Media Content Management Systems

Magyar, Gábor
Szűcs, Gábor

Media Content Management Systems

írta Magyar, Gábor és Szűcs, Gábor

Publication date 2014

Tartalom

Media Content Management Systems 1
1. 1 BASICS 1
1.1. Data, information, knowledge 1
1.2. Data and information 1
1.3. Data management 1
1.4. Information management 2
1.5. Knowledge management 2
1.6. Media value chain 2
1.7. Multiscreen world 5
1.8. Participative media 8
1.9. digital literacy 8
2. 2 METADATA 8
2.1. Metadata, schemes and standards 8
2.2. Metadata 8
2.3. Basic types of metadata 9
2.4. Metadata (structure) Standards 9
2.5. Dublin Core Metadata Initiative (DCMI) 9
2.6. DC Metadata Element Set 9
2.7. Design goals for DCMES 9
2.8. DCMES: 15 elements 10
2.9. Commonly accepted: 10
2.10. Element name semantics 11
2.11. DC qualifiers 11
2.12. DCMI Terms 11
2.13. Dublin Core evaluation 11
2.14. Application profiles 12
2.15. "Levels of interoperability" 12
2.16. Learning Object Metadata (LOM) 12
2.17. IEEE-LOM 13
2.18. IEEE-LOM evaluation 14
2.19. Uses of IEEE-LOM 14
2.20. METS 14
2.21. METS structure 14
2.22. METS metadata 15
2.23. METS Descriptive Metadata 15
2.24. METS Administrative Metadata 16
2.25. METS File Inventory 17
2.26. METS Structural Map 17
2.27. MODS 18
2.28. MODS 19
2.29. MODS evaluation 19
2.30. MODS User Guidelines 20
2.31. element and attribute 20
2.32. element and sub element 20
2.33. MODS Elements 21
2.34. EAD: Encoded Archival Description 21
2.35. EAD features 21
2.36. MPEG standards 21
2.37. MPEG-7 21
2.38. Application model 22
2.39. main elements of MPEG-7 standard 22
2.40. MPEG-7 DDL 22
2.41. MPEG-7 Visual 23
2.42. MPEG-7 Visual examples 23
2.43. MPEG-7 Audio 23
2.44. MPEG-7 and MPEG-21 community 24

3. 3 SEMANTIC CONTENT MANAGEMENT 24
3.1. Semantic content management/1 24
3.2. RDF 26
3.3. Why we need RDF? 26
3.4. URI 26
3.5. RDF 27
3.6. Subject, Predicate, Object 27
3.7. statement 27
3.8. abbreviations 27
3.9. expressive strength 28
3.10. Common concepts 29
3.11. Common concepts 29
3.12. RDF vocabulary 30
3.13. Semantic Graph Data Model 31
3.14. RDF Schema 32
3.15. RDF Schema 32
3.16. Classes 32
3.17. Defining Classes 33
3.18. Defining Instances 33
3.19. rdf:type 33
3.20. Defining Classes (cont'd) 34
3.21. Notation 34
3.22. Defining Subclasses 34
3.23. Classes and Instances 34
3.24. Properties of rdfs:subClassOf 35
3.25. RDF Schema Predefined Classes 35
3.26. Properties 35
3.27. Defining Properties 35
3.28. Defining Properties 36
3.29. Domain and Range 36
3.30. Datatypes for Ranges 37
3.31. The Property rdfs:label 37
3.32. The Property rdfs:comment 37
3.33. The Property rdfs:seeAlso 37
3.34. RDFS vs. Types (cont'd) 38
3.35. Schema Languages 38
3.36. The Semantic Web "Layer Cake" 38
3.37. URI, Unicode layer 39
3.38. XML layer 39
3.39. RDF layer 40
3.40. RDF Scheme 40
3.41. Ontology layer 41
3.42. Controlled vocabularies 41
3.43. Thesaurus 42
3.44. Thesaurus standards 42
3.45. Widely known thesauri 43
4. 4 CMS types 44
4.1. CMS types 44
4.2. DAM: Digital Asset Management 44
4.3. Broad categories of DAM 44
4.4. Subtypes of DAM 44
4.5. DAM file types 44
4.6. Application (areas) of DAM 45
4.7. DM: Document Management 45
4.8. Components of DM 46
4.9. Capture 46
4.10. Functionalities in DM 46
4.11. DM file types 46
4.12. Application (areas) of DM 47
4.13. LMS: Learning Management System 47

4.14. Functionalities in LMS 47
4.15. LMS standards 48
4.16. Most important LMS softwares 48
4.17. Web CMS 48
4.18. Functionalities in WCMS 48
4.19. Most important WCMS softwares 49
4.20. WCMS file types 49
4.21. KM: Knowledge Management 49
4.22. Knowledge Management software 50
4.23. ECM: Enterprise Content Management 50
4.24. Components of ECM 50
4.25. Capture component 51
4.26. Manage component 51
4.27. Store component 51
4.28. Preserve component 52
4.29. Deliver component 52
4.30. Output and distribution media 52
4.31. Most important ECM systems 52
4.32. Axes of Magic Quadrant 53
4.33. CMS plan, design, integration 54
4.34. Plan of functionalities (1) 54
4.35. Plan of functionalities (2) 55
4.36. Plan of functionalities (3) 55
4.37. Analyzing the content lifecycle 55
4.38. Roles 56
4.39. CMS adopting, integration 56
4.40. Content Migration 56
4.41. Two approaches 56
5. 5 Information Retrieval (IR) 56
5.1. Text Databases and IR 56
5.2. Information Retrieval (IR) 57
5.3. IR vs DBMS 57
5.4. Basic Measures in the IR 57
5.5. Keyword-Based Retrieval 58
5.6. Similarity-Based Retrieval in Text Databases 58
5.7. Retrieval : Ad hoc and Filtering 58
5.8. architecture 59
5.9. IR System and Tasks Involved 59
5.10. Models for IR - Taxonomy 59
5.11. Formal Specification of the Task 60
5.12. Formal Specific. of the Task (Cont.) 60
5.13. Boolean Retrieval Model - Queries 60
5.14. Boolean Retr. Model - Definition 61
5.15. Boolean Retrieval Model 61
5.16. Vector Model - Definition 61
5.17. Vector Model - Similarity 61
5.18. Vector Model 62
5.19. Weights 62
5.20. TF-IDF ranking 62
5.21. Classic Probabilistic Model (first) 63
5.22. Classic Probabilistic Model 63
5.23. Classic Probabilistic Model 63
5.24. Classic Probabilistic Model 64
5.25. Classic Probabilistic Model 64
5.26. Classic Probabilistic Model 65
5.27. Classic Probabilistic Model 65
5.28. Estimation of Term Relevance 65
5.29. (Dis) advantages of Classic Probabilistic model (last) 66
5.30. Summary: taxonomy of IR models 67
5.31. Alternative Probabilistic Models 67

5.32. Bayesian Network 68
5.33. Belief Network Model (1/6) 68
5.34. Belief Network Model (2/6) 68
5.35. Belief Network Model (3/6) 69
5.36. Belief Network Model (4/6) 69
5.37. Belief Network Model (5/6) 69
5.38. Belief Network Model (6/6) 70
5.39. Summary 70
5.40. Alternative Set Theoretic models 70
5.41. Extended Boolean logic 71
5.42. Extended Boolean Model: 71
5.43. Extended Boolean Model: 71
5.44. Extend the idea to m terms 72
5.45. Properties: 72
5.46. Example: 72
6. 6 Information Retrieval on Web 73
6.1. Mining the World-Wide Web 73
6.2. Web search engines 73
6.3. Web Mining: A more challenging task 74
6.4. Web Mining Taxonomy 74
6.5. Mining the World-Wide Web 74
6.6. Mining the World-Wide Web 75
6.7. Mining the World-Wide Web 75
6.8. Mining the World-Wide Web 76
6.9. Mining the World-Wide Web 76
6.10. Mining the World-Wide Web 77
6.11. Multilayered Web Information Base 78
6.12. Multiple Layered Web Architecture 78
6.13. Mining the World-Wide Web 78
6.14. Mining the World-Wide Web 78
6.15. Benefits of Multi-Layer Meta-Web 79
6.16. Web Search Topics 79
6.17. Search on the Web 79
6.18. Classic IR vs. Web Search 80
6.19. General Web Search Engine Architecture 80
6.20. The Indexer Module 81
6.21. Storage Issues 81
6.22. Update Strategies 81
6.23. Batch vs. Steady 82
6.24. Partial vs. Complete Crawls 82
6.25. 1^{st} Generation Web Search Engines 82
6.26. The evolution of search engines 83
6.27. 2^{nd} generation web IR ranking criteria 83
6.28. Considering the link structure 83
6.29. Considering the link structure 83
6.30. Simple PageRank - Definition 84
6.31. Simple PageRank - Example 84
6.32. Simple PageRank - Calculation 84
6.33. Problems if graph is not strongly connected 85
6.34. What happens with Leaks? 85
6.35. What happens with Sinks? 85
6.36. Intuitive Interpretation 85
6.37. PageRank - conclusion 86
6.38. PageRank in the 1^{st} Google engine 86
6.39. HITS: (Hyperlink-Induced Topic Search) 86
6.40. HITS 86
6.41. Systems Based on HITS 87
6.42. HITS: Identifying The Focused Subgraph 87
6.43. HITS: Link Analysis 87
6.44. HITS:Link Analysis Computation 88

6.45. Research issues 88
6.46. Summary of previous lessons: taxonomy of IR models 88
6.47. Search Engine Marketing 89
6.48. PPC and SEO 90
6.49. SEO: Search Engine Optimization 90
6.50. PPC vs. "Natural" SEO 90
6.51. What Are Searchers Looking For? 90
6.52. KEI values at different search engines 91
6.53. SEO techniques: White Hat vs. Black Hat 91
6.54. Black Hat (Spamming) 92
6.55. White Hat: The Seven Steps to Higher Rankings 92
6.56. 1 Get Your Site Fully Indexed 93
6.57. 2 Get Your Pages Visible 93
6.58. Dynamic Site Leaves Session IDs in URLs: 93
6.59. Optimize the use of Images 94
6.60. 3 Build Links and PageRank 95
6.61. Ideal internal site linking hierarchies: 95
6.62. 4 Leverage Your PageRank 95
6.63. Avoid PageRank dilution 95
6.64. Write better anchor text 95
6.65. 5) Encourage Clickthrough 95
6.66. 6) Track the Right Metrics 96
6.67. 7) Avoid Worst Practices 96
6.68. Not Spam, But Bad for Rankings 96
7. 7 Search improvement 96
7.1. Search improvement 96
7.2. Metasearch (1) 97
7.3. Metasearch (2) 97
7.4. Merging algorithms: 97
7.5. Re-ranking method based on inter-document distances 97
7.6. Introduce a function g 98
7.7. Goal of the estimation 98
7.8. Relationships 98
7.9. Changing matrix 98
7.10. Document Weight Improving (DWI) 99
7.11. Example 99
7.12. Original relevance values 99
7.13. Calculations 100
7.14. Final results 101
7.15. Multimedia retrieval 101
7.16. Multimedia retrieval: Description vs. Content-based retrieval 101
7.17. Queries in Content-Based Retrieval Systems 102
7.18. Approaches Based on Image Signature 102
7.19. Mining Multimedia Databases 102
7.20. Multidimensional Analysis of Multimedia Data 102
7.21. Mining Associations in Multimedia Data 102
7.22. Mining Multimedia Databases 103
7.23. Mining Multimedia Databases 103
7.24. Preludes and application areas of CBIR 103
7.25. Content-based Image Retrieval (CBIR) 103
7.26. Specification of picture attributes 104
7.27. Comparison of images 104
7.28. Feature extraction for images 106
7.29. Shape Based methods for images 106
7.30. Scale-invariant feature transform (SIFT) 107
7.31. Matching by SIFT 107
7.32. Optical Character Recognition (OCR) 108
7.33. Importance of context 108
7.34. Role of the context 109
7.35. Text retrieval vs. Image retrieval 109

7.36. Experiment of concept search in Text retrieval 109
7.37. Trend of CBIR 110
7.38. New in CBIR: context 110
7.39. Video Retrieval 112
7.40. Multimedia Information Retrieval 113
7.41. Dimension reduction 113
7.42. Information Gain (IG) for Text Mining 113
7.43. X^2 Chi-square distribution for Text Mining 114
7.44. Principal components analysis 114
7.45. SVD 114
7.46. Latent Semantic Indexing for Text Mining 114
7.47. Latent Semantic Indexing (LSI) 115
7.48. LSI 115
7.49. Text Index Building 115
7.50. Bag of words 115
7.51. Query 115
7.52. Term-document incidence 116
7.53. Incidence vectors 116
7.54. Answers to query 116
7.55. Bigger corpora 116
7.56. Can't build the matrix 116
7.57. Inverted index 116
7.58. Inverted index 117
7.59. Inverted index construction 118
7.60. Indexer steps 118
7.61. The index we just built 118
7.62. Query processing 118
7.63. The merge 119
7.64. Boolean queries: Exact match 119
7.65. More general merges 119
7.66. Merging 119
7.67. Query optimization 120
7.68. Query optimization example 120
7.69. More general optimization 120
7.70. Exercise 120
7.71. Query processing exercises 121
7.72. Skip pointers for faster postings 121
7.73. Recall basic merge 121
7.74. Augment postings with skip pointers (at indexing time) 121
7.75. Query processing with skip pointers 121
7.76. Where do we place skips? 122
7.77. Placing skips 122
7.78. Problems and questions about term search 122
7.79. Detour: problems and questions 122
7.80. Beyond term search 122
7.81. Evidence accumulation 123
7.82. Ranking search results 123
7.83. Structured vs unstructured data 123
7.84. Unstructured data 123
7.85. Semi-structured data 123
7.86. More sophisticated semi-structured search 124
7.87. Tokenization 124
7.88. Tokenization 124
7.89. Parsing a document 124
7.90. Format/language stripping 124
7.91. Tokenization 125
7.92. Language issues 125
7.93. Tokenization: language issues 125
7.94. Normalization 125
7.95. Punctuation 126

7.96. Numbers 126
7.97. Case folding 126
7.98. Lemmatization 126
7.99. Stemming 127
7.100. Stemming 127
7.101. Types of stemmers 127
7.102. Mistakes of stemming 127
7.103. Measuring the stemmers 127
7.104. Evaluation of stemmers 128
7.105. Porter's algorithm 128
7.106. Phases and typical rules in Porter 128
7.107. Examples for Porter algorithm 128
7.108. Snowball 129
7.109. Other stemmers 129
7.110. Exercise for stemming 129
7.111. Exercise for stemming 130
7.112. Dictionary entries - first cut 130
7.113. Language detection 130
7.114. Language detection using n -grams 130
7.115. N -grams 130
7.116. Start: Zipf's law 131
7.117. Zipf's law: Reuters-21578 131
7.118. Zipf's law: web 2.2 (Hungarian) 131
7.119. Method 131
7.120. Creating profiles 131
7.121. Observations 132
7.122. Comparing profiles 132
7.123. Example 132
7.124. Flowchart 132
7.125. Results 132
7.126. Phrase queries, Positional indexes 132
7.127. Phrase queries 133
7.128. A first attempt: Biword indexes 133
7.129. Longer phrase queries 133
7.130. Extended biwords 133
7.131. Query processing 133
7.132. Other issues 134
7.133. Positional indexes 134
7.134. Positional index example 134
7.135. Processing a phrase query 134
7.136. Rules of thumb 134
8. 8 Exercise 135
8.1. General information 135
8.2. 135
8.3. 135
8.4. Exercise 135
8.5. General information 135
8.6. Task 135
8.7. Exercise 136
8.8. General information 136
8.9. 136
8.10. Exercise 136
8.11. General information 136
8.12. Task 1 137
8.13. Comparison of two stemmers 137
8.14. Task 2 137

Media Content Management Systems

1. 1 BASICS

1.1. Data, information, knowledge

- Data: facts of the World, mirrors reality(without meaning).Data on its own carries no meaning.
- Information: interpreted data(for data to become information, it must be interpreted and take on a meaning)
- Knowledge: information in context(by human being; or artificial intelligence?)

- Data
- + interpretation
- Information
- + context
- Knowledge

(â€žfactsâ€ť)

1.2. Data and information

- 4 dimensions:you should know the subject/phenomenon(the data refers to)in general and in concrete;+ attribute(s) of subject/phenomenon,in general and in concrete;
- Example: "X car type achieved in NCAP safety test *****",you should know in general what a car is, in concrete X type;in general what is car safety, in concrete NCAP test

1.3. Data management

Storage, query, manipulation of raw data.

[data model, DB query, data manipulation languages, etc.]

