Information Retrieval (Part I)

[DAT640] Information Retrieval and Text Mining

Krisztian Balog
University of Stavanger

September 16, 2019

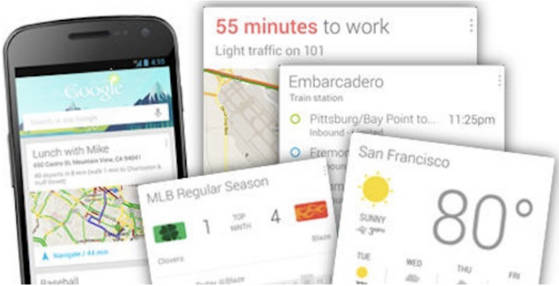
Information Retrieval (IR)

"Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information."

(Salton, 1968)

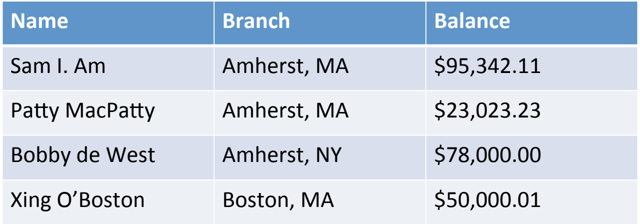
Modern definition

"Making the **right information** available to the **right person** at the **right time** in **the right form**."



Searching in databases

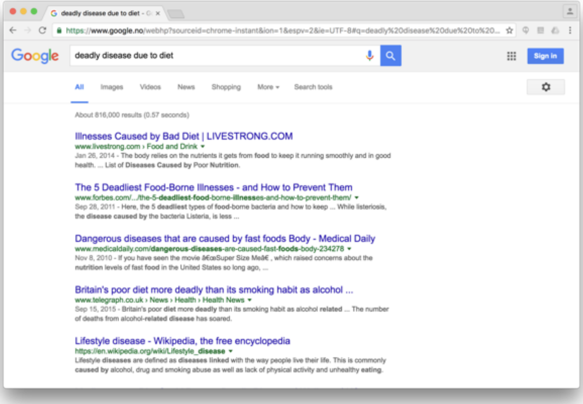
Query: *records with balance* $>$ *$50,000 in branches located in Amherst, MA.*



Name	Branch	Balance
Sam I. Am	Amherst, MA	$95,342.11
Patty MacPatty	Amherst, MA	$23,023.23
Bobby de West	Amherst, NY	$78,000.00
Xing O’Boston	Boston, MA	$50,000.01

Searching in text

Query: *deadly disease due to diet*


deadly disease due to diet - G ×
https://www.google.no/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=deadly%20disease%20due%20to%20...
Google
deadly disease due to diet
Sign in
All
Images
Videos
News
Shopping
More ▾
Search tools

About 816,000 results (0.57 seconds)

Illnesses Caused by Bad Diet | LIVESTRONG.COM
www.livestrong.com › Food and Drink ▾
Jan 26, 2014 - The body relies on the nutrients it gets from **food** to keep it running smoothly and in good health. ... List of **Diseases Caused** by Poor **Nutrition**.

The 5 Deadliest Food-Borne Illnesses - and How to Prevent Them
www.forbes.com/.../the-5-**deadliest-food**-borne-**illnesses**-and-how-to-prevent-them/ ▾
Sep 28, 2011 - Here, the 5 **deadliest** types of **food**-borne bacteria and how to keep ... While listeriosis, the **disease caused** by the bacteria Listeria, is less ...

Dangerous diseases that are caused by fast foods Body - Medical Daily
www.medicaldaily.com/**dangerous-diseases**-are-caused-fast-**foods**-body-234278 ▾
Nov 8, 2010 - If you have seen the movie â€œSuper Size Meâ€ , which raised concerns about the **nutrition** levels of fast **food** in the United States so long ago, ...

Britain's poor diet more deadly than its smoking habit as alcohol ...
www.telegraph.co.uk › News › Health › Health News ▾
Sep 15, 2015 - Britain's poor **diet** more **deadly** than its smoking habit as alcohol **related** ... The number of deaths from alcohol-**related disease** has soared.

