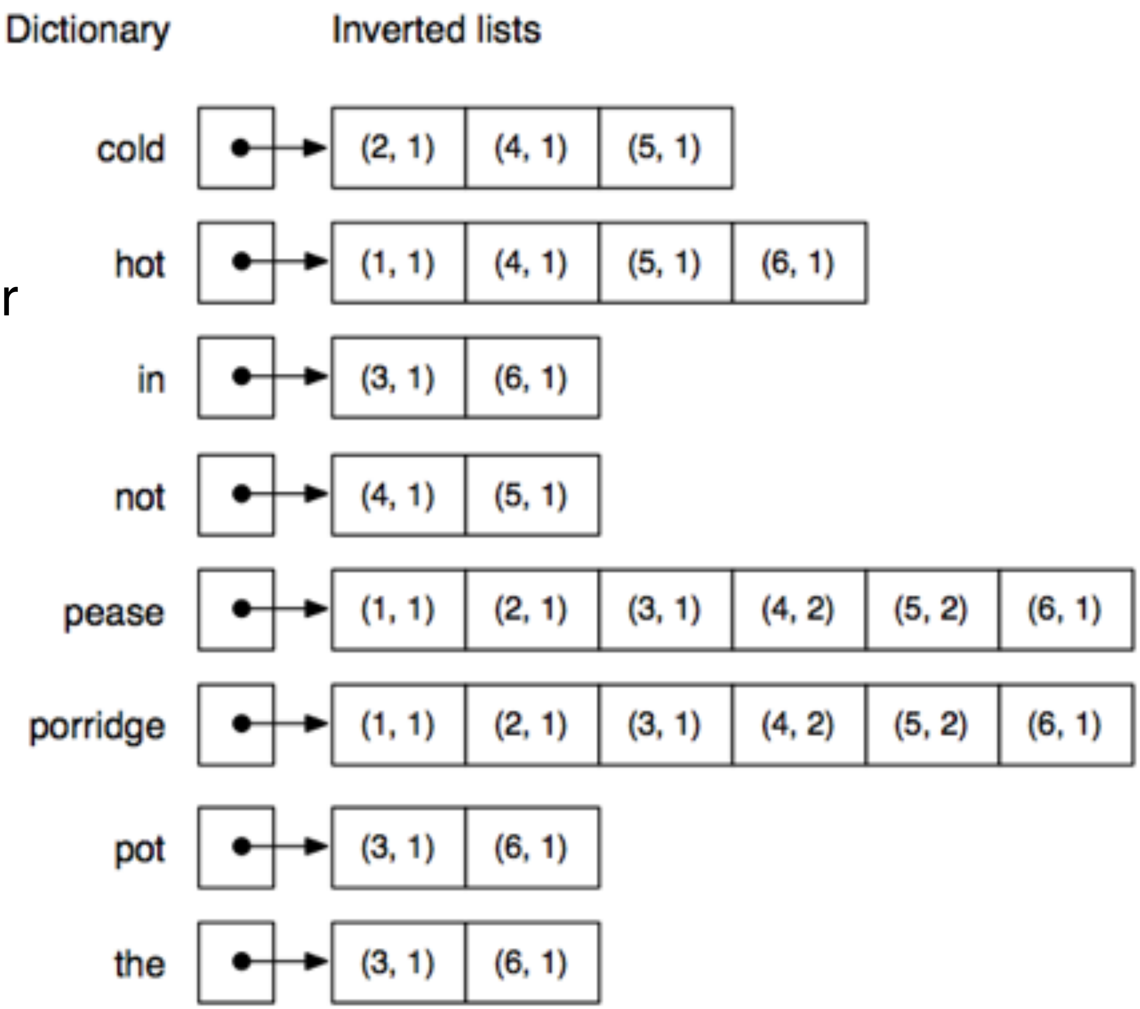
- Text processing may involve
  - tokenization
  - stopword removal
  - case conversion
  - stemming
- Example
  - original text: *Living in America*
  - applying all: *liv america*
  - in practice: *living in america*

Sample Document Collection

Doc id	Text
1	pease porridge hot
2	pease porridge cold
3	pease porridge in the pot
4	pease porridge hot, pease porridge not cold
5	pease porridge cold, pease porridge not hot
6	pease porridge hot in the pot

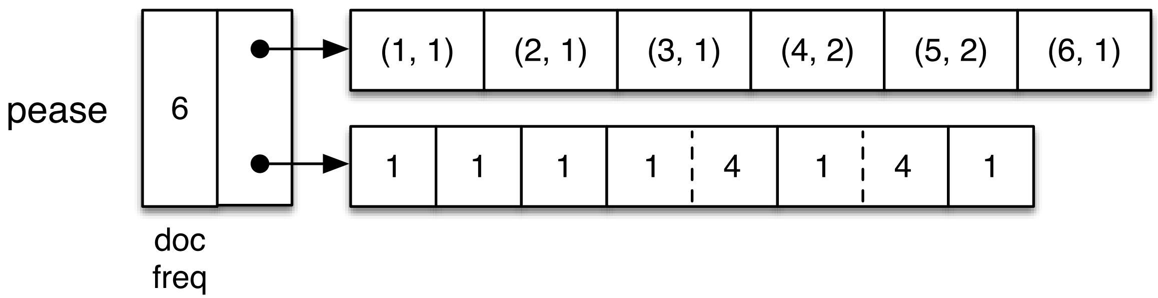
Inverted Index

- An inverted index is a representation for the document collection over which user queries are evaluated.
- It has two parts
  - a vocabulary index (dictionary)
  - inverted lists
    - document id
    - term information

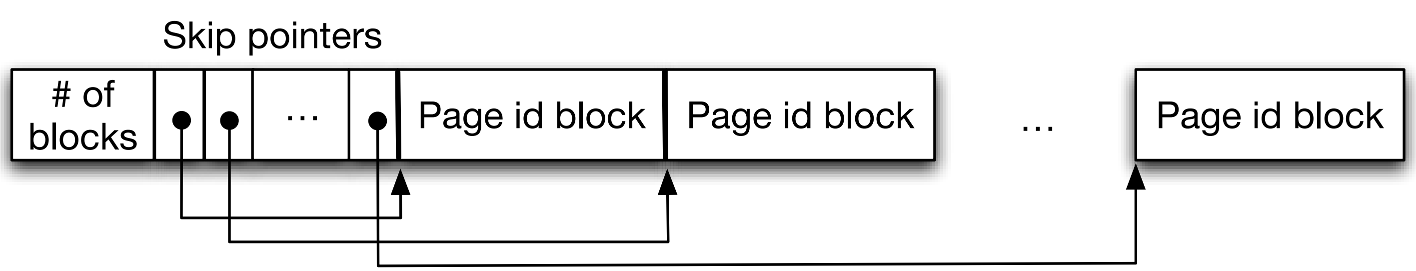


Extended Inverted Index

- Extensions
  - position lists: list of all positions a term occurs in a document



  - skipping



  - title, body, header, anchor text (inbound, outbound links)

Success Measures

- Quality measures
  - spam rate: fraction of spam pages in the index
  - duplicate rate: fraction of near duplicate web pages in the index
- Performance measures
  - compactness: size of the index in bytes
  - deployment cost: effort needed to create and deploy a new inverted index from scratch
  - update cost: time and space overhead of updating a document entry in the index

Inverted List Compression

- Benefits
  - reduced space consumption
  - reduced transfer costs
  - increased posting list cache hit rate

- Gap encoding
  - original: 17 18 28 40 44 47 56 58
  - gap encoded: 17 $\underbrace{1 \quad 10 \quad 12 \quad 4 \quad 3 \quad 9 \quad 2}_{\text{gaps}}$

Compression Algorithms

Compression algorithm	Input sequence	Output	Parameters	Encoded values
Unary	d-gaps	bit-aligned	non-parametric	individual values
Gamma	d-gaps	bit-aligned	non-parametric	individual values
Delta	d-gaps	bit-aligned	non-parametric	individual values
Variable byte	d-gaps	byte-aligned	non-parametric	individual values
Golomb	d-gaps	bit-aligned	parametric	individual values
Simple-9	d-gaps	word-aligned	parametric	blocks of values
PForDelta	d-gaps	bit-aligned	parametric	blocks of values
Binary interpolation	monotonic sequences	bit-aligned	parametric	bisections
Elias-Fano	monotonic sequences	bit-aligned	parametric	entire sequence

Document Identifier Reordering

- Goal: reassign document identifiers so that we obtain many small d-gaps, facilitating compression

Id mapping:

1 → 1
2 → 9
3 → 2
4 → 7
5 → 8
6 → 3
7 → 5
8 → 6
9 → 4

Original lists:

L1: 1, 3, 6, 8, 9 L2: 2, 4, 5, 6, 9 L3: 3, 6, 7, 9

Original d-gaps:

L1: 2, 3, 2, 1 L2: 2, 1, 1, 3 L3: 3, 1, 2

Reordered lists:

L1: 1, 2, 3, 4, 6 L2: 3, 4, 7, 8, 9 L3: 2, 3, 4, 5

New d-gaps:

L1: 1, 1, 1, 2 L2: 1, 3, 1, 1 L3: 1, 1, 1

Document Identifier Reordering

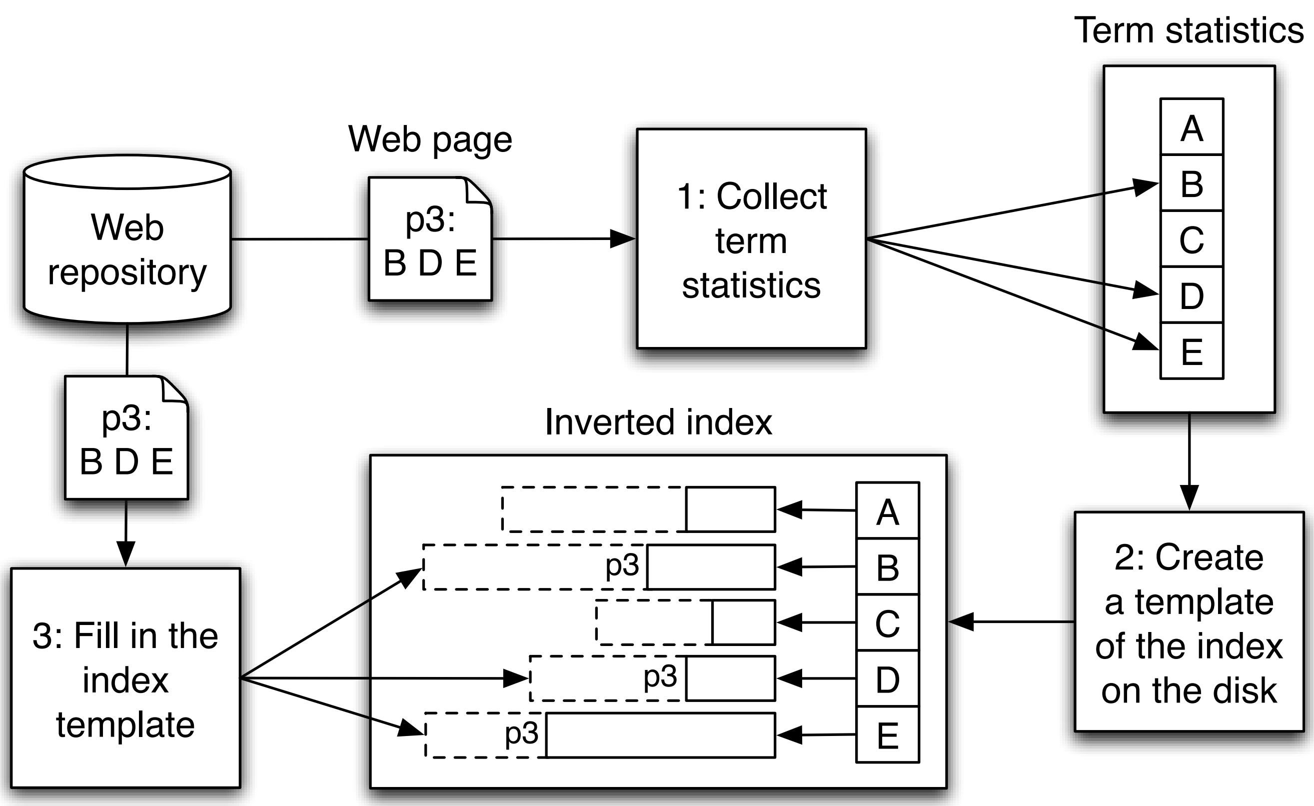
- Techniques
  - traversal of document similarity graph
    - formulated as the traveling salesman problem
  - clustering similar documents
    - assigns nearby ids to documents in the same cluster
  - sorting URLs alphabetically and assigning ids in that order
    - idea: pages from the same site have high textual overlap
    - simple yet effective
    - only applicable to web page collections