1.4. Information management

interpretation,

systematization,

evaluation,

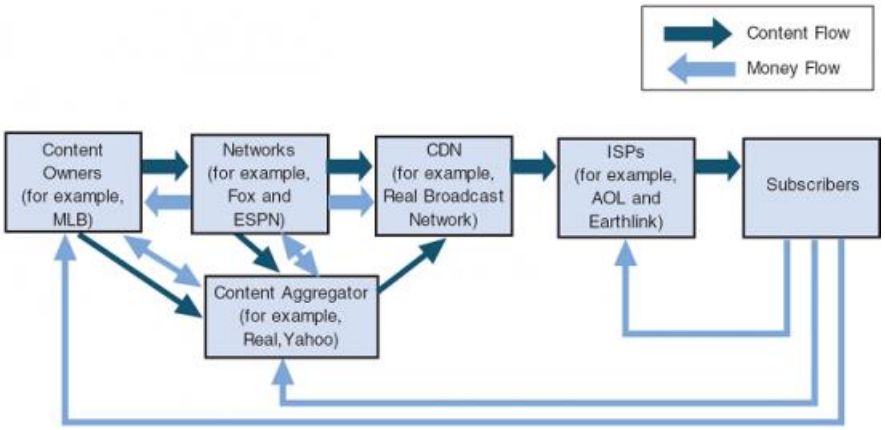
retrieval

of information

1.5. Knowledge management

- Knowledge management: efforts, measures in order to increase corporate knowledge capital(Knowledge capital is part of the corporate assets.)

1.6. Media value chain



DE! 2012 NJSZT

1.7. Multiscreen world

81%

66%

Source: Google: The New Multiscreen World, 2012

1.8. Participative media

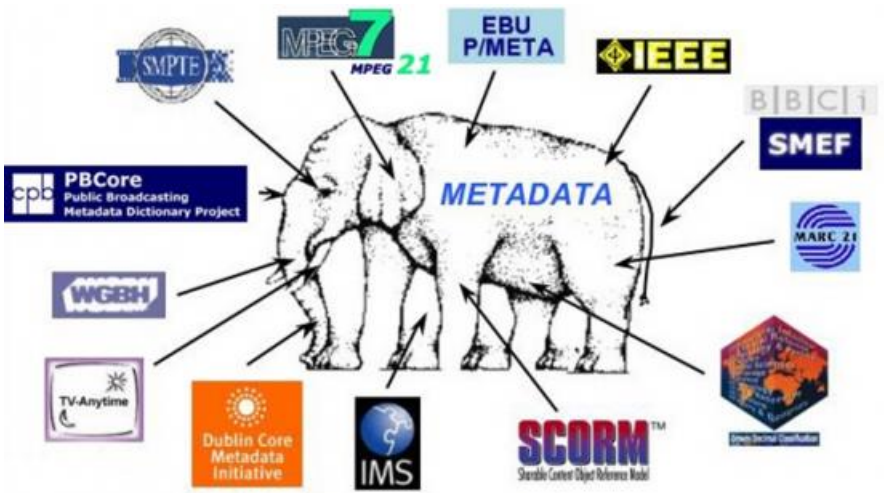
- What do we need to participate?
  - access
  - content
  - digital literacy

1.9. digital literacy

- multi-tasking
- preferred: image, graphics, video
- linearity vs. navigation
- "copy-paste"
- real-time communcation
- personality (blog, twit)
- "shorter user-concentration"

2. 2 METADATA

2.1. Metadata, schemes and standards



2.2. Metadata

Metadata: data from data

(all data, about other data, e.g. catalogue data)

"Metadata articulates a context for objects of interest - "resources" such as MP3 files, library books, or satellite images - in the form of "resource descriptions". As a tradition, resource description dates back to the earliest archives and library catalogs. The modern "metadata" field that gave rise to Dublin Core and other recent standards emerged with the Web revolution of the mid-1990s."

(source: Dublin Core Metadata Initiative)

2.3. Basic types of metadata

- descriptive metadata: describes a resource for purposes such as discovery and identification. It can include elements such as title, abstract, author, keywords, date of last modification, etc.
- structural metadata: indicates how compound objects are put together, for example, how pages are ordered to form chapters
- administrative metadata provides information to help manage a resource, such as when and how it was created, file type and other technical information, and who can access it. There are several subsets of administrative data; two that are sometimes listed as separate metadata types are

Source: NISO (2004) Understanding Metadata

2.4. Metadata (structure) Standards

- Dublin Core (http://dublincore.org)
- IEEE-LOM (http://ltsc.ieee.org/wg12/)
- MODS (www.loc.gov/standards/mods/)
- EAD (http://www.loc.gov/ead/)
- MPEG-7 (http://www.mpeg.org/)

2.5. Dublin Core Metadata Initiative (DCMI)

- fostering the widespread adoption of interoperable metadata standards, development of specialized metadata vocabularies (to enable intelligent resource discovery systems -> make it easier to find resources in the Internet)

2.6. DC Metadata Element Set

- 15 descriptive semantic definitions
  - a core set of elements: could be shared across disciplines, any type of organization (to organize and classify information)
- version 1.1:
  - Internet RFC 2413 (1998)
  - NISO Standard Z39.85-2001 (2001)
  - ISO Standard 15836-2003 (2003)

2.7. Design goals for DCMES

- Simplicity of creation and maintenance
  - small and simple metadata element set: non-specialists can create descriptive records (for effective retrieval of information resources)
- Commonly understood semantics
  - the semantics of DCMES are universally understood
- International scope
  - standard to the multilingual and multicultural information universe.
- Extensibility
  - mechanisms for extending the DC element set for additional resource discovery needs

2.8. DCMES: 15 elements

- Title (Title):
- Creator (Creator)
- Subject (Subject and Keywords)
- Description (Description):
- Publisher (Publisher)
- Contributor (Contributor)
- Date (Date)
- Type (Resource Type)
- Format (Format)
- Identifier (Resource Identifier)
- Source (Source)
- Language (Language)
- Relation (Relation)
- Coverage (Coverage)
- Rights (Rights Management)

All elements are optional and repeatable

2.9. Commonly accepted:

- Elements and Semantics
  - Definitions for the content of the elements (e.g., what does it mean: 'a title', 'a creator', etc.)
- Content Rules

  - guidelines for inputting the element (what to capitalize, order of elements, etc.)
- Syntax
  - rules for structuring and expressing the elements for machine processing

2.10. Element name semantics

- Element Name: Title
  - Label: Title
  - Semantics: a name given to the resource.
  - Comment: typically, Title will be a name by which the resource is formally known.
- Element Name: Creator
  - Label: Creator
  - Semantics: An entity primarily responsible for making the content of the resource.
  - Comment: examples of Creator include a person, an organization, or a service. Typically, the name of a Creator should be used to indicate the entity

2.11. DC qualifiers

- to extend and refine the 15 DCMES elements
  - Element Refinement - these qualifiers make the meaning of an element narrower or more specific ("more restricted scope")
  - Encoding Scheme - these qualifiers identify schemes that aid in the interpretation of an element value (controlled vocabularies, formal notations, parsing rules)

2.12. DCMI Terms

- Authoritative specification of all metadata terms related to DC, including elements, element refinements, encoding schemes, vocabulary terms
- Maintained by the DC Usage Board
- Contained in the DCMI Metadata Registry

DC Element	Element Refinements	Element Encoding Schemes
Date	DateCreated Valid Available Issued Modified Date Copyrighted Date Submitted	DCMI Period W3C-DTF

2.13. Dublin Core evaluation

- International and cross-domain
- Increased efficiency of the discovery/retrieval of digital objects
- qualified DC element set aids the management of information
- easy mapping to other metadata standards

2.14. Application profiles

- Consist of data elements drawn from one or more namespace schemas combined together by implementors and optimised for a particular local application.
- Application profiles are useful as they allow the implementor to declare how they are using standard schemas
- Characteristics:
  - May draw on one or more existing namespaces
  - Introduce no new data elements
  - May specify permitted schemes and values
  - Can refine standard definitions
- Application profiles enable implementors "to share information about their schemas in order to inter-work with wider groupings...Communities can start to align practice and develop common approaches by sharing their application profiles."

2.15. "Levels of interoperability"

- Level 1: shared term definitions (interoperability among metadata-using applications is based on shared natural-language definitions) Terms are "hard-wired" into applications.
- Level 2: formal semantic interoperability (interoperability among metadata-using applications is based on the shared formal model provided by RDF)
- Level 3: description set syntactic interoperability (applications are compatible with the Linked Data model and share an abstract syntax for validatable metadata records, the "description set")
- Level 4: description set profile interoperability (records exchanged among metadata-using applications follow a common set of constraints, use the same vocabularies, and reflect a shared model of the world.

2.16. Learning Object Metadata (LOM)

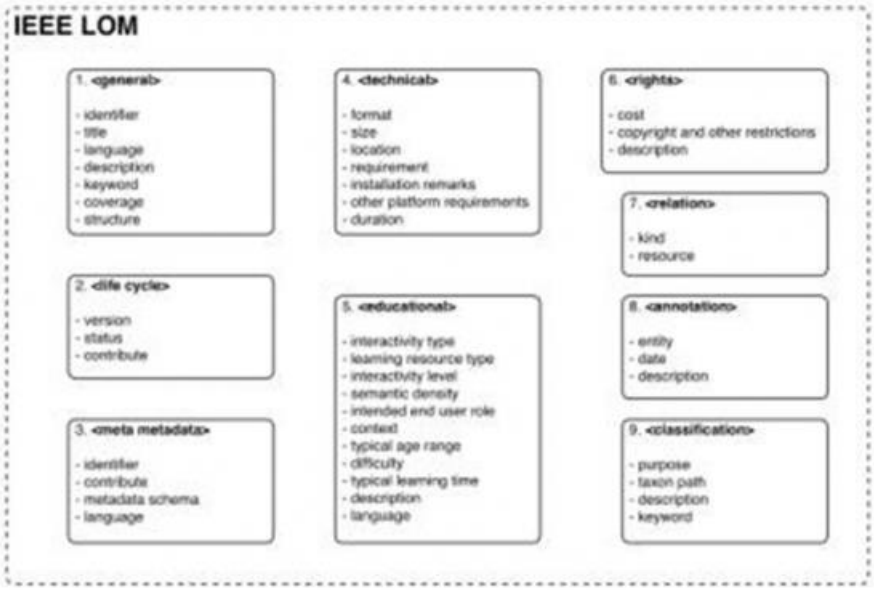
An array of related standards for description of 'learning objects' or 'learning resources'

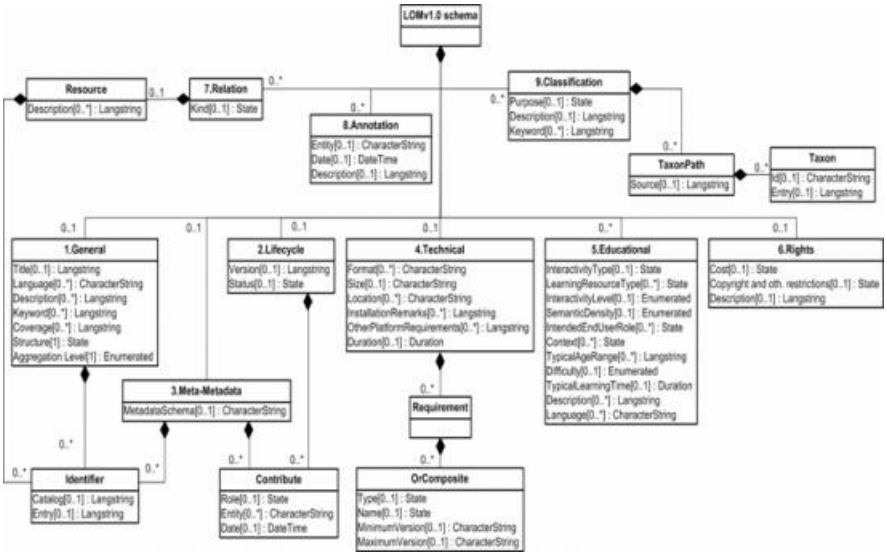
- IEEE LTSC (Institute of Electrical and Electronics Engineers Learning Technology Standards Committee)
- IMS Global Learning Consortium
- Dublin Core Metadata Initiative
  - http://www.dublincore.org
- DCMI Schemas (XML and RDF)
  - http://dublincore.org/schemas/

- ANSI/NISO Z39.85-2001: The Dublin Core Metadata Element Set
  - http://www.niso.org/standards/resources/Z39-85.pdf

2.17. IEEE-LOM

- top level categories
  - General
  - Life Cycle
  - Meta-Metadata
  - Technical
  - Educational
  - Rights
  - Relation
  - Annotation
  - Classification





2.18. IEEE-LOM evaluation

- Built on an explicit data model
  - Specifies which aspects of a learning object should be described
- International community contributes to standard
  - Education and learning sector in Europe is particularly invested in using the standard; is required in certain circumstances
- Applicable to a broad array of learning objects

2.19. Uses of IEEE-LOM

- Describe and share information about learning objects individually or as a group
- Export as LOM in XML or RDF
- Most descriptive elements mapped to Dublin Core

2.20. METS

- Metadata Encoding and Transmission Standard (METS)a digital library standard for encoding descriptive, administrative, and structural metadata

2.21. METS structure

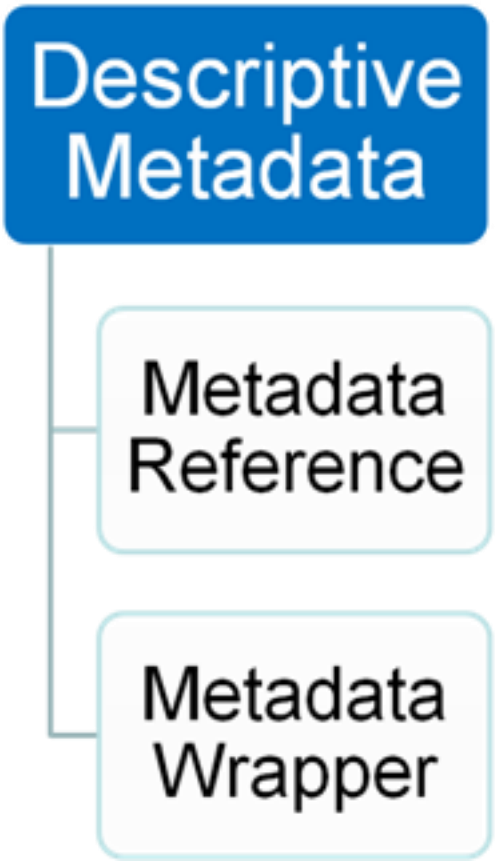
- METS Header: Contains metadata describing the METS document itself
- Descriptive Metadata: May point to descriptive metadata external to the METS document or contain internally embedded descriptive metadata, or both.
- Administrative Metadata: Provides information regarding how the files were created and stored, intellectual property rights, etc.
- File Section: Lists all files containing content which comprise the electronic versions of the digital object.Â

- Structural Map: Outlines a hierarchical structure for the digital library object, and links the elements of that structure to content files and metadata that pertain to each element.
- Structural Links: Records the existence of hyperlinks between nodes in the hierarchy outlined in the Structural Map.
- Behavior: A behavior section can be used to associate executable behaviors with content in the METS object

2.22. METS metadata

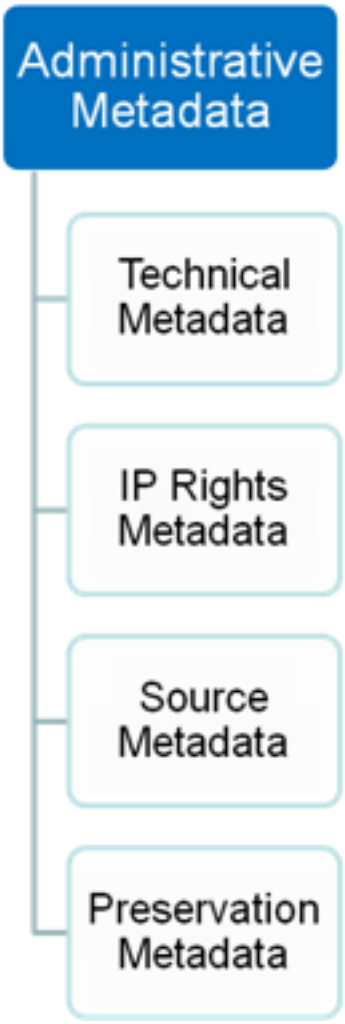
- METS
- Header
- Administrative
- metadata
- File
- Inventory
- Structure
- map
- Descriptive
- metadata
- Behavioral
- metadata
- optional
- optional
- optional
- required
- optional
- optional

2.23. METS Descriptive Metadata



- Metadata Reference: link to external descriptive metadata. type of link: included as an attribute
- Metadata Wrapper: Included descriptive metadata, binary data or arbitrary XML using namespace metadata type: specified as an attribute.

