Chapter 21: Information Retrieval

Database System Concepts, 6th Ed.

- Chapter 1: Introduction
- **Part 1: Relational databases**
  - Chapter 2: Introduction to the Relational Model
  - Chapter 3: Introduction to SQL
  - Chapter 4: Intermediate SQL
  - Chapter 5: Advanced SQL
  - Chapter 6: Formal Relational Query Languages
- **Part 2: Database Design**
  - Chapter 7: Database Design: The E-R Approach
  - Chapter 8: Relational Database Design
  - Chapter 9: Application Design
- **Part 3: Data storage and querying**
  - Chapter 10: Storage and File Structure
  - Chapter 11: Indexing and Hashing
  - Chapter 12: Query Processing
  - Chapter 13: Query Optimization
- **Part 4: Transaction management**
  - Chapter 14: Transactions
  - Chapter 15: Concurrency control
  - Chapter 16: Recovery System
- **Part 5: System Architecture**
  - Chapter 17: Database System Architectures
  - Chapter 18: Parallel Databases
  - Chapter 19: Distributed Databases
- **Part 6: Data Warehousing, Mining, and IR**
  - Chapter 20: Data Mining
  - Chapter 21: Information Retrieval
- **Part 7: Specialty Databases**
  - Chapter 22: Object-Based Databases
  - Chapter 23: XML
- **Part 8: Advanced Topics**
  - Chapter 24: Advanced Application Development
  - Chapter 25: Advanced Data Types
  - Chapter 26: Advanced Transaction Processing
- **Part 9: Case studies**
  - Chapter 27: PostgreSQL
  - Chapter 28: Oracle
  - Chapter 29: IBM DB2 Universal Database
  - Chapter 30: Microsoft SQL Server
- **Online Appendices**
  - Appendix A: Detailed University Schema
  - Appendix B: Advanced Relational Database Model
  - Appendix C: Other Relational Query Languages
  - Appendix D: Network Model
  - Appendix E: Hierarchical Model

Chapter 21: Information Retrieval

- 21.1 Overview
- 21.2 Relevance Ranking Using Terms
- 21.3 Relevance Using Hyperlinks
- 21.4 Synonyms, Homonyms, and Ontologies
- 21.5 Indexing of Documents
- 21.6 Measuring Retrieval Effectiveness
- 21.7 Crawling and Indexing the Web
- 21.8 Information Retrieval: Beyond Ranking of Pages
- 21.9 Directories and Categories

Information Retrieval Systems [1/2]

- **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

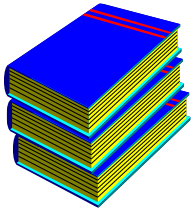
Information Retrieval Systems [2/2]

- 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

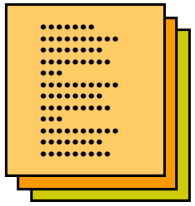
- In **full text** retrieval, all the words in each document are considered to be keywords.
  - We use the word **term** to refer to the words in a document
- Information-retrieval systems typically allow query expressions formed using keywords and the logical connectives *and, or,* and *not*
  - *"And*s" 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 (TF)
      - Frequency of occurrence of query keyword in document
    - Inverse document frequency (IDF)
      - 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

Text-Based Full Text Retrieval System



Original Texts

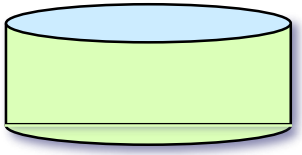
Digitization

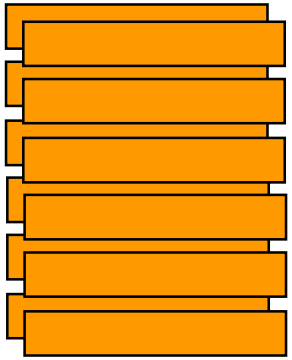


Digitized Full Texts

Information Retrieval System

User Query





Relevant Digitized Full Texts

Chapter 21: Information Retrieval

- 21.1 Overview
- 21.2 Relevance Ranking Using Terms
- 21.3 Relevance Using Hyperlinks
- 21.4 Synonyms, Homonyms, and Ontologies
- 21.5 Indexing of Documents
- 21.6 Measuring Retrieval Effectiveness
- 21.7 Crawling and Indexing the Web
- 21.8 Information Retrieval: Beyond Ranking of Pages
- 21.9 Directories and Categories

Relevance Ranking using Terms [1/3]

- **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
    - One naïve way of defining TF: Just count the number of occurrences

$$TF\,(d, t) = \frac{n(d, t)}{n(d)}$$

    - The number of occurrences depends on the length of the document
    - A document containing 10 occurrences of a term may not be 10 times as relevant as a document containing one word

Relevance Ranking using Terms [2/3]

- Applying log factor is to avoid excessive weight to frequent terms

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

- IDF $IDF(t) = 1 / n(t)$
  - n(t) is number of documents containing term t

- Relevance of a document *d* to a *query Q*

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

Relevance Ranking Using Terms [3/3]

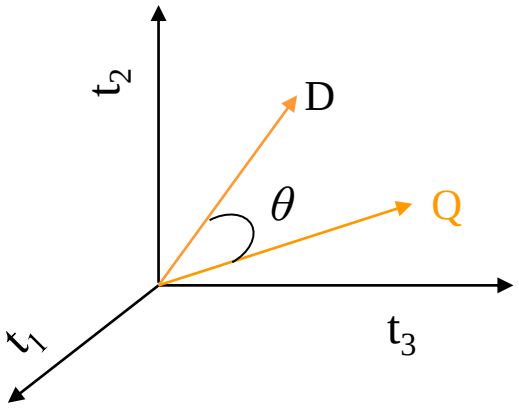
- Most systems add to the above model
  - Words that occur in title, author list, section headings, etc. are given greater importance
  - Words whose first occurrence is late in the document are given lower importance
  - Very common words such as “a”, “an”, “the”, “it” etc. are eliminated
    - Called **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 values of *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* having terms t1, t2, … tn goes from origin to a point
    - *i* th coordinate of the point is *r(d, ti) = TF* (*d, ti*) * *IDF* (*ti*)
  - The cosine of the angle between the vectors of two documents is used as a measure of their similarity

Vector Space Model

- 문서와 질의를 가중치가 부여된 색인어들의 벡터로 표현
  - $D = \{(t_1, w_{d1}), (t_2, w_{d2}), \ldots, (t_n, w_{dn})\}$ w_{di} : 문서 D 에서 i번째 색인어 t_i 의 가중치
  - $Q = \{(t_1, w_{q1}), (t_2, w_{q2}), \ldots, (t_n, w_{qn})\}$ w_{ti} : 질의 Q 에서 i번째 색인어 t_i 의 가중치



