UFMG UNIVERSIDADE FEDERAL DE MINAS GERAIS

Ranking Models

Search Architecture

Rodrygo L. T. Santos
rodrygo@dcc.ufmg.br

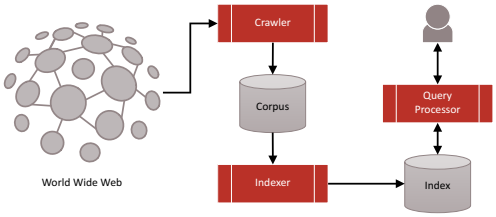
Search architecture

A software architecture consists of software components, the interfaces provided by those components, and the relationships between them

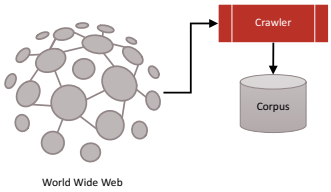
For search, we are concerned about

- Effectiveness (quality of results)
- Efficiency (response time and throughput)

Search components



Search components



Crawling overview

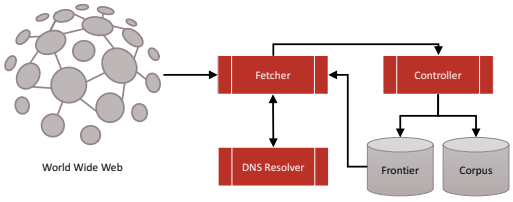
Document acquisition

- Builds a local corpus for searching
- Many types – Web, enterprise, desktop

Web crawlers follow links to find documents

- Must efficiently find huge numbers of web pages (coverage) and keep them up-to-date (freshness)

Crawling overview



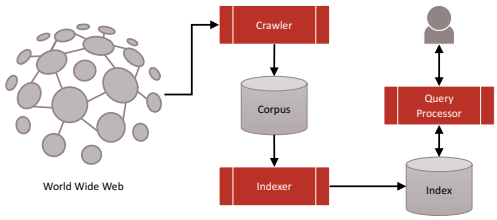
Key challenges

Web is huge and constantly changing
- Not under the control of search providers

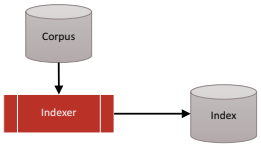
A lot of time is spent waiting for responses
- Parallel crawling is essential

Could potentially flood sites with requests
- To avoid this problem, use politeness policies

Search components



Search components



Indexing overview

Document representation
- From raw text to index terms
- Plus annotations (e.g., entities, categories)

Off-document evidence
- Anchor text, link analysis
- Social network signals

Document representation

Fred's Tropical Fish Shop is the best place to find tropical fish at low, low prices. Whether you're looking for a little fish or a big fish, we've got what you need. We even have fake seaweed for your fishtank (and little surfboards too).

Document representation

Fred's Tropical Fish Shop is the best place to find tropical fish at low, low prices. Whether you're looking for a little fish or a big fish, we've got what you need. We even have fake seaweed for your fishtank (and little surfboards too).

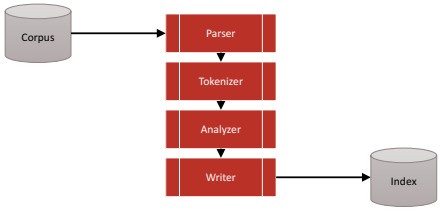
Topical features

9.7 fish
4.2 tropical
22.1 tropical fish
8.2 seaweed
4.2 surfboards

Quality features

14 incoming links
3 days since last update

Indexing overview



Key challenges

Support effective retrieval
- Extract meaningful document features
- Both topical and quality features

Support efficient retrieval
- Quick scoring of matched documents

Index structures

Indexes are designed to make search faster
- Unique requirements, unique data structures

Most common structure is the inverted index
- General name for a class of structures
- "Inverted" because documents are associated with words, rather than words with documents

Example "corpus"

d_1	Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.
d_2	Fish keepers often use the term tropical fish to refer only those requiring fresh water, with saltwater tropical fish referred to as marine fish.
d_3	Tropical fish are popular aquarium fish, due to their often bright coloration.
d_4	In freshwater fish, this coloration typically derives from iridescence, while salt water fish are generally pigmented.

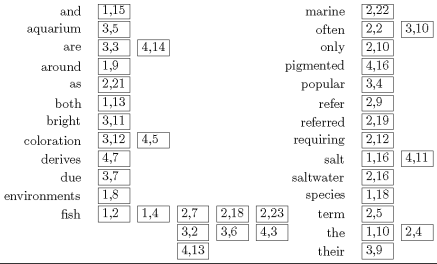
Inverted index: occurrences

and	1	only	2
aquarium	3	pigmented	4
are	3 4	popular	3
around	1	refer	2
as	2	referred	2
both	1	requiring	2
bright	3	salt	1 4
coloration	3 4	saltwater	2
derives	4	species	1
due	3	term	2
environments	1	the	1 2
fish	1 2 3 4	their	3
fishkeepers	2	this	4
found	1	those	2

Inverted index: counts

and	1:1	only	2:1
aquarium	3:1	pigmented	4:1
are	3:1 4:1	popular	3:1
around	1:1	refer	2:1
as	2:1	referred	2:1
both	1:1	requiring	2:1
bright	3:1	salt	1:1 4:1
coloration	3:1 4:1	saltwater	2:1
derives	4:1	species	1:1
due	3:1	term	2:1
environments	1:1	the	1:1 2:1
fish	1:2 2:3 3:2 4:2	their	3:1
fishkeepers	2:1	this	4:1
found	1:1	those	2:1