2.24. METS Administrative Metadata

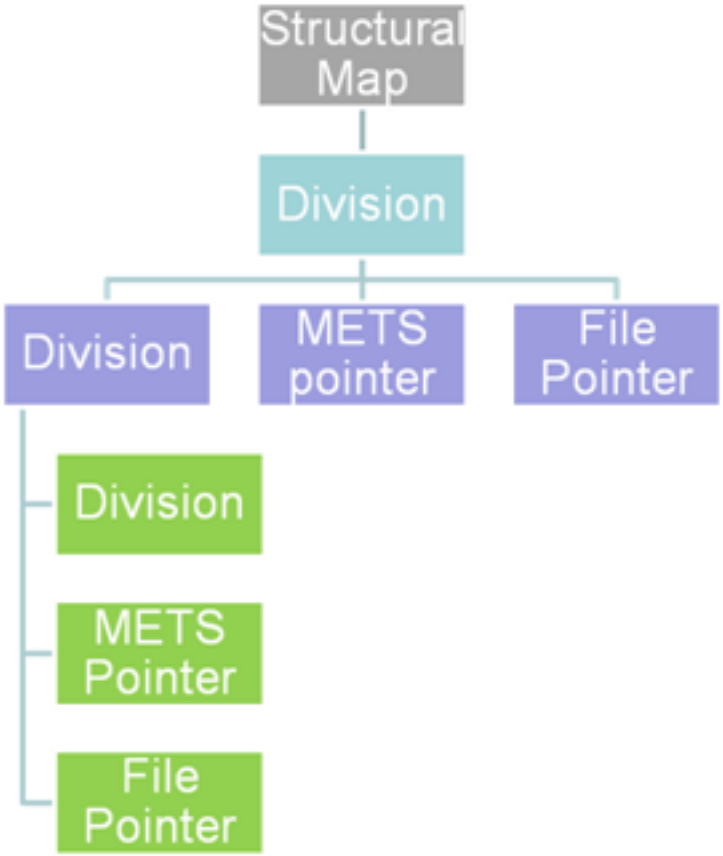


- Technical Metadata: technical metadata regarding content files
- IP Rights Metadata: rights metadata regarding content files or primary source material
- Source Metadata: provenance information for content files.
- Preservation Metadata: metadata to assist in preservation of digital content

2.25. METS File Inventory

- File Group (fileGrp): hierarchically subdivids physical files (e.g. by type)
- File : pointer to an external file (or includes file content internally)

2.26. METS Structural Map



provides a tree structure describing the original document

each division element is a node of the tree (identifies content files associated with that division by a METS Pointer or a File Pointer

- XML extension "schemas"
  - descriptive metadata
    - Dublin Core, MARC, VRA...
  - administrative metadata
    - NISO image technical metadata
    - LC schemas for A/V technical metadata
    - Rights metadata
    - etc.

2.27. MODS

MODS : Metadata Object Description Schema

a schema for a bibliographic element set that may be used for different purposes, particularly for library applications. maintained by:

Network Development and MARC Standards Office

Library of Congress, USA

2.28. MODS

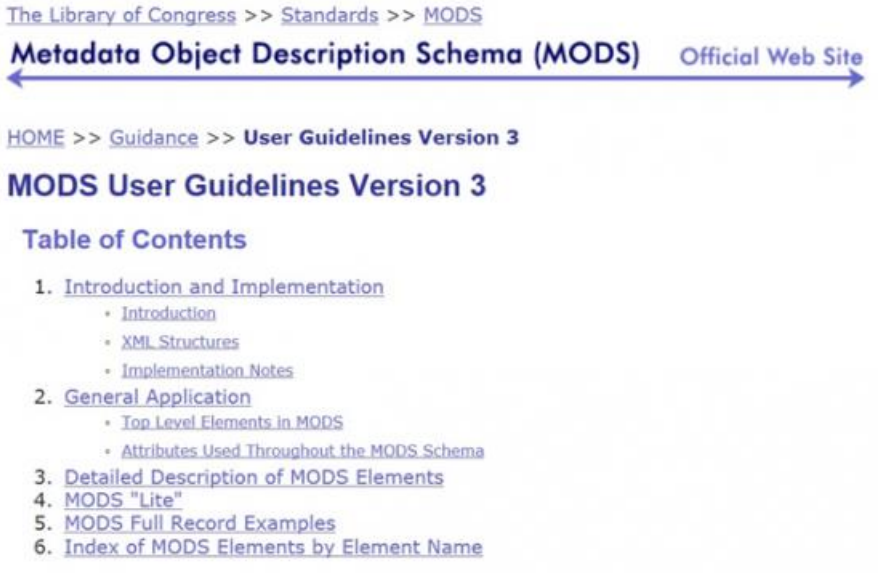
- MARC21 compatible XML descriptive metadata standard
- MARC21 derivative, simpler than MARC21 but compatible
- MODS descriptive records can be contained inside METS record
- richer than Dublin Core
- Title Info
- Name
- Type of resource
- Genre
- Origin Info
- Language
- Physical description
- Abstract
- Table of contents
- Target audience
- Note
- Subject
- Classification
- Related item
- Identifier
- Location
- Access conditions
- Part
- Extension
- Record Info

2.29. MODS evaluation

- Element set is compatible with existing descriptions in large library databases
- Element set is richer than Dublin Core but simpler than full MARC
- Language tags are more user-friendly than MARC numeric tags
- Hierarchy allows for rich description, especially of complex digital objects
- Rich description that works well with hierarchical METS objects

2.30. MODS User Guidelines

http://www.loc.gov/standards/mods/v3/mods-userguide.html



[fragile]

2.31. element and attribute

```
<genre authority = "EIT"> portraits
</genre>
```

Element

Attribute

Controlled vocabulary stipulated as the "authority"

Value (i.e. Controlled term)

Element

[fragile]

2.32. element and sub element

Element

<originInfo<place> <placeTerm type="text">BME, Budapest </placeTerm>

Sub elements

Attribute

Text stipulated as the "type"

Value (i.e. Place name)

Sub element

sub elements clarify, explain source of terms/codes, or provide instructions for data manipulation

2.33. MODS Elements

- elements and attributes are optional
- elements and sub elements are repeatable
- some top level elements may also serve as a sub element under another element

2.34. EAD: Encoded Archival Description

- standard for archival and manuscript collections
- XML DTD
- supports archival descriptive practices for discovery, exchange and use of data
  - Single-level description at collection level
  - Multilevel description from collection through file and item levels
- maintained by Society of American Archivists
  - http://www.loc.gov/ead/

2.35. EAD features

- Documents the interrelated descriptive information of an archival finding aid
- Preserves the hierarchical relationships existing between levels of description
- Represents descriptive information that is inherited by one hierarchical level from another
- Supports element-specific indexing and retrieval of descriptive information

2.36. MPEG standards

- MPEG-1: coding of moving pictures and associated audio for digital storage media (at up to about 1.5 Mbit/s)
- MPEG-2: coding of moving pictures and associated audio for broadcast-quality (digital) television
- MPEG-3: was intended for HDTV compression but was merged with MPEG-2 -> no MPEG-3 standard today not(not to be confused with MP3 (MPEG-1 Audio Layer III)
- MPEG-4: coding of audio-visual objects
- MPEG-7: multimedia content description interface
- MPEG-21: multimedia framework (iidentification, copyright, protection, etc.)

2.37. MPEG-7

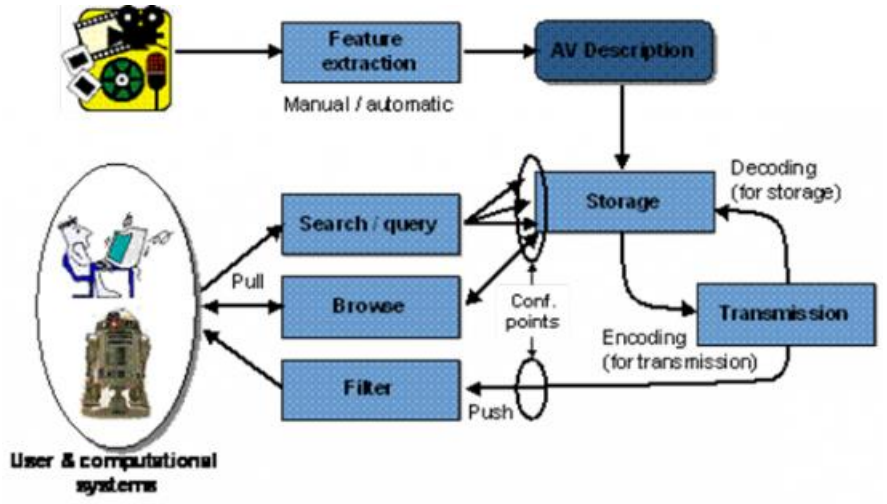
- content-based description for audiovisual information in multimedia environments

- Content may include: still pictures, graphics, 3D models, audio, speech, video, and composition information about how these elements are combined (scenarios).
- May cover facial expressions, personal characteristics, etc.

MPEG-7 Description tools allow to create descriptions of content that may include:

- Information describing the creation and production processes of the content (director, title, short feature movie)
- Information related to the usage of the content (copyright pointers, usage history, broadcast schedule)
- Information of the storage features of the content (storage format, encoding)
- Structural information on spatial, temporal or spatio-temporal components of the content (scene cuts, segmentation in regions, region motion tracking)
- Information about low level features in the content (colors, textures, sound timbres, melody description)
- Conceptual information of the reality captured by the content (objects and events, interactions among objects)

2.38. Application model



2.39. main elements of MPEG-7 standard

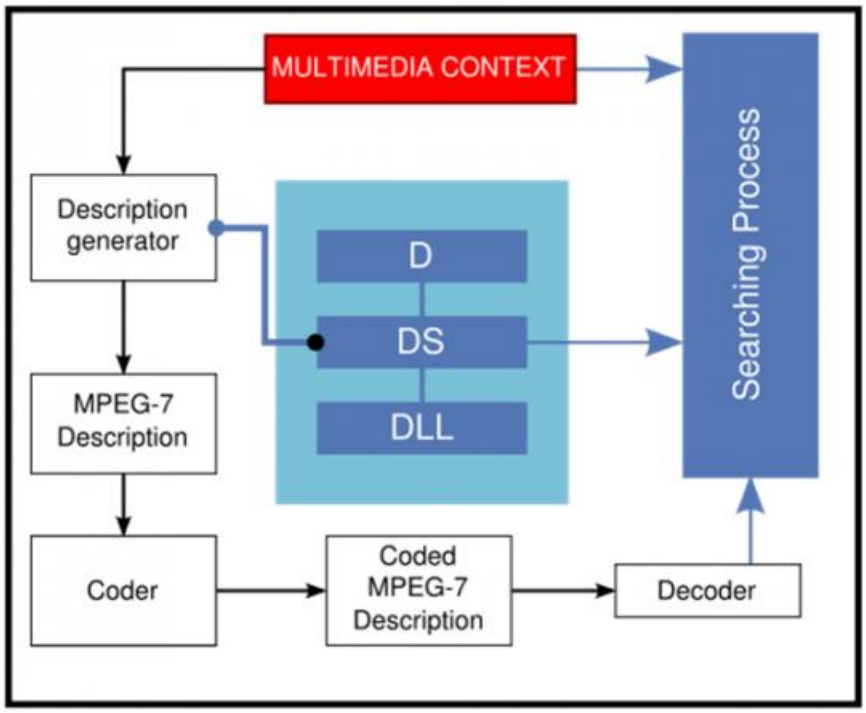
- Descriptors (D): representations of Features, that define the syntax and the semantics of each feature representation,
- Description Schemes (DS): specify the structure and semantics of the relationships between their components.(components can be both Descriptors and Description Schemes)

2.40. MPEG-7 DDL

DDL : Description Definition Language

- defines the syntactic rules to express and combine Description Schemes and Descriptors
- DDL is a schema language to represent the results of modeling audiovisual data, i.e. DSs and Ds.

- DDL is an XML Schema application, with some specific extensions (such as array and matrix datatypes).DDL allows to define complexTypes and simpleTypes. The complexTypes specify the structural constraints while simpleTypes express datatype constraints.



2.41. MPEG-7 Visual

- MPEG-7 Visual Description Tools:basic structures and Descriptors that cover the visual features: Color, Texture, Shape, Motion,Localization, Face recognition.Each category consists of elementary and sophisticated Descriptors

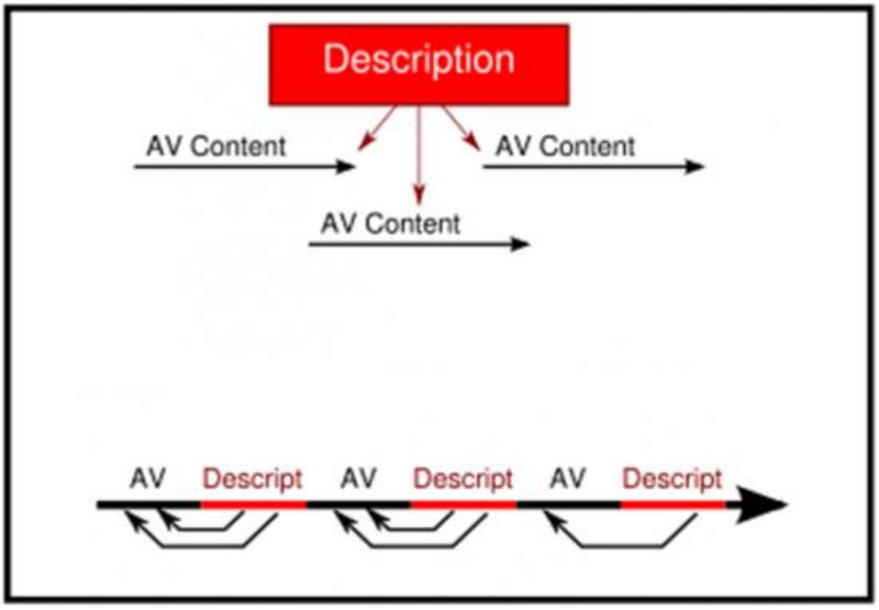
2.42. MPEG-7 Visual examples

- Shape: the shape of an object may consist of either a single connected region or a set of disjoint regions, as well as some holes in the object
- (Camera) motion: characterizes 3-D camera motion parameters. (can be automatically extracted or generated by capture devices

2.43. MPEG-7 Audio

MPEG-7 Audio Description Tools:

- Audio Framework. The main hook into a description for all audio description schemes and descriptors
- Spoken Content DS. A DS representing the output of Automatic Speech Recognition (ASR).
- Timbre Description. A collection of descriptors describing the perceptual features of instrument sounds
- Audio Independent Components. A DS containing an Independent Component Analysis (ICA) of audio



2.44. MPEG-7 and MPEG-21 community

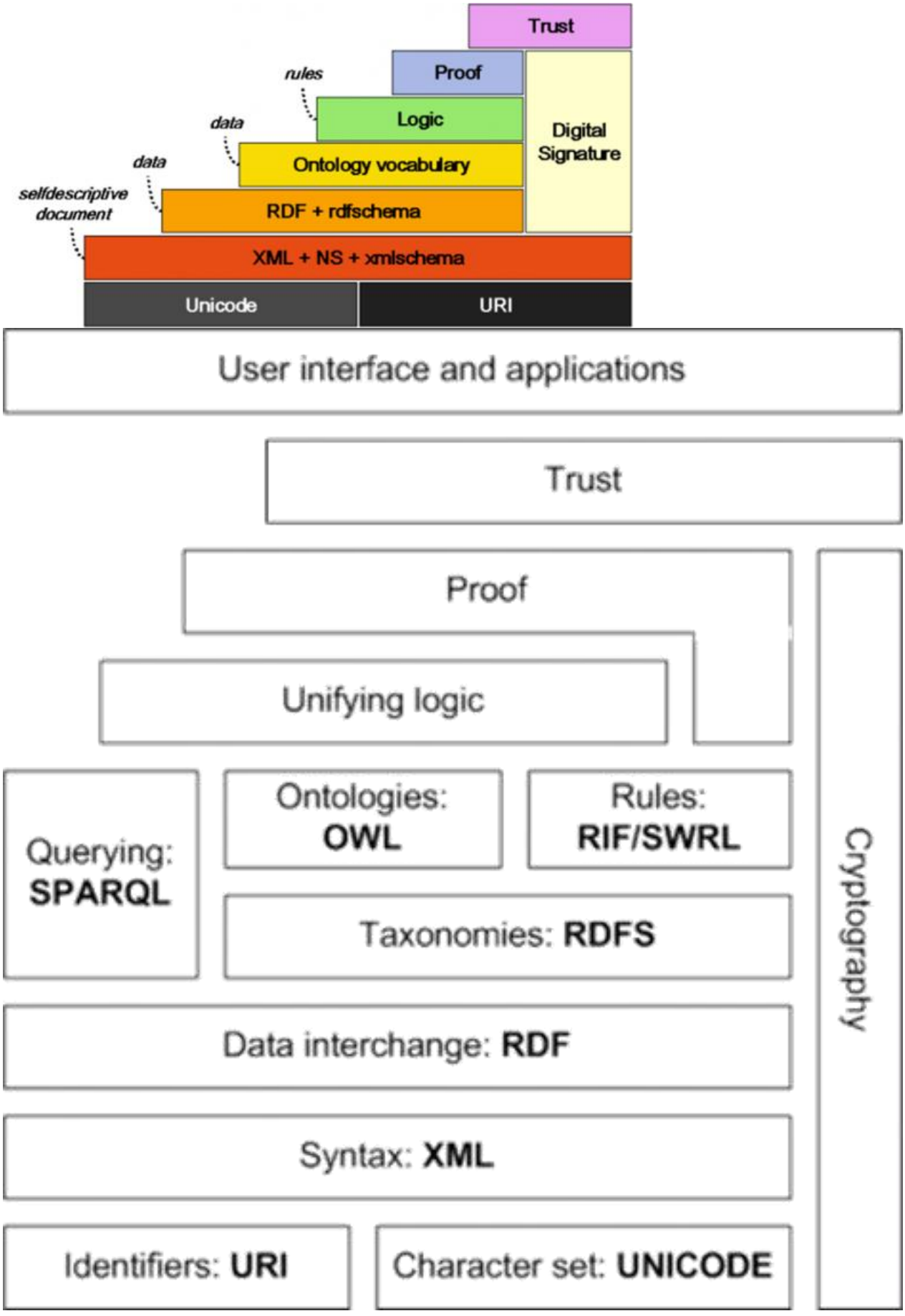
- http://www.multimedia-metadata.info/

3. 3 SEMANTIC CONTENT MANAGEMENT

3.1. Semantic content management/1

- Semantic content management (RDF, RDF Schema, ontologies).
- Semantic Web Layers.
- Thesaurus.