Lifestyle disease - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/Lifestyle_**disease** ▾
Lifestyle **diseases** are defined as **diseases linked** with the way people live their life. This is commonly **caused by** alcohol, drug and smoking abuse as well as lack of physical activity and unhealthy **eating**.

Which of the results are relevant?

Core problem in IR

How to match information needs (“queries”) and information objects (“documents”)

Core issues in IR

- **Relevance**
  - Simple (and simplistic) definition: A relevant document contains the information that a person was looking for when they submitted a query to the search engine
    - Many factors influence a person's decision about what is relevant (task, context, novelty, ...)
    - Distinction between *topical relevance* vs. *user relevance* (all other factors)
  - *Retrieval models* define a view of relevance
  - *Ranking algorithms* used in search engines are based on retrieval models
  - Most models are based on statistical properties of text rather than linguistic
  - Exact matching of words is not enough!

Core issues in IR

- **Evaluation**
  - Experimental procedures and measures for comparing system output with user expectations
  - Typically use test collection of documents, queries, and relevance judgments
  - *Recall* and *precision* are two examples of effectiveness measures

Core issues in IR

- **Information needs**
  - Keyword queries are often poor descriptions of actual information needs
  - Interaction and context are important for understanding user intent
  - Query modeling techniques such as query expansion, aim to refine the information need and thus improve ranking

Dimensions of IR

- IR is more than just text, and more than just web search
    - Although these are central
- Content
    - Text, images, video, audio, scanned documents, ...
- Applications
    - Web search, vertical search, enterprise search, desktop search, social search, legal search, chatbots and virtual assistants, ...
- Tasks
    - Ad hoc search, filtering, question answering, response ranking, ...

Search engines in operational environments

- Performance
  - Response time, indexing speed, etc.
- Incorporating new data
  - Coverage and freshness
- Scalability
  - Growing with data and users
- Adaptibility
  - Tuning for specific applications

Outline for the coming lectures

- **Search engine architecture, indexing** ⇐ today
- Evaluation
- Retrieval models
- Query modeling
- Learning-to-rank, Neural IR
- Semantic search

Search engine architecture

- A software architecture consists of software components, the interfaces provided by those components, and the relationships between them
  - Describes a system at a particular level of abstraction
- Architecture of a search engine determined by 2 requirements
  - Effectiveness (quality of results)
  - Efficiency (response time and throughput)
- Two main processes:
  - Indexing (offline)
  - Querying (online)

Indexing process

Indexing

Indexing is the process that makes a document collection searchable

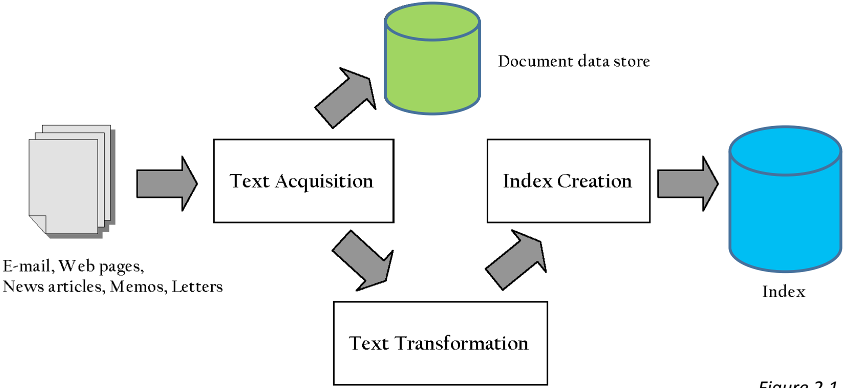
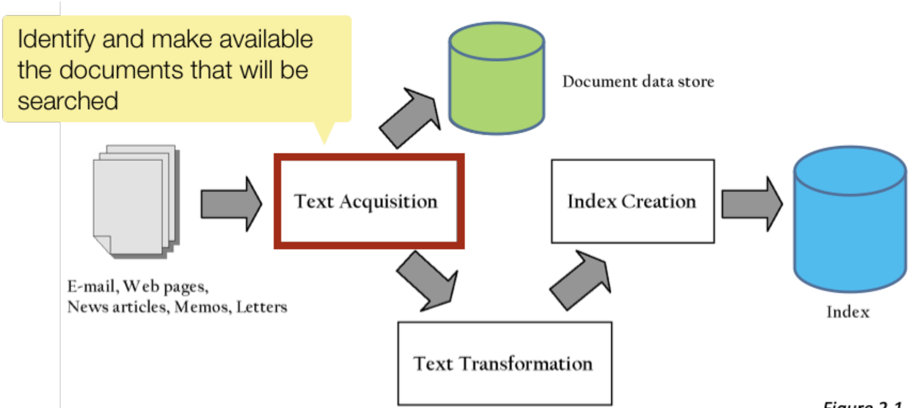


