Information Retrieval

Outline

- Relevance Ranking Using Terms
- Relevance Using Hyperlinks
- Synonyms., Homonyms, and Ontologies
- Indexing of Documents
- Measuring Retrieval Effectiveness
- Web Search Engines
- Information Retrieval and Structured Data
- Directories

Information Retrieval Systems

- **Information retrieval (IR)** systems use a simpler data model than database systems
  - Information organized as a collection of documents
  - Documents are unstructured, no schema
- Information retrieval locates relevant documents, on the basis of user input such as keywords or example documents
  - e.g., find documents containing the words "database systems"
- Can be used even on textual descriptions provided with non-textual data such as images
- Web search engines are the most familiar example of IR systems

Information Retrieval Systems (Cont.)

- Differences from database systems
  - IR systems don’t deal with transactional updates (including concurrency control and recovery)
  - Database systems deal with structured data, with schemas that define the data organization
  - IR systems deal with some querying issues not generally addressed by database systems
    - Approximate searching by keywords
    - Ranking of retrieved answers by estimated degree of relevance

Keyword Search

- In **full text** retrieval, all the words in each document are considered to be keywords.
  - We use the word **term** to refer to the words in a document
- Information-retrieval systems typically allow query expressions formed using keywords and the logical connectives *and*, *or*, and *not*
  - *And*s are implicit, even if not explicitly specified
- Ranking of documents on the basis of estimated relevance to a query is critical
  - Relevance ranking is based on factors such as
    - Term frequency
      - Frequency of occurrence of query keyword in document
    - Inverse document frequency
      - How many documents the query keyword occurs in
        - Fewer ➔ give more importance to keyword
    - Hyperlinks to documents
      - More links to a document ➔ document is more important

Relevance Ranking Using Terms

- **TF-IDF** (Term frequency/Inverse Document frequency) ranking:
  - Let *n*(*d*) = number of terms in the document *d*
  - *n*(*d*, *t*) = number of occurrences of term *t* in the document *d*.
  - Relevance of a document *d* to a *term t*

$$TF(d, t) = \log\left(1 + \frac{n(d, t)}{n(d)}\right)$$

    - The log factor is to avoid excessive weight to frequent terms
  - Relevance of document to *query Q*

$$r(d, Q) = \sum_{t \in Q} \frac{TF(d, t)}{n(t)}$$

Relevance Ranking Using Terms (Cont.)

- Most systems add to the above model
  - Words that occur in title, author list, section headings, etc. are given greater importance
  - Words whose first occurrence is late in the document are given lower importance
  - Very common words such as “a”, “an”, “the”, “it” etc. are eliminated
    - Called **stop words**
  - **Proximity**: if keywords in query occur close together in the document, the document has higher importance than if they occur far apart
- Documents are returned in decreasing order of relevance score
  - Usually only top few documents are returned, not all

Similarity Based Retrieval

- Similarity based retrieval - retrieve documents similar to a given document
  - Similarity may be defined on the basis of common words
    - E.g., find *k* terms in A with highest *TF* (*d*, *t*) / *n* (*t*) and use these terms to find relevance of other documents.
- **Relevance feedback**: Similarity can be used to refine answer set to keyword query
  - User selects a few relevant documents from those retrieved by keyword query, and system finds other documents similar to these
- Vector space model: define an *n*-dimensional space, where *n* is the number of words in the document set.
  - Vector for document *d* goes from origin to a point whose *i* th coordinate is *TF* (*d*,*t*) / *n* (*t*)
  - The cosine of the angle between the vectors of two documents is used as a measure of their similarity.

Relevance Using Hyperlinks

- Number of documents relevant to a query can be enormous if only term frequencies are taken into account
- Using term frequencies makes “spamming” easy
    - E.g., a travel agency can add many occurrences of the words “travel” to its page to make its rank very high
- Most of the time people are looking for pages from popular sites
- Idea: use popularity of Web site (e.g., how many people visit it) to rank site pages that match given keywords
- Problem: hard to find actual popularity of site
    - Solution: next slide

Relevance Using Hyperlinks (Cont.)

- Solution: use number of hyperlinks to a site as a measure of the popularity or **prestige** of the site
  - Count only one hyperlink from each site (why? - see previous slide)
  - Popularity measure is for site, not for individual page
    - But, most hyperlinks are to root of site
    - Also, concept of "site" difficult to define since a URL prefix like cs.yale.edu contains many unrelated pages of varying popularity
- Refinements
  - When computing prestige based on links to a site, give more weight to links from sites that themselves have higher prestige
    - Definition is circular
    - Set up and solve system of simultaneous linear equations
  - Above idea is basis of the Google **PageRank** ranking mechanism

Relevance Using Hyperlinks (Cont.)