Semantic web layers



3.2. RDF

is a W3C standard for describing Web resources

is a general method to decompose any type of knowledge into small pieces

method for expressing knowledge

the foundation of the Semantic Web

[fragile]

3.3. Why we need RDF?

```
<?xml version="1.0" encoding="UTF-8"?>
<Adatbazis>
<Szemely>
<Nev>Fekete Péter</Nev>
<Feleseg>Fehér Mária</Feleseg>
<Munkahely>NevenincsKft.</Munkahely>
<Kereset penznem="Ft"
tipus="brutto">300000</Kereset>
</Szemely>
</Adatbazis>
```

[fragile]

```
<?xml version="1.0" encoding="UTF-8"?>
<DB>
<Persons>
<Person ID="p1" wife="p2" worksfor="c1">
<PName>Fekete Péter</PName>
<Salary currency=''USD,, type="gross">
1304</Salary>
</Person>
<Person ID="p2">
<PName>Fehér Mária</PName>
</Person>
</Persons>
<Companies>
<Company ID="c1">
<CName>NevenicsKft.</CName>
</Company>
</Companies>
</DB>
```

- N3 (Notation3) language
- primer - getting into the semantic web and rdf using n3.htm

[fragile]

3.4. URI

Uniform Resource Identifier

RDF:

qualified URI

(URI + optional detail description: #text)

[fragile]

3.5. RDF

A fact is expressed as a triple of the form

Subject, Predicate, Object

(a simple English sentence)

Subjects, predicates, and objects are names for entities, whether concrete or abstract, in the real world

```
<#pat> <#knows> <#jo> .
```

Subject / Verb / Predicate / Property / Object

3.6. Subject, Predicate, Object

- Subject, Predicate (Verb/Property) and an Object value forms a Statement

[fragile]

3.7. statement

```
<#pat> <#knows> <#jo> .
```

- RDF uses Web identifiers (URI: Web identifier, Uniform Resource Identifier)
- Object can be a string: <#pat> <#age> ,,34" .
- Predicate expresses the other 2 elements relation:

```
<#pat> <#child> <#al> .
<#pat> has <#child> <#al> .
<#al> is <#child> of <#pat> .
```

[fragile]

3.8. abbreviations

Semicolon (;)

>1 predicate to the same subject,

comma (,)

>1 object related to the same subject-predicate pair

```
<#pat> <#child> <#al>, <#chaz>, <#mo>;
<#age> ,,34" ;
<#eyecolor> "blue" .
```

[fragile]

3.9. expressive strength

	age	eyecolor
pat	34	blue
al	3	green
jo	5	green

```
<#pat> <#age> ,,34"; <#eyecolor> "blue" .
<#al> <#age> "3"; <#eyecolor> "green" .
<#jo> <#age> "5"; <#eyecolor> "green" .
```

[fragile]

```
<#pat><#child>[<#age> "4"],[<#age> "3"].
```

(Means: #pat has a #child which has #age of "4" and a #child which has an #age of "3")

No identified object in this statement.

What you find in [ ] : refers to an existing object, but no URI for that.

More precisely: [ ] declares that there is something with that property, but you can't get right to refer to that.

If you need to do so:

```
[ <#name> "Pat"; <#age> "24"; <#eyecolor> "blue" ].
[ <#name> "Al" ; <#age> "3"; <#eyecolor> "green" ].
[ <#name> "Jo" ; <#age> "5"; <#eyecolor> "green" ].
```

[fragile]

"pat", "child", "age", e.t.c. are just characters, do not carry "meaning" to the computer!

until we do not state this (and <#name> is defined somewhere):

```
<#pat> <#name> "Pat".
```

[fragile]

3.10. Common concepts

"title" (e.g. in web document, library catalogue) is just a concept. In case >1 resource (e.g. documents) will use that:

- we need a a common understanding
- we should use exactly the same "lexical item" to identify that concept

```
<> <#title> ,,RDF basics for EIT Master Course".
```

(<> refers to the current document. In this example #title refers to a concept, what is defined by the document itself.)

3.11. Common concepts

Let's use Dublin Core (DC) definition!

http://purl.org/dc/elements/1.1/title "RDF basics for EIT Master Course".

make it shorter in N3:

@prefix dc: <http://purl.org/dc/elements/1.1/> .

<> dc:title " RDF basics for EIT Master Course ".

[fragile]

Use reliable, widely accepted prefixes, e.g.:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix ont: <http://www.daml.org/2001/03/daml-ont#>.
```

[fragile]

Let

```
@prefix : <#> .
```

Using this formalism the previous statement can be shorter:

```
:pat :child [ :age "4" ] , [ :age "3" ].
```

[fragile]

3.12. RDF vocabulary

is a defined set of predicates that can be used in an application.

dc:title = predicate

new vocabulary -> definition of classes and propeties

Class: what type is that something rdf:type

There is an abbreviation for that in N3: 'a'

Let's define a class of persons:

:Person a rdfs:Class.

We will use that in a document like this:

:Pat a :Person.

[fragile]

object can be in many classes (doesn't have to be any hierarchical relationship - but you can state this in RDF Schema, example. :Woman a rdfs:Class; rdfs:subClassOf :Person)

- when a subject of any property must be in a class:domain (What can have this kind of property?)
- when an object must be in a class range (What type of values can have)

```
:sister rdfs:domain :Person;
rdfs:range :Woman.
```

Convention:

- Class IDs begin with capitals
- Properties lower case letters

http://tmit.bme.hu/index.html

mailto::whois@bme.hu

http://vhol.org/DC/Creator

http://tmit.bme.hu/index.html

mailto::whois@bme.hu

http://vhol.org/DC/Creator

mailto::tome@bme.hu

http://tmit.bme.hu/photo.html

http://vhol.org/schema/content

http://vhol.org/schema/together

http://vhol.org/DC/Creator

3.13. Semantic Graph Data Model

RDF has an abstract syntax that reflects a simple graph-based data model.

Any statement in RDF is a collection of triples, each consisting of a subject, a predicate and an object

Triples together form RDF graphs, by a node and directed-arc diagram.

Nodes are subjects and objects; direction of the arc -> toward the object

A node may be a URI with optional fragment identifier, a literal, or blank. Properties are URI references

A URI reference or literal used as a node identifies what that node represents.

Simple logical statements are coded in the graph.

Directed arc ('P') from node 'A' to node 'B'

'P' predicate arc

"property of node 'A' is 'B'

Statement representation: (P,A,B) triple.

(where 'P' is a property, the predicate of the statement, 'A' is the subject, 'B' is the object.

http://tmit.bme.hu/index.html

mailto::whois@bme.hu

http://vhol.org/DC/Creator

http://tmit.bme.hu/index.html

mailto::whois@bme.hu

http://vhol.org/DC/Creator

mailto::tome@bme.hu

http://tmit.bme.hu/photo.html

http://vhol.org/schema/content

http://vhol.org/schema/together

http://vhol.org/DC/Creator

http://tmit.bme.hu/index.html

mailto::whois@bme.hu

http://vhol.org/DC/Creator

OBJECT PREDICATE SUBJECT

"The creator of index.html is whois."

http://tmit.bme.hu/index.html

mailto::whois@bme.hu

http://vhol.org/DC/Creator

mailto::tome@bme.hu

http://tmit.bme.hu/photo.html

http://vhol.org/schema/content

http://vhol.org/schema/together

http://vhol.org/DC/Creator

"The creator of index.html is whois and tome (works together). index.html contains photo.html"

3.14. RDF Schema

- RDF is a data model that provides a way to express simple statements about resources, using named properties and values.
- The RDF Vocabulary Description Language 1.0 (or RDF Schema or RDFS) is a language that can be used to define the vocabulary (i.e., the terms) to be used in an RDF graph.
- The RDF Vocabulary Description Language 1.0 is used to indicate that we are describing specific kinds or classes of resources, and will use specific properties in describing those resources.
- The RDF Vocabulary Description Language 1.0 is an ontology definition language (a simple one, compared with other languages such as OWL; we can only define taxonomies and do some basic inference about them).
- The RDF Vocabulary Description Language is like a schema definition language in the relational or object-oriented data models (hence the alternative name RDF Schema - we will use this name and its shorthand RDFS mostly!).

[fragile]

3.15. RDF Schema

- The RDF Schema concepts are themselves provided in the form of an RDF vocabulary; that is, as a specialized set of predefined RDF resources with their own special meanings.
- The resources in the RDF Schema vocabulary have URIrefs with the prefix http://www.w3.org/2000/01/rdf-schema (associated with the QName prefix rdfs:).
- Vocabulary descriptions (schemas, ontologies) written in the RDF Schema language are legal RDF graphs. In other words, we use RDF to represent RDFS information.

3.16. Classes

- A basic step in any kind of description process is identifying the various kinds of things to be described. RDF Schema refers to these "kinds of things" as classes.

- A class in RDF Schema corresponds to the generic concept of a type or category, somewhat like the notion of a class in object-oriented programming languages such as Java or object-oriented data models.

[fragile]

3.17. Defining Classes

- Suppose an organization example.org wants to use RDF Schema to provide information about motor vehicles, vans and trucks.
- To define classes that represent these categories of vehicles, we write the following statements (triples):

```
ex:MotorVehicle rdf:type rdfs:Class .
ex:Van rdf:type rdfs:Class .
ex:Truck rdf:type rdfs:Class .
```

- In RDFS, a class C is defined by a triple of the form

```
C rdf:type rdfs:Class .
```

  using the predefined class rdfs:Class and the predefined property rdf:type.

[fragile]

3.18. Defining Instances

- Now suppose example.org wants to define an individual car (e.g., the company car) and say that it is a motor vehicle.
- This can be done with the following RDF statement:

```
exthings:companyCar rdf:type ex:MotorVehicle .
```

[fragile]

3.19. rdf:type

- The predefined property rdf:type is used as a predicate in a statement

```
I rdf:type C
```

  to declare that individual I is an instance of class C.

- In statements of the form

```
C rdf:type rdfs:Class .
```

  rdf:type

is used to declare that class C (viewed as an individual object) is an instance of the predefined class rdfs:Class.

[fragile]

3.20. Defining Classes (cont'd)

- Defining a class explicitly is optional; if we write the triple

```
I rdf:type C
```

then C is inferred to be a class (an instance of rdfs:Class) in RDFS.

3.21. Notation

- Class names will be written with an initial uppercase letter.
- Property and instance names are written with an initial lowercase letter.

[fragile]

3.22. Defining Subclasses

- Now suppose example.org wants to define that vans and trucks are specialized kinds of motor vehicle.
- This can be done with the following RDF statements:

```
ex:Van  rdfs:subClassOf  ex:MotorVehicle .
ex:Truck  rdfs:subClassOf  ex:MotorVehicle .
```

- The predefined property rdfs:subclassOf is used as a predicate in a statement to declare that a class is a specialization of another more general class.
- A class can be a specialization of multiple superclasses (e.g., the graph defined by the rdfs:subclassOf property is a directed graph not a tree).

[fragile]

3.23. Classes and Instances

- The meaning of the predefined property rdfs:subClassOf in a statement of the form

```
C1 rdfs:subClassOf C2
```

is that any instance of class C1 is also aninstance of class C2.

- Example: If we have the statements

```
ex:Van  rdfs:subClassOf  ex:MotorVehicle .
exthings:myCar rdf:type ex:Van .
```

then RDFS allows us to infer the statement

```
exthings:myCar rdf:type ex:MotorVehicle .
```

[fragile]

3.24. Properties of rdfs:subClassOf

- The rdfs:subClassOf property is reflexive and transitive.
- Examples:
  - If we have a class ex:MotorVehicle then RDFS allows us to infer the statement

    ```
    ex:MotorVehicle rdfs:subClassOf ex:MotorVehicle .
    ```

  - If we have the statements

    ```
    ex:Van rdfs:subClassOf ex:MotorVehicle .
    ex:MiniVan rdfs:subClassOf ex:Van .
    ```

  then RDFS allows us to infer the statement

```
ex:MiniVan rdfs:subClassOf ex:MotorVehicle .
```

[fragile]

3.25. RDF Schema Predefined Classes

- The group of resources that are RDF Schema classes is itself a class called rdfs:Class. All classes are instances of this class.
- In the literature, classes such as rdfs:Class that have other classes as instances are called meta-classes.
- All things described by RDF are called resources, and are instances of the class rdfs:Resource.
- rdfs:Resource is the class of everything. All other classes are subclasses of this class. For example, rdfs:Class is a subclass of rdfs:Resource.

[fragile]

3.26. Properties

- In addition to defining the specific classes of things they want to describe, user communities also need to be able to define specific properties that characterize those classes of things (such as author to describe a book).

[fragile]

3.27. Defining Properties

- A property can be defined by stating that it is an instance of the predefined class

```
rdf:Property.
```

- Example:

```
ex:author rdf:type rdf:Property .
```

- Then, property ex:author can be used as a predicate in an RDF triple such as the following:

```
ex:john ex:author ex:book123 .
```

[fragile]

3.28. Defining Properties

- Defining a property explicitly is optional; if we write the RDF triple

```
S P O .
```

  then P is inferred to be a property by RDFS.

[fragile]

- Properties are resources too (this makes RDF and RDFS different than many other KR formalisms).
- Therefore, properties can appear as subjects or objects of triples.
- Example (provenance):

```
ex:author prov:definedBy ke:john
ke:john prov:defined ex:author
```

- We will see many more examples like the above.
- In RDFS property definitions are independent of class definitions. In other words, a property definition can be made without any reference to a class.
- Optionally, properties can be declared to apply to certain instances of classes by defining their domain and range.

[fragile]

3.29. Domain and Range

- RDFS provides vocabulary for describing how properties and classes are intended to be used together in RDF data.
- The rdfs:domain predicate can be used to indicate that a particular property applies to instances of a designated class (i.e., it defines the domain of the property).

- The rdfs:range predicate is used to indicate that the values of a particular property are instances of a designated class (i.e., it defines the range of the property).

[fragile]

3.30. Datatypes for Ranges

- The rdfs:range property can also be used to indicate that the value of a property is given by a typed literal.
- Example:

```
ex:age rdf:type rdf:Property .
ex:age rdfs:range xsd:integer .
```

- Optionally, we can also assert that xsd:integer is a datatype as follows:

```
xsd:integer rdf:type rdfs:Datatype .
```

[fragile]

3.31. The Property rdfs:label

- rdfs:label is an instance of rdf:Property that may be used to provide a human-readable version of a resource's name.
- The rdfs:domain of rdfs:label is rdfs:Resource. The rdfs:range of rdfs:label is rdfs:Literal.
- Multilingual labels are supported using the language tagging facility of RDF literals.

[fragile]

3.32. The Property rdfs:comment

- rdfs:comment is an instance of rdf:Property that may be used to provide a human-readable description of a resource.
- The rdfs:domain of rdfs:comment is rdfs:Resource. The rdfs:range of rdfs:comment is rdfs:Literal.
- Multilingual documentation is supported through use of the language tagging facility of RDF literals.

[fragile]

3.33. The Property rdfs:seeAlso

- rdfs:seeAlso is an instance of rdf:Property that is used to indicate a resource that might provide additional information about the subject resource.
- A triple of the form S rdfs:seeAlso O states that the resource O may provide additional information about S. It may be possible to retrieve representations of O from the Web, but this is not required. When such representations may be retrieved, no constraints are placed on the format of those representations.
- The rdfs:domain of rdfs:seeAlso is rdfs:Resource. The rdfs:range of rdfs:seeAlso is rdfs:Resource.

[fragile]

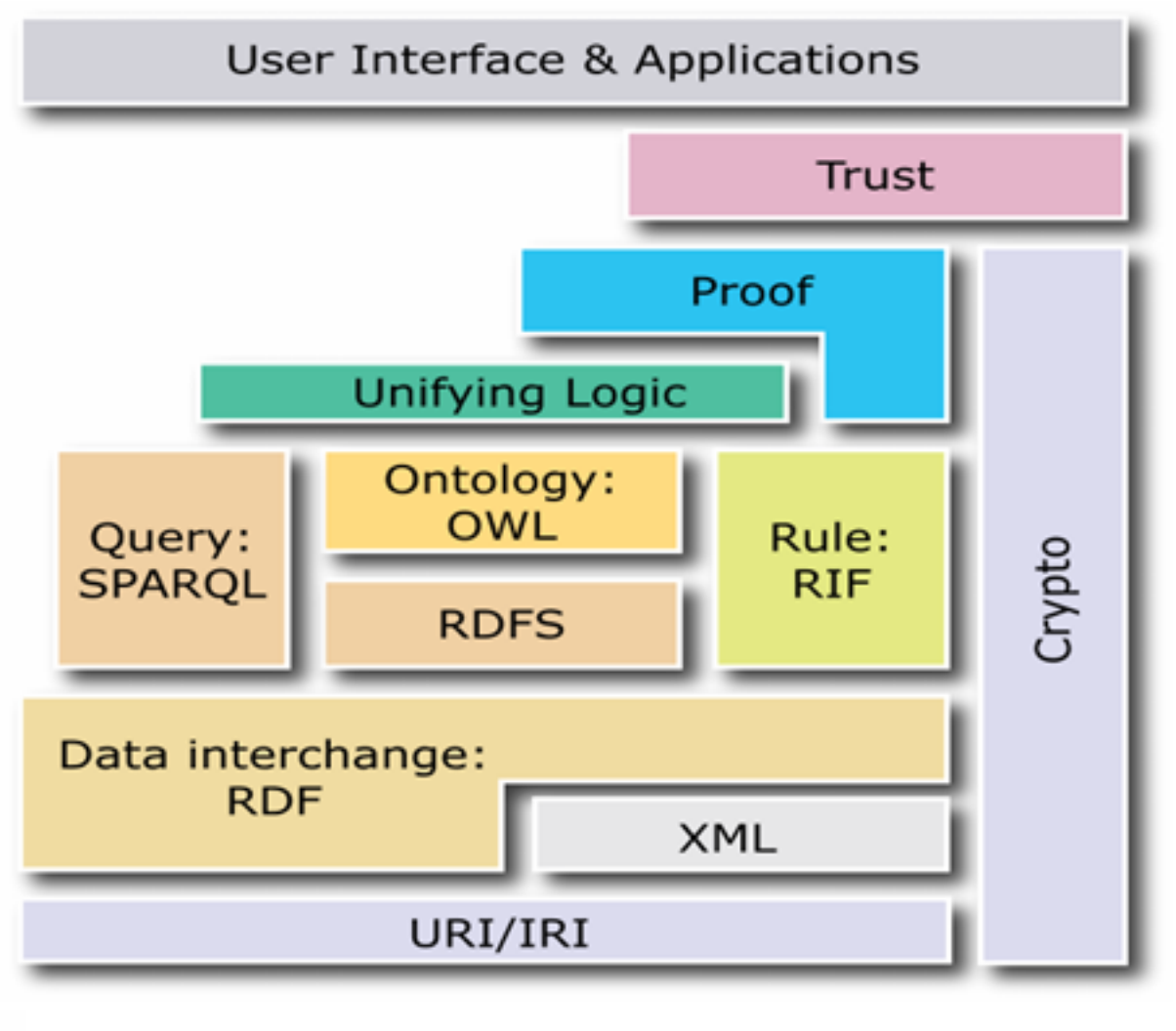
3.34. RDFS vs. Types (cont'd)

- Benefits of the RDF approach: One can start with a property definition and then extend it to other uses that might not have been anticipated.
- Shortcoming: In RDFS, it is not possible to say, for example, that if the property ex:hasParent is used to describe a resource of class ex:Human, then the range of the property is also a resource of class ex:Human, while if the property is used to describe a resource of class ex:Tiger, then the range of the property is also a resource of class ex:Tiger. This can be done in ontology languages that we will define later in the course.

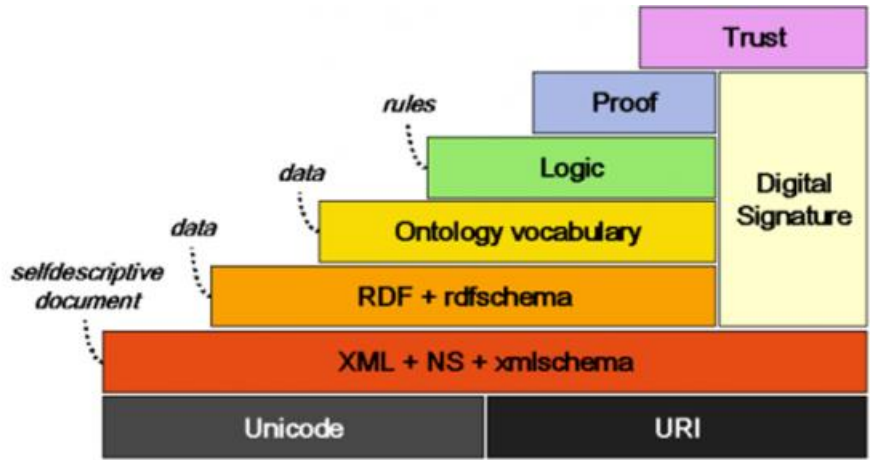
3.35. Schema Languages

- RDF Schema provides basic capabilities for describing RDF vocabularies, but additional capabilities are also possible, and can be useful.
- These capabilities may be provided through further development of RDF Schema, or in other languages (for example, ontology languages such as OWL).

3.36. The Semantic Web "Layer Cake"

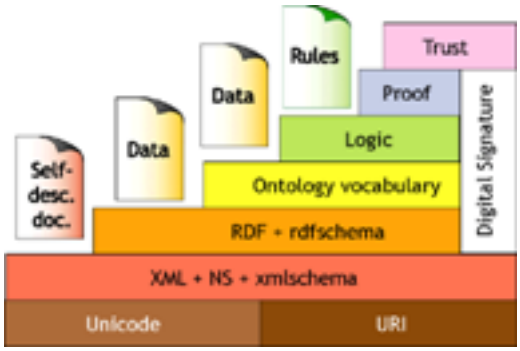


- Semantic web layers - overview



3.37. URI, Unicode layer

- URI and Unicode layer: ensures to identify objects using international character set IDs
- unambiguous identification is essential to consistent statements
- Typical URIsProtocol-dependent (http://, mailto:)Protocol/position independent (URN-s, PURL)"Identifiers and character set"



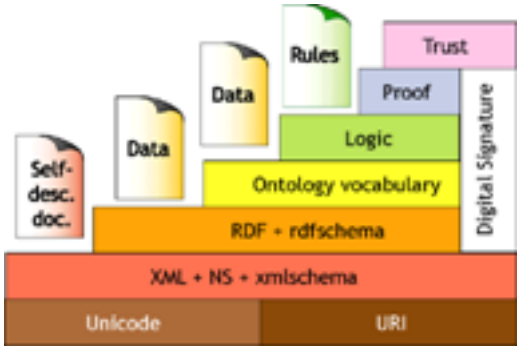
3.38. XML layer

To organize the resources (e.g. documents)

into structures

nothing is about meaning

"syntax"



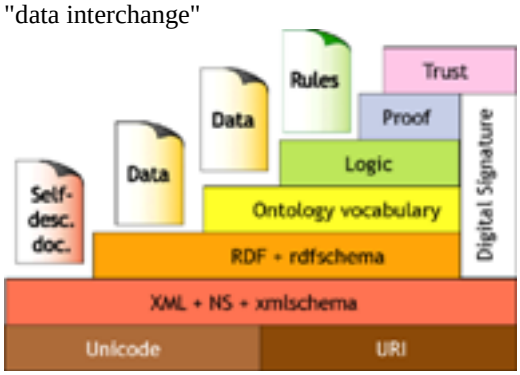
3.39. RDF layer

"meaning layer":

triples,

all triples: subject, predicate, object

"data interchange"

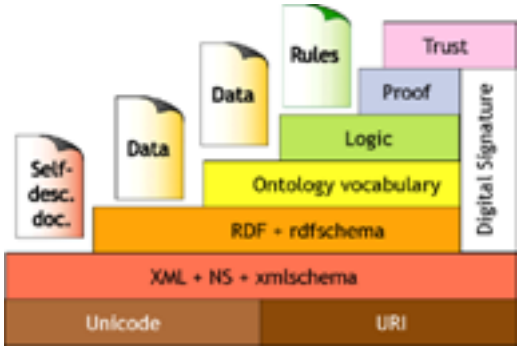


3.40. RDF Scheme

Why we need scheme

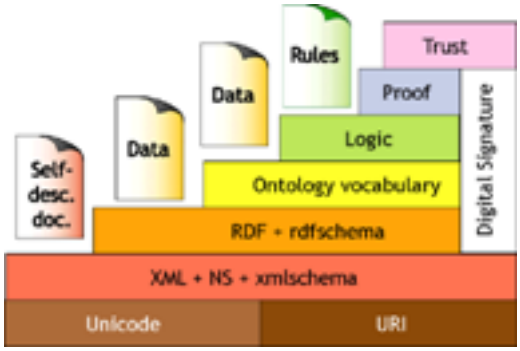
- in RDF no data types
- Users should agree in given lexical units (used in documents)
- RDF Schemas: what relations are accepted? to what resources are valid those relations?
- RDFS provides basic vocabulary for RDF. RDFS: create hierarchies of classes and properties

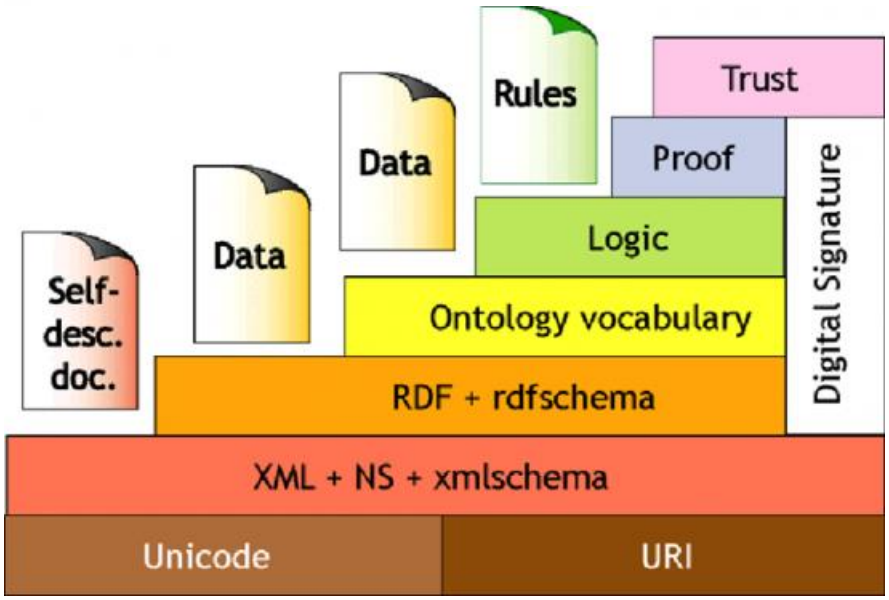
"taxonomies"



3.41. Ontology layer

- RDF schemas gives the "basics" (no inferences about concepts)"if A is true, than B is true""if C is true, than D is not true"
- Ontology layer extends RDFS by adding more advanced constructs to describe semantics of RDF statements.Allows stating additional constraints (e.g. cardinality, restrictions of values, or characteristics of properties)Based on description logic. Brings reasoning power to the semantic web.





3.42. Controlled vocabularies

- Vocabularies and other language constructions of native or artificial languages to support information management
  - Term list, free phrase list
  - Classification (categorization)
  - Relationship Schemes
- Term list
  - Glossaries, Dictionaries, Authority Files
- Classification
  - Subject Classification
  - Taxonomy
- Relationship Schemes
  - Thesaurus
  - Semantic Network (e.g. WordNet)
  - Ontology

3.43. Thesaurus

- A structured wordlist used to standardise terminology(more than a standard wordlist - helps with indexing and acts as a guide to a subject area)
- Reasoning over thesaurus relationships:
  - automatic suggestion of terms to be considered for query
  - query reformulation ('like this' option)
  - semantic expansion of queries

3.44. Thesaurus standards

- ISO 2788:1986 Documentation -Guidelines for the establishment and development of monolingual thesauri
- ISO 5964:1985 Documentation -Guidelines for the establishment and development of multilingual thesauri
- 4 basic relationships in a thesaurus
  - Equivalence (preferred and non-preferred terms)
  - Hierarchy (narrower (more specific) / broader (more general) term)
  - Association (establishes non-hierarchical relationships, e.g. related term)
  - Scope notes (provide guidance and clarification)

DATABASE

M: Data stored electronically, available for search, distribution

F Technical document

X CD ROM

DATA STORAGE

M: A storage medium on which certain physical aspects describe data

A Celluloid film

Vinyl

Sound tape

Video disc

Magnetic storage

Optical storage

Sheet of paper

T Document

THESIS

M: A certification of a qualification or obtained title

F Certificate

H Diploma Thesis

L Bachelor's Thesis

DISSERTATION

M: A written work to acquire an academic title in a higher education program

H Doctoral Thesis

F Study

X Diploma Thesis

Thesis

3.45. Widely known thesauri

- The Art and Architecture Thesaurus, Getty Institutehttp://www.getty.edu/research/conducting_research/vocabularies/aat/
- Union List of Artist's Names, Getty Institutehttp://www.getty.edu/research/conducting_research/vocabularies/ulan/
- British Museum Object Names Thesaurus http://www.mda.org.uk/bmobj/Objintro.htm

- NASA Thesaurushttp://www.sti.nasa.gov/thesfrm1.htm
- USA Library of Congresshttp://www.loc.gov/lexico/servlet/lexico/

4. 4 CMS types

4.1. CMS types

- DAM: Digital Asset Management
- DM: Document Management
- LMS: Learning Management System
- WCMS: Web CMS
- KM: Knowledge Management
- ECM: Enterprise Content Management

4.2. DAM: Digital Asset Management

Task of DAM is to handle, store, retrieve, annotate, distribute of large amounts of digital assets.

- broad categories of DAM based on purpose of management handling
- subtypes of DAM based on content types

4.3. Broad categories of DAM

- Brand asset management systems focus on marketing- or sales-related content re-use within large organizations (e.g. product imagery, logos, etc.).
- Library asset management systems handle with storage and retrieval of large amounts of infrequently changing media assets (e.g. text, video or photo archiving).
- Production asset management systems focus on managing assets as they are being created for a digital media production (video game, music, animation, etc.).

4.4. Subtypes of DAM

- Gereral DAMÂ
- Media Asset Management (MAM) (see also DMMS - Digital Media Management Systems), e.g. Convera, Virage, Cinebase. This contains high levels service to assist in creating, managing, distributing and using rich media contents / assets (audio, video).

4.5. DAM file types

- Images
- Audio

- Video
- Logos and line art
- Animation like Flash
- CADÂ
- 3-D
- HTML

4.6. Application (areas) of DAM

- Multimedia press Kits
- Multimedia sales
- Video on demand (VOD)
- Rich Media libraries
- Advertising and marketing
- Tutorials (multimedia)
- Film production
- TV content production

4.7. DM: Document Management

The task of DM is to store and track of the (different versions of) electronic documents.

- Document management systems commonly provide storage, versioning, metadata, security, as well as indexing and retrieval capabilities.

4.8. Components of DM

Metadata, Integration, Capture, Indexing, Storage, Retrieval, Distribution, Security, Workflow, Collaboration, Versioning, Searching, Publishing, Reproduction

4.9. Capture

- Capture primarily involves accepting and processing images of paper documents from scanners.
- Optical character recognition (OCR) software is often used in order to convert digital images into machine readable text.

4.10. Functionalities in DM

- Integration with text processing tools (Word, Adobe).
- Definition of content components within a document.
- Assemble documents or document segments for reuse.
- Modifying documents.
- Storing of text effectively, often in a variety of different formats, including native formats and XML.
- Provide text specific search features such as natural language search.
- Apply metadata at varying levels within the document.

4.11. DM file types

- Paper
- File output from office automation tools (e.g. PPT)
- PDF
- Text output
- Imaged documents
- XML documents and fragments
- Other text files, including HTML
- Other images and multimedia files, typically as generic binaries

4.12. Application (areas) of DM

- Contracts
- Documentation/Manuals
- Policy and Procedures
- Forms
- Research
- Applications for regulatory product approval
- Statements (program codes)
- Articles
- Reports

4.13. LMS: Learning Management System

is a software application for the

- administration,
- documentation,
- tracking,
- reporting and
- delivery

of education courses or training programs.

4.14. Functionalities in LMS

- centralize and automate administration
- assemble and deliver learning contents
- personalize content and enable knowledge reuse

- deliver online training
- support portability and standards

4.15. LMS standards

- AICC
- SCORM: Sharable Content Object Reference Model is a specification of the Advanced Distributed Learning (ADL) Initiative

4.16. Most important LMS softwares

- Blackboard Learning System (leading provider), WebCT (acquired Blackboard in 2005)
- Moodle (Open source)

4.17. Web CMS

is a software system that provides

- website authoring,
- collaboration,
- administration tools

to create and manage website content with relative ease.

4.18. Functionalities in WCMS

- Automated templates
- Access control
- Easily editable content
- Web standards upgrades
- Workflow management

- Collaboration
- Content syndication
- Multilingual
- Versioning

4.19. Most important WCMS softwares

- WordPress originated as a blogging CMS, but later evolved into a fully CMS.
- Joomla! can be used to easily create and edit webpages.
- Drupal is more difficult to learn and understand than the above two CMSs, but is the most secure.

4.20. WCMS file types

Large variety in types

- HTML,
- XML,
- PDF,
- JPG, GIF, PNG, BMP,
- WAV, MP3,
- MPEG2, MP4

4.21. KM: Knowledge Management

is based on a range of practices used by an individual or organization to

- identify,
- create,
- represent and
- redistribute

knowledge.

4.22. Knowledge Management software

collects and contains information to then build knowledge that can be searched through specialised search tools including

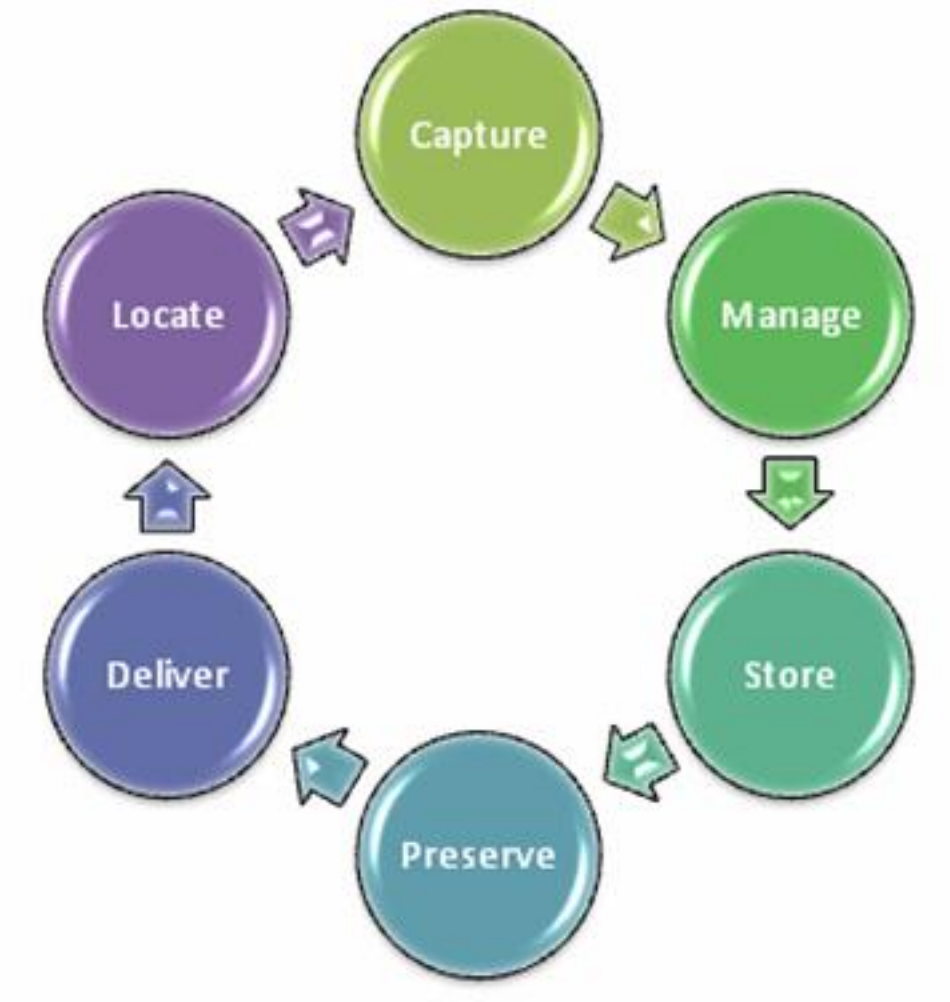
- concept building tools and or
- visual search tools.

4.23. ECM: Enterprise Content Management

- organizes and stores (enterprise's) documents, and other content, that relate to the organization's processes.
- This deals with strategies, methods, and tools used throughout the lifecycle of the content.

4.24. Components of ECM

- capture
- manage
- store
- preserve
- deliver



4.25. Capture component

- Recognition technologies, like OCR, Optical mark recognition (OMR such as checkmarks or dots), Barcode recognition.
- Aggregation combines documents from different applications.
- Indexing (automatic or manual)

4.26. Manage component

- Document management
- Collaboration (or collaborative software, a.k.a. groupware)
- Web content management (including web portals)
- Records management
- Workflow and business process management (BPM)

4.27. Store component

- Repositories as storage locations,
- Library Services as administration components for repositories, and
- Storage technologies.

4.28. Preserve component

involves the long-term, safe storage and backup of static, unchanging information.

- Long term storage media (Write once read many: WORM technology)
- Long term preservation strategies with conversion and migration tools (migration of applications, emulation of older software)

4.29. Deliver component

- Methods
  - On-premise as a traditional software application that companies implemented on their own networks.
  - Software as a service (SaaS) users access the application and their data online. It is also known as cloud computing, hosted, and on demand.
  - Hybrid
- Transformation technologies (Personalization, Syndication)
- Security technologies
- Distribution

4.30. Output and distribution media

- E-mail
- Fax
- Data transfer by EDI, XML or other formats