Figure 2.1

Text acquisition



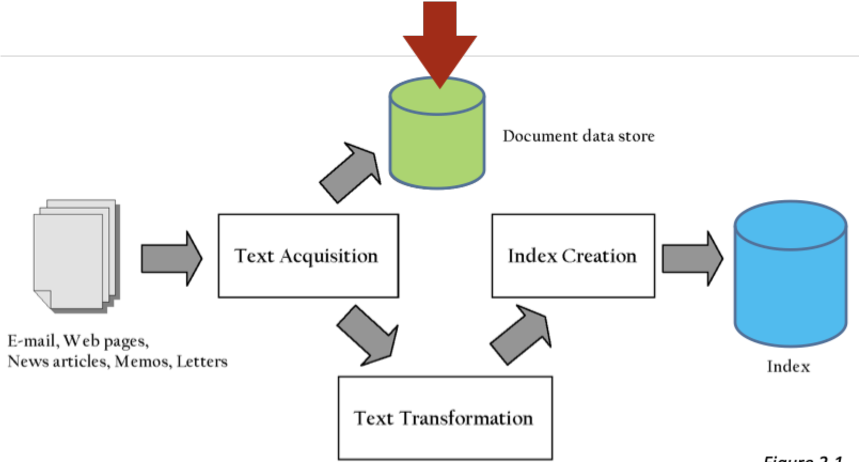
Identify and make available the documents that will be searched

Figure 2.1

Text acquisition

- **Crawler**: identifies and acquires documents for search engine
    - Many types: web, enterprise, desktop, etc.
    - Web crawlers follow links to find documents
        - Must efficiently find huge numbers of web pages (coverage) and keep them up-to-date (freshness)
        - Single site crawlers for site search
        - Topical or focused crawlers for vertical search
    - Document crawlers for enterprise and desktop search
        - Follow links and scan directories
- **Feeds**: real-time streams of documents
    - E.g., web feeds for news, blogs, video, radio, TV
    - RSS is common standard

Document data store



Document data store

E-mail, Web pages,
News articles, Memos, Letters

Text Acquisition

Index Creation

Index

Text Transformation

Figure 2.1

Document data store

- Stores text, metadata, and other related content for documents
  - Metadata is information about document such as type and creation date
  - Other content includes links, anchor text
- Provides fast access to document contents for search engine components
  - E.g. result list generation
- Could use relational database system
  - More typically, a simpler, more efficient storage system is used due to huge numbers of documents

Text transformation

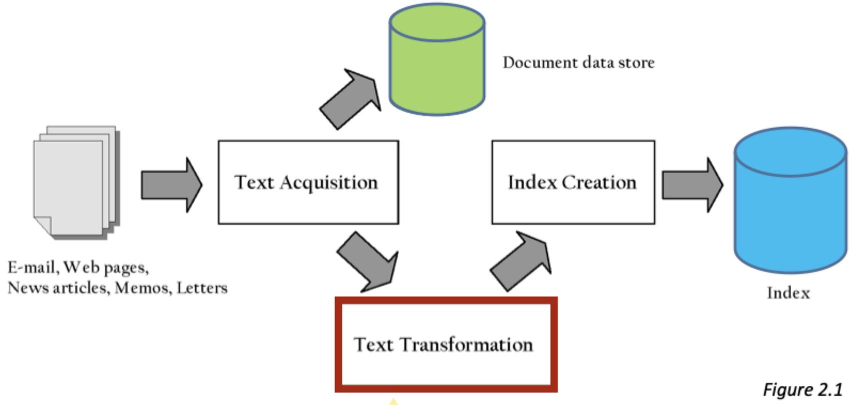


Figure 2.1

Transform documents into index terms or features

Text transformation

- Tokenization, stopword removal, stemming
- Semantic annotation
  - Named entity recognition
  - Text categorization
  - ...
- Link analysis
  - Anchor text extraction
  - ...