Index Construction

- Equivalent to computing the transpose of a matrix
- In-memory techniques do not work well with web-scale data
- Techniques
  - two-phase
  - one-phase

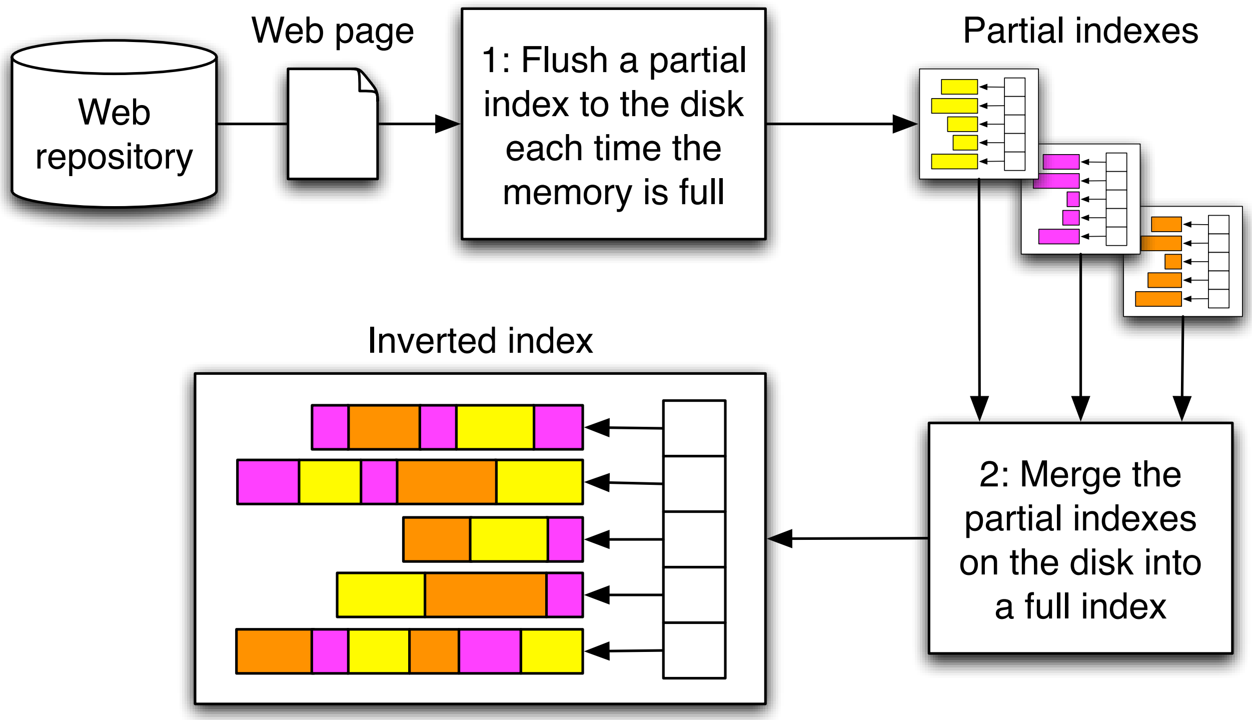
Two-Phase Index Construction

- First phase: read the collection and allocate a skeleton for the index
- Second phase: fill the posting lists



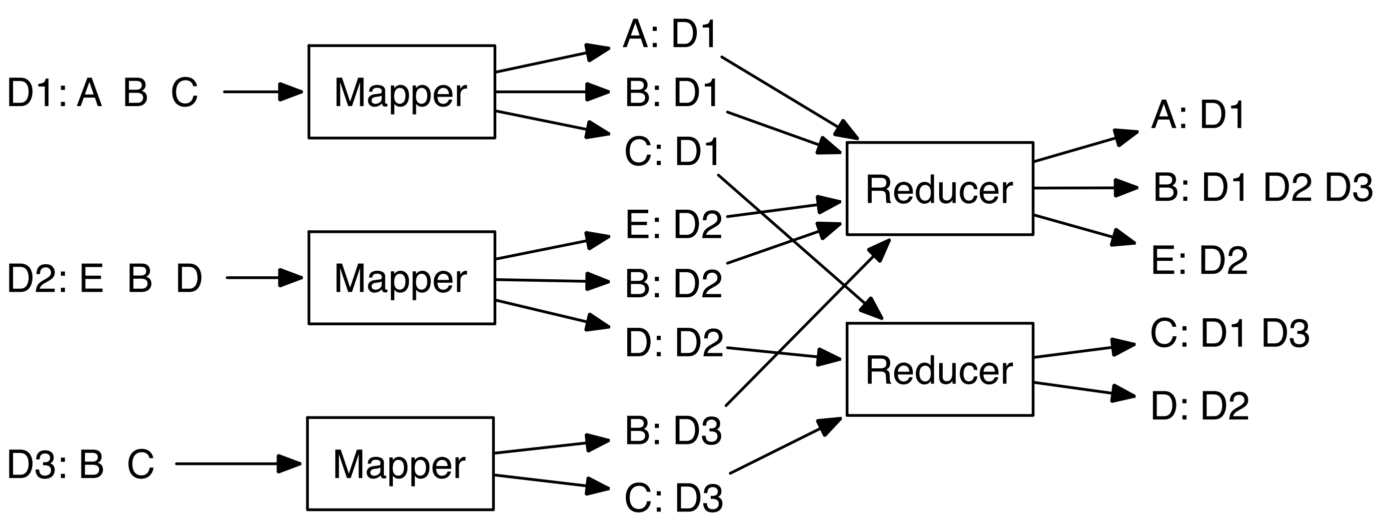
One-Phase Index Construction

- Keep reading documents and building an in-memory index
- Each time the memory is full, flush the index to the disk
- Merge all on-disk indexes into a single index in a final step



Parallel Index Construction

- Possible alternatives for creating an inverted index in parallel
  - message passing paradigm
  - MapReduce framework

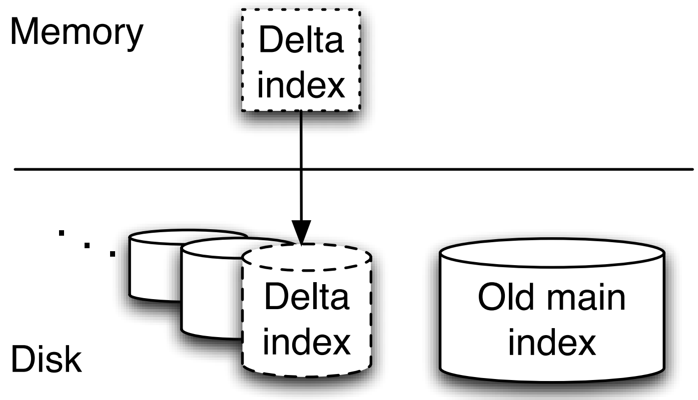


Index Maintenance

- Grow a new (delta) index in the memory; each time the memory is full, flush the in-memory index to disk
- Techniques
  - no merge
  - incremental update
  - immediate merge
  - lazy merge

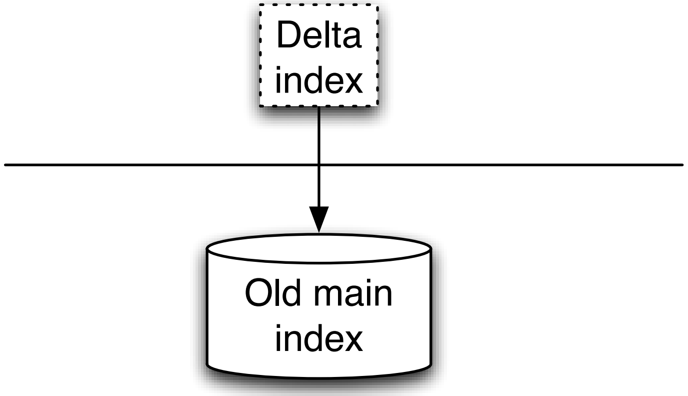
No Merge

- Flushed index is written to disk as a separate index
- Increases fragmentation and query processing time
- Eventually requires merging all on-disk indexes or rebuilding



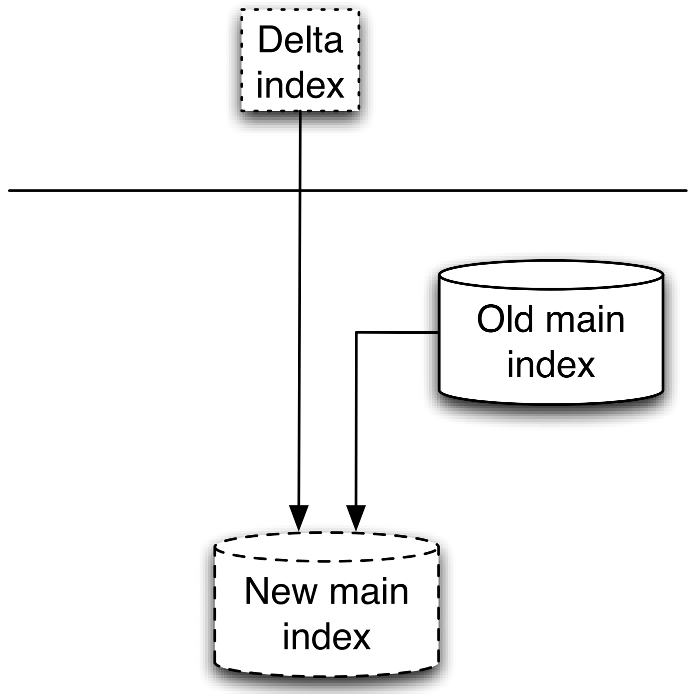
Incremental Update

- Each inverted list contains additional empty space at the end
- New documents are appended to the empty space in the list
- If the extra space allocated in an inverted list is full
  - inverted list may be reallocated on disk
  - inverted list is maintained in multiple fragments on disk



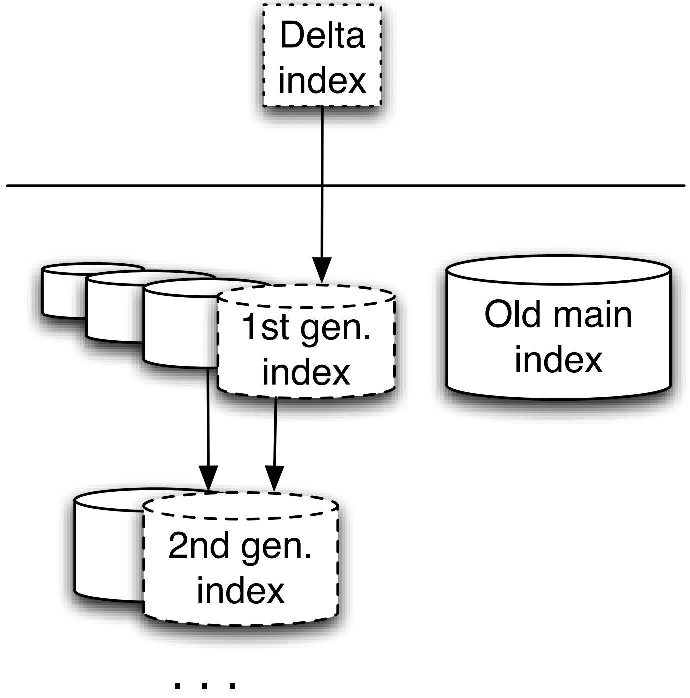
Immediate Merge

- Delta index is immediately merged to the old index and written to a new location on disk
- Only one copy of the index is maintained on disk