- 문서D 와 질의 Q 의 유사도

$$Sim(D,Q) = \sum_{i=1}^{n} (w_{di} \times w_{qi})$$

- 예제) 다음 문서 D 와 Q의 유사도 계산

D = {(정보, 0.3), (검색, 0.5), (시스템, 0.2)}

Q = {(정보, 0.4), (검색, 0.7)}

Sim (D,Q) = 0.3*0.4 + 0.5*0.7 = 0.47

Chapter 21: Information Retrieval

- 21.1 Overview
- 21.2 Relevance Ranking Using Terms
- 21.3 Relevance Using Hyperlinks
- 21.4 Synonyms, Homonyms, and Ontologies
- 21.5 Indexing of Documents
- 21.6 Measuring Retrieval Effectiveness
- 21.7 Crawling and Indexing the Web
- 21.8 Information Retrieval: Beyond Ranking of Pages
- 21.9 Directories and Categories

Relevance Using Hyperlinks [1/3]

- If only term frequencies are taken into account
  - Number of documents relevant to a query can be enormous
  - Using high 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

- The advent of WWW
  - Observation: 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 [2/3]

- 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

PageRank Concept

- At any time-step the random surfer
  - jumps (teleport) to any other node with probability δ
  - jumps to its direct neighbors with total probability $1 - \delta$

[wikipedia.org]

$$P[j] = \frac{\delta}{N} + (1 - \delta) * \sum_{i=1}^{N} T[i,j] * P[i]$$

N: total number of pages

$T[i,j] = 1/N_i$

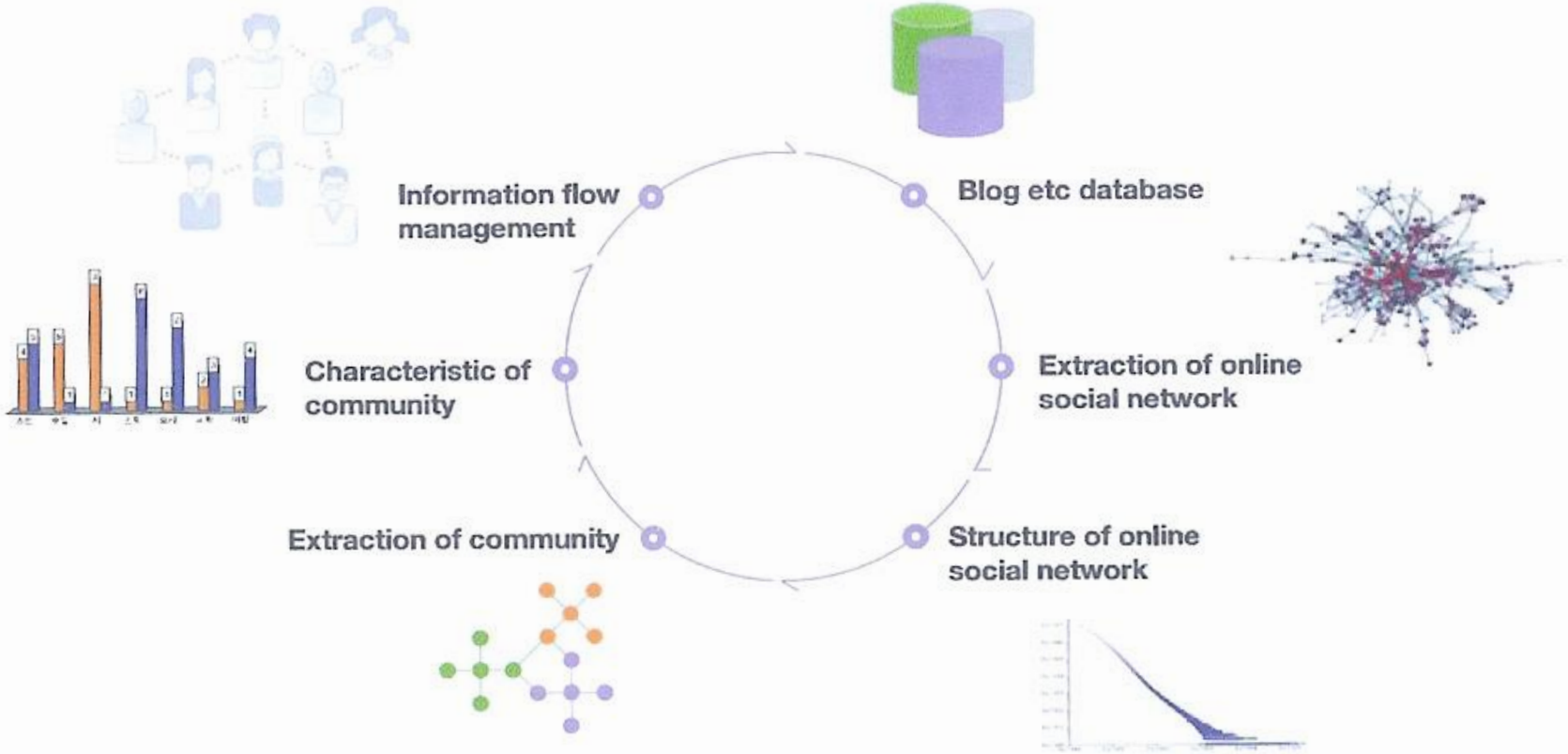
N_i: number of links out of page i

Relevance Using Hyperlinks [3/3]

- 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

Social Network Analysis

- Extract hidden information from on-line social networks in blog or other services
- Use them to provide new and more intelligent services to customers



Chapter 21: Information Retrieval

- 21.1 Overview
- 21.2 Relevance Ranking Using Terms
- 21.3 Relevance Using Hyperlinks
- 21.4 Synonyms, Homonyms, and Ontologies
- 21.5 Indexing of Documents
- 21.6 Measuring Retrieval Effectiveness
- 21.7 Crawling and Indexing the Web
- 21.8 Information Retrieval: Beyond Ranking of Pages
- 21.9 Directories and Categories

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
  - Gene Ontology
  - Ontology for home appliances
- Ontologies can link multiple languages
  - WordNet for English
  - WordNet for Korean
- Foundation of the **Semantic Web** (not covered here)

Chapter 21: Information Retrieval

- 21.1 Overview
- 21.2 Relevance Ranking Using Terms
- 21.3 Relevance Using Hyperlinks
- 21.4 Synonyms, Homonyms, and Ontologies
- 21.5 Indexing of Documents
- 21.6 Measuring Retrieval Effectiveness
- 21.7 Crawling and Indexing the Web
- 21.8 Information Retrieval: Beyond Ranking of Pages
- 21.9 Directories and Categories

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

참고자료

Inverted Index Example

ID	Text
1	Baseball is played during summer months.
2	Summer is the time for picnics here.
3	Months later we found out why.
4	Why is summer so hot here

