Search Engine Architecture
Software Architecture

• The high level structure of a software system
• Software components
• The interfaces provided by those components
• The relationships between those components
Primary Goals of Search Engines

• Effectiveness (quality): to retrieve the most relevant set of documents for a query
  – Process text and store text statistics to improve relevance
• Efficiency (speed): process user queries as quickly as possible
  – Use specialized data structures
• Specific goals usually fall into the above primary goals
  – Example: handling changing document collections – both an effectiveness issue and an efficiency issue
Two Major Functions

• Search engine components support two major functions

• The index process: building data structures that enable searching

• The query process: using those data structures to produce a ranked list of documents for a user’s query
The Indexing Process

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

- Text Transformation

- Index Creation

- Document data store

- Index
Text Acquisition

- Identifying and making available the documents that will be searched

How?
- Crawling or scanning the web, a corporate intranet, or other sources of information
- Building a document data store containing the text and metadata for all the documents
  - Metadata: document type, document structure, document length, ...
Crawlers

- Identifying and acquiring documents for the search engine
- Web crawler: following links on web pages to discover new pages
  - Efficiency: how to handle a huge volume of new pages and updated pages?
- Web crawler restricted to a single site supports site search
- Topic-based/focused crawlers: using classification techniques to restrict pages that are likely relevant to a specific topic
  - Used in vertical or topical search
- Enterprise document crawler: following links to discover both internal and external pages
Document Feeds

• A mechanism for accessing a real-time stream of documents

• RSS: a common standard used for web feeds for content such as news, blogs, or video
  – An RSS reader subscribes to RSS feeds, and provides new content when it arrives
  – RSS feeds are formatted in XML
Conversion

• Converting a variety of formats (e.g., HTML, XML, PDF, …) into a consistent text and metadata format

• Resolving encoding problem
  – Using ASCII (7 bits) or extended ASCII (8 bits) for English
  – Using Unicode (16 bits) for international languages
Document Data Stores

- A simple database to manage large numbers of documents and structured data
- Document components are typically stored in a compressed form for efficiency
- Structured data consists of document metadata and other information extracted from the documents such as links and anchor text
Discussion

• Why do we need document data stores local to search engines?
  – The original documents are available on the web …
The Indexing Process

Text Acquisition

Text Transformation

Index Creation

Document data store

Index

E-mail, Web pages, News articles, Memos, Letters
Text Transformation / Index Creation

- Text transformation: transforming documents into index terms or features
  - Index terms: the parts of a document that are stored in the index and used in searching
  - Features: parts of a text document that is used to represent its content
  - Examples: phrases, names, dates, links, ...
  - Index vocabulary: the set of all the terms that are indexed for a document collection

- Index creation: creating the indexes or data structures
  - Example: building inverted list indexes
Parsers, Stopping, and Stemming

• Processing the sequence of text tokens in the document to recognize structural elements such as titles, figures, links, and headings
  – Tokenization: identifying units to be indexed
  – Using syntax of markup languages to identify structures
• Stopping: removing common words from the stream of tokens, e.g., the, of, to, …
  – Reducing index size considerably
• Stemming: group words that are derived from a common stem
  – Example: fish, fishes, fishing
  – Increase the likelihood that words used in queries and documents will match
Other Text Transformation Tasks

• Link extraction and analysis
  – Links can be indexed separately from the general text content
  – Using link analysis algorithms, e.g., PageRank, to quantify page popularity and find authority pages
  – Using anchor text to enhance the text content of a page that the link points to

• Information extraction: identifying index terms that are more complex than single words
  – Entity identification, e.g., finding names

• Classifiers: identifying class-related metadata for documents or parts of documents
  – Example: finding spam documents and non-content parts of documents (e.g., ads)
  – Alternatively, clustering related documents
The Indexing Process

Text Acquisition

Text Transformation

Index Creation

Document data store

Index

E-mail, Web pages, News articles, Memos, Letters
Collecting Document Statistics

- Gathering and recording statistical information about words, features, and documents
  - Statistics will be used to compute scores of documents
  - Stored in lookup tables
- Examples
  - Counts of index term occurrences
  - Positions in the documents where the index terms occurred
  - Counts of occurrences over groups of documents
  - Lengths of documents in terms of the number of tokens
- Actual data depend on the retrieval model and the associated ranking method
Weighting

• Calculating index term weights using the document statistics and storing them in lookup tables
  – Pre-computation can improve query answering efficiency
• TF/IDF weighting
  – TF (the term frequency): the frequency of index term occurrences in a document
  – IDF (inverse document frequency): the inverse of the frequency of index term occurrences in all documents – \(N/n\) (\(N\): # documents indexed, \(n\): # documents containing a particular term)
Inversion and Index Distribution