Lazy Merge

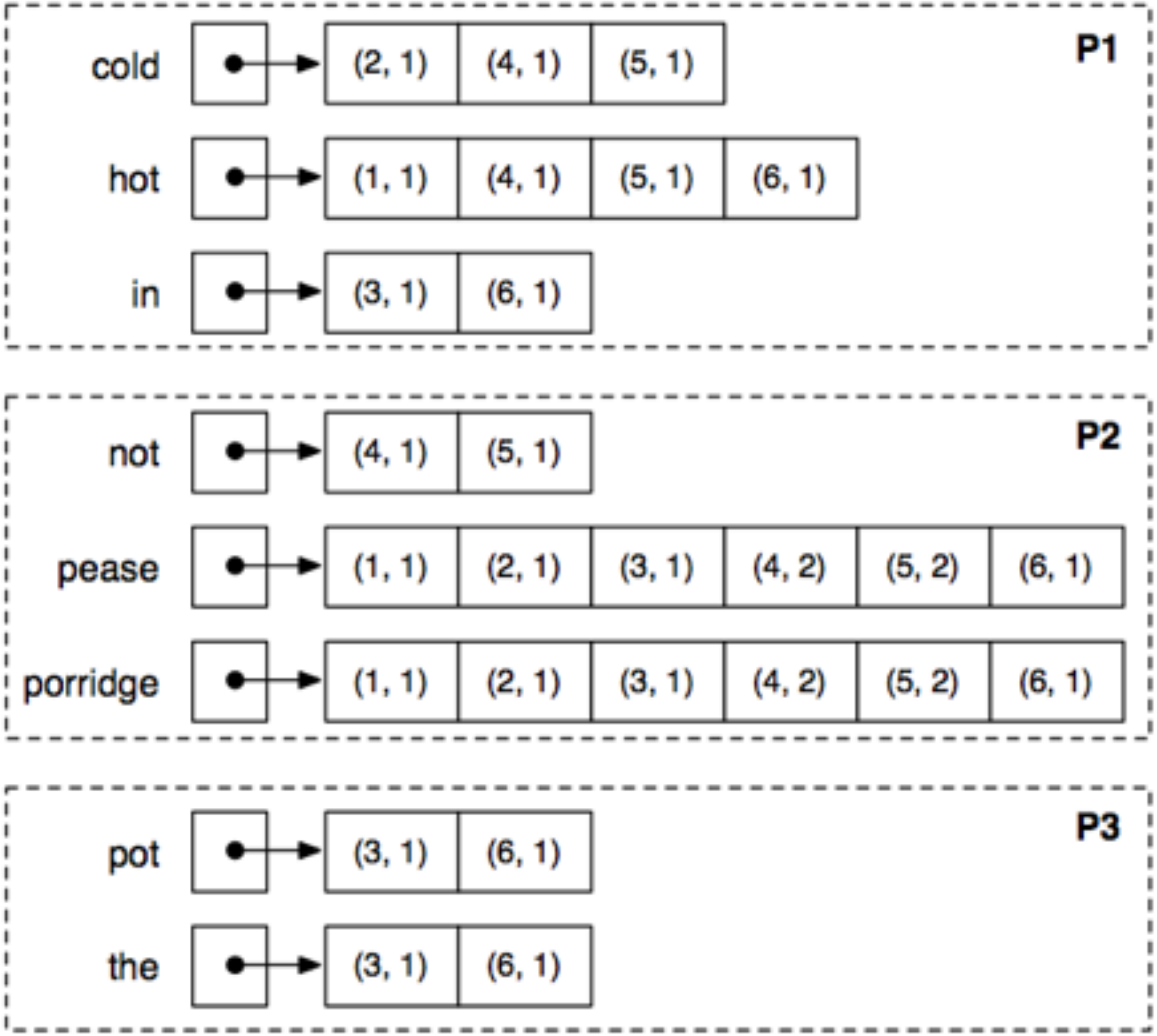
- Maintains multiple generations of the index on disk
- Index generations are lazily merged



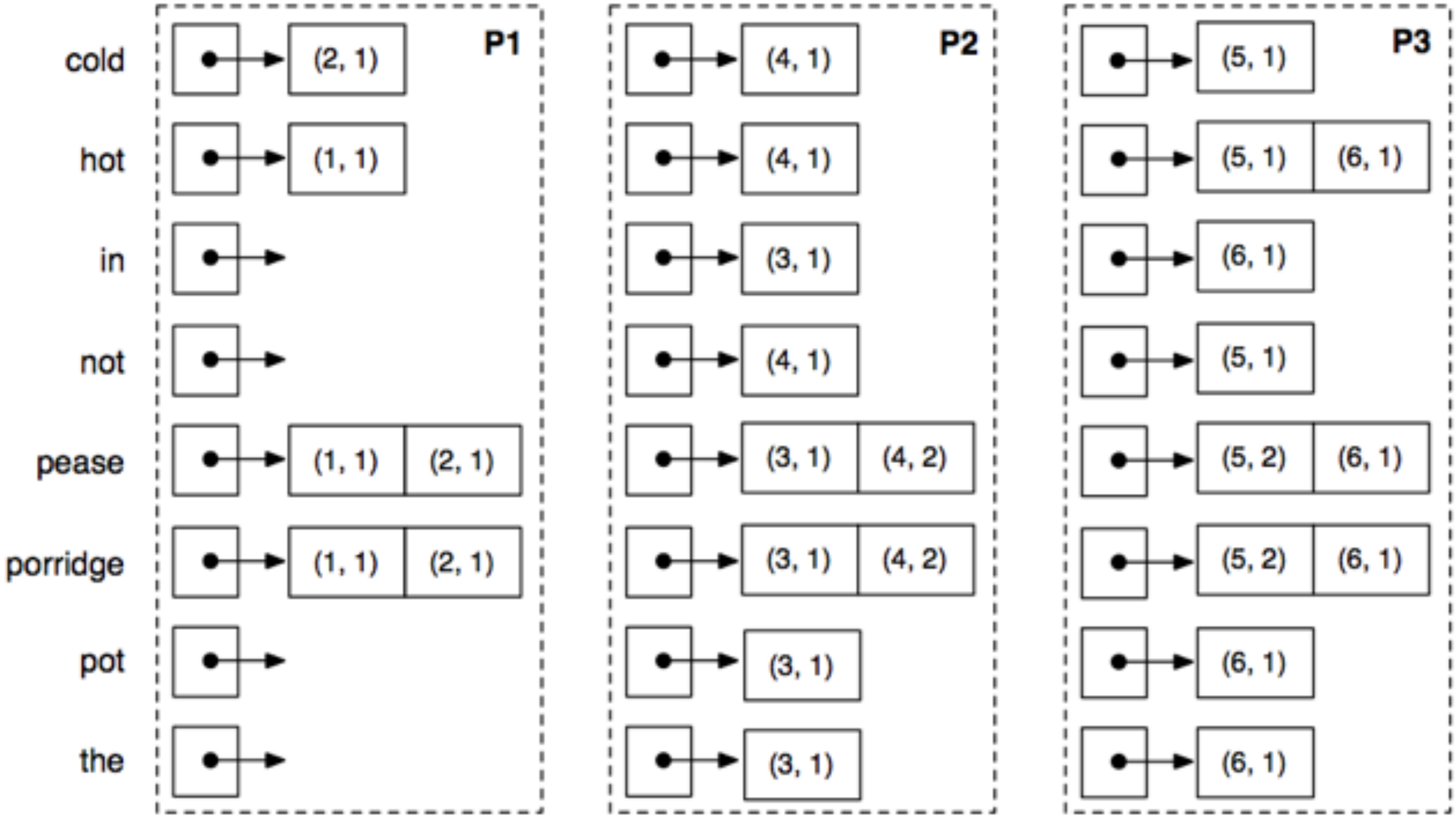
Inverted Index Partitioning

- Two alternatives for partitioning an inverted index
  - term-based partitioning
    - *T* inverted lists are distributed across *P* processors
    - each processor is responsible for processing the postings of a mutually disjoint subset of inverted lists assigned to itself
    - single disk access per query term
  - document-based partitioning
    - *N* documents are distributed across *P* processors
    - each processor is responsible for processing the postings of a mutually disjoint subset of documents assigned to itself
    - multiple (parallel) disk accesses per query term

Term-Based Index Partitioning



Document-Based Index Partitioning



Comparison of Index Partitioning Approaches

	Document-based	Term-based
Space consumption	Higher	Lower
Number of disk accesses	Higher	Lower
Concurrency	Lower	Higher
Computational load imbalance	Lower	Higher
Max. posting list I/O time	Lower	Higher
Cost of index building	Lower	Higher
Maintenance cost	Lower	Higher

Inverted Index Partitioning

- In practice, document-based partitioning is used
  - easier to build and maintain
  - low inter-query-processing concurrency, but good load balance
  - low query processing time
  - high throughput is achieved by replication
  - more fault tolerant
- Hybrid techniques are possible (e.g., term partitioning inside a document sub-collection)

Key Papers

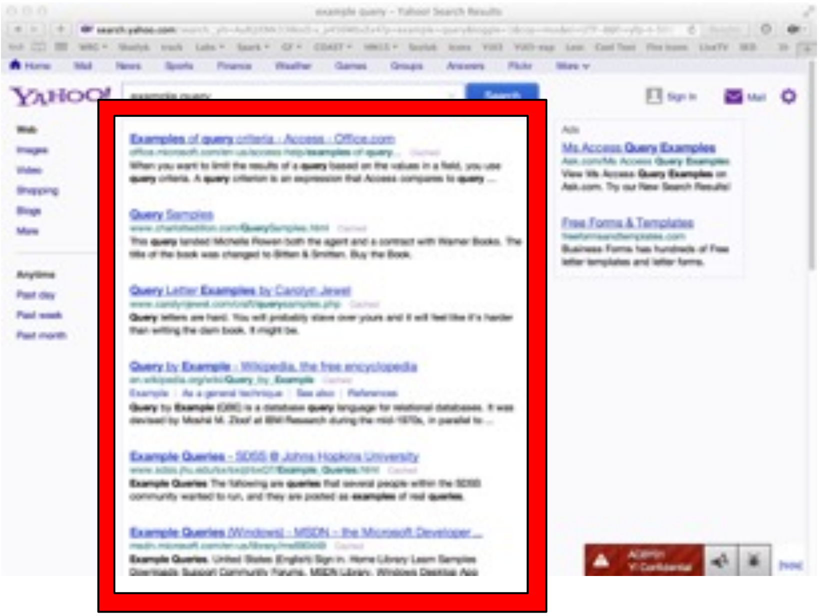
- Brin and Page, "The anatomy of a large-scale hypertextual Web search engine", Computer Networks and ISDN Systems, 1998.
- Zobel, Moffat, and Ramamohanarao, "Inverted files versus signature files for text indexing". ACM Transactions on Database Systems, 1998.
- Page, Brin, Motwani, and Winograd, "The PageRank citation ranking: bringing order to the Web", Technical report, 1998.
- Kleinberg, "Authoritative sources in a hyperlinked environment", Journal of the ACM, 1999.
- Ribeiro-Neto, Moura, Neubert, and Ziviani, "Efficient distributed algorithms to build inverted files", SIGIR, 1999.
- Carmel, Cohen, Fagin, Farchi, Herscovici, Maarek, and Soffer, "Static index pruning for information retrieval systems, SIGIR, 2001.
- Scholer, Williams, Yiannis, and Zobel, "Compression of inverted indexes for fast query evaluation", SIGIR, 2002.

