Content

©HAGEN 2017

Educational Objectives

- Get the big picture of a search engine
- Know the building blocks
- Understand their relationships
- Comprehend the abstract indexing process
- Comprehend the abstract query process
- Know the key components and their functions

©HAGEN 2017

Why Architecture?

Software architecture

Describes

- Components
- Interfaces
- Relationships

Idea

- Present high-level descriptions
- Not code level!
- Enables to see the broader context when discussing specific techniques

Architecture usually designed to

- Ensure that a system will satisfy application requirements

©HAGEN 2017

Why Architecture?

Requirements for search engines

Effectiveness

- Quality
- Retrieve the most relevant set of documents possible for a query

Efficiency

- Speed
- Process queries as quickly as possible

Other goals usually also fall into these two categories, e.g.:

- Freshness of results corresponds to effectiveness
- Bigger indexed collection has to do with efficiency

©HAGEN 2017

Why Architecture?

Requirements determine architecture

Effectiveness

- Sophisticated text statistics to improve relevance

Efficiency

- Special data structures for fast retrieval

©HAGEN 2017

Why Architecture?

The two processes

General design of search engines established over decades

Supports two major functions

- Indexing process
  Build structures that enable searching
- Query process
  Use structures + query to rank documents

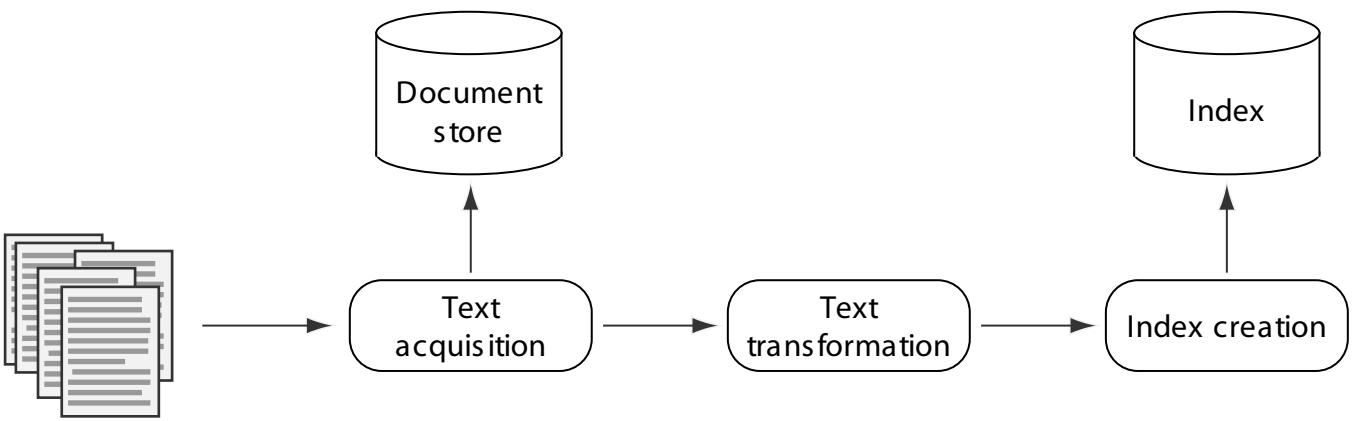
Asides:

- Specific components will be mentioned and briefly explained
- Discussed in much more detail in later parts of the course