↑ Sample document data

Dictionary and posting lists →

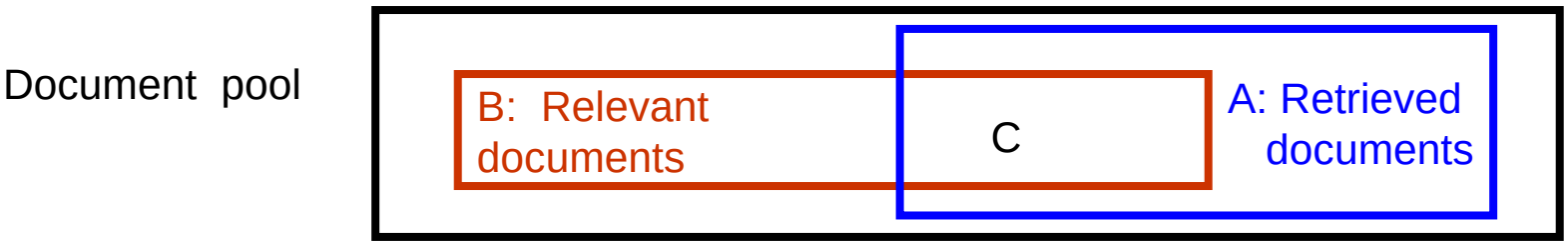
Term	Freq	Document ids
baseball	1	[1]
during	1	[1]
found	1	[3]
here	2	[2], [4]
hot	1	[4]
is	3	[1], [2], [4]
months	2	[1], [3]
summer	3	[1], [2], [4]
the	1	[2]
why	2	[3], [4]

Chapter 21: Information Retrieval

- 21.1 Overview
- 21.2 Relevance Ranking Using Terms
- 21.3 Relevance Using Hyperlinks
- 21.4 Synonyms, Homonyms, and Ontologies
- 21.5 Indexing of Documents
- 21.6 Measuring Retrieval Effectiveness
- 21.7 Crawling and Indexing the Web
- 21.8 Information Retrieval: Beyond Ranking of Pages
- 21.9 Directories and Categories

Measuring Retrieval Effectiveness [1/2]

- 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 = C / A
  - **recall** - what percentage of the documents relevant to the query were retrieved = C / B

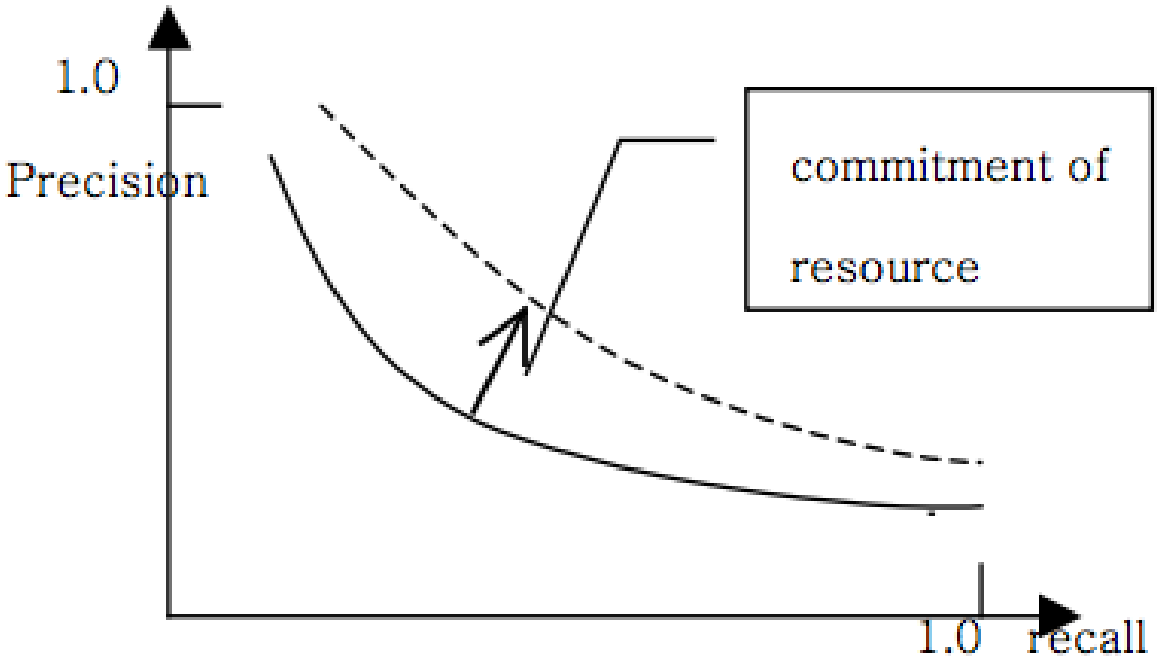


Measuring Retrieval Effectiveness [2/2]

- **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

참고자료

Recall vs Precision tradeoff



The F-measure

- a single measure to *combine* precision and recall
- *the harmonic mean* of precision and recall (F1-measure)

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

  - dominated by the smaller of the two

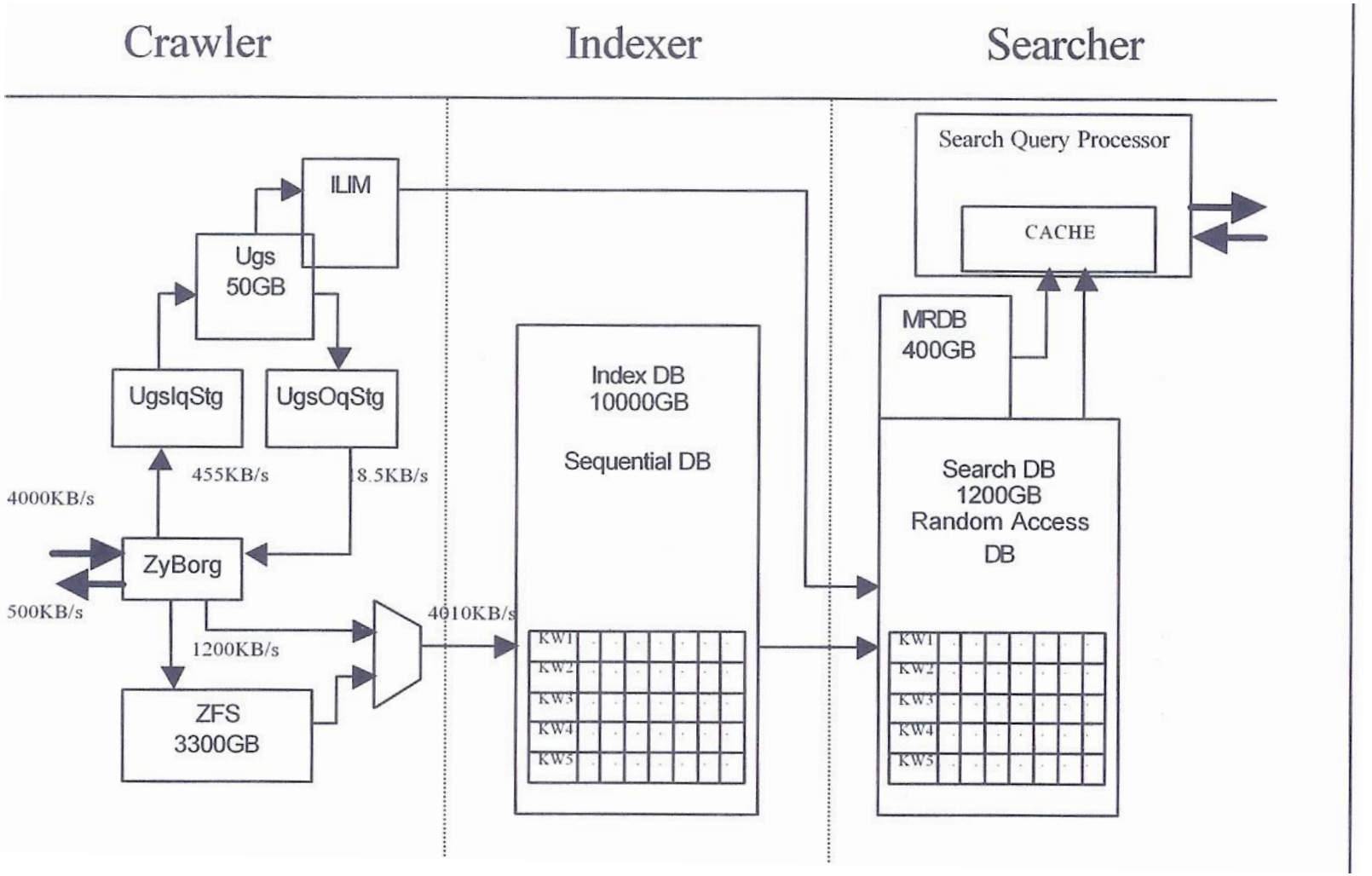
- General form $F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$