Index creation

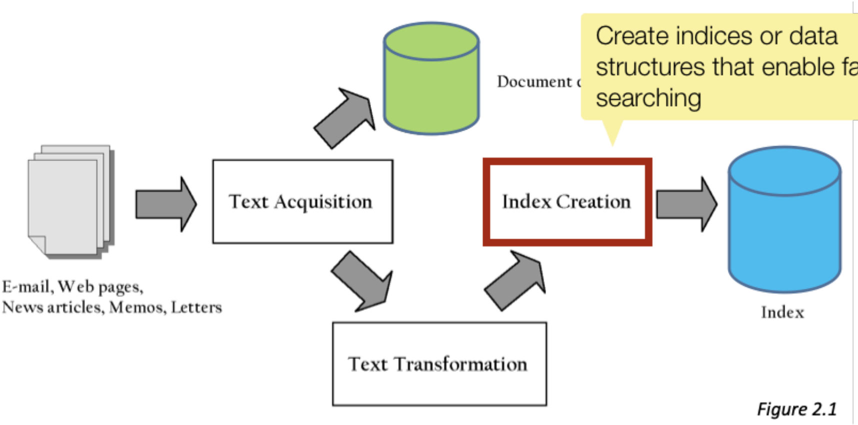


Figure 2.1

Create indices or data structures that enable fast searching

Index creation

- Gathers counts and positions of words and other features used in ranking algorithm
- Format is designed for fast query processing
- Index may be distributed across multiple computers and/or multiple sites
- (More in a bit)