©HAGEN 2017

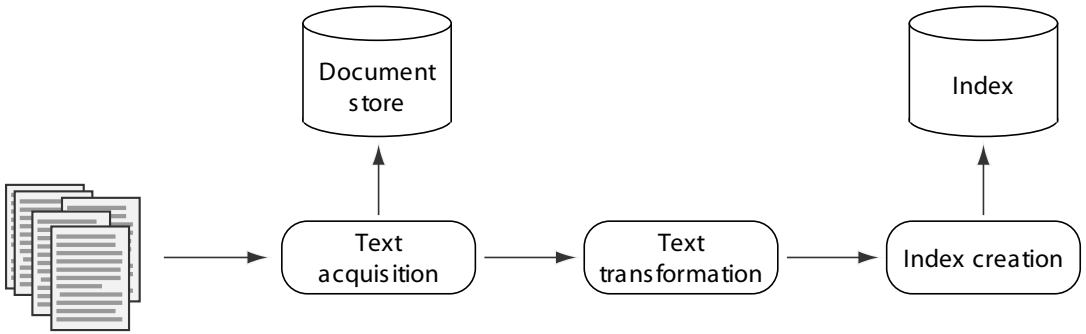
Indexing Process

The high-level building blocks



©HAGEN 2017

Indexing Process



Text acquisition

Task: Identify and make available documents to be searched

- Sometimes existing collection used (e.g., our search engine ChatNoir)
- Often collection needs to be build

Components:

- Crawler (for the web, an intranet, a desktop, etc.)
- Feed reader
- Converter (for file formats)
- Document store (documents + metadata (type, length, etc.))

©HAGEN 2017

Indexing Process

Text acquisition: Crawler

Task: Responsible to identify and acquire documents

- Web crawler: Follows links to discover new web pages
  - Sounds dead simple but there are challenges

©HAGEN 2017

Indexing Process

2-minute exercise

What are challenges for web crawling?

©HAGEN 2017

Indexing Process

Text acquisition: Crawler

Task: Responsible to identify and acquire documents

- Web crawler: Follows links to discover new web pages
  - Sounds dead simple but there are challenges
  - Huge volume of new pages $\rightarrow$ efficiency
  - Pages change; need re-crawl $\rightarrow$ when?
- Site crawler: like web crawler but restricted to a site
- Focused / topical crawler: Only crawl specific topics
  - Classifier needed
  - Used for vertical searches: GoogleScholar, medical information
- Enterprise crawler: Only crawl enterprise related documents
  - Follow links to find web and intranet pages
  - Scan corporate directories, emails, databases, ...
- Document crawler: Scan personal directories

©HAGEN 2017

Indexing Process

Text acquisition: Feed reader

Task: Similar to crawling; identify and acquire documents

- For documents provided by some feed
- Might even be audio or video
- Monitors feed and provides content when it arrives
- Might also involve classification for specific topics

©HAGEN 2017

Indexing Process

Text acquisition: Converter

Task: Create plain text from crawls and feeds

- Documents come in a variety of formats (HTML, XML, PDF, DOC, PPT, . . .)
- To search text and metadata, conversion needed
- Various utilities available for different formats

- Other problem: Text encoding
- ASCII and Unicode as two standards (128/256 vs. 65,536 characters)
- Problems:
  - Documents often “lie” about their encoding
  - UNIX vs. Windows
- Search engine needs consistent internal encoding

©HAGEN 2017

Indexing Process

Text acquisition: Document store

Task: Manage large document collection and metadata

- Document contents typically stored compressed
- “Hey, but the original documents are available on the web!?”

©HAGEN 2017

Indexing Process

2-minute exercise

Why redundant local mirroring of the documents?

©HAGEN 2017

Indexing Process

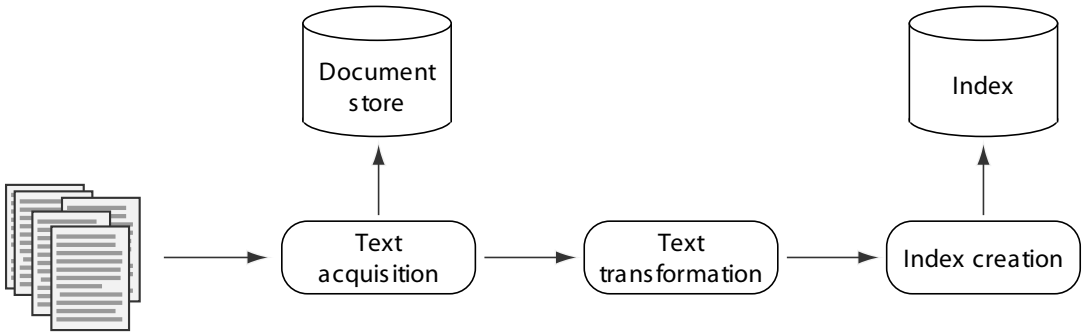
Text acquisition: Document store

Task: Manage large document collection and metadata

- Document contents typically stored compressed
- “Hey, but the original documents are available on the web!?”
  - Document store enables very fast access
  - Original might not always be accessible