- Connections to **social networking** theories that ranked prestige of people
  - E.g., the president of the U.S.A has a high prestige since many people know him
  - Someone known by multiple prestigious people has high prestige
- Hub and authority based ranking
  - A **hub** is a page that stores links to many pages (on a topic)
  - An **authority** is a page that contains actual information on a topic
  - Each page gets a **hub prestige** based on prestige of authorities that it points to
  - Each page gets an **authority prestige** based on prestige of hubs that point to it
  - Again, prestige definitions are cyclic, and can be got by solving linear equations
  - Use authority prestige when ranking answers to a query

Synonyms and Homonyms

- **Synonyms**
  - E.g., document: “motorcycle repair”, query: “motorcycle maintenance”
    - Need to realize that “maintenance” and “repair” are synonyms
  - System can extend query as “motorcycle *and* (repair *or* maintenance)”
- Homonyms
  - E.g., “object” has different meanings as noun/verb
  - Can disambiguate meanings (to some extent) from the context
- Extending queries automatically using synonyms can be problematic
  - Need to understand intended meaning in order to infer synonyms
    - Or verify synonyms with user
  - Synonyms may have other meanings as well

Concept-Based Querying

- Approach
  - For each word, determine the **concept** it represents from context
  - Use one or more **ontologies**:
    - Hierarchical structure showing relationship between concepts
    - E.g., the ISA relationship that we saw in the E-R model
- This approach can be used to standardize terminology in a specific field
- Ontologies can link multiple languages
- Foundation of the **Semantic Web** (not covered here)

Indexing of Documents

- An inverted index maps each keyword K_i to a set of documents S_i that contain the keyword
  - Documents identified by identifiers
- Inverted index may record
  - Keyword locations within document to allow proximity based ranking
  - Counts of number of occurrences of keyword to compute TF
- **and** operation: Finds documents that contain all of K_1, K_2, ..., K_n.
  - Intersection $S_1 \cap S_2 \cap ..... \cap S_n$
- **or** operation: documents that contain at least one of K_1, K_2, …, K_n
  - union, $S_1 \cap S_2 \cap ..... \cap S_n$,.
- Each S_i is kept sorted to allow efficient intersection/union by merging
  - "**not**" can also be efficiently implemented by merging of sorted lists

Measuring Retrieval Effectiveness

- Information-retrieval systems save space by using index structures that support only approximate retrieval. May result in:
  - **false negative (false drop)** - some relevant documents may not be retrieved.
  - **false positive** - some irrelevant documents may be retrieved.
  - For many applications a good index should not permit any false drops, but may permit a few false positives.
- Relevant performance metrics:
  - **precision** - what percentage of the retrieved documents are relevant to the query.
  - **recall** - what percentage of the documents relevant to the query were retrieved.

Measuring Retrieval Effectiveness (Cont.)

- Recall vs. precision tradeoff:
    - Can increase recall by retrieving many documents (down to a low level of relevance ranking), but many irrelevant documents would be fetched, reducing precision
- Measures of retrieval effectiveness:
    - Recall as a function of number of documents fetched, or
    - **Precision as a function of recall**
        - Equivalently, as a function of number of documents fetched
    - E.g., "precision of 75% at recall of 50%, and 60% at a recall of 75%"
- Problem: which documents are actually relevant, and which are not

Web Search Engines

- **Web crawlers** are programs that locate and gather information on the Web
  - Recursively follow hyperlinks present in known documents, to find other documents
    - Starting from a *seed* set of documents
  - Fetched documents
    - Handed over to an indexing system
    - Can be discarded after indexing, or store as a *cached* copy
- Crawling the entire Web would take a very large amount of time
  - Search engines typically cover only a part of the Web, not all of it
  - Take months to perform a single crawl

Web Crawling (Cont.)