Query process

Query process

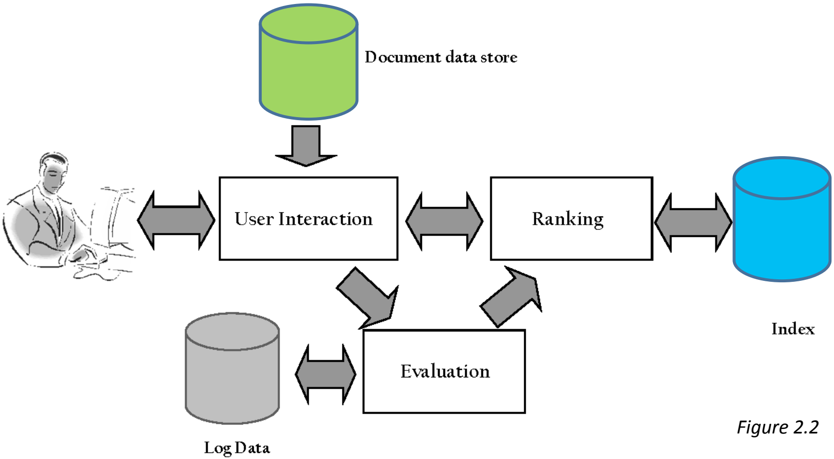


Figure 2.2

User interaction

Interface between the person doing the searching and the search engine

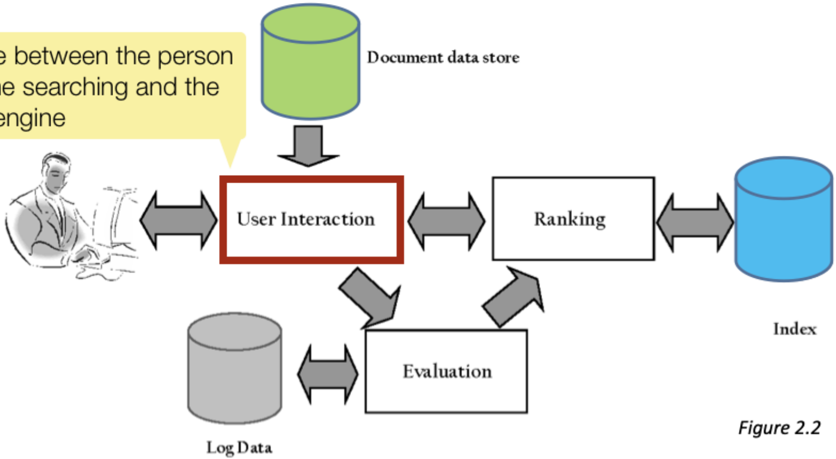
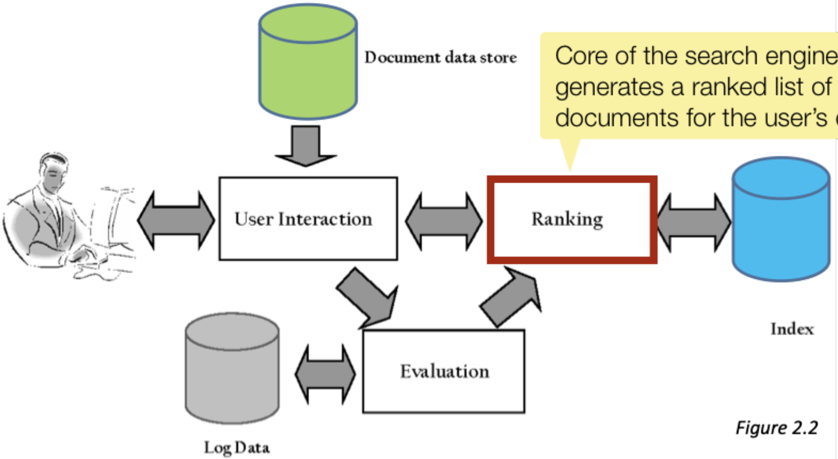


Figure 2.2

User interaction

- **Query input**: accepting the user's query and transforming it into index terms
  - Most web search query languages are very simple (i.e., small number of operators)
  - There are more complicated query languages (proximity operators, structure specification, etc.)
- **Results output**: taking the ranked list of documents from the search engine and organizing it into the results shown to the user
  - Generating *snippets* to show how queries match documents
  - *Highlighting* matching words and passages
  - May provide *clustering* of search results and other visualization tools

Ranking



Core of the search engine: generates a ranked list of documents for the user's query

Figure 2.2

Ranking

- Calculates scores for documents using a *ranking algorithm*, which is based on a *retrieval model*
- Core component of search engine
- Many variations of ranking algorithms and retrieval models exist
- **Performance optimization**: designing ranking algorithms for efficient processing
  - *Term-at-a-time* vs. *document-at-a-time* processing
  - *Safe* vs. *unsafe* optimizations
- **Distribution**: processing queries in a distributed environment
  - *Query broker* distributes queries and assembles results