©HAGEN 2017

Indexing Process



Text transformation

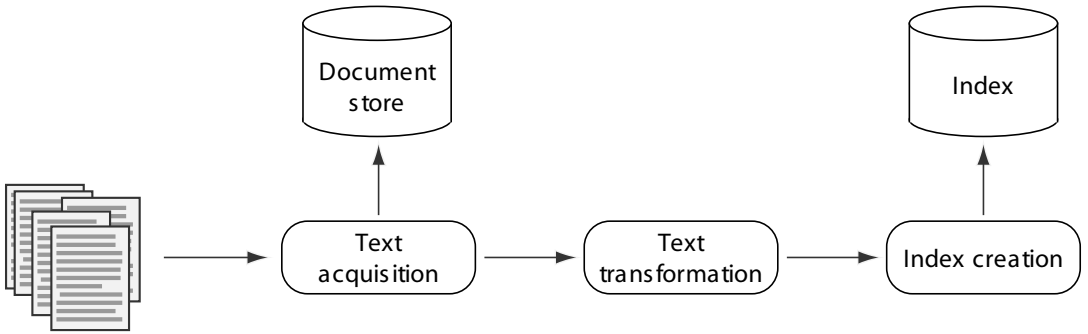
Task: Create index terms and features from the text

Index terms and features

- Parts of the document stored in the index and used for searching
- The term “feature” comes from the field of machine learning
- Simplest index terms are words (but not every word might be used for searching!)
- Other examples: phrases, named entities, dates, links, etc.
- Index terms often just named “terms”
- They form the index vocabulary

©HAGEN 2017

Indexing Process



Text transformation

Task: Create index terms and features from the text

Components:

- Parser
- Stopping
- Stemmer / Lemmatizer
- Link extraction
- Information extraction
- Classifier

©HAGEN 2017

Indexing Process

Text transformation: Parser

Task: Process token sequence and recognize document structure (title, body, . . .)

- First step: Tokenizing the text
- Tokens often the same as words
- Subtlety: Documents and queries have to be tokenized the same way
- Again, the exact details do matter a lot
  - Let a token be any alphanumeric character sequence
  -
  -
  -
-
-

©HAGEN 2017

Indexing Process

2-minute exercise

What might be problems with alphanumeric tokens?

©HAGEN 2017

Indexing Process

Text transformation: Parser

Task: Process token sequence and recognize document structure (title, body, . . .)

- First step: Tokenizing the text
- Tokens often the same as words
- Subtlety: Documents and queries have to be tokenized the same way
- Again, the exact details do matter a lot
  - Let a token be any alphanumeric character sequence
  - What about special characters? (capital letters, apostrophe, hyphen)
  - What about “apple” vs. “Apple”?
  - Apostrophe in “O’Connor” the same as in “owner’s”?
- Chinese does not even have a good tokenizer
- Markup tags need special care to not loose the respective information (e.g., words in headings more important than in body)

©HAGEN 2017

Indexing Process

Text transformation: Stopping

Task: Remove common words from token stream

- Most common words typically function words
- Form sentence structure but not really content relevant
- Examples: the, of, to, for
- Removal reduces index size
- Depending on retrieval model, usually not much influence on relevance
- But how many and what words to exclude?
- Stop word list used (some contain hundreds of words)
- Problem: How to search for “to be or not to be?”
- Stop word list often small for documents and larger for queries
- To stop aggressively, conservatively, or not at all?
- Effectiveness usually not improved that much

©HAGEN 2017

Indexing Process

Text transformation: Stemmer / Lemmatizer

Task: Group words from a common stem

- Example: “statistics” should also match “statistic” and “statistical”
- Replace word occurrences from such group by one designated term
- Likelihood of match during retrieval increased
- Stemming usually removes affixes (which can lead to non-word terms)
  - Suffixes: work*er*
  - Prefixes: *mega*volt
  - Infixes: un-*bloody*-likely
- Lemmatizing: replacements are real words
- Aggressive stemming can cause problems

©HAGEN 2017

Indexing Process

2-minute exercise

What are problems related to aggressive stemming?