Query Processing

Query Processing

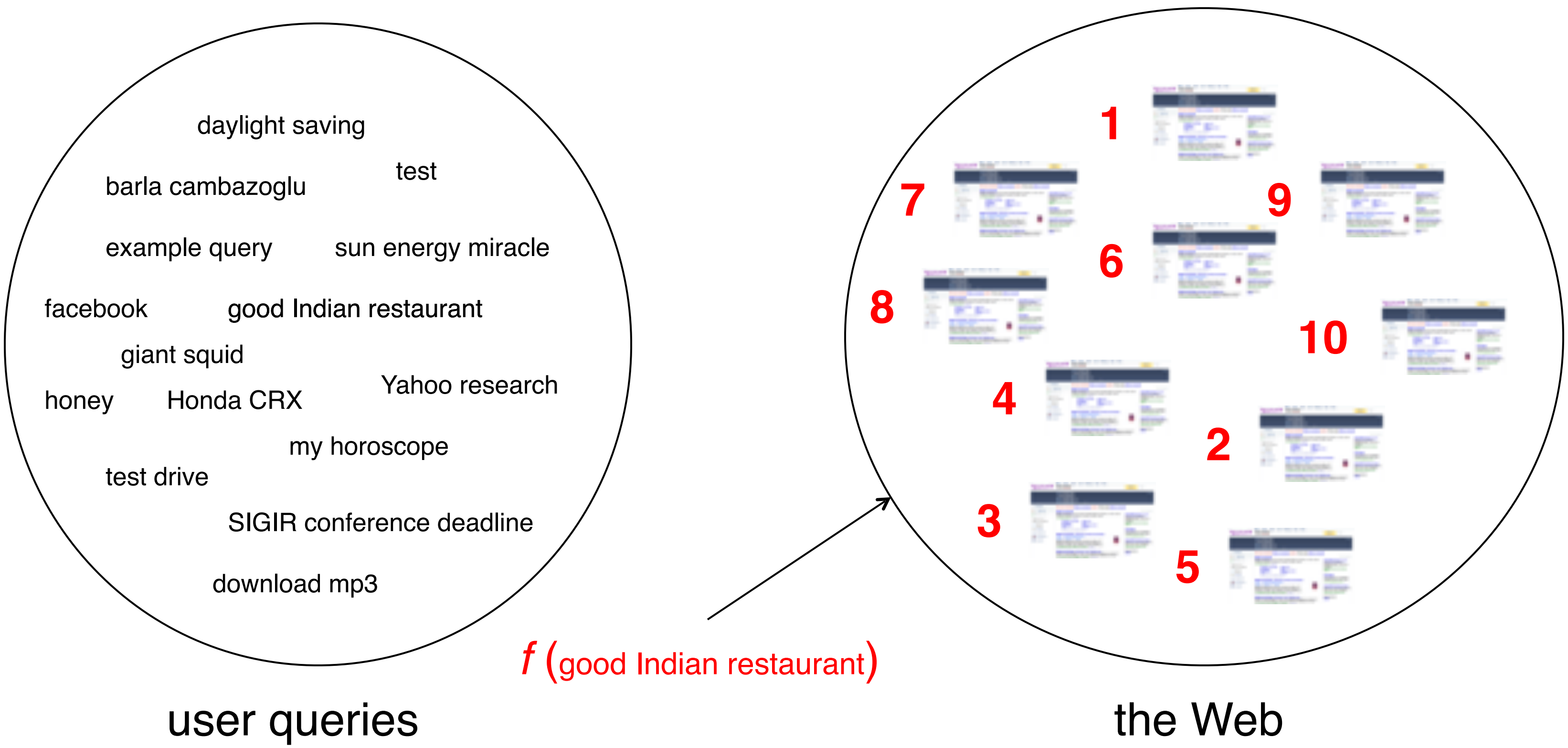
- Query processing is the problem of generating the best-matching answers (typically, top 10 documents) to a given user query, spending the least amount of time

- Our focus: creating 10 blue links as an answer to a user query

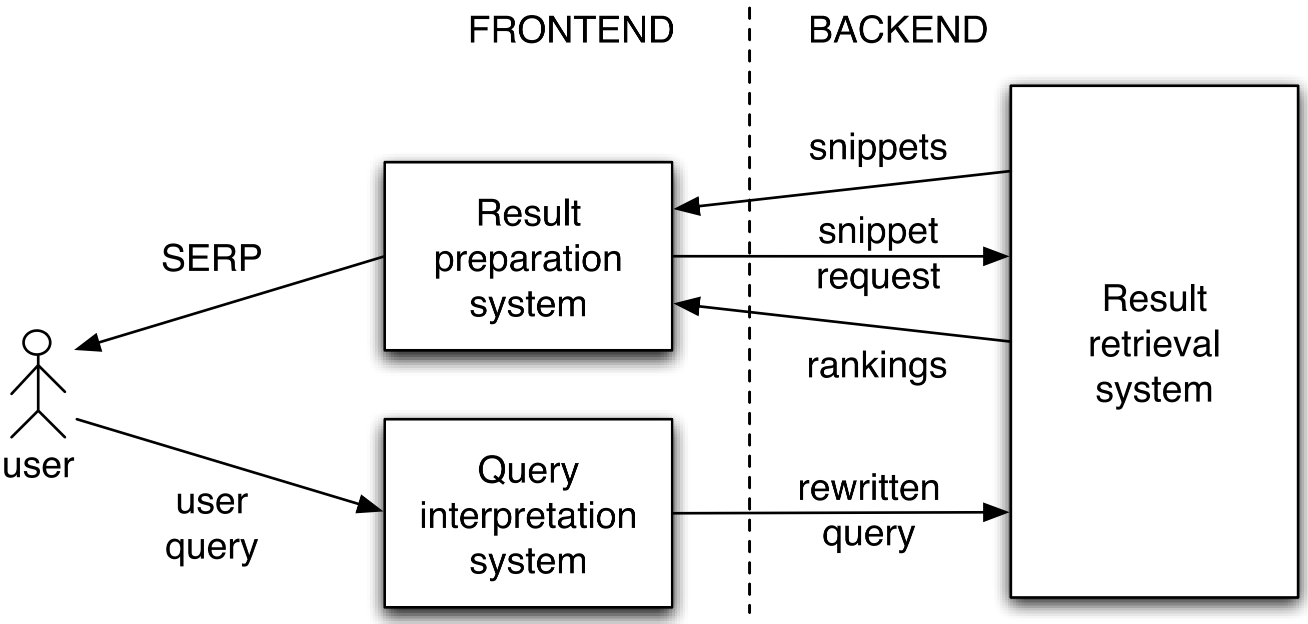


Web Search

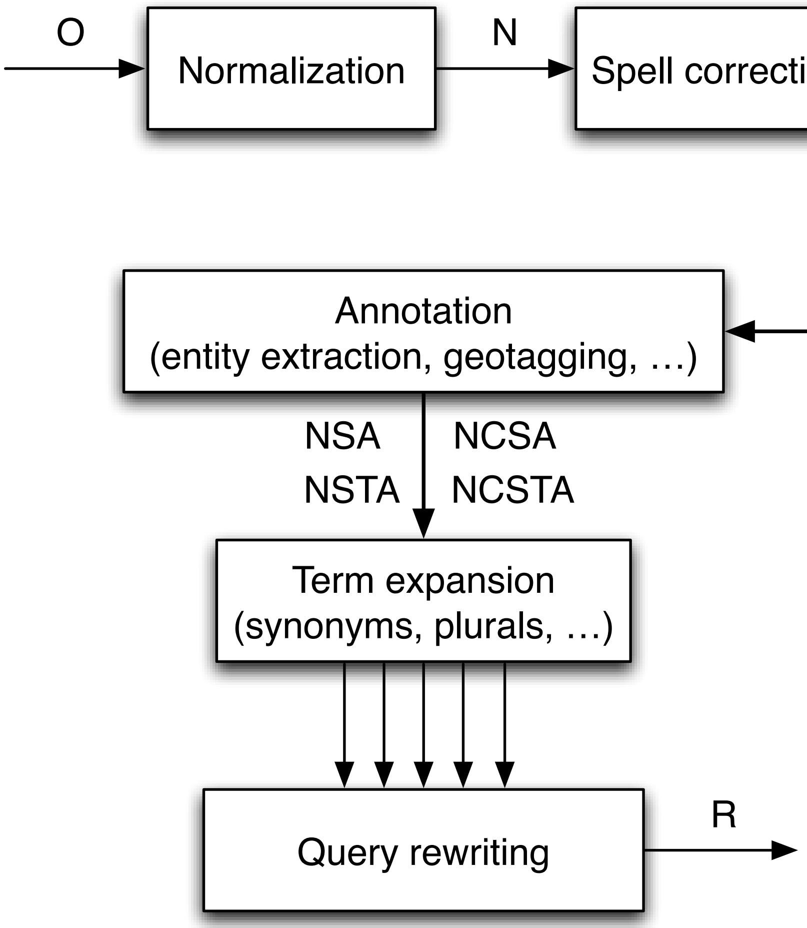
- Web search is a sorting problem!



Query Processing



Query Interpretation System



O: Original

N: Normalized

C: Spell corrected

S: Segmented

T: Stemmed

A: Annotated

R: Rewritten

Query Rewriting Example

- Original user query: *amusement arcades in New York*
- Internal system query: AND(OR(PHRASE(*amusement arcade*),

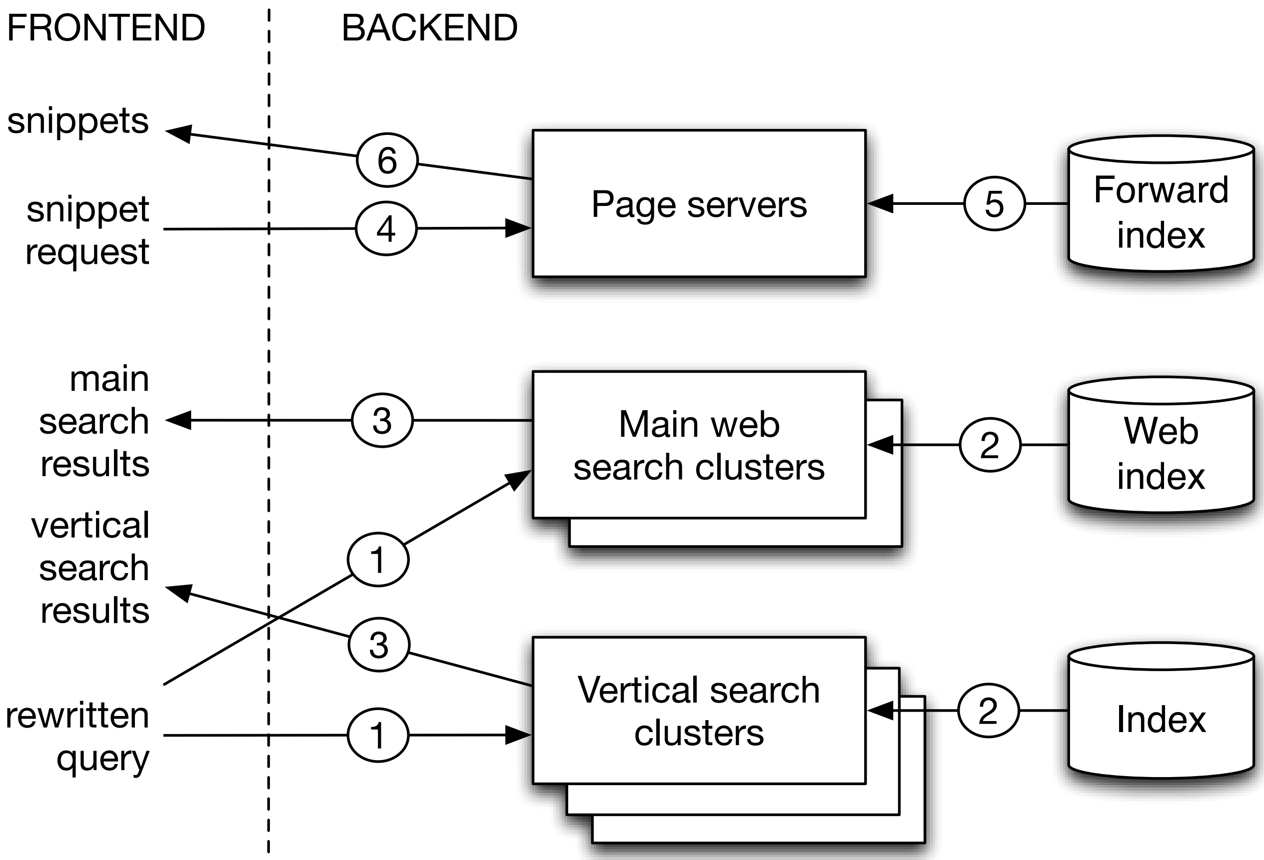
  PHRASE(*video arcade*)

)

  LOCATION(*new york*)

)
- Applied modifications
  1. stop word "*in*" is removed
  2. term "*arcades*" is converted into its singular form "*arcade*"
  3. "*amusement arcade*" is detected as a phrase and expanded to "*video arcade*"
  4. "*New York*" is detected as a location and converted to lower case

Result Preparation



Snippet Generation

- Search result snippets (a.k.a., summary or abstract)
  - important for users to correctly judge the relevance of a web page to their information need before clicking on its link