Evaluation

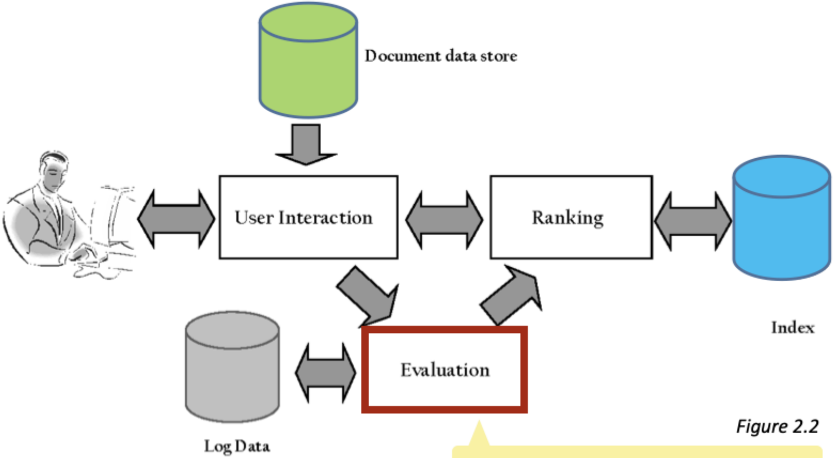


Figure 2.2

Measure and monitor effectiveness and efficiency. Record and analyze usage data

Evaluation

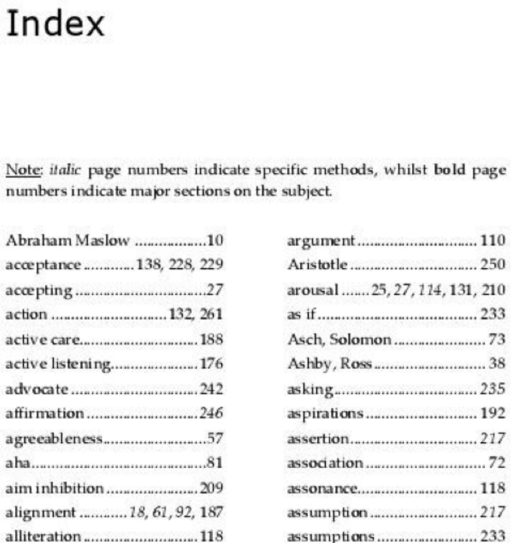
- **Logging** user queries and interaction is crucial for improving search effectiveness and efficiency
    - *Query logs* and *clickthrough data* used for query suggestion, spell checking, query caching, ranking, advertising search, and other components
- **Ranking analysis**: measuring and tuning ranking effectiveness
- **Performance analysis**: measuring and tuning system efficiency

Indexing

Indices

- Text search has unique requirements, which leads to unique data structures
- Indices are data structures designed to make search faster
- Most common data structure is the *inverted index*
    - General name for a class of structures
    - "Inverted" because documents are associated with words, rather than words with documents
    - Similar to a concordance

Motivation



Index

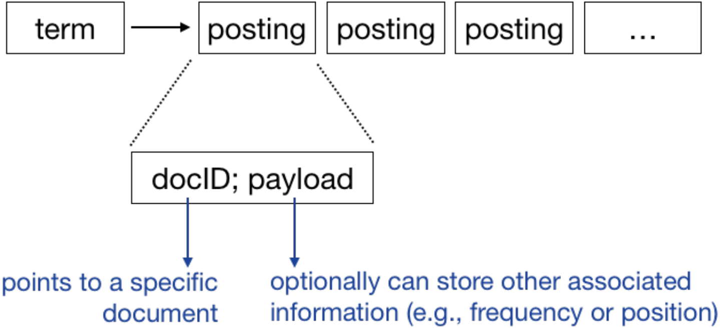
Note: *italic* page numbers indicate specific methods, whilst **bold** page numbers indicate major sections on the subject.

Abraham Maslow	10
acceptance	138, 228, 229
accepting	*27*
action	**132, 261**
active care	188
active listening	176
advocate	242
affirmation	*246*
agreeableness	*57*
aha	81
aim inhibition	209
alignment	*18, 61, 92,* 187
alliteration	118
argument	110
Aristotle	250
arousal	25, *27, 114,* **131**, 210
as if	233
Asch, Solomon	73
Ashby, Ross	38
asking	*235*
aspirations	192
assertion	*217*
association	*72*
assonance	118
assumption	*217*
assumptions	233

Inverted Index

- Each index term is associated with a *postings list* (or *inverted list*)
    - Contains lists of documents, or lists of word occurrences in documents, and other information
    - Each entry is called a *posting*
    - The part of the posting that refers to a specific document or location is called a *pointer*
        - Each document in the collection is given a unique number (docID)
    - The posting can store additional information, called the *payload*
    - Lists are usually *document-ordered* (sorted by docID)

Postings list



points to a specific document

optionally can store other associated information (e.g., frequency or position)

Example

S_1 Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.

S_2 Fishkeepers often use the term tropical fish to refer only those requiring fresh water, with saltwater tropical fish referred to as marine fish.

S_3 Tropical fish are popular aquarium fish, due to their often bright coloration.

S_4 In freshwater fish, this coloration typically derives from iridescence, while salt water fish are generally pigmented.

Four sentences from the Wikipedia entry for *tropical fish*

Simple inverted index

Each document that contains the term is a posting. No additional payload.

docID

and	1			
aquarium	3			
are	3	4		
around	1			
as	2			
both	1			
bright	3			
coloration	3	4		
derives	4			
due	3			
environments	1			
fish	1	2	3	4
fishkeepers	2			
found	1			
fresh	2			
freshwater	1	4		
from	4			
generally	4			
in	1	4		
include	1			
including	1			
iridescence	4			
marine	2			
often	2	3		
only	2			
pigmented	4			
popular	3			
refer	2			
referred	2			
requiring	2			
salt	1	4		
saltwater	2			
species	1			
term	2			
the	1	2		
their	3			
this	4			
those	2			
to	2	3		
tropical	1	2	3	
typically	4			
use	2			
water	1	2	4	
while	4			
with	2			
world	1			

Inverted index with counts

The payload is the frequency of the term in the document.