- Mobile devices, like mobile phones, PDAs
- Data media like CDs and DVDs
- Digital TV and other multimedia services

4.31. Most important ECM systems

- Autonomy
- EMC Documentum
- IBM/FileNet
- Microsoft
- Open Text/Hummingbird

- Oracle/Stellent

Figure 1. Magic Quadrant for Enterprise Content Management

4.32. Axes of Magic Quadrant

Vision

- market understanding
- marketing strategy
- sales strategy

- product strategy
- business model
- industry strategy
- innovation
- geographic strategy

Ability

- product, service
- overall viability
- pricing
- market responsiveness
- marketing execution
- customer experience
- operations

4.33. CMS plan, design, integration

- Plan of CMS functionalities
- Analyzing the content lifecycle and designing content components
- Roles: different authors
- Integration

4.34. Plan of functionalities (1)

- Document Capture: Includes import and transformation.
- Authoring: Facilities for content creation.
- Metadata Tagging: Describing the content.
- Editing: Changing and updating the content.
- Indexing: indexes of document and content items.
- Searching: search capabilities both at document and content levels.
- Viewing: content is viewed correctly in all devices.
- Collaboration: allow users to collaborate.
- Workflow: routing of content between individuals and processes.
- Database Integration.
- Security: Prevent unauthorized modification.

4.35. Plan of functionalities (2)

- Version Control: Enables the tracking of changes to documents and content, providing an audit trail, and the possibility of rollback to previous versions.
- Scheduling: Deciding when to display content.
- Templating: Allows separation of content from its presentation.
- Syndication: Allowing content to be displayed by others.
- Personalisation: Displaying content differently dependent on the visitor.
- Filtering: Ability to filter out content for publication or filter out specific users content retrieval.
- Categorisation: Break the content of documents into components, categorise them and store the relationships.
- Application Integration: Integration into existing infrastructure, suppliers, customers, partners (B2B) and third party product.

4.36. Plan of functionalities (3)

- Storage: handling a wide variety of content.
- Records Management: store and manage all records including physical records.
- Content Aggregation: different pieces of content to put together .
- Content Delivery: delivering content to users in the most efficient manner (caching).
- Multiple Web Site Management (synchronisation, localisation, branding, and content delivery).
- Platform Viewing Independence: Enables content to easily be reused and delivered to different channels (Internet, Mobile Devices and TV).

4.37. Analyzing the content lifecycle

- Product content
  - The evolution of reuse in publications
  - The role of content strategy
- Identifying content lifecycle
- Identifying the players, processes, and issues

4.38. Roles

- Content strategist
- Content owners
- External, internal authors
- Business owners
- Editors
- Information architect
- Publishing roles

4.39. CMS adopting, integration

- CMS beta testing
- CMS adopted as a solution
- Formal plan developed to migrate content into the system

4.40. Content Migration

- Schedule for the migration of current content into the CMS system
- Migration occurred in small groups
- Thousands of pages, images, and resource records were copied

4.41. Two approaches

- Integration of CMSs - This approach enables an organisation to capitalise on the strengths of different providers rather than looking to one platform to manage all unstructured information.
- Full ECM Suite (the acquisition of full ECM suites from a single vendor - This approach enables one platform to fulfill an organisation's enterprise wide needs.

5. 5 Information Retrieval (IR)

5.1. Text Databases and IR

- Text databases (document databases)
  - Large collections of documents from various sources: news articles, research papers, books, digital libraries, e-mail messages, and Web pages, library database, etc.
  - Data stored is usually semi-structured
  - Traditional search techniques become inadequate for the increasingly vast amounts of text data
- Information retrieval
  - A field developed in parallel with database systems
  - Information is organized into (a large number of) documents
  - Information retrieval problem: locating relevant documents based on user input, such as keywords or example documents

5.2. Information Retrieval (IR)

- Typical IR systems
  - Online library catalogs
  - Online document management systems
- Information retrieval vs. database systems
  - Some DB problems are not present in IR, e.g., update, transaction management, complex objects
  - Some IR problems are not addressed well in DBMS, e.g., unstructured documents, approximate search using keywords and relevance

5.3. IR vs DBMS

	DBMS	IR
match	exact	partial or best match
inference	deduction	induction
model	deterministic	probabilistic
data	record/field	text document
query language	artificial	natural?
query	complete	incomplete
items wanted	matching	relevant
error response	sensitive	insensitive

5.4. Basic Measures in the IR

- Precision: the percentage of retrieved documents that are in fact relevant to the query (i.e., "correct" responses)

$$precision = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Retrieved\}|}$$

- Recall: the percentage of documents that are relevant to the query and were, in fact, retrieved

$$recall = \frac{|\{Relevant\} \cap \{Retrieved\}|}{|\{Relevant\}|}$$

5.5. Keyword-Based Retrieval

- A document is represented by a string, which can be identified by a set of keywords
- Queries may use expressions of keywords
  - E.g., car and repair shop, tea or coffee, DBMS but not Oracle
  - Queries and retrieval can consider synonyms, e.g., repair and maintenance
- Major difficulties of the model
  - Synonymy: A keyword T does not appear anywhere in the document, even though the document is closely related to T, e.g., data mining.
  - Polysemy: The same keyword may mean different things in different contexts, e.g., mining.

5.6. Similarity-Based Retrieval in Text Databases

- Finds similar documents based on a set of common keywords
- Answer should be based on the degree of relevance based on the nearness of the keywords, relative frequency of the keywords, etc.
- Stop list
  - Set of words that are deemed "irrelevant", even though they may appear frequently
  - E.g., a, the, of, for, with, etc.
  - Stop lists may be different at different corpus
- Word stem
  - Several words are small syntactic variants of each other since they share a common word stem
  - E.g., drug, drugs, drugged

5.7. Retrieval : Ad hoc and Filtering

- Ad hoc (Search): The documents in the collection remain relatively static while new queries are submitted to the system.

- Filtering: The queries remain relatively static while new documents come into the system

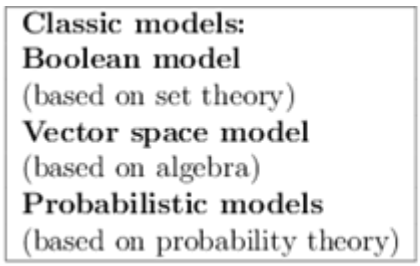
5.8. architecture

- user
- Query matching
- Learning component
- Object base
- (objects and their descriptions)
- hits
- query
- feedback

5.9. IR System and Tasks Involved

- INDEX
- INFORMATION NEED
- DOCUMENTS
- User Interface
- PERFORMANCE EVALUATION
- QUERY
- QUERY PROCESSING (PARSING and TERM PROCESSING)
- LOGICAL VIEW OF THE INFORM. NEED
- SELECT DATA FOR INDEXING
- PARSING and TERM PROCESSING

5.10. Models for IR - Taxonomy



Other models:
Fuzzy set model
Extended Boolean model
Generalized vector model
Latent semantic indexing
Neural networks
Inference networks
Belief network

5.11. Formal Specification of the Task

Definition: An information retrieval model is a quadrupel $[D, Q, F, R(q_i, d_j)]$ where

D is a set composed of logical views (or representations) for the documents in the set

Q is a set composed of logical views for the user information needs: queries.

F is a framework for modeling document representations, queries, and their relationships.

$R(q_i, d_j)$ is a ranking function which associates a real number with a query q_i in Q and a document representation d_j in D. Such ranking defines an ordering among the documents with regard to the query q_i.

5.12. Formal Specific. of the Task (Cont.)

Generally, we represent the query and documents through a set of terms $T = \{t_1, ..., t_k\}$ where k is the number of all unique index terms in the system.

We assume $w_{i,j}$ to be a weight for term t_i in document d_j with $w_{i,j} = 0$ if t_i is not in d_j.

Document d_j can be represented as an index term vector $\overrightarrow{d_j} = (w_{1,j}, w_{2,j}, ..., w_{k,j})$.

g_i represents a function for which $g_i(\overrightarrow{d_j}) = w_{i,j}$ (i.e. given a document $\overrightarrow{d_j}$, g_i delivers the weight of term t_i in $\overrightarrow{d_j}$).

5.13. Boolean Retrieval Model - Queries

Based on set theory and Boolean algebra

Documents:

Index term vector $\overrightarrow{d_j} = (w_{1,j}, ..., w_{k,j})$ with $w_{i,j} \in \{0, 1\}$

Queries:

Terms combined with AND, OR, NOT

Boolean expression in disjunctive normal form (DNF)

A query $\overrightarrow{q}$ is defined as a Boolean expression $\overrightarrow{q}_{dnf}$ in DNF with $\overrightarrow{q}_{cc}$ being the conjunctive elements from $\overrightarrow{q}_{dnf}$.

$w_{i,j} = 0$ or 1 are the index term weight variables.

5.14. Boolean Retr. Model - Definition

We define the similarity sim of a document d_j with query q as

$$sim(d_j,q) = \begin{cases} 1 \text{ if } \exists \vec{q}_{cc} \mid (\vec{q}_{cc} \in \vec{q}_{dnf}) \wedge (\forall t_i, g_i(\vec{d}_j) = g_i(\vec{q}_{cc})) \\ 0 \text{ otherwise} \end{cases}$$

(A document is considered relevant if $sim = 1$ and irrelevant otherwise)

5.15. Boolean Retrieval Model

Advantages:

Precise, clean formalism

Offers great control and transparency,

Simplicity, easy math, easy implementation

Good for domains with ranking by other

means than relevance, i.e. chronological

Disadvantages:

Query might be hard to specify

Binary decision (relevant or not)

Often too many or too few results

5.16. Vector Model - Definition

Based on vector algebra

Formal Definition:

$w_{i,q}$ is defined as the weight associated with the pair (t_i, q) and $w_{i,q} = 0$ or > 0

k describes the number of all unique index terms

With this, we can define

Query vector $\vec{q} = (w_{1,q}, w_{2,q}, \ldots, w_{k,q})$

Document vector $\vec{d_j} = (w_{1,j}, w_{2,j}, \ldots, w_{k,j})$

5.17. Vector Model - Similarity

We should define similarity.

Using the inner product (arithmetical):

$$\vec{x}\vec{y} = x_1 y_1 + x_2 y_2 + \ldots + x_n y_n$$

Using the cosinus of the angle between the 2 vectors

$$sim(d_m, q) = \frac{\vec{d_m} \bullet \vec{q}}{|\vec{d_m}| \times |\vec{q}|} = \frac{\sum_{i=1}^{x} w_{i,m} \times w_{i,q}}{\sqrt{\sum_{i=1}^{x} w_{i,m}^2} \times \sqrt{\sum_{i=1}^{x} w_{i,q}^2}}$$

Weights: Often $TF*IDF$ (or variants of it)

5.18. Vector Model

Advantages:

Fast and easy,

Finds similar documents (no binary decision),

Ranking based on similarity

Often better results than Boolean search

(because of the term weighting)

Disadvantages:

Terms are assumed to be independent

5.19. Weights

- How are the weights w_{ij} obtained? Many variants.

One way: TF-IDF balance

- TF: Term frequency
  - How well the term is related to the doc?
- IDF: Inverse document frequency
  - How important is the term to distinguish documents?
- Contradictory. How to balance?

5.20. TF-IDF ranking

- TF: Term frequency

$$tf_{i,j} = \frac{n_{ij}}{\sum_i n_{ij}}$$

- IDF: Inverse documentfrequency

$$idf_i = \log\left(\frac{N}{df_i}\right)$$

- Balance: $TF \times IDF$

$$w_{i,j} = tf_{i,j} \times idf_i = tf_{i,j} \times \log(N / df_i)$$

5.21. Classic Probabilistic Model (first)

- Also called binary independence retrieval (BIR) model
- Idea: Given a user query q, and the ideal answer set of the relevant documents, the problem is to specify the properties for this set.
  - i.e. the probabilistic model tries to estimate the probability that the user will find the document d_j relevant with ratio $P(d_j$ relevant to $q)/P(d_j$ nonrelevant to $q)$

5.22. Classic Probabilistic Model

- An initial set of documents is retrieved somehow
- User inspects these docs looking for the relevant ones (in truth, only top 10-20 need to be inspected)
- IR system uses this information to refine description of ideal answer set
- By repeting this process, it is expected that the description of the ideal answer set will improve

5.23. Classic Probabilistic Model

- Definition

- All index term weights are all binary i.e., $w_{i,j} \in \{0, 1\}$
- Let R be the set of documents know to be relevant to query q
- Let $\overline{R}$ be the complement of R
- Let $P(R|d_j)$ be the probability that the document d_j is relevant to the query q
- Let $P(\overline{R}|d_j)$ be the probability that the document d_j is nonelevant to query q

5.24. Classic Probabilistic Model

- The similarity $sim(d_j, q)$ of the document d_j to the query q isdefined as the ratio

$$sim(\vec{d}_j, q) = \frac{\Pr(R \mid \vec{d}_j)}{\Pr(\overline{R} \mid \vec{d}_j)}$$

- Using Bayes' rule,

