CS6200
Information Retrieval

David Smith
College of Computer and Information Science
Northeastern University
IR and Search Engines

Information Retrieval
- Relevance
  - Effective ranking
- Evaluation
  - Testing and measuring
- Information needs
  - User interaction

Search Engines
- Performance
  - Efficient search and indexing
- Incorporating new data
  - Coverage and freshness
- Scalability
  - Growing with data and users
- Adaptability
  - Tuning for applications
- Specific problems
  - e.g. Spam
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) and efficiency (response time and throughput)
Query Process

Document data store

User Interaction

Ranking

Evaluation

Index

Log Data
Indexing Process

Text Acquisition

Index Creation

Text Transformation

Document data store

Index

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

• Crawler
  – Identifies and acquires documents for search engine
  – 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*)
    • 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
Text Acquisition

• Feeds
  – Real-time streams of documents
    • e.g., web feeds for news, blogs, video, radio, TV
  – RSS is common standard
    • RSS “reader” can provide new XML documents to search engine

• Conversion
  – Convert variety of documents into a consistent text plus metadata format
    • e.g. HTML, XML, Word, PDF, etc. → XML
  – Convert text encoding for different languages
    • Using a Unicode standard like UTF-8
Text Acquisition

• 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
Indexing Process

Text Acquisition

Index Creation

Text Transformation

Document data store

Index

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

• Parser
  – Processing the sequence of text tokens in the document to recognize structural elements
    • e.g., titles, links, headings, etc.
  – *Tokenizer* recognizes “words” in the text
    • must consider issues like capitalization, hyphens, apostrophes, non-alpha characters, separators
  – *Markup languages* such as HTML, XML often used to specify structure (also JSON, PDF, closed captions, …)
    • *Tags* used to specify document elements
      – E.g., <h2> Overview </h2>
    • Document parser uses syntax of markup language (or other formatting) to identify structure
Text Transformation

• Stopping
  – Remove common words
    • e.g., “and”, “or”, “the”, “in”
  – Some impact on efficiency and effectiveness
  – Can be a problem for some queries

• Stemming
  – Group words derived from a common stem
    • e.g., “computer”, “computers”, “computing”, “compute”
  – Usually effective, but not for all queries
  – Benefits vary for different languages
Text Transformation

• Link Analysis
  – Makes use of *links* and *anchor text* in web pages
  – Link analysis identifies *popularity* and *community* information
    • e.g., PageRank
  – Anchor text can significantly enhance the representation of pages pointed to by links
  – Significant impact on web search
    • Less importance in other applications
Text Transformation

• Information Extraction
  – Identify classes of index terms that are important for some applications
  – *Named entity recognizers* identify classes such as *people, locations, companies, dates*, etc.
  – Other parsers for business addresses, event information, job postings, etc.

• Classifier
  – Identifies class-related metadata for documents
    • i.e., assigns labels to documents
    • e.g., topics, reading levels, sentiment, genre
  – Use depends on application
Indexing Process

Text Acquisition

Index Creation

Text Transformation

Document data store

Index

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

• Document Statistics
  – Gathers counts and positions of words and other features
  – Used in ranking algorithm

• Weighting
  – Computes weights for index terms
  – Used in ranking algorithm
  – e.g., *tf.idf* weight
    • Combination of *term frequency* in document and *inverse document frequency* in the collection
Index Creation

• Inversion
  – Core of indexing process
  – Converts document-term information to term-document for indexing
    • Difficult for very large numbers of documents
  – Format of inverted file is designed for fast query processing
    • Must also handle updates
    • Compression used for efficiency
Index Creation

• Index Distribution
  – Distributes indexes across multiple computers and/or multiple sites
  – Essential for fast query processing with large numbers of documents
  – Many variations
    • Document distribution, term distribution, replication
  – P2P and distributed IR involve search across multiple sites
    • For efficiency or for data encapsulation/hiding
Query Process

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

• Query input
  – Provides interface and parser for *query language*
  – Most web queries are very simple, other applications may use forms
  – Query language used to describe more complex queries and results of query transformation
    • e.g., Boolean queries, Indri and Galago query languages
    • similar to SQL language used in database applications
    • IR query languages also allow content and structure specifications, but focus on content
User Interaction

• Query transformation
  – Improves initial query, both before and after initial search
  – Includes text transformation techniques used for documents
  – *Spell checking* and *query suggestion* provide alternatives to original query
  – *Query expansion* and *relevance feedback* modify the original query with additional terms
User Interaction

• Results output
  – Constructs the display of ranked documents for a query
  – Generates *snippets* to show how queries match documents
  – *Highlights* important words and passages
  – Retrieves appropriate *advertising* in many applications
  – May provide *clustering* and other visualization tools
Query Process

1. User Interaction
2. Ranking
3. Log Data
4. Document data store
5. Index
6. Evaluation
Ranking

• Scoring
  – Calculates scores for documents using a ranking algorithm
  – Core component of search engine
  – Basic form of score is $\sum q_i d_i$
    • $q_i$ and $d_i$ are query and document term weights for term $i$
  – Many variations of ranking algorithms and retrieval models
Ranking

• 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
  – Caching is a form of distributed searching
Query Process

User Interaction → Document data store

Ranking → Evaluation

Evaluation → Log Data

Log Data → Index
Evaluation

• Logging
  – 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
How Does It *Really* Work?

- This course explains these components of a search engine in more detail
- Often many possible approaches and techniques for a given component
  - Focus is on the most important alternatives
  - i.e., explain a small number of approaches in detail rather than many approaches
  - “Importance” based on research results and use in actual search engines
  - Follow up references in text for alternatives
Topics

- *Overview*
- Architecture of a search engine
- **Data acquisition**
- Text representation
- Information extraction
- Indexing
- Query processing
- Ranking
- Evaluation
- Classification and clustering
- Social search
- More...
Topics

• For background read:
  – *Search Engines* chapter 3, or
  – *Intro to IR*, chapters 19 and 20
Exercise

• Write down 2 queries for a web search engine, each between 2 and 5 words.
• **Before** you run the queries, write down what you expect to find.
• Run these queries on 2 search engines and compare the top 10 results.
• How are the search engines different?
• What criteria did you use?