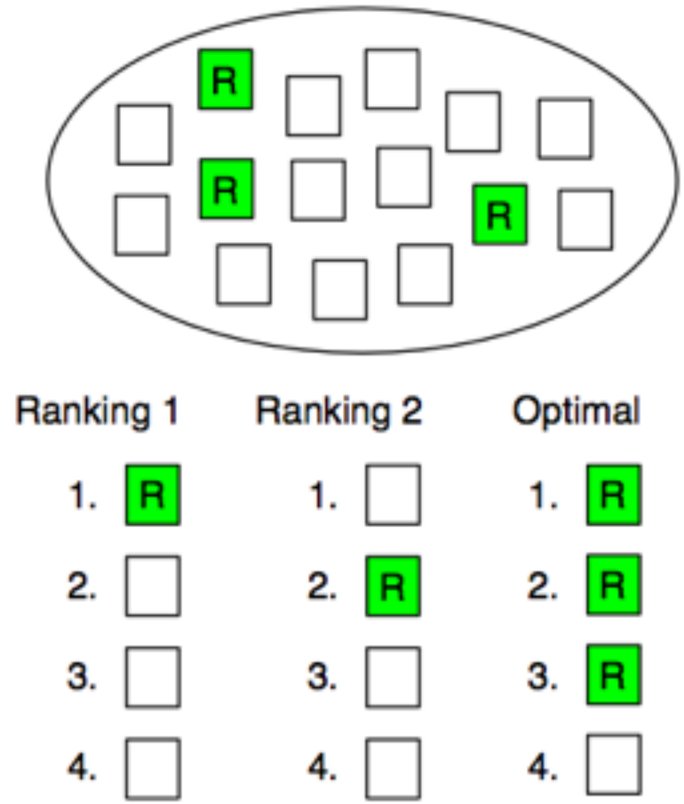
- Snippet generation
  - snippets are computed using the page content or position lists only for the top 10 result pages
  - efficiency of this step is important
  - entire page as well as snippets can be cached

Success Measures

- Quality measures
  - result quality: the degree to which returned answers meet user’s information need.
- Performance measures
  - latency: the response time delay experienced by the user
  - peak throughput: number of queries that can be processed per unit of time without any degradation on other metrics

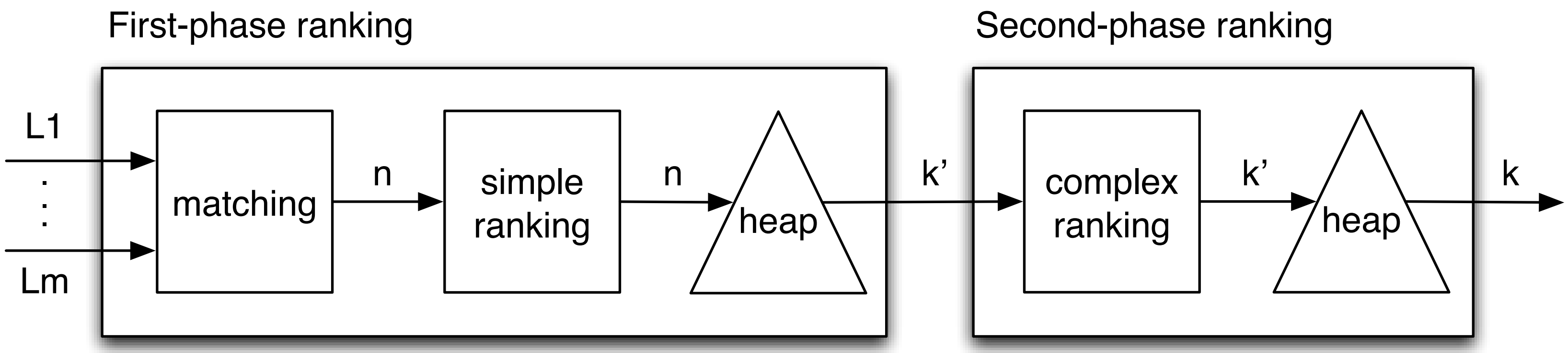
Measuring Relevance

- It is not always possible to know the user's intent and his information need
- Commonly used relevance metrics in practice
  - recall
  - precision
  - DCG
  - NDCG



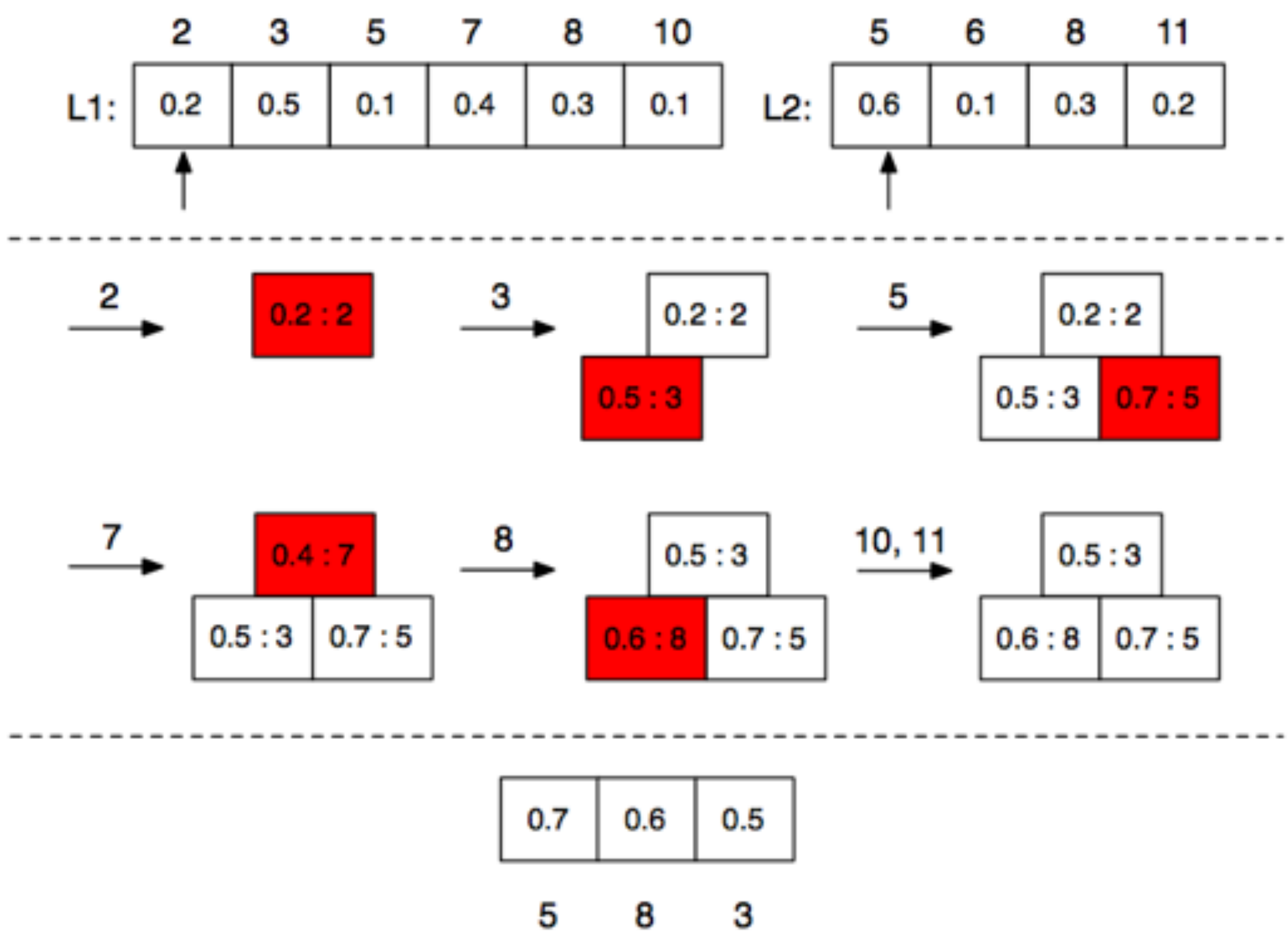
	Ranking 1	Ranking 2	Optimal
Recall:	1/3	1/3	1
Precision:	1/4	1/4	3/4
DCG:	1	0.63	1+0.63+0.5=2.13
NDCG:	1/2.13	0.63/2.13	

Two-Phase Ranking in a Search Node



- Two-phase ranking
  - simple ranking
    - linear combination of query-dependent and query-independent scores potentially with score boosting
    - main objective: efficiency
  - complex ranking
    - machine learned
    - main objective: quality

Efficient Score Computation (using a min heap)



Design Alternatives in First-Phase Ranking

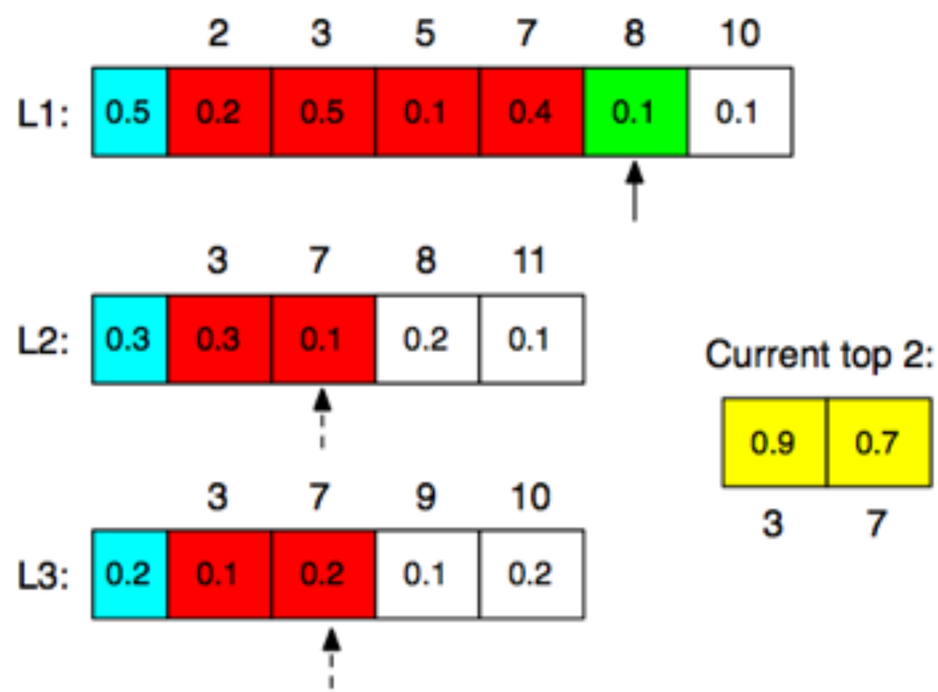
- In practice
  - term frequencies: enables compression
  - doc-id sorted lists: enables compression
  - document-at-a-time list traversal: enables better optimizations
  - AND mode: faster and leads to better results in web search
- Alternative models for score computation
  - vector-space model
  - statistical models
  - language models
- They all pretty much boil down to the same thing

Scoring Optimizations

- Techniques
  - bounding the number of accumulators
  - dynamic index pruning
    - WAND
    - MaxScore
  - early termination

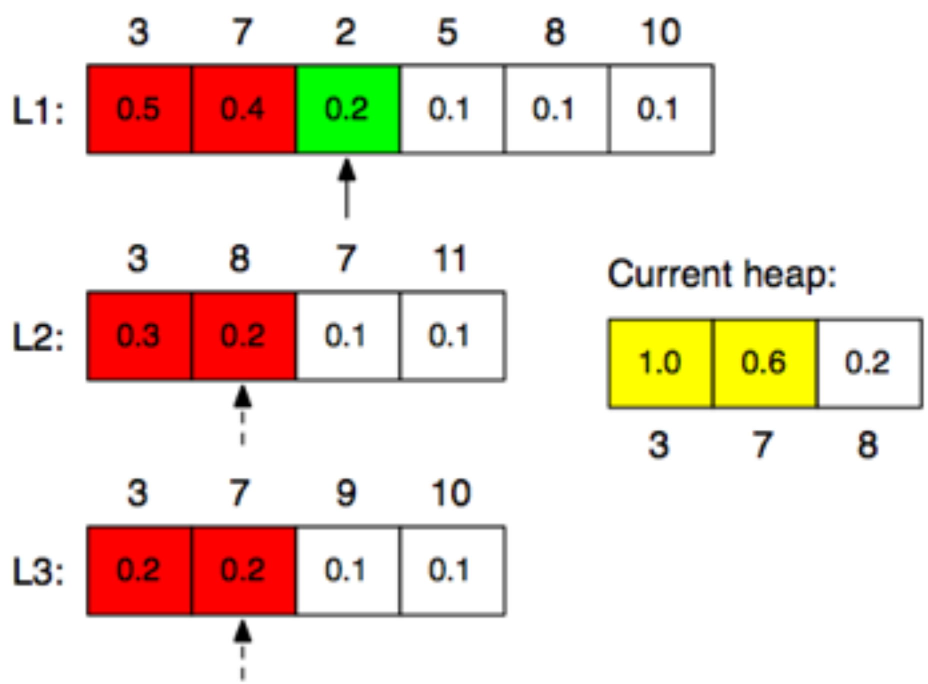
Scoring Optimizations

- Dynamic index pruning
  - store the maximum possible score contribution of each list
  - compute the maximum possible score for the current document
  - compare with the lowest score in the heap
  - gains in scoring and decompression time