  - β times as much importance to recall as precision
  - van Rijsbergen (1979)
  - Commonly used values for β = 0.5, 1, 2

Chapter 21: Information Retrieval

- 21.1 Overview
- 21.2 Relevance Ranking Using Terms
- 21.3 Relevance Using Hyperlinks
- 21.4 Synonyms, Homonyms, and Ontologies
- 21.5 Indexing of Documents
- 21.6 Measuring Retrieval Effectiveness
- 21.7 Crawling and Indexing the Web
- 21.8 Information Retrieval: Beyond Ranking of Pages
- 21.9 Directories and Categories

Web Search Engine Architecture



NAVER: more than 170,000 servers

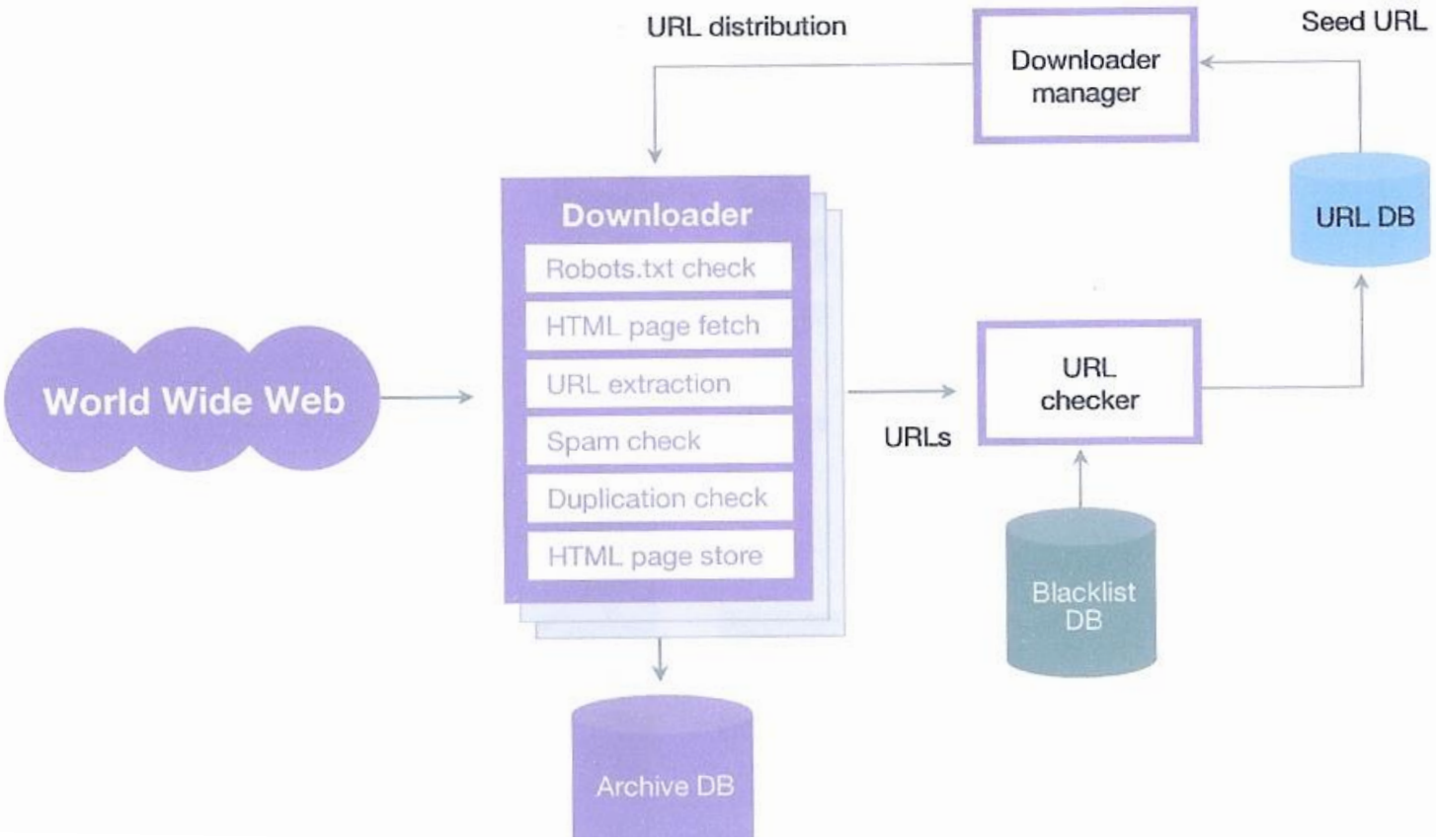
Google: more than 1000,000 servers

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 Crawler

Crawling



Web Crawling

- 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

Chapter 21: Information Retrieval

- 21.1 Overview
- 21.2 Relevance Ranking Using Terms
- 21.3 Relevance Using Hyperlinks
- 21.4 Synonyms, Homonyms, and Ontologies
- 21.5 Indexing of Documents
- 21.6 Measuring Retrieval Effectiveness
- 21.7 Crawling and Indexing the Web
- 21.8 Information Retrieval: Beyond Ranking of Pages
- 21.9 Directories and Categories