$$\mathrm{sim}(\vec{d}_j, q) = \frac{P(\vec{d}_j \mid R) \times P(R)}{P(\vec{d}_j \mid \overline{R}) \times P(\overline{R})}$$

- $P(R)$ stands for the probability that a document randomly selected from the entire collection is relevant
- $P(d_j|R)$ stands for the probability of randomly selecting the document d_j from the set R of relevant documents

5.25. Classic Probabilistic Model

$$sim\ (\vec{d}_j, q) \approx \log \frac{\Pr(\vec{d}_j \mid R)}{\Pr(\vec{d}_j \mid \overline{R})} + \log \frac{\Pr(R)}{\Pr(\overline{R})}$$

- Assuming independence of index terms and given $q = (d_1, d_2, \ldots, d_t)$,

$$\Pr(\vec{d}_j \mid R) = \prod_{i=1}^{t} \Pr(k_i = d_i \mid R)$$

$$\Pr(\vec{d}_j \mid \overline{R}) = \prod_{i=1}^{t} \Pr(k_i = d_i \mid \overline{R})$$

$$sim(\vec{d}_j, q) \approx \log \frac{\prod_{i=1}^{t} \Pr(k_i = d_i \mid R)}{\prod_{i=1}^{t} \Pr(k_i = d_i \mid \overline{R})}$$

5.26. Classic Probabilistic Model

- $P(k_i|R)$ stands for the probability that the index term k_i is present in a document randomly selected from the set R
- $P(\overline{k_i}|R)$ stands for the probability that the index term k_i is not present in a document randomly selected from the set R

5.27. Classic Probabilistic Model

$$\mathrm{sim}\ (\vec{d}_j, q) \approx \frac{\prod_{g_i(d_j)=1} P(k_i \mid R) \prod_{g_i(d_j)=0} P(\overline{k}_i \mid R)}{\prod_{g_i(d_j)=1} P(k_i \mid \overline{R}) \prod_{g_i(d_j)=0} P(\overline{k}_i \mid \overline{R})}$$

$$P(k_i \mid R) + P(\overline{k_i} \mid R) = 1$$

$$sim(d_j, q) \approx \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i \mid R)}{1 - P(k_i \mid R)} + \log \frac{1 - P(k_i \mid \overline{R})}{P(k_i \mid \overline{R})} \right)$$

5.28. Estimation of Term Relevance

In the very beginning:

$$P(k_i \mid R) = 0.5$$

$$P(\overline{k_i} \mid R) = \frac{df_i}{N}$$

Next, the ranking can be improved as follows:

$$P(k_i \mid R) = \frac{V_i}{V}$$

$$P(\overline{k_i} \mid R) = \frac{df_i - V_i}{N - V}$$

For small values for V

$$P(k_i \mid R) = \frac{V_i + 0.5}{V + 1}$$

$$P(\overline{k_i} \mid R) = \frac{df_i - V_i + 0.5}{N - V + 1}$$

$$P(k_i \mid R) = \frac{V_i + \frac{df_i}{N}}{V + 1}$$

$$P(\overline{k_i} \mid R) = \frac{df_i - V_i + \frac{df_i}{N}}{N - V + 1}$$

Let V be a subset of the documents initially retrieved

5.29. (Dis) advantages of Classic Probabilistic model (last)

Advantage:

- Theoretical adequacy: ranks by probabilities

Disadvantages:

- Need to guess the initial ranking
- Binary weights, ignores frequencies
- Independence assumption

5.30. Summary: taxonomy of IR models

- U
- S
- E
- R
- T
- A
- S
- K
- Retrieval:
- Ad hoc
- Filtering
- Classic Models
- Browsing
- Boolean
- Vector
- Probabilistic
- Fuzzy
- Extended Boolean
- Set Theoretic
- Algebraic
- Generalized Vector
- Lat. Semantic Index
- Neural Networks
- Inference Network
- Belief Network
- Probabilistic

5.31. Alternative Probabilistic Models

- Inference Network Model
- Belief Network Model

5.32. Bayesian Network

- Let x_i be a node in a Bayesian network G and Γ_{x_i} be the set of parent nodes of x_i.
- The influence of Γ_{x_i} on x_i can be specified by any set of functions that satisfy:

$$\sum_{\forall x_i} F_i(x_i, \Gamma_{x_i}) = 1$$

$$0 \leq F_i(x_i, \Gamma_{x_i}) \leq 1$$

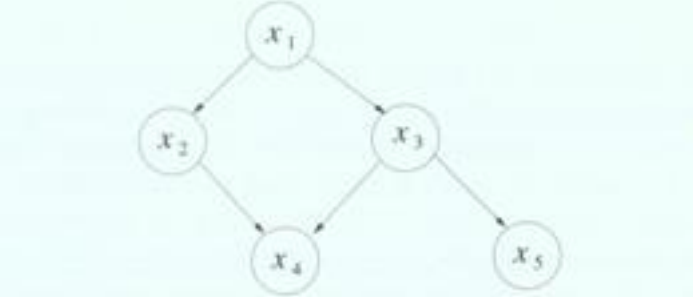


Figure 2.8 An example of a Bayesian network.

- $P(x_1, x_2, x_3, x_4, x_5) = P(x_1)P(x_2|x_1)P(x_3|x_1)P(x_4|x_2, x_3)P(x_5|x_3)$

5.33. Belief Network Model (1/6)

- The probability spaceThe set $K = k_1, k_2, \ldots, k_t$ is the universe. To each subset u is associated a vector such that $g_i() = 1 \Leftrightarrow k_i \in u$.
- Random variables
  - To each index term k_i is associated a binary random variable.

5.34. Belief Network Model (2/6)

- Concept space
  - A document d_j is represented as a concept composed of the terms used to index d_j.
  - A user query q is also represented as a concept composed of the terms used to index q.
  - Both user query and document are modeled as subsets of index terms.
- Probability distribution P over K

$$P(c) = \sum_{u} P(c \mid u) \times P(u)$$

$$P(u) = (\frac{1}{2})^t$$

5.35. Belief Network Model (3/6)

- A query is modeled as a network node
  - This variable is set to 1 whenever q completely covers the concept space K
  - $P(q)$ computes the degree of coverage of the space K by q
- A document d_j is modeled as a network node
  - This random variable is 1 to indicate that d_j completely covers the concept space K
  - $P(d_j)$ computes the degree of coverage of the space K by d_j

5.36. Belief Network Model (4/6)

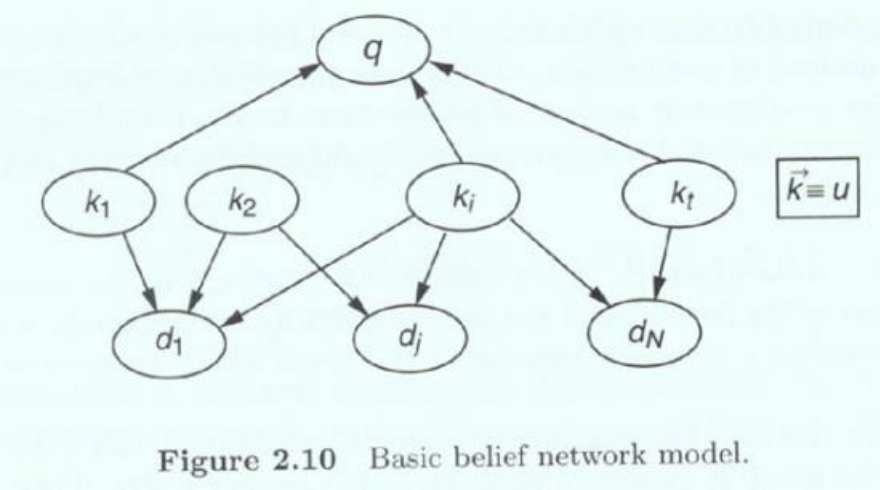


Figure 2.10 Basic belief network model.

5.37. Belief Network Model (5/6)

- Assumption
  - $P(d_j|q)$ is adopted as the rank of the document d_j with respect to the query q.

$$P(d_j \mid q) = P(d_j \wedge q) / P(q)$$
$$\approx P(d_j \wedge q) \approx \sum_{\forall u} P(d_j \wedge q \mid u) \times P(u)$$
$$\approx \sum_{\forall u} P(d_j \mid u) \times P(q \mid u) \times P(u)$$
$$\approx \sum_{\forall \vec{k}} P(d_j \mid \vec{k}) \times P(q \mid \vec{k}) \times P(\vec{k})$$

5.38. Belief Network Model (6/6)

- Specify the conditional probabilities as follows:

$$P(d_j \mid \vec{k}) = \begin{cases} \frac{w_{i,j}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2}} & \text{if } \vec{k} = \vec{k}_i \quad \wedge g_i(d_j) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$P(q \mid \vec{k}) = \begin{cases} \frac{w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,q}^2}} & \text{if } \vec{k} = \vec{k}_i \quad \wedge g_i(q) = 1 \\ 0 & \text{otherwise} \end{cases}$$

- Thus, the belief network model can be tuned to subsume the vector model.

5.39. Summary

- Belief network model
  - Belief network model is based on set-theoretic view
  - Belief network model provides a separation between the document and the query
  - Belief network model is able to reproduce any ranking strategy generated by the inference network model

5.40. Alternative Set Theoretic models

Extended Boolean model

- Combination of Boolean and Vector
- In comparison with Boolean model, adds "distance from query"
  - some documents satisfy the query better than others

- In comparison with Vector model, adds the distinction between AND and OR combinations
- There is a parameter (degree of norm) allowing to adjust the behavior between Boolean-like and Vector-like
- This can be even different within one query
- Not investigated well. Why not investigate it?

5.41. Extended Boolean logic

considering the space composed of two terms kx and ky only.

- ky
- ky
- kx
- kx

5.42. Extended Boolean Model:

- For query $q = K_x$ or $K_y, (0,0)$ is the point we try to avoid.Thus, we can use

$$sim(q_{or}, d) = \sqrt{\frac{x^2 + y^2}{2}}$$

  to rank the documents

- The bigger the better.

5.43. Extended Boolean Model:

- For query $q = K_x$ and $K_y, (1,1)$ is the most desirable point.
- We use

$$sim(q_{and}, d) = 1 - \sqrt{\frac{(1-x)^2 + (1-y)^2}{2}}$$

  to rank the documents.

- The bigger the better.

5.44. Extend the idea to m terms

- $q_{or} = k_1 \vee^p k_2 \vee^p \cdots \vee^p k_m$

$$sim(q_{or}, d_j) = \left(\frac{x_1^p + x_2^p + \ldots + x_m^p}{m} \right)^{1/p}$$

- $q_{and} = k_1 \wedge^p k_2 \wedge^p \cdots \wedge^p k_m$

$$sim(q_{and}, d_j) = 1 - \left(\frac{(1-x_1)^p + (1-x_2)^p + \ldots + (1-x_m)^p}{m} \right)^{1/p}$$

5.45. Properties:

- The p norm as defined above enjoys a couple of interesting properties as follows. First, when $p = 1$ it can be verified that

$$sim(q_{or}, d_j) = sim(q_{and}, d_j) = \frac{x_1 + \ldots + x_m}{m}$$

- Second, when $p =^{\circ}$ it can be verified that
- $Sim(q_{or}, d_j) = \max(x_i)$
- $Sim(q_{and}, d_j) = \min(x_i)$

5.46. Example:

- For instance, consider the query $q = (k_1 \wedge k_2) \vee k_3$. The similarity $sim(q, d_j)$ between a document d_j and this query is then computed as

$$sim(q,d) = \left(\frac{\left(1 - \left(\frac{(1-x_1)^p + (1-x_2)^p}{2}\right)^{1/p}\right)^p + x_3^p}{2} \right)^{1/p}$$

- Any boolean can be expressed as a numeral formula.

6. 6 Information Retrieval on Web

6.1. Mining the World-Wide Web

- The WWW is huge, widely distributed, global information service center for
  - Information services
  - Hyper-link information
  - Access and usage information
- WWW provides rich sources for data mining
- Challenges
  - Too huge for effective data warehousing and data mining
  - Too complex and heterogeneous: no standards and structure
  - Only a small portion of the information on the Web is truly relevant or useful

6.2. Web search engines

- Index-based: search the Web, index Web pages, and build and store huge keyword-based indices
- Help locate sets of Web pages containing certain keywords
- Deficiencies
  - A topic may easily contain thousands of documents
  - Many documents that are highly relevant to a topic may not contain keywords defining them (polysemy)

6.3. Web Mining: A more challenging task

- Searches for
  - Web access patterns
  - Web structures
  - Regularity and dynamics of Web contents
- Problems
  - The "abundance" problem
  - Limited coverage of the Web: hidden Web sources, majority of data in DBMS
  - Limited query interface based on keyword-oriented search
  - Limited customization to individual users

6.4. Web Mining Taxonomy

Mining Mining Content Mining Mining Mining Pattern Tracking Usage Tracking

6.5. Mining the World-Wide Web

- Web Structure
- Mining
- Web Content
- Mining
- Web Page Content Mining
- Web Page Summarization
- WebLog (Lakshmanan et.al.), WebOQL(Mendelzon et.al.) â€¦:
- Web Structuring query languages;
- Can identify information within given web pages
- Ahoy! (Etzioni et.al.):Uses heuristics to distinguish personal home pages from other web pages
- ShopBot (Etzioni et.al.): Looks for product prices within web pages
- Search Result
- Mining
- Web Usage
- Mining
- General Access
- Pattern Tracking

- Customized
- Usage Tracking

6.6. Mining the World-Wide Web

- Web Usage
- Mining
- General Access
- Pattern Tracking
- Customized
- Usage Tracking
- Web Structure
- Mining
- Web Content
- Mining
- Web Page
- Content Mining
- Search Result Mining
- Search Engine Result Summarization
- Clustering Search Result (Leouski and Croft" Zamir and Etzioni,):
- Categorizes documents using phrases in titles and snippets

6.7. Mining the World-Wide Web

- Web Content
- Mining
- Web Page
- Content Mining
- Search Result
- Mining
- Web Usage
- Mining
- General Access
- Pattern Tracking
- Customized

- Usage Tracking
- Web Structure Mining
- Using Links
- PageRank (Brin et al.)
- CLEVER (Chakrabarti et al.,)
- Use interconnections between web pages to give weight to pages.
- Using Generalization
- MLDB , VWV
- Uses a multi-level database representation of the Web. Counters (popularity) and link lists are used for capturing structure.

6.8. Mining the World-Wide Web

- Web Structure
- Mining
- Web Content
- Mining
- Web Page
- Content Mining
- Search Result
- Mining
- Web Usage
- Mining
- General Access Pattern Tracking
- Web Log Mining (Za $\ddot{i}$ ane, Xin and Han)
- Uses KDD techniques to understand general access patterns and trends.
- Can shed light on better structure and grouping of resource providers.
- Customized
- Usage Tracking

6.9. Mining the World-Wide Web

- Web Usage
- Mining
- General Access

- Pattern Tracking
- Customized Usage Tracking
- Adaptive Sites (Perkowitz and Etzioni)
- Analyzes access patterns of each user at a time.
- Web site restructures itself automatically by learning from user access patterns.
- Web Structure
- Mining
- Web Content
- Mining
- Web Page
- Content Mining
- Search Result
- Mining

6.10. Mining the World-Wide Web

- Design of a Web Log Miner
  - Web log is filtered to generate a relational database
  - A data cube is generated form database
  - OLAP is used to drill-down and roll-up in the cube
  - OLAM is used for mining interesting knowledge
- 1
- Data Cleaning
- 2
- Data Cube Creation
- 3
- OLAP
- 4
- Data Mining
- Web log
- Database
- Data Cube
- Sliced and diced cube
- Knowledge

6.11. Multilayered Web Information Base

- $Layer_0$: the Web itself
- $Layer_1$: the Web page descriptor layer
  - Contains descriptive information for pages on the Web
  - An abstraction of $Layer_0$: substantially smaller but still rich enough to preserve most of the interesting, general information
  - Organized into dozens of semistructured classes
    - document, person, organization, ads, directory, sales, software, game, stocks, $library_catalog$, $geographic_data$, $scientific_data$, etc.
- $Layer_2$ and up: various Web directory services constructed on top of $Layer_1$
  - provide multidimensional, application-specific services

6.12. Multiple Layered Web Architecture

- Generalized Descriptions
- More Generalized Descriptions
- $Layer_0$
- $Layer_1$
- $Layer_n$
- ...

6.13. Mining the World-Wide Web

Layer-0: Primitive data

Layer-1: dozen database relations representing types of objects (metadata)

document, organization, person, software, game, map, image, ...

- document ($file_addr$, authors, title, publication, $publication_date$, abstract, language, $table_of_contents$, $category_description$, keywords, index, $multimedia_attached$, num_pages, format, $first_paragraphs$, $size_doc$, timestamp, $access_frequency$, $links_out$, ...)