Scoring Optimizations

- Early termination
  - stop scoring documents when it is guaranteed that neither document can make it into the top *k* list
  - gains in scoring and decompression time



Machine Learned Ranking

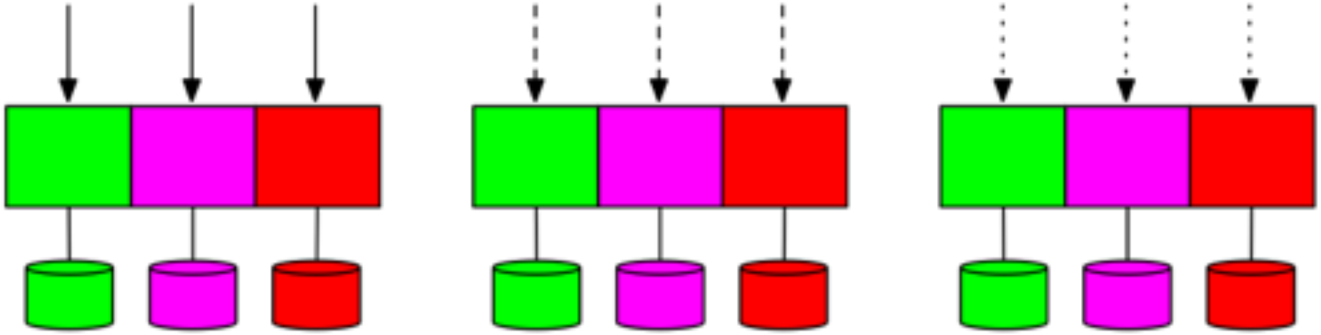
- Modeling quality
  - relevance
  - popularity
  - recency
  - quality
- Many features
  - term statistics (e.g., BM25)
  - term proximity
  - link analysis (e.g., PageRank)
  - spam detection
  - click data
  - search session analysis
- Popular learners used in commercial search engines
  - neural networks
  - boosted decision trees
- Ranking models
  - pointwise
  - pairwise
  - listwise

Query Processing Architectures

- Single node
  - not scalable in terms of response time
- Multiple nodes (search cluster)
  - large search clusters (low response time)
  - replicas of clusters (high query throughput)
- Multiple data centers (multi-site search engine)
  - reduces user-to-center latencies

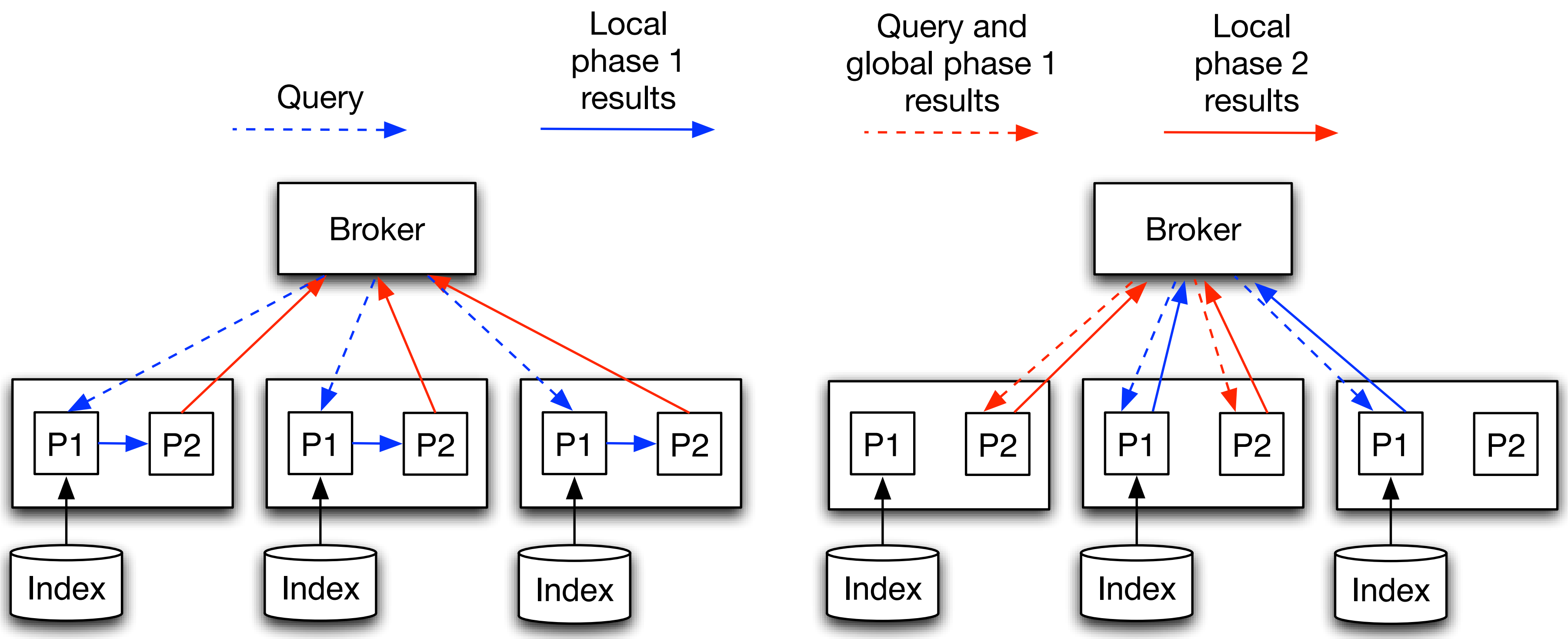
Inverted Index Partitioning/Replication

- In practice, the inverted index is
  - partitioned on thousands of computers in a large search cluster
    - reduces query response times
    - allows scaling with increasing collection size
  - replicated on tens of search clusters
    - increases query processing throughput
    - allows scaling with increasing query volume
    - provides fault tolerance



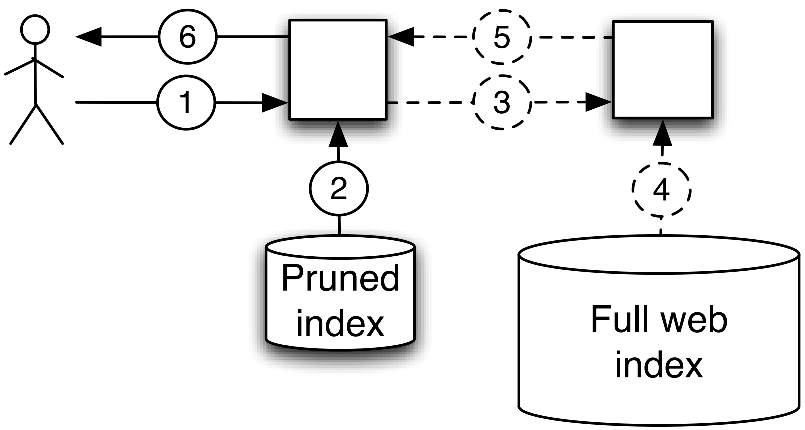
Parallel Query Processing

- Document-based partitioning
- Term-based partitioning



Static Index Pruning

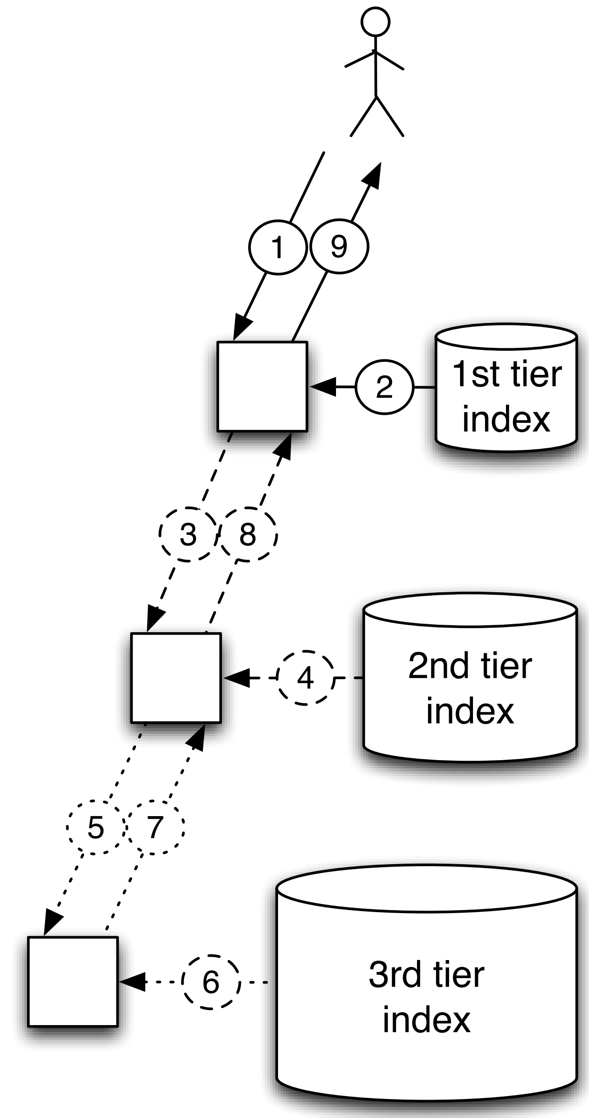
- Idea: to create a small version of the search index that can accurately answer most search queries



- Techniques
  - term-based pruning
  - doc-based pruning
- Result quality
  - guaranteed
  - not guaranteed

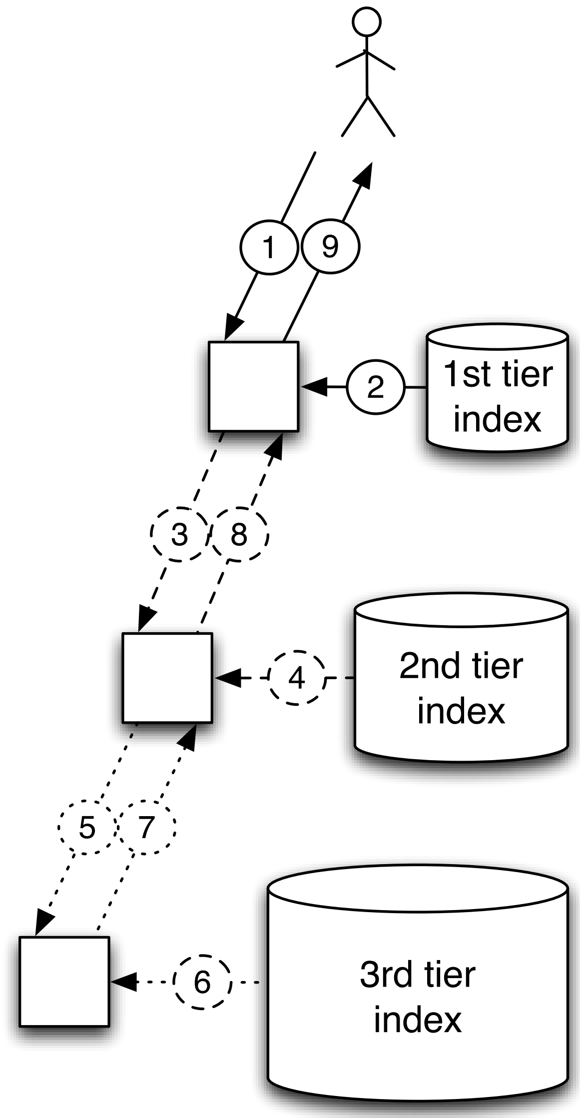
Tiering

- A sequence of sub-indexes
  - former sub-indexes are small and keep more important documents
  - later sub-indexes are larger and keep less important documents
  - a query is processed selectively only on the first *n* tiers
- Two decisions need to be made
  - tiering (offline): how to place documents in different tiers
  - fall-through (online): at which tier to stop processing the query