©HAGEN 2017

Indexing Process

Text transformation: Stemmer / Lemmatizer

Task: Group words from a common stem

- Example: “statistics” should also match “statistic” and “statistical”
- Replace word occurrences from such group by one designated term
- Likelihood of match during retrieval increased
- Stemming usually removes affixes (which can lead to non-word terms)
  - Suffixes: work*er*
  - Prefixes: *mega*volt
  - Infixes: un-*bloody*-likely
- Lemmatizing: replacements are real words
- Aggressive stemming can cause problems
- Often conservative stemming used (e.g., remove plural-s)
- Or no stemming at all but query expansion with related words
- More complicated morphology $\rightarrow$ stemming more important (e.g., Arabic)
- For Chinese, stemming not effective

©HAGEN 2017

Indexing Process

Text transformation: Link extraction

Task: Extract links and anchor texts

- Information stored in document store
- Link analysis like PageRank makes extensive use of link structure
- Anchor text: clickable text of a web link
- Anchor text often describes the page the link points to
- Anchors enhance text content of the linked document
- Links and anchors can significantly improve effectiveness

©HAGEN 2017

Indexing Process

Text transformation: Information extraction

Task: Identify more complex index terms

- For example, words in bold or in headings (= more important)
- In general, additional computation needed
- Noun phrase detection requires part-of-speech tagging (POS)
- Named entity recognition also important (person names, companies, locations, dates)
- Research problems from the field of natural language processing (NLP)

©HAGEN 2017

Indexing Process

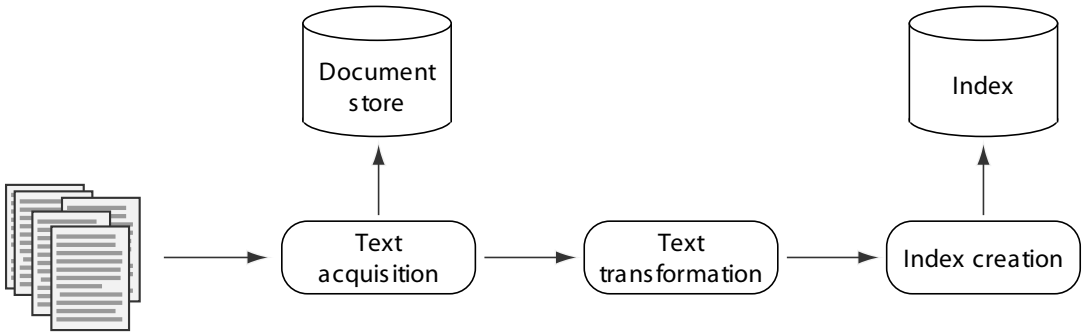
Text transformation: Classifier

Task: Identify and assign class-related metadata

- Assign labels like “sports” or “politics” to documents
- Spam classification very important for effectiveness
- Non-content part identification (e.g., ads)
- Clustering used to group related documents without predefined categories
- Such categories important for ranking and user interaction