• Inversion: changing the stream of document-term information coming from the text transformation component into term-document information for the creation of inverted indexes
  – Core of the indexing process
  – The number of documents is large
  – The indexes are updated with new documents from feeds and crawls, and are often compressed for high efficiency

• Index distribution: distributing indexes across multiple computers/sites on a network
  – Document distribution, term distribution, and replication
The Indexing Process

Text Acquisition

Document data store

Index Creation

Index

Text Transformation

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

• Providing the interface between users and the search engine

• Tasks
  – Accepting a user query and transforming it into index terms
  – Taking the ranked list of documents from the search engine and organizing it into the results shown to the user
    • Generating snippets to summarize documents
  – Refining a user query to better represent the information need
Query Input

- Query input: providing an interface and a parser for a query language
- Simple languages: some Boolean operations on keywords
- More complex languages may consider issues like proximity of keywords
Discussion

• What is the language used in Google and Bing?
• Do you think using punctuations in queries a good idea?
Transformation

- Query transformation: improving the initial query before and after producing a document ranking
- Tokenizing, stopping, and stemming queries
- Spell checking and query suggestion using query logs
- Query expansion based on analysis of term occurrences in documents
- Relevance feedback based on term occurrences in documents that are identified as relevant by a user
Results Output

- Constructing the display of ranked documents
- Generating snippets
- Highlighting important words and passages
- Clustering documents to identify related groups
- Adding appropriate ads
- Translating multiple language documents into the common language
The Query Process

- Document data store
- User Interaction
- Ranking
- Log Data
- Evaluation
- Index
Ranking and Evaluation

• Ranking: taking the transformed query from the user interaction component and generating a ranked list of documents using scores based on a retrieval model
  – The core of a search engine
  – Both effectiveness and efficiency matter

• Evaluation: measuring and monitoring the effectiveness and efficiency
  – Recording and analyzing user behavior using log data
  – Evaluation results are used to tune and improve the ranking component
Scoring

• Assigning a score to a document according to the query and a retrieval model
• Some models are efficient: Boolean model
• Some models are costly: language model – compute the probability that a document is relevant to the query based on the words in the document
Performance Optimization

- Find an efficient way to compute the scores
- Using indexes term by term, term-at-a-time scoring
- Accessing all indexes for the query terms simultaneously, document-at-a-time
- Safe optimizations guarantee that the score calculated will be the same as the scores without optimization
- Unsafe optimization (approximation) may be faster in the tradeoff of accuracy
Distributed Processing

- Queries can be distributed to processors in a network
  - Distributed by a query broker
- Caching: indexes or even ranked document lists from previous queries are left in local memory
  - Reusing previous query results for popular queries
Evaluation Tasks

- **Logs**: clickthrough data and dwell time (time spent looking at a document)
  - Can be used for spell checking, query suggestions, query caching, mapping ads to users, ...
- **Ranking analysis**: measuring the ranking method and comparing to alternatives
  - Top ranked documents rather than the whole ranked list are more important
- **Performance analysis**: monitoring and improving overall system performance
  - Response time, throughput, network usage
  - Simulation methods
Summary

- A high-level description of search engine software architecture
- The indexing process and the query process
- Building blocks and their functionalities
Data in Search Engines

- Documents
- Queries
- Logs: history of queries and answers, possibly including user interaction data, such as click-through data
- Search engine task can be tackled using individual types of data and their combinations
Discussion

• Identify the types of data used in the components of search engine architecture
Architecture of Google Search
Baidu Interactive Query Architecture

**Compute Layer**
- Meta Store
- HiveQL
- UDFs
- SerDes
- Spark SQL
- Apache Spark

**Hot Data Layer**
- Tachyon

**Storage Layer**
- Data Warehouse
- Baidu File System
Search Engine in Apps
To-Do List

• Expand the figures of the index process and query process to include the detailed functionalities.

• A “more-like-this” query occurs when a user clicks on a particular document in the result list and tells the search engine to find documents that are similar to this one. Describe which low-level components are used to answer this type of query and the sequence they are used in.

• Document filtering is an application that stores a large number of queries or user profiles and compares these profiles to every incoming document on a feed. Documents that are sufficiently similar to the profile are forwarded to that person via email or some other mechanism. Describe the architecture of a filtering engine and how it may differ from a search engine.

• Read textbook Chapter 2.