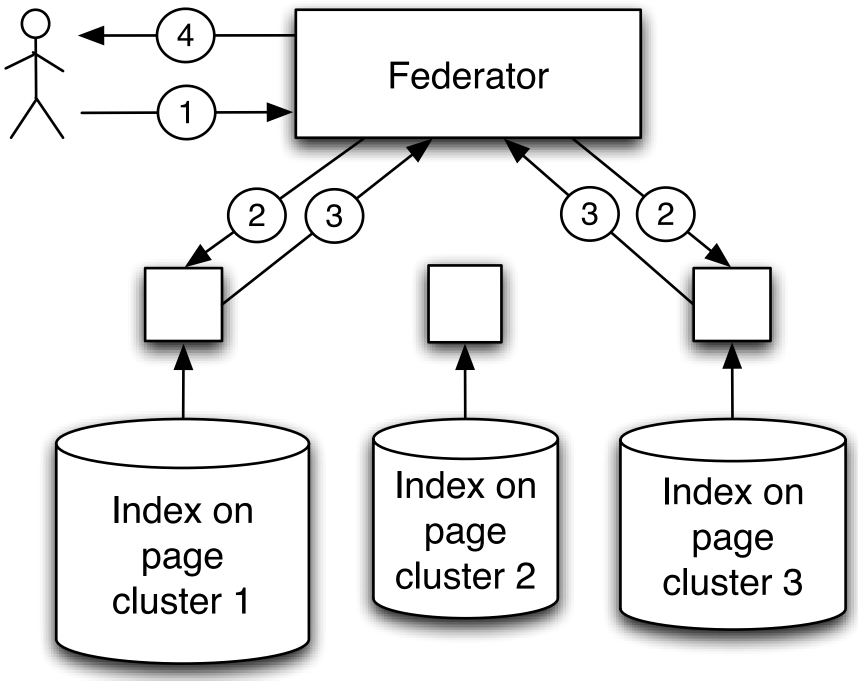
Tiering

- Tiering strategy is based on some document importance metric
  - PageRank
  - click count
  - spam score
- Fall-through strategy
  - query the next index until there are enough results
  - query the next index until search result quality is good
  - predict the next tier's result quality by machine learning



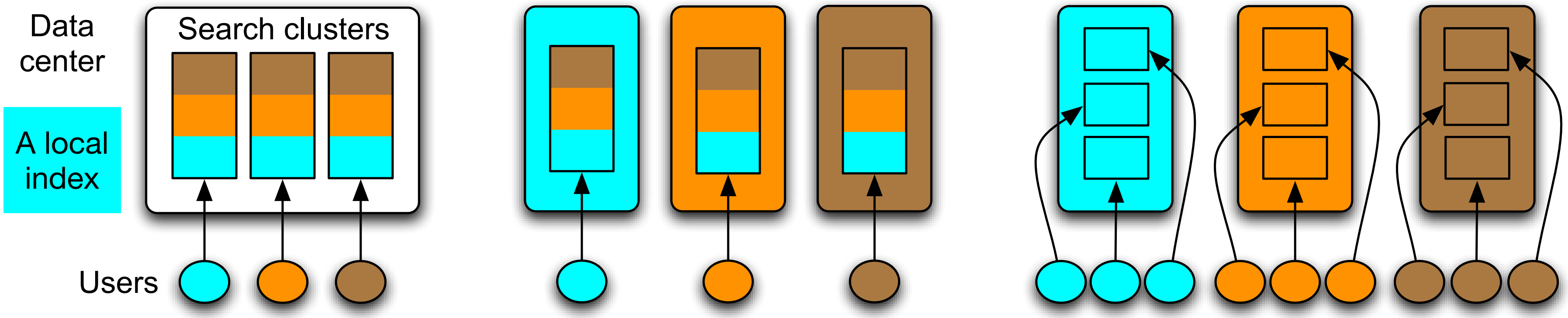
Selective Search

- Documents are clustered and a separate index is built
  - similarity between documents
  - co-click likelihood
- A query is processed on the indexes associated with the most similar *n* clusters
- Reduces the workload
- Suffers from the load imbalance problem
  - query topic distribution may be skewed
  - certain indexes have to be queried much more often



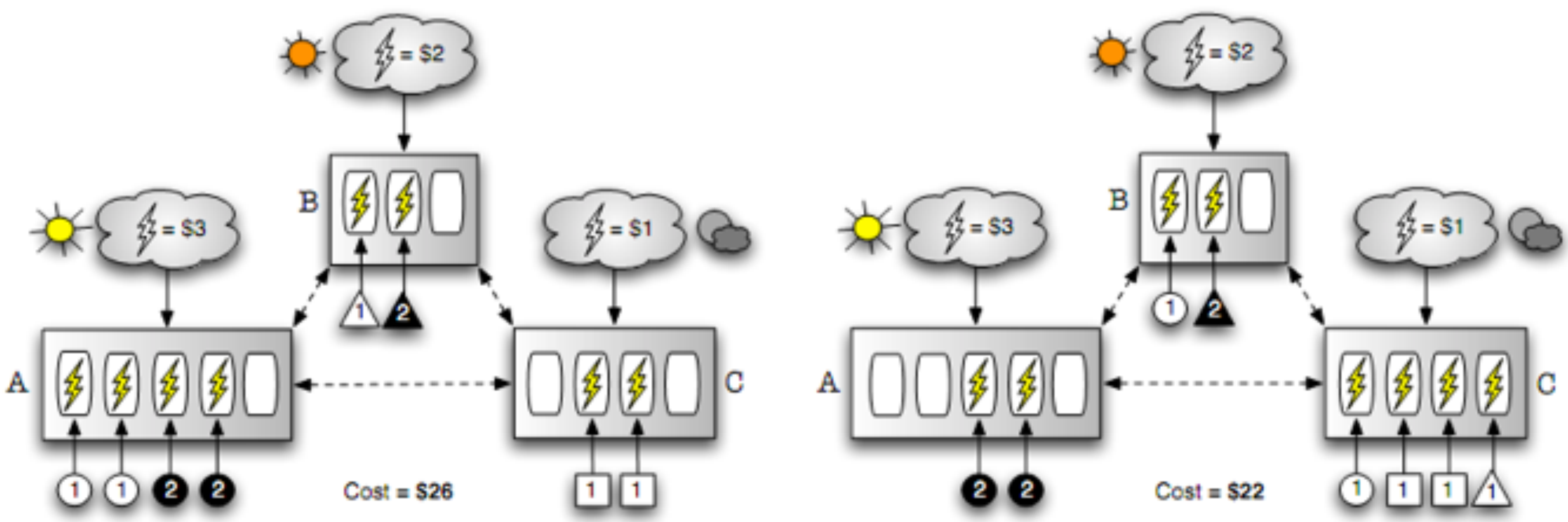
Multi-site Web Search Architectures

- Centralized
- Replicated
- Partitioned



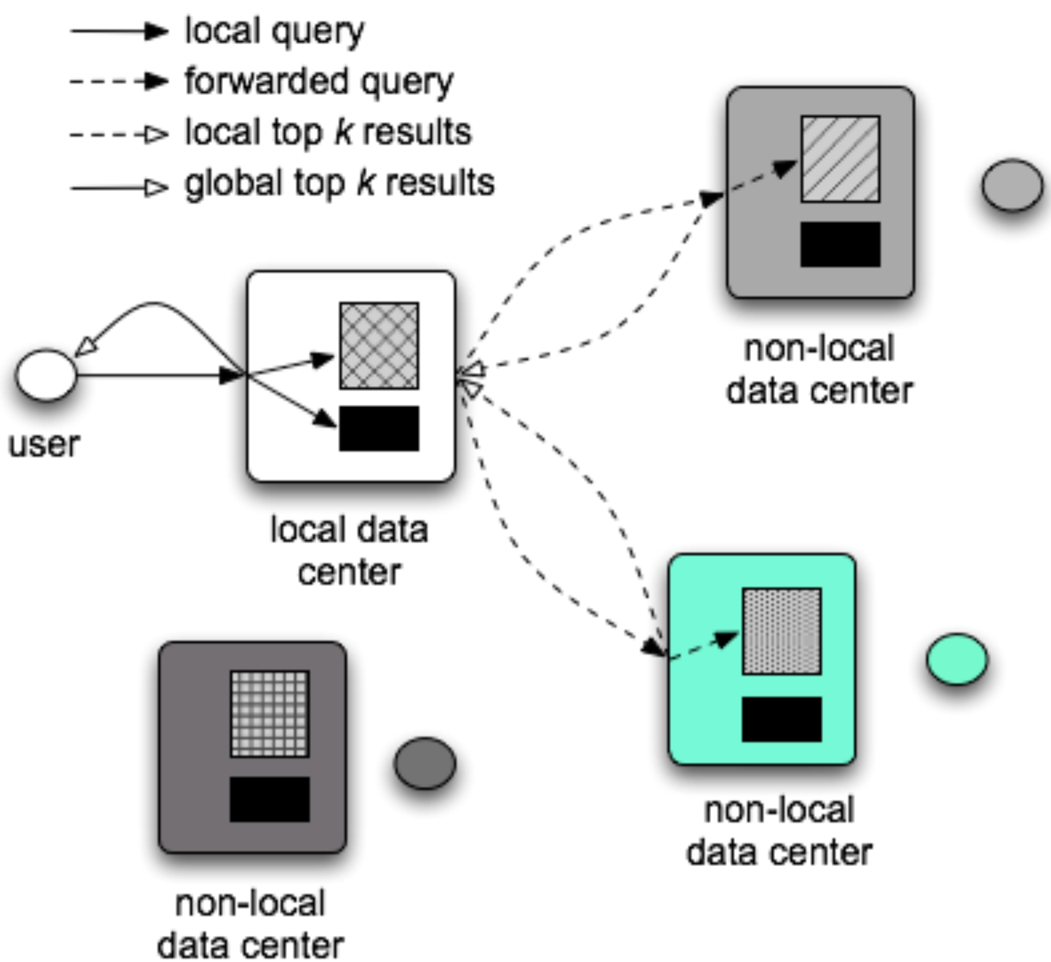
Replicated Index

- Key points
  - multiple, global data centers (sites)
  - user-to-center assignment
  - replicated web index
- Enables
  - local web crawling
  - energy price optimizations



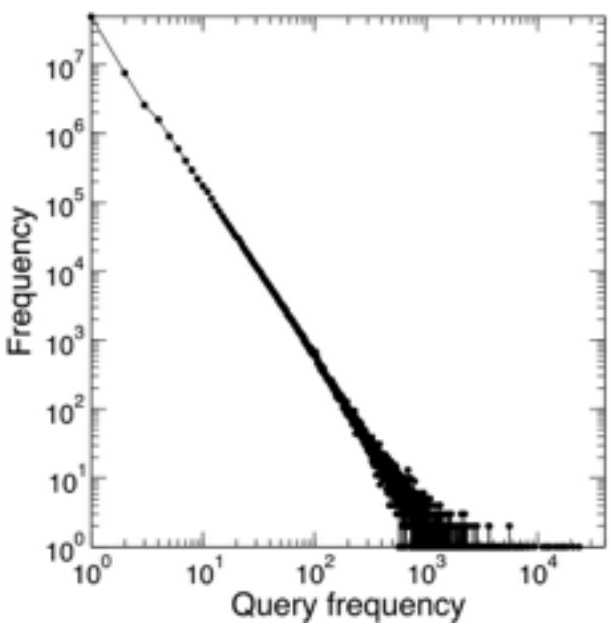
Partitioned Search Architectures

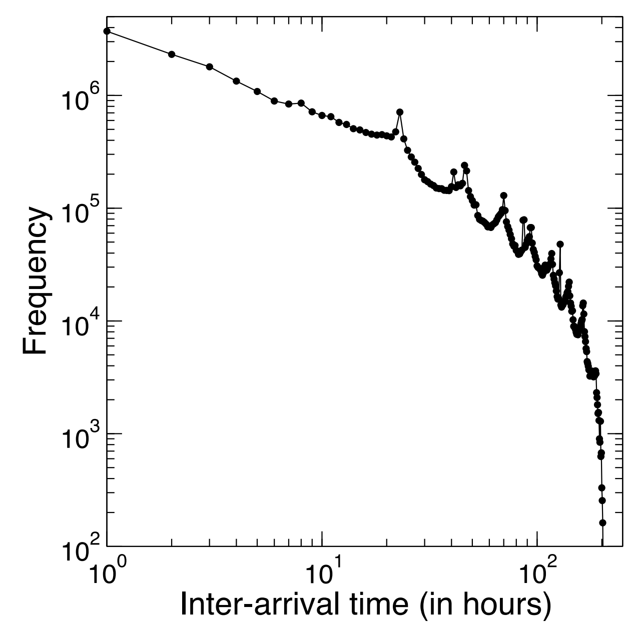
- Key points
  - multiple, regional data centers (sites)
  - user-to-center assignment
  - partitioned web index
  - partial document replication
- Enables
  - local web crawling
  - query processing with selective forwarding