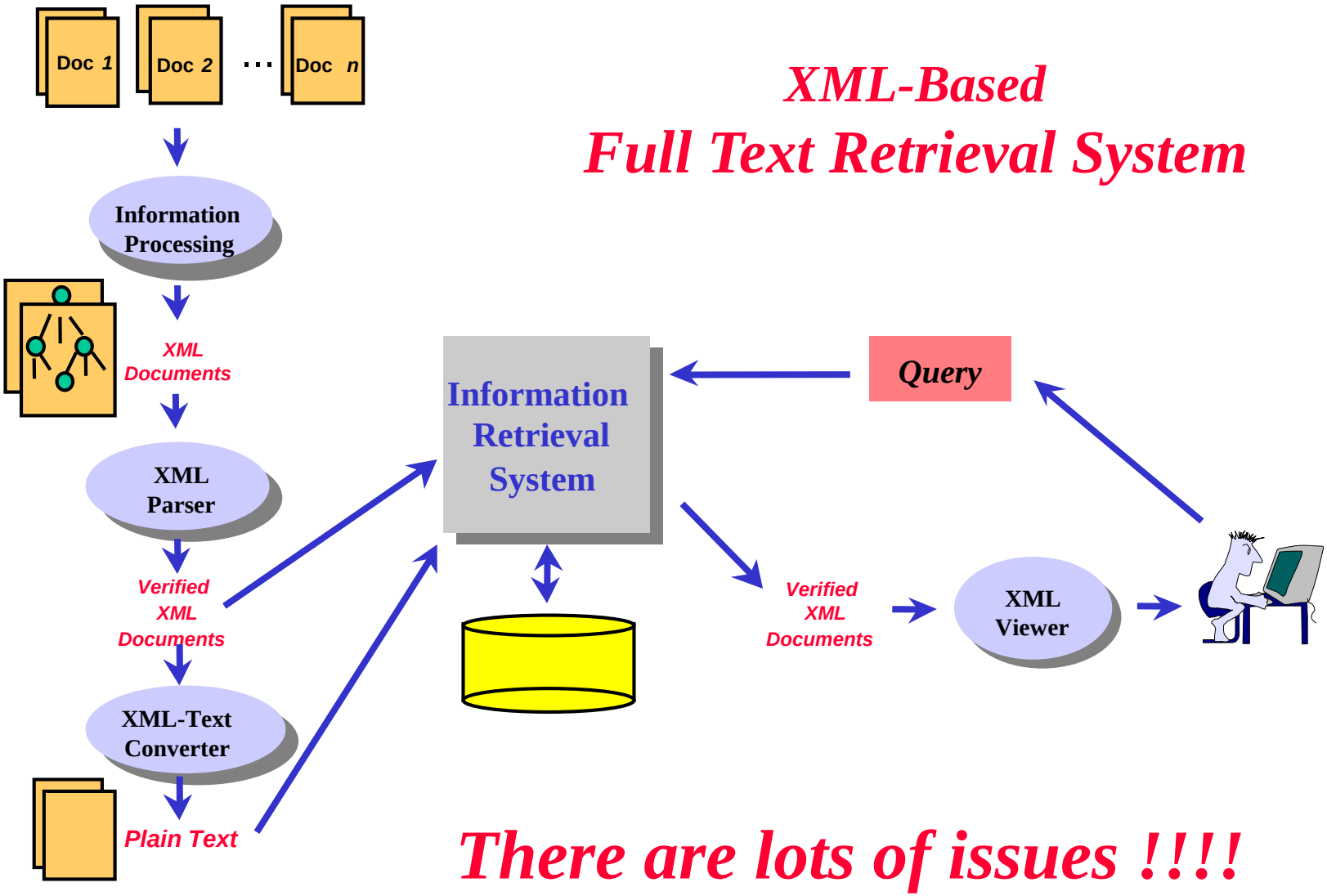
Information Retrieval and Structured Data

- 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**

Information Retrieval and Structured Data

- Querying Structured Data (Keyword search in relational data and XML data)
  - Keyword “Smith Queens” may be “Smith” in customer tuple or “Queens” in branch tuple
    - Don’t care schema / Don’t care SQL
  - Techniques using connections among keywords or assigning popularity to keywords

XML-Based Full Text Retrieval System



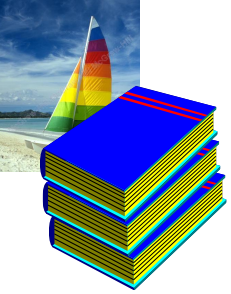
There are lots of issues !!!!

IR and Question Answering System

- Question answering in web search engine
  - Question ➔ “Who killed Lincoln?”

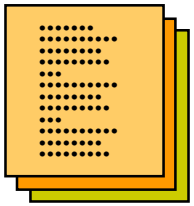
    Answer ➔ “Abraham Lincoln was shot by John Wilkes Booth in 1865”
  - Steps of QA system
    - Extract some keywords from the submitted question
    - Execute keyword searching against Web search engine
    - Parse the returned documents and generate the answer
      - A number of linguistic techniques and heuristics from AI Natural Language Processing

Passage-Based Full Text Retrieval System



Original Texts

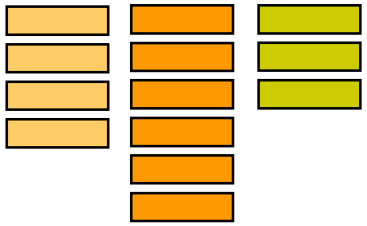
Digitization



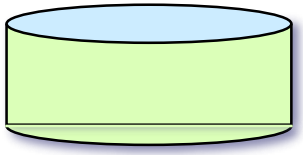
Digitized Full Texts

Passage Generation

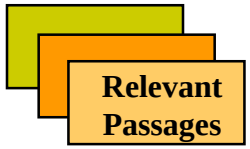
Generated Passages



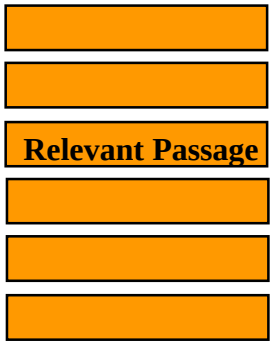
Information Retrieval System



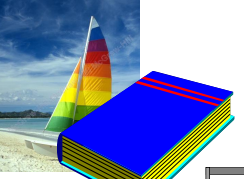
User Query



Relevant Passages

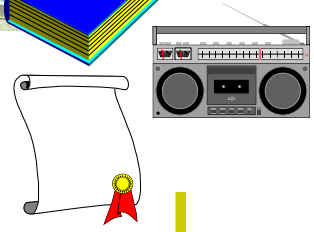


Relevant Passage



Original Information Items

Advanced Information Systems



Digitization

Text Summ.
OCR
Color Ext.
Feature Ext.
Voice Rec.