Inverted index: positions



Inverted index: fields

Document structure is useful in search
- Field restrictions (e.g., date, from:)
- Some fields more important (e.g., title, h1)

A couple of options
- Separate inverted lists for each field type
- Add information about fields to postings

Auxiliary structures

Vocabulary, dictionary, or lexicon
- Lookup table from term to inverted list
- Either hash table in memory or B-tree for disk

Additional structures for document data
- Basic statistics, static features, metadata

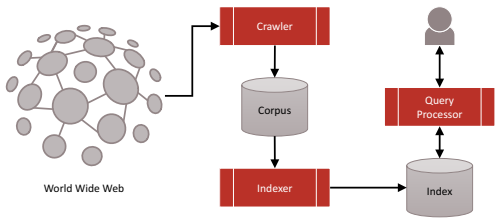
Additional structure for corpus statistics

Index compression

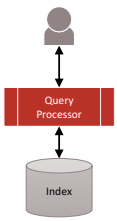
Inverted indexes are very large
- Typically 25-50% of corpus size
- Much higher if n-grams are indexed

Compression saves disk and/or memory space
- Typically have to decompress lists to use them
- Best techniques have good trade-offs

Search components



Search components



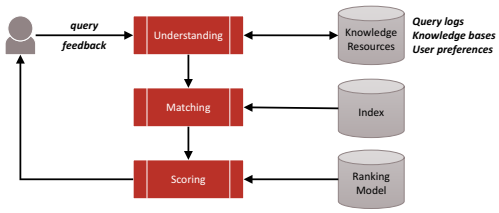
Query processing overview

Query representation
- Infers user's need from a keyword query

Document ranking
- Matches and scores indexed documents

Feedback handling
- Both explicit and implicit signals

Query processing overview



Key challenges

Queries are typically short, ill-specified
- Long queries tend to be difficult

Finding matching documents can be expensive
- Particularly for common terms or long queries

Ranking is a tough business
- Different queries, different requirements

Query understanding: expand matches

Query relaxation

[information about tropical fish]

↳ [tropical fish]

Query expansion

[tropical fish]

↳ [tropical fish aquarium]

Query understanding: narrow results

Query segmentation

[tropical fish captive breeding]

↳ [”tropical fish” AND ”captive breeding”]

Query scoping

[tropical fish hawaii]

↳ [category:”tropical fish” place:hawaii]

Document matching

Scan posting lists for all query terms

fish	1:2	2:3	3:2	4:2
tropical	1:2	2:2	3:1	

Score matching documents

$$f(q,d) = \sum_{t \in q} f(t,d)$$

Matching semantics

Disjunctive matching
- Documents must contain at least one query term
- More matches, lower precision

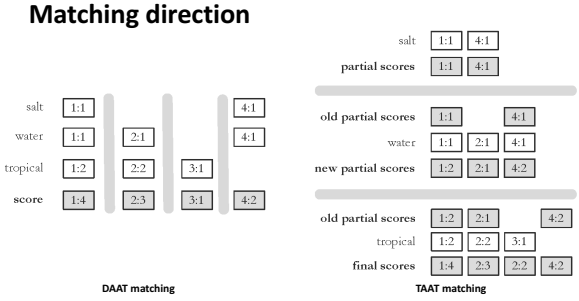
Conjunctive matching
- Documents must contain all query terms
- Fewer matches, higher precision

Matching direction

Document-at-a-time (DAAT)
- Inverted lists processed in parallel
- One document scored at a time

Term-at-a-time (TAAT)
- Inverted lists processed in sequence
- Partial document scores accumulated

Matching direction



Optimization techniques

No clear winner
- DAAT uses less memory (no accumulators)
- TAAT is more memory efficient (sequential access)

Both can be improved
- Read less data from inverted lists (skipping)
- Calculate scores for fewer documents (thresholding)

Other approaches

Unsafe early termination
- Ignore high-frequency word lists in TAAT
- Ignore documents at end of lists in DAAT

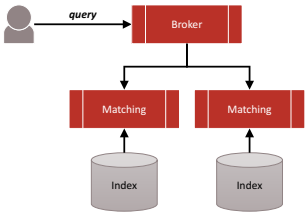
Can be improved with index tiering
- Postings ordered by quality (e.g., PageRank)
- Postings ordered by score (e.g., BM25)

Distributed matching

Indexes are often distributed in a cluster
- Too large to fit in one machine
- Replication helps load balancing

Two main approaches
- Document partitioning (most common)
- Term partitioning

Distributed matching



Distributed matching

Problem
- Ranking models typically leverage global statistics (e.g., number of documents where a term appears)
- Shards only have local statistics

Solution
- Share or approximate global statistics

Multi-stage ranking

Some ranking models can be expensive
- Infeasible to score billions of documents

Ranking as a multi-stage cascade
- Stage #1: Boolean matching (billions)
- Stage #2: Unsupervised scoring (millions)
- Stage #3: Supervised scoring (thousands)

Caching

Query distributions similar to Zipf
- About 15% of never seen queries each day
- Popular queries account for majority of traffic

Caching can significantly improve effectiveness
- For popular queries, cache search results
- For unpopular queries, cache inverted lists

References

Search Engines: Information Retrieval in Practice, Ch. 5
Croft et al., 2009

Scalability Challenges in Web Search Engines
Cambazoglu and Baeza-Yates, 2015

UFMG UNIVERSIDADE FEDERAL DE MINAS GERAIS

Coming next…

Document Understanding

Rodrygo L. T. Santos
rodrygo@dcc.ufmg.br