Caching

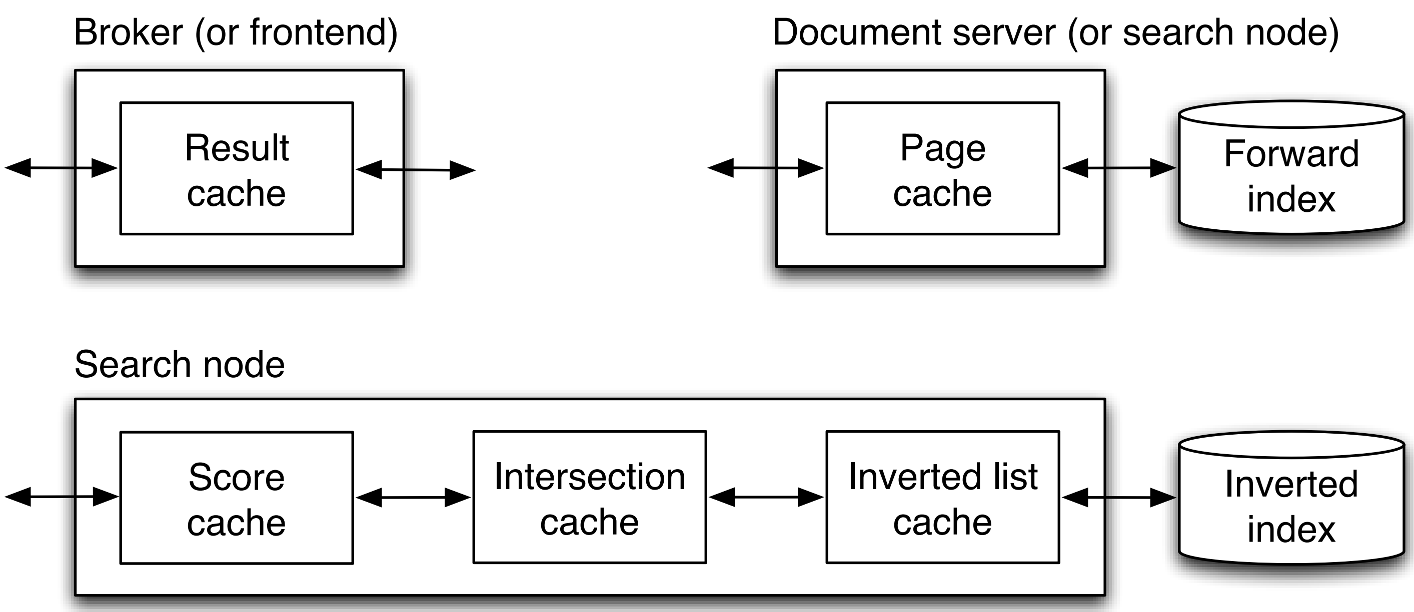
- Skewed distribution in query frequency
  - few queries are issued many times (head queries)
  - many queries are issued rarely (tail queries)
- Skewed distribution in query inter-arrival time
  - low inter-arrival time is for many queries
  - high inter-arrival time for few queries





Caches Available in a Web Search Engine

- Main caches in search engines: result cache, score cache, intersection cache, inverted list cache, page cache

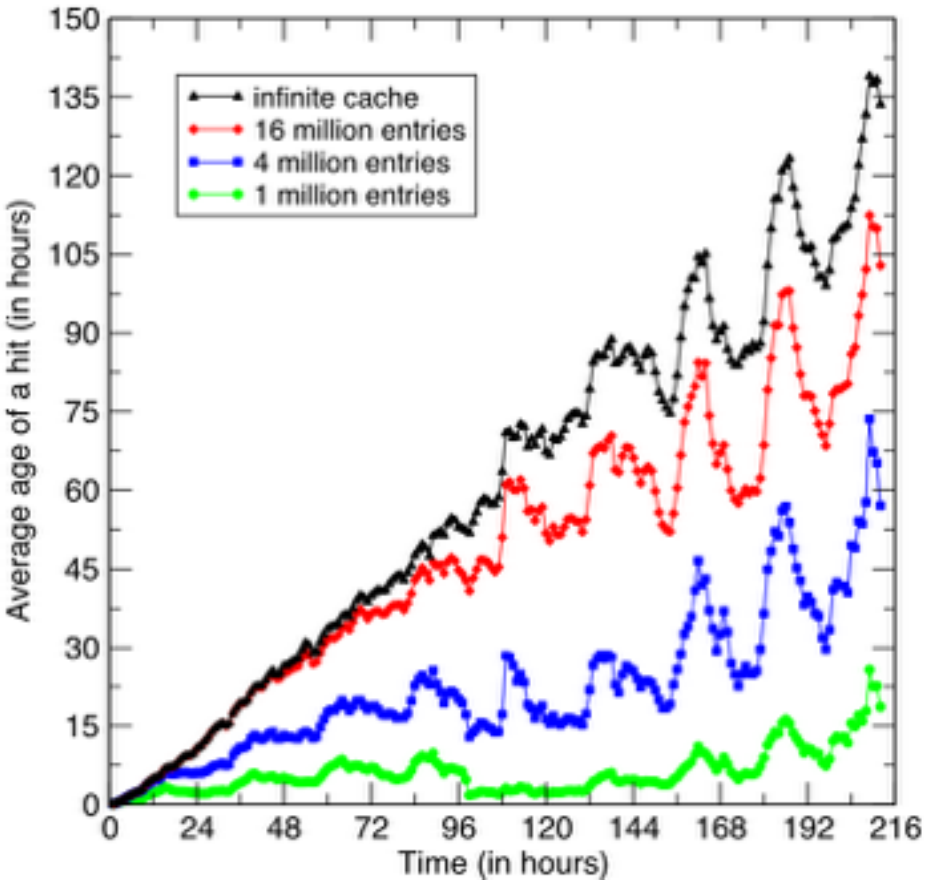


Caching Techniques

- Static caching
  - built in an offline manner
  - prefers items that are accessed often in the past
  - periodically re-deployed
- Dynamic caching
  - maintained in an online manner
  - prefers items that are recently accessed
  - requires removing items from the cache (eviction)
- Static/dynamic caching
  - shares the cache space between a static and a dynamic cache

Result Cache Freshness

- In practice
  - index is continuously updated or re-built
  - result caches are almost infinite capacity
  - staleness problem



Solutions

- Naïve solution: flushing the cache at regular time intervals
- Common solution: setting a time-to-live value for each item
- Advanced solutions
  - cache refreshing: stale results are predicted and scheduled for re-computation in idle cycles of the backend search system
    - easy to implement
    - little computational overhead
    - not very accurate
  - cache invalidation
    - hard to implement
    - incurs communication and computation overheads
    - highly accurate

Open Source Search Engines

- DataparkSearch: GNU general public license
- Lemur Toolkit & Indri Search Engine: BSD license
- Lucene: Apache software license
- mnoGoSearch: GNU general public license
- Solr: based on Lucene
- Elasticsearch: based on Lucene
- Seeks: Affero general public license
- Sphinx: free software/open source
- Terrier Search Engine: open source
- Zettair: open source

Key Papers

- Turtle and Flood, "Query evaluation: strategies and optimizations", Information Processing and Management, 1995.
- Barroso, Dean, and Holzle, "Web search for a planet: the Google cluster architecture", IEEE Micro, 2003.
- Broder, Carmel, Herscovici, Soffer, and Zien, "Efficient query evaluation using a two-level retrieval process", CIKM, 2003.
- Chowdhury and Pass, “Operational requirements for scalable search systems”, CIKM, 2003.
- Moffat, Webber, Zobel, and Baeza-Yates, "A pipelined architecture for distributed text query evaluation", Information Retrieval, 2007.

Key Papers

- Turpin, Tsegay, Hawking, and Williams, "Fast generation of result snippets in web search. SIGIR, 2007.
- Baeza-Yates, Gionis, Junqueira, Plachouras, and Telloli, "On the feasibility of multi-site web search engines", CIKM, 2009.
- Cambazoglu, Zaragoza, Chapelle, Chen, Liao, Zheng, and Degenhardt, "Early exit optimizations for additive machine learned ranking systems", WSDM, 2010.
- Wang, Lin, and Metzler, "Learning to efficiently rank", SIGIR, 2010.
- Cambazoglu, Junqueira, Plachouras, Banachowski, Cui, Lim, and Bridge, “A refreshing perspective of search engine caching”, WWW, 2010.
- Macdonald, Tonellotto, Ounis, "Learning to predict response times for online query scheduling", SIGIR, 2012.

Concluding Remarks

Summary

- We presented a high-level overview of the challenges faced by search engines.
- We provided a summary of commonly used success measures.
- We discussed some architectural and algorithmic optimizations employed in search engines.
- We provided references to available software and key research work in literature.

Observations

- Unlike the past research, the current research on information retrieval is mainly driven by the needs of commercial search engine companies.
- Lack of hardware resources, real-life query logs, and ground-truth datasets render information retrieval research somewhat difficult, especially for researchers in academia.
- Efficiency and effectiveness of web retrieval systems are likely to be a research challenge for some more time (at least, in the foreseeable future). But, we believe that certain limits will be reached at some point in time.

Suggestions to Newcomers

- Follow the trends in the Web, user bases, and hardware parameters to identify the real bottlenecks in web retrieval efficiency effectiveness.
- Watch out newly emerging techniques whose primary target is to improve the search quality and think about their impact on search performance.
- Reuse or adapt existing solutions in more mature research fields, such as databases, computer networks, distributed computing, and natural language processing.
- Know the key people in the field (the community is small) and follow their work.

Surveys on Related Topics

- M. R. Henzinger, R. Motwani, and C. Silverstein. “Challenges in web search engines”, SIGIR Forum, 2002.
- J. Zobel and A. Moffat, “Inverted files for text search engines”, ACM Computing Surveys, 2006.
- T-Y. Liu, “Learning to rank for information retrieval”, Foundations and Trends in Information Retrieval, 2009.
- C. Olston and M. Najork: “Web Crawling”, Foundations and Trends in Information Retrieval, 2010.
- F. Silvestri, “Mining Query Logs: Turning Search Usage Data into Knowledge”, Foundations and Trends in Information Retrieval, 2010.
- B. B. Cambazoglu and Ricardo Baeza-Yates, “Scalability Challenges in Web Search Engines”, The Information Retrieval Series, 2011.
- N. Spirin and J. Han. “Survey on web spam detection: Principles and algorithms”. SIGKDD Exploration Newsletter, 2012.

Related Books

	Coverage			
Reference	Web crawling	Indexing	Query processing	Perspective
[Witten et al., 1999]	None	High	Medium	Information retrieval
[Chakrabarti, 2002]	High	High	High	Web retrieval
[Grossman and Frieder, 2004]	None	Medium	High	Information retrieval
[Manning et al., 2008]	Low	High	High	Information retrieval
[Croft et al., 2009]	Low	High	High	Web retrieval
[Chowdhury, 2010]	None	High	Medium	Library science
[Buttcher et al., 2010]	Low	High	High	Information retrieval
[Baeza-Yates and Ribeiro-Neto, 2011]	High	High	High	Information retrieval