Text
SGML
Tiff
JPEG
MPEG
WAV
et al.

Manual Information Processing

Automatic Information Processing

Digitized Information

Secondary Information

Information Retrieval System

Query

Relevant Secondary Information

Relevant Digitized Information

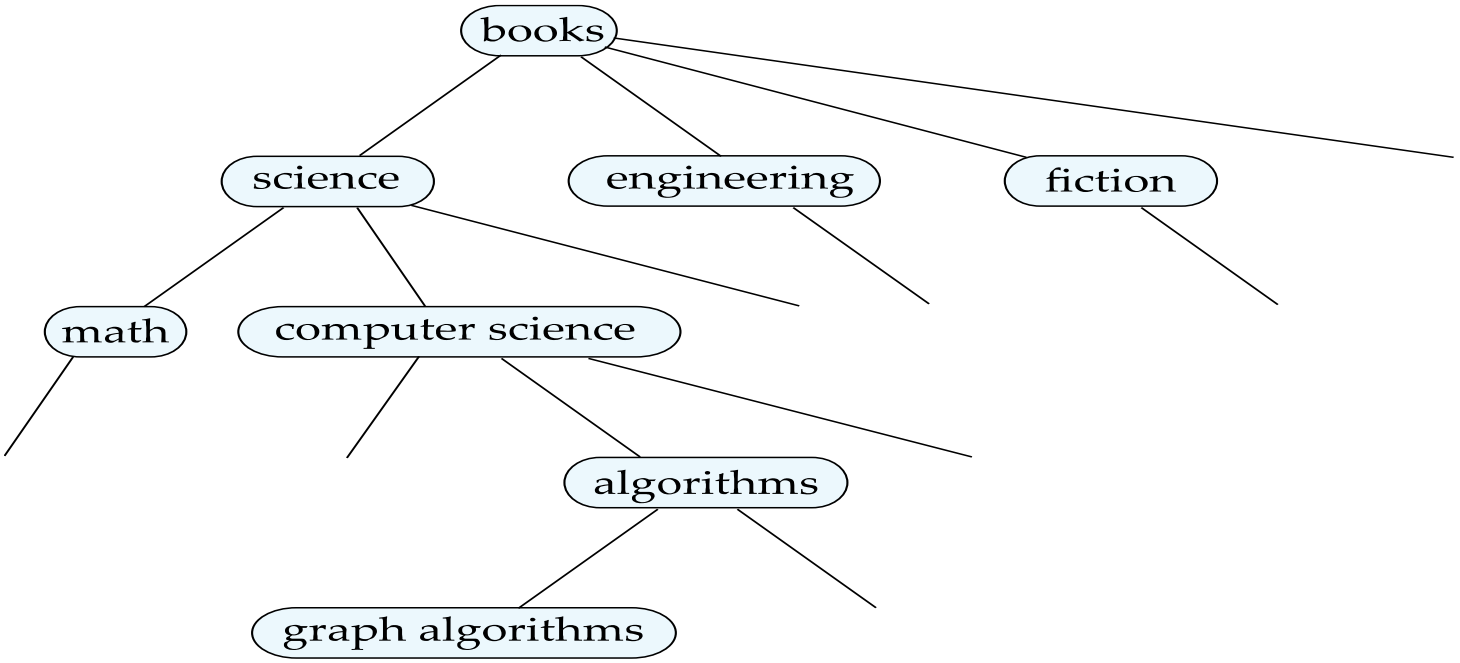
Chapter 21: Information Retrieval

- 21.1 Overview
- 21.2 Relevance Ranking Using Terms
- 21.3 Relevance Using Hyperlinks
- 21.4 Synonyms, Homonyms, and Ontologies
- 21.5 Indexing of Documents
- 21.6 Measuring Retrieval Effectiveness
- 21.7 Crawling and Indexing the Web
- 21.8 Information Retrieval: Beyond Ranking of Pages
- 21.9 Directories and Categories

Directory in IR System [1/2]

- 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**

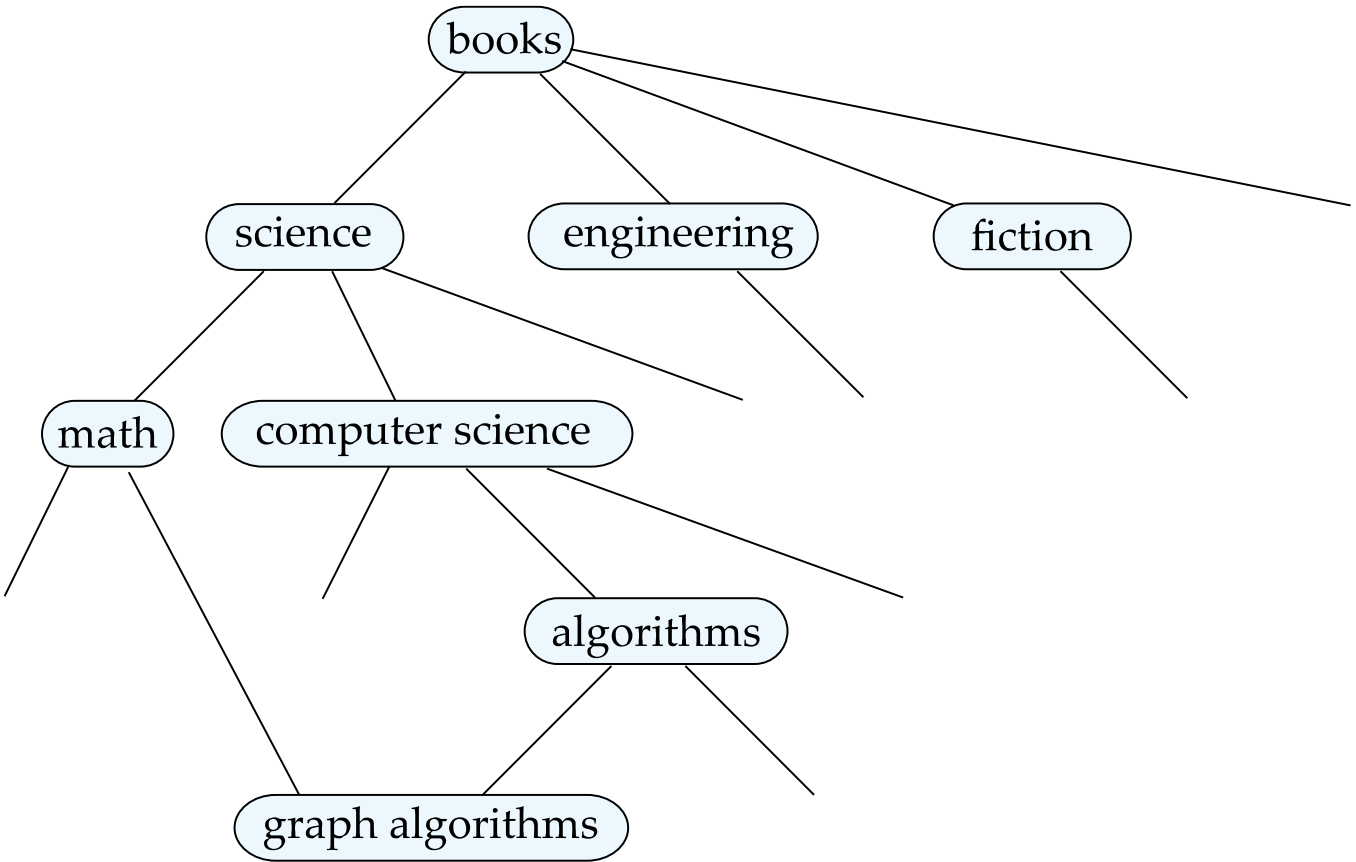
A Classification Hierarchy For A Library System