Supports better ranking algorithms.

docID: freq

and	1:1			
aquarium	3:1			
are	3:1	4:1		
around	1:1			
as	2:1			
both	1:1			
bright	3:1			
coloration	3:1	4:1		
derives	4:1			
due	3:1			
environments	1:1			
fish	1:2	2:3	3:2	4:2
fishkeepers	2:1			
found	1:1			
fresh	2:1			
freshwater	1:1	4:1		
from	4:1			
generally	4:1			
in	1:1	4:1		
include	1:1			
including	1:1			
iridescence	4:1			
marine	2:1			
often	2:1	3:1		
only	2:1			
pigmented	4:1			
popular	3:1			
refer	2:1			
referred	2:1			
requiring	2:1			
salt	1:1	4:1		
saltwater	2:1			
species	1:1			
term	2:1			
the	1:1	2:1		
their	3:1			
this	4:1			
those	2:1			
to	2:2	3:1		
tropical	1:2	2:2	3:1	
typically	4:1			
use	2:1			
water	1:1	2:1	4:1	
while	4:1			
with	2:1			
world	1:1			

Inverted index with term positions

There is a separate posting for each term occurrence in the document. The payload is the term position.

Supports proximity matches. E.g., find “tropical” within 5 words of“fish”

docID. position

and	1,15				
aquarium	3,5				
are	3,3	4,14			
around	1,9				
as	2,21				
both	1,13				
bright	3,11				
coloration	3,12	4,5			
derives	4,7				
due	3,7				
environments	1,8				
fish	1,2	1,4	2,7	2,18	2,23
			3,2	3,6	4,3
			4,13		
fishkeepers	2,1				
found	1,5				
fresh	2,13				
freshwater	1,14	4,2			
from	4,8				
generally	4,15				
in	1,6	4,1			
include	1,3				
including	1,12				
iridescence	4,9				

marine	2,22	
often	2,2	3
only	2,10	
pigmented	4,16	
popular	3,4	
refer	2,9	
referred	2,19	
requiring	2,12	
salt	1,16	4
saltwater	2,16	
species	1,18	
term	2,5	
the	1,10	2
their	3,9	
this	4,4	
those	2,11	
to	2,8	2
tropical	1,1	1
typically	4,6	
use	2,3	
water	1,17	2
while	4,10	
with	2,15	
world	1,11	

Issues

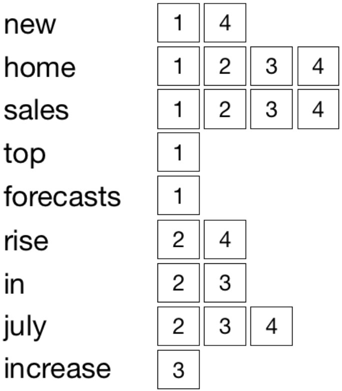
- Compression
  - Inverted lists are very large
  - Compression of indexes saves disk and/or memory space
- Optimization techniques to speed up search
  - Read less data from inverted lists
    - "Skipping" ahead
  - Calculate scores for fewer documents
    - Store highest-scoring documents at the beginning of each inverted list
- Distributed indexing

Example

Create a simple inverted index for the following document collection

Doc 1	new home sales top forecasts
Doc 2	home sales rise in july
Doc 3	increase in home sales in july
Doc 4	july new home sales rise

Solution



new	1	4		
home	1	2	3	4
sales	1	2	3	4
top	1			
forecasts	1			
rise	2	4		
in	2	3		
july	2	3	4	
increase	3			

Exercise #1

- Build an inverted index
- Code skeleton on GitHub: `exercises/lecture_07/exercise_1.ipynb` (make a local copy)

Reading

- Text Data Management and Analysis (Zhai&Massung)
  - Sections 5.3, 5.4
  - Sections 8.1, 8.2
  - Sections 10.1, 10.2
  - (optional) Sections 8.5, 8.6