- Crawling is done by multiple processes on multiple machines, running in parallel
  - Set of links to be crawled stored in a database
  - New links found in crawled pages added to this set, to be crawled later
- Indexing process also runs on multiple machines
  - Creates a new copy of index instead of modifying old index
  - Old index is used to answer queries
  - After a crawl is “completed” new index becomes “old” index
- Multiple machines used to answer queries
  - Indices may be kept in memory
  - Queries may be routed to different machines for load balancing

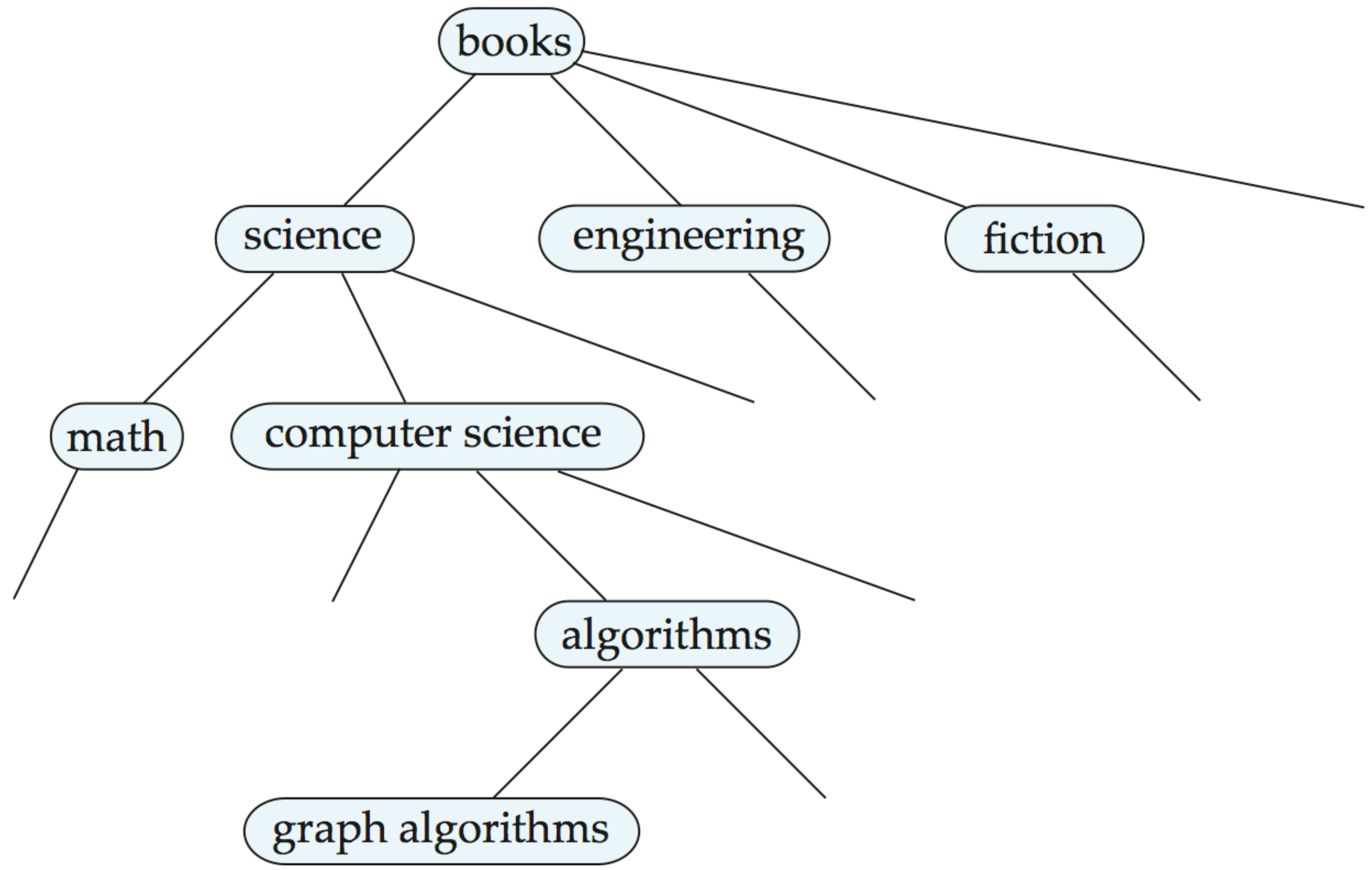
Information Retrieval and Structured Data

- Information retrieval systems originally treated documents as a collection of words
- **Information extraction systems** infer structure from documents, e.g.:
  - Extraction of house attributes (size, address, number of bedrooms, etc.) from a text advertisement
  - Extraction of topic and people named from a new article
- Relations or XML structures used to store extracted data
  - System seeks connections among data to answer queries
  - **Question answering systems**

Directories

- Storing related documents together in a library facilitates browsing
  - Users can see not only requested document but also related ones.
- Browsing is facilitated by classification system that organizes logically related documents together.
- Organization is hierarchical: **classification hierarchy**