Directory in IR System [2/2]

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

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: Yahoo's Open Directory project
    - Classification of documents into a hierarchy may be done based on term similarity in an automatic tool

- Tagging vs. Directory

Web Crawling using Python

Dongjun Lee

Database System Concepts, 6th Ed.

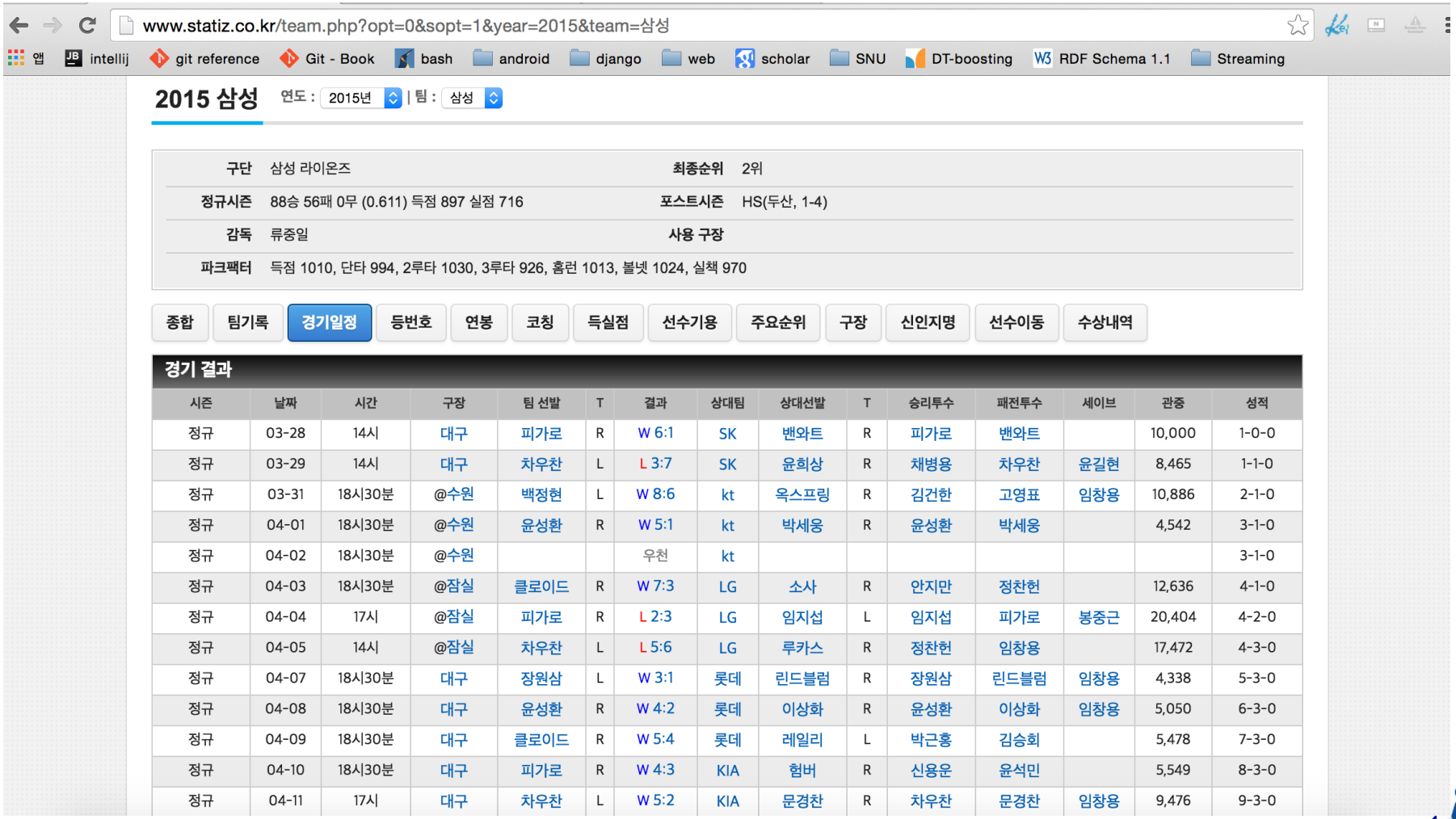
Library

- requests
  - Library for making a request and getting response from url
  - Reference
    - http://docs.python-requests.org/en/master/user/quickstart/

- BeautifulSoup
  - HTML and XML parser
  - Library for pulling data out of HTML and XML files
  - Reference
    - https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Example

- http://statiz.co.kr에서 년도별 각 팀의 경기결과 crawling
  - (예) http://www.statiz.co.kr/team.php?opt=0&sopt=1&year=2015&team=삼성



www.statiz.co.kr/team.php?opt=0&sopt=1&year=2015&team=삼성

앱 intellij git reference Git - Book bash android django web scholar SNU DT-boosting RDF Schema 1.1 Streaming

2015 삼성

연도 : 2015년 | 팀 : 삼성

구단	삼성 라이온즈	최종순위	2위
정규시즌	88승 56패 0무 (0.611) 득점 897 실점 716	포스트시즌	HS(두산, 1-4)
감독	류중일	사용 구장	
파크팩터	득점 1010, 단타 994, 2루타 1030, 3루타 926, 홈런 1013, 볼넷 1024, 실책 970		