- person ($last_name$, $first_name$, $home_page_addr$, position, $picture_attached$, phone, e-mail, $office_address$, education, $research_interests$, publications, $size_of_home_page$, timestamp, $access_frequency$, ...)
- image ($image_addr$, author, title, $publication_date$, $category_description$, keywords, size, width, height, duration, format, $parent_pages$, $colour_histogram$, $Colour_layout$, $Texture_layout$, $Movement_vector$, $localisation_vector$, timestamp, $access_frequency$, ...)

6.14. Mining the World-Wide Web

Layer-2: simplification of layer-1

- doc_brief ($file_addr$, authors, title, publication, $publication_date$, abstract, language, $category_description$, key_words , $major_index$, num_pages , format, $size_doc$, $access_frequency$, $links_out$)
- $person_brief$ ($last_name$, $first_name$, publications,affiliation, e-mail, $research_interests$, $size_home_page$, $access_frequency$)

Layer-3: generalization of layer-2

- cs_doc ($file_addr$, authors, title, publication, $publication_date$, abstract, language, $category_description$, keywords, num_pages , form, $size_doc$, $links_out$)
- $doc_summary$ (affiliation, field, $publication_year$, count, $first_author_list$, $file_addr_list$)
- doc_author_brief ($file_addr$, authors, affiliation, title, publication, pub_date , $category_description$, keywords, num_pages , format, $size_doc$, $links_out$)
- $person_summary$ (affiliation, $research_interest$, year, $num_publications$, count)

6.15. Benefits of Multi-Layer Meta-Web

- Benefits:
  - Multi-dimensional Web info summary analysis
  - Approximate and intelligent query answering
  - Web high-level query answering (WebSQL, WebML)
  - Web content and structure mining
  - Observing the dynamics/evolution of the Web
- Is it realistic to construct such a meta-Web?
  - Benefits even if it is partially constructed
  - Benefits may justify the cost of tool development, standardization and partial restructuring

6.16. Web Search Topics

- Web Search
  - Architecture
  - Crawling
  - Search engines
  - Link analysis

6.17. Search on the Web

- Corpus:The publicly accessible Web: static + dynamic
- Goal: Retrieve high quality results relevant to the user's need
  - (not docs!)
- Need
  - Informational - want to learn about something (40%)
  - Navigational - want to go to that page (25%)
  - Transactional - want to do something (web-mediated) (35%)
    - Access a service
    - Downloads
    - Shop
  - Grey areas
    - Find a good hub
    - Exploratory search "see what's there"

Bank information systems

National Airlines

Weather forecast

BUX data

Nikon CoolPix

Bank credits

6.18. Classic IR vs. Web Search

Classic IR: Generally a 2-step process

- Select subset of relevant documents
- Rank / sort by estimated relevance

Problems with web search:

- Size of the WWW
- Heterogeneity (length, quality, format, . . .)
- Not always self descriptive (e.g. search engine)
- Can be manipulated (spam)
- Ill-defined queries

Therefore: Pure content based ranking critical

6.19. General Web Search Engine Architecture

- CLIENT
- QUERY ENGINE
- RANKING
- CRAWL CONTROL
- CRAWLER(S)
- USAGE FEEDBACK
- RESULTS
- QUERIES
- WWW
- COLLECTION ANALYSIS MOD.
- INDEXER MODULE
- PAGE REPOSITORY
- INDEXES
- STRUCTURE
- UTILITY
- TEXT

6.20. The Indexer Module

Creates Two indexes :

- Text (content) index : Uses "Traditional" indexing methods like Inverted Indexing
- Structure(Links) index : Uses a directed graph of pages and links. Sometimes also creates an inverted graph

6.21. Storage Issues

- Scalability and seamless load distribution
- Dual access modes
  - Random access (used by the query engine for cached pages)
  - Streaming access (used by the Indexer and Collection Analysis)
- Large bulk update - reclaim old space, avoid access/update conflicts
- Obsolete pages - remove pages no longer on the web

6.22. Update Strategies

- Updates are generated by the crawler

- Several characteristics
  - Time in which the crawl occurs and the repository receives information
  - Whether the crawl's information replaces the entire database or modifies parts of it

6.23. Batch vs. Steady

- Batch mode
  - Periodically executed
  - Allocated a certain amount of time
- Steady mode
  - Run all the time
  - Always send results back to the repository

6.24. Partial vs. Complete Crawls

- A batch mode crawler can
  - Do a complete crawl every run, and replace entire collection
  - Recrawl only a specific subset, and apply updates to the existing collection - partial crawl
- The repository can implement
  - In place update
    - Quickly refresh pages
  - Shadowing, update as another stage
    - Avoid refresh-access conflicts

6.25. 1^{st} Generation Web Search Engines

Use only "on page" text data

1995-1997 (AltaVista, Excite, Lycos, etc.)

- Extended Boolean model
  - Matches: Exact, prefix, phrase, . . .
  - Operators: AND, OR, AND NOT, NEAR, . . .
  - Fields: TITLE, URL, HOST, . . .
  - AND is somewhat easier to implement, maybe preferable as default for short queries
- Ranking
  - TF like factors: TF, explicit keywords, words in the title, explicit emphasis (headers), etc.
  - IDF factors: IDF, total word count in corpus, frequency in query log, frequency in language

6.26. The evolution of search engines

1^{st} generation: Use only "on page", text data

- Word frequency, language

1995-1997 (AltaVista, Excite, Lycos, etc.)

2^{nd} gen.: Use off-page, web-specific data

- Link (or connectivity) analysis
- Click-through data (what results people click on)
- Anchor-text (how people refer to a page)

From 1998 (made popular by Google but everyone now)

6.27. 2^{nd} generation web IR ranking criteria

- Extended Boolean model
- Content-based techniques (variants of term vector or probabilistic model)
- Ad-hoc factors (anti-porn heuristics, publication/location data, url depth, ...)
- Human annotations (in Yahoo, ODP, etc.)
- Popularity: Click-through feedback ("People vote with their clicks", DirectHit)
- Anchor text
- Connectivity-based techniques

6.28. Considering the link structure

Idea: Use link structure to get a content-independent feature to estimate relevance

Why? Because a link from page A to page B

- Often represents a relationship of contents
- Can be interpreted as a reference or recommendation

Compare citations in scientific literature

- Articles with lots of citations:Considered to have a high scientific value(Bibliometric laws)

6.29. Considering the link structure

First approach: Use the number of backlinks as a measure for a page's relevance

Problem: web vs. scientific publications

- Publications: rigorous review process, high quality, well-defined length and style, same goal, similar number of citations, etc.

- Web pages: no quality control, no publication costs, can be manipulated (self-citations), large variance regarding quality, length, style and intended usage.

Idea: Consider reputation and credibility of a link, i.e. the quality and importance of a web page

6.30. Simple PageRank - Definition

Goal: Measure for the importance and quality of a web page

Definition: Simple PageRank

- Assume the web is a strongly connected graph
- Define the Simple PageRank R of a page p as

$$R(p) = \sum_{q \in B_p} \frac{R(q)}{N_q}$$

6.31. Simple PageRank - Example

- R(1) = 0,286
- R(2) = 0,286
- R(3) = 0,143
- R(5) = 0,143
- R(4) = 0,143
- 1
- 2
- 3
- 5
- 4

6.32. Simple PageRank - Calculation

Mathematical representation:

If $\underline{R} = mx1$ vector $[R(1), R(2), \ldots, R(m)]$

and $A =$ matrix with a_{ij}

$= 1/N(j)$ if link from i to j exists

$= 0$ otherwise

Then $\underline{R} = A\ \underline{R}$

$$R(p) = \sum_{q \in B_p} \frac{R(q)}{N_q}$$

Calculation:

- Set of linear equations, but direct calculation of a solution too expensive
- But: R is eigenvector of A (with eigenvalue 1)
- Hence: Efficient approaches for iterative calculation exist (e.g. power iteration)

6.33. Problems if graph is not strongly connected

Page Sink = group of pages that link to each other but have no outside links Page Leak = single page with no outside links (special case of a page sink)

6.34. What happens with Leaks?

- 1
- 2
- 3
- 4

$$R(p) = \sum_{q \in B_p} \frac{R(q)}{N_q}$$

Possible Solutions:

- Remove all Leaks before the PageRank is calculated
- Or: Insert back links for each forward link to a Leak

6.35. What happens with Sinks?

- 1
- 2
- 3
- 5
- 4

Possible Solutions:

- Introduce a damping factor d

$$\mathbf{R(p) = d \cdot \sum_{q \in B(p)} \frac{R(q)}{Nq} + (1-d) \cdot \frac{1}{m}}$$

6.36. Intuitive Interpretation

The Random Surfer Model: A random surfer performs a random walk on the web graph.

- Surfer on page $q->$ probability to click on any of the links $= 1/N_q$
- Probability to get to page p: Sum of the probabilities of all clicks from page q to p (q in B_p) times the probability of being on page q.
- The damping factor d represents the probability of going to a random new page.

6.37. PageRank - conclusion

PageRank = Quality or importance measure for a web page

Harder to manipulate than pure backlinks.

Content and query independent.

Ranking:

- The higher the PageRank,
- the higher a page's quality or importance,
- the higher the page's relevance

6.38. PageRank in the 1^{st} Google engine

In practice: Combination with other features

- Hits classified in different types (anchor, title, ...), represented by a vector of type-weights.
- Term distribution represented by a vector of count-weights.
- Combination of vectors $= IR$ score
- Final rank $=$ combination of IR score and PageRank

6.39. HITS: (Hyperlink-Induced Topic Search)

- A query dependent technique
- Produces two scores:
  - Authority: A most likely to be relevant page to a given query
  - Hub: Points to many Authorities
- Contains two part:
  - Identifying the focused subgraph
  - Link analysis

6.40. HITS

- Use an index-based search engine to form the root set

- Many of these pages are presumably relevant to the search topic
- Some of them should contain links to most of the prominent authorities

- Expand the root set into a base set
  - Include all of the pages that the root-set pages link to, and all of the pages that link to a page in the root set
- Apply weight-propagation
  - An iterative process that determines numerical estimates of hub and authority weights

6.41. Systems Based on HITS

- Output a short list of the pages with large hub weights, and the pages with large authority weights for the given search topic
- Systems based on the HITS algorithm
  - Clever: achieve better quality search results than those generated by term-index engines such as AltaVista and those created by human ontologists such as Yahoo!

6.42. HITS: Identifying The Focused Subgraph

Subgraph creation from t-sized page set:

1. $R \leftarrow$ t initial pages

2. $S \leftarrow R$

3. for each page $p \in R$

(a) Include all the pages that p points to in S

(b) Include (up to maximum d) all pages that points to p in S

4. S holds the focused graph

(d reduces the influence of extremely

popular pages like yahoo.com)

6.43. HITS: Link Analysis

- Calculates Authorities and Hubs scores ($a_i and h_i$) for each page in S

1. Initialize a_i, h_i arbitrarily.

2. Repeat until convergence

(a) $a_i = \sum_{j \in B(i)} h_j$ (for all pages)

(b) $h_i = \sum_{j \in F(i)} a_j$ (for all pages)

6.44. HITS:Link Analysis Computation

- Eigenvectors computation can be used by:

$$\left.\begin{aligned} a &= Ah \\ h &= A^{tr} a \end{aligned}\right\} \Rightarrow \begin{aligned} a &= AA^{tr} a \\ h &= A^{tr} Ah \end{aligned}$$

Where a : Vector of Authorities' scores h : Vector of Hubs' scores A : Adjacency matrix in which $a_{i,j} = 1$ if page i points to j

6.45. Research issues

- How people judge relevance?
  - ranking strategies
- What interfaces can help users to understand and formulate their Information Need?
  - user interfaces: an open issue
- Meta-search engines: combine results from different Web search engines
  - They almost do not intersect
  - How to combine ranking?

6.46. Summary of previous lessons: taxonomy of IR models

- U
- S
- E
- R
- T
- A
- S
- K
- Retrieval:
- Ad hoc
- Filtering
- Classic Models
- Browsing
- Boolean
- Vector
- Probabilistic
- Fuzzy
- Extended Boolean
- Set Theoretic
- Algebraic
- Generalized Vector
- Lat. Semantic Index
- Neural Networks
- Inference Network
- Belief Network
- Probabilistic

6.47. Search Engine Marketing

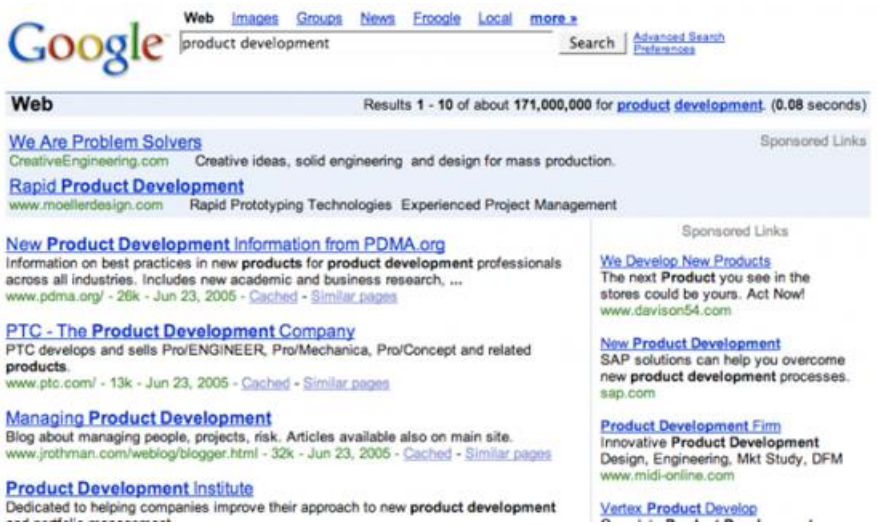
Search Engine Marketing (SEM): the entire set of techniques and strategies used to direct more visitors from search engines to marketing web sites, including all of the tactics and strategies defined below.

- PPC (or Paid Placement): Text ads targeted to keyword search results on search engines, through programs such as Google AdWords and Yahoo Search "Precision Match".
- SEO: Search Engine Optimization

Why is Search Engine Marketing important?

- 80% of Internet user sessions begin at the search engines(Source: Internetstats.com)

6.48. PPC and SEO



6.49. SEO: Search Engine Optimization

Definition