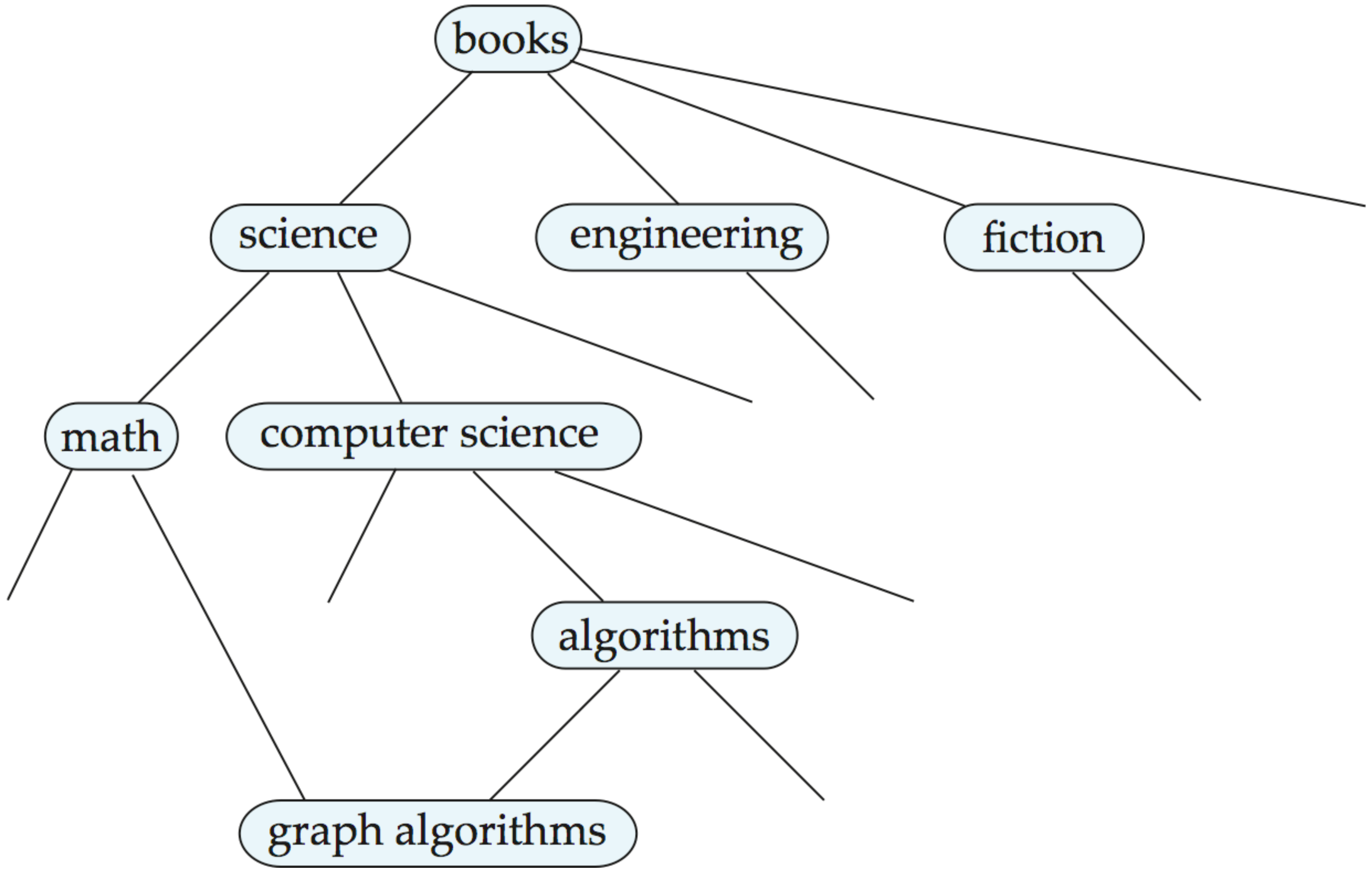
A Classification Hierarchy For A Library System



Classification DAG

- Documents can reside in multiple places in a hierarchy in an information retrieval system, since physical location is not important.
- Classification hierarchy is thus Directed Acyclic Graph (DAG)

A Classification DAG For A Library Information Retrieval System



Web Directories

- A **Web directory** is just a classification directory on Web pages
  - E.g., Yahoo! Directory, Open Directory project
  - Issues:
    - What should the directory hierarchy be?
    - Given a document, which nodes of the directory are categories relevant to the document
  - Often done manually
    - Classification of documents into a hierarchy may be done based on term similarity

End of Chapter