©HAGEN 2017

Indexing Process



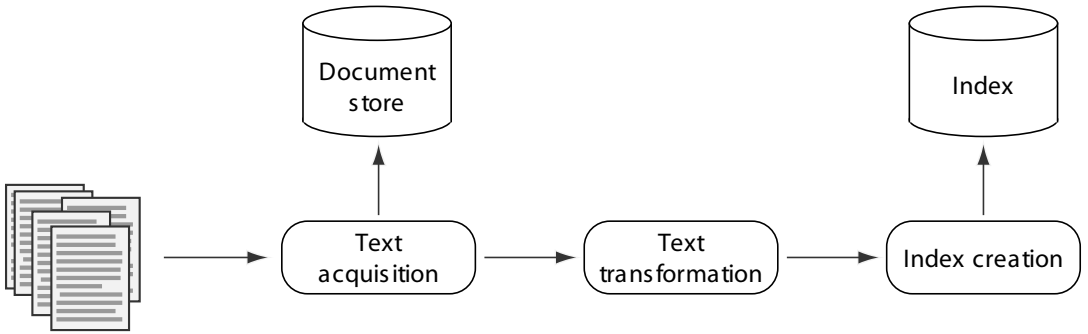
Index creation

Task: Create data structures from transformed text

- Indexes enable fast searching
- Creation must be efficient with respect to time and space
- Efficient updates often necessary (e.g., new or changed documents)
- Most common form: inverted index / inverted file
  - For every index term a list of documents that contain it
  - Inverted in the sense of opposite of a document file (list of terms)
- Many variations of inverted indexes
- Particular form used is very important

©HAGEN 2017

Indexing Process



Index creation

Task: Create data structures from transformed text

Components:

- Document statistics
- Weighting
- Inversion
- Distribution

©HAGEN 2017

Indexing Process

Index creation: Document statistics

Task: Gather and record statistical information about words, features, documents

- Information used by ranking component to compute scores for documents
- Frequencies of index terms (words and more complex features) in individual documents
- Position of index terms in documents
- Frequencies over document groups (e.g., all labeled with “sports”)
- Document lengths (in number of tokens)
- Retrieval model and ranking algorithm determine data actually required
- Statistics stored in lookup tables (e.g., key-value stores)

©HAGEN 2017

Indexing Process

Index creation: Weighting

Task: Calculate weights for words in documents

- Index term weights reflect relative importance in documents
- Weights are used in score computation for ranking
- Weights stored in lookup tables
- Weights could be calculated during query process
  - Some actually are (the ones that need query information)
  - Most are pre-computed $\rightarrow$ efficiency
- Most common in older retrieval models: *tf* $\cdot$ *idf*
  - *tf*: term frequency (number of occurrences in a document)
  - *idf*: inverse document frequency (in the collection)
    - High weights for terms that occur in few documents
    - Typical formula: $\log(N/n)$
    - N is the number of indexed documents
    - n is the number of documents that contain the term

©HAGEN 2017

Indexing Process

Index creation: Inversion

Task: Change document-term information to term-document information

- Core of the indexing process
- Challenge: efficiency; not only for initial creation but also on updates
- Inverted index format designed for fast query processing
- Inverted index format depends on ranking algorithm
- Indexes typically compressed $\rightarrow$ efficiency

©HAGEN 2017

Indexing Process

Index creation: Distribution

Task: Distribute index over multiple computers and network sites

- Essential for efficiency; often also called “sharding” or “partitioning”
- Document distribution
  - Split collection; smaller indexes for sub-collections on different machines
  -
  -
- Term distribution
  - Split the index for the entire collection by terms
  - Different machines serve different terms
  -
- Replication
  - Copies of indexes or parts on multiple sites
  -
  -

©HAGEN 2017

Indexing Process

2-minute exercise

What are arguments for sharding by documents, by terms, and replication?

©HAGEN 2017

Indexing Process

Index creation: Distribution

Task: Distribute index over multiple computers and network sites

- Often called “sharding” and essential for efficiency
- Document distribution
  - Split collection; smaller indexes for sub-collections on different machines
  - Enables parallelism for indexing and query processing
  - Smaller indexes often faster
- Term distribution
  - Split the index for the entire collection by terms
  - Different machines serve different terms
  - Not all machines have to process each query
- Replication
  - Copies of indexes or parts on multiple sites
  - Reduced delays during query processing
  - Fault tolerance

©HAGEN 2017