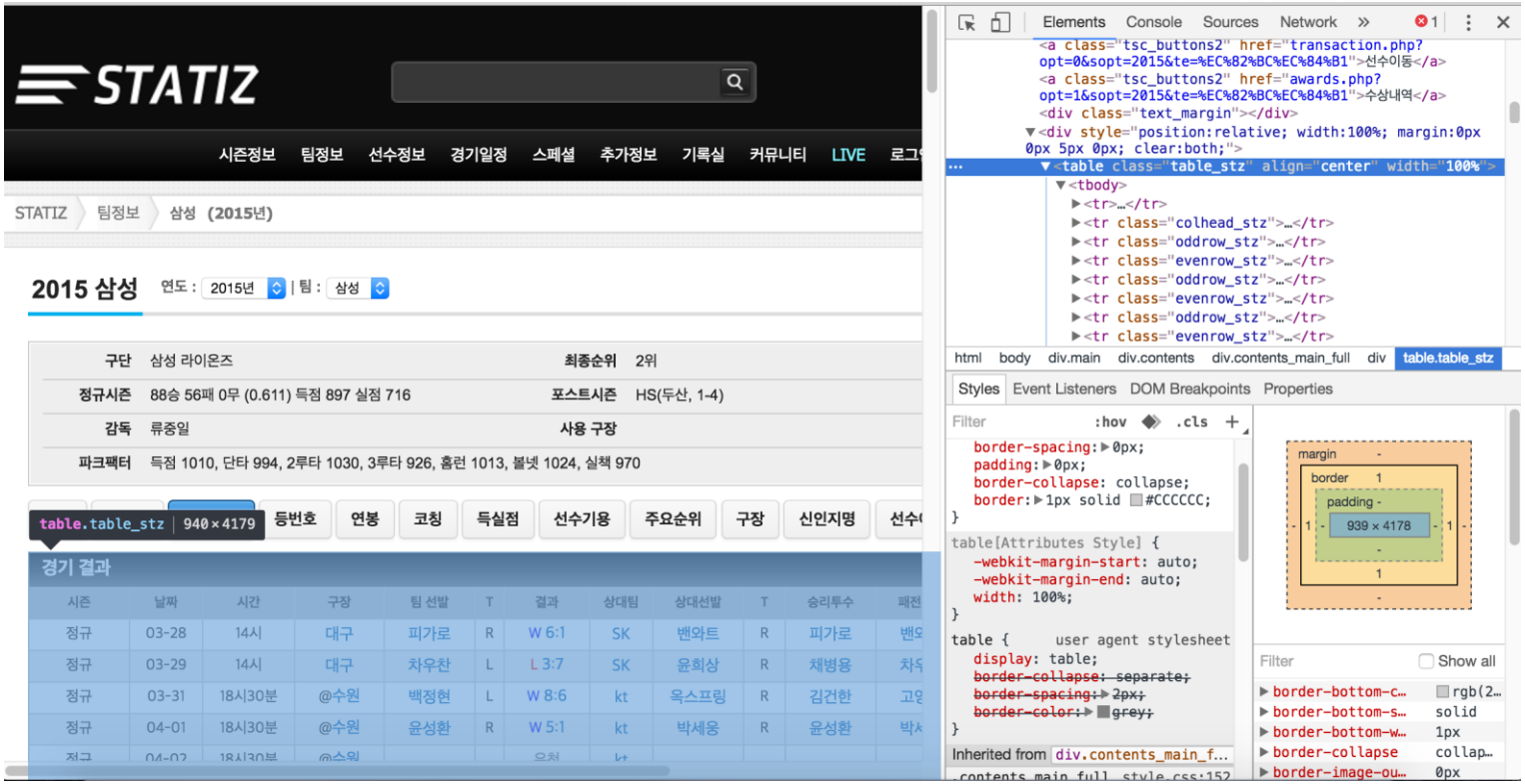
종합 | 팀기록 | 경기일정 | 등번호 | 연봉 | 코칭 | 득실점 | 선수기용 | 주요순위 | 구장 | 신인지명 | 선수이동 | 수상내역

경기 결과

시즌	날짜	시간	구장	팀 선발	T	결과	상대팀	상대선발	T	승리투수	패전투수	세이브	관중	성적
정규	03-28	14시	대구	피가로	R	W 6:1	SK	밴와트	R	피가로	밴와트		10,000	1-0-0
정규	03-29	14시	대구	차우찬	L	L 3:7	SK	윤희상	R	채병용	차우찬	윤길현	8,465	1-1-0
정규	03-31	18시30분	@수원	백정현	L	W 8:6	kt	옥스프링	R	김건한	고영표	임창용	10,886	2-1-0
정규	04-01	18시30분	@수원	윤성환	R	W 5:1	kt	박세웅	R	윤성환	박세웅		4,542	3-1-0
정규	04-02	18시30분	@수원			우천	kt							3-1-0
정규	04-03	18시30분	@잠실	클로이드	R	W 7:3	LG	소사	R	안지만	정찬헌		12,636	4-1-0
정규	04-04	17시	@잠실	피가로	R	L 2:3	LG	임지섭	L	임지섭	피가로	봉중근	20,404	4-2-0
정규	04-05	14시	@잠실	차우찬	L	L 5:6	LG	루카스	R	정찬헌	임창용		17,472	4-3-0
정규	04-07	18시30분	대구	장원삼	L	W 3:1	롯데	린드블럼	R	장원삼	린드블럼	임창용	4,338	5-3-0
정규	04-08	18시30분	대구	윤성환	R	W 4:2	롯데	이상화	R	윤성환	이상화	임창용	5,050	6-3-0
정규	04-09	18시30분	대구	클로이드	R	W 5:4	롯데	레일리	L	박근홍	김승회		5,478	7-3-0
정규	04-10	18시30분	대구	피가로	R	W 4:3	KIA	험버	R	신용운	윤석민		5,549	8-3-0
정규	04-11	17시	대구	차우찬	L	W 5:2	KIA	문경찬	R	차우찬	문경찬	임창용	9,476	9-3-0

Example

- 수집하려는 data는 “table_stz” 라는 이름의 table class
- table은 <tr></tr>의 row들로 구성됨
- 각 row는 <td></td>의 column들로 구성됨



Example

- python code

```python
import requests
from bs4 import BeautifulSoup


def match_crawl(team_name, year):
    data = []

    url = "http://www.statiz.co.kr/team.php?opt=0&sopt=1&year=" + str(year)
        + "&team=" + team_name
    result = requests.get(url)
    html = BeautifulSoup(result.text, 'html.parser')
    table = html.find('table', {'class': 'table_stz'})
    trs = table.find_all('tr')

    for i in range(2, len(trs)):
        row = []
        tr = trs[i]
        tds = tr.find_all('td')
        for td in tds:
            row.append(td.text)
        data.append(row)

    return data
```

Example

- python code 설명

```
8       url = "http://www.statiz.co.kr/team.php?opt=0&sopt=1&year=" + str(year)
9           + "&team=" + team_name
10
11      # requests.get method를 사용하여 url의 request 객체를 가져옴.
12      result = requests.get(url)
13      # result.text를 'html.parser'를 사용하여 parsing
14      html = BeautifulSoup(result.text, 'html.parser')
15      # html에서 class 이름이 'table_stz'인 table을 찾아서 table 변수에 저장
16      table = html.find('table', {'class': 'table_stz'})
17      # table의 모든 row(tr)을 변수 trs에 저장
18      trs = table.find_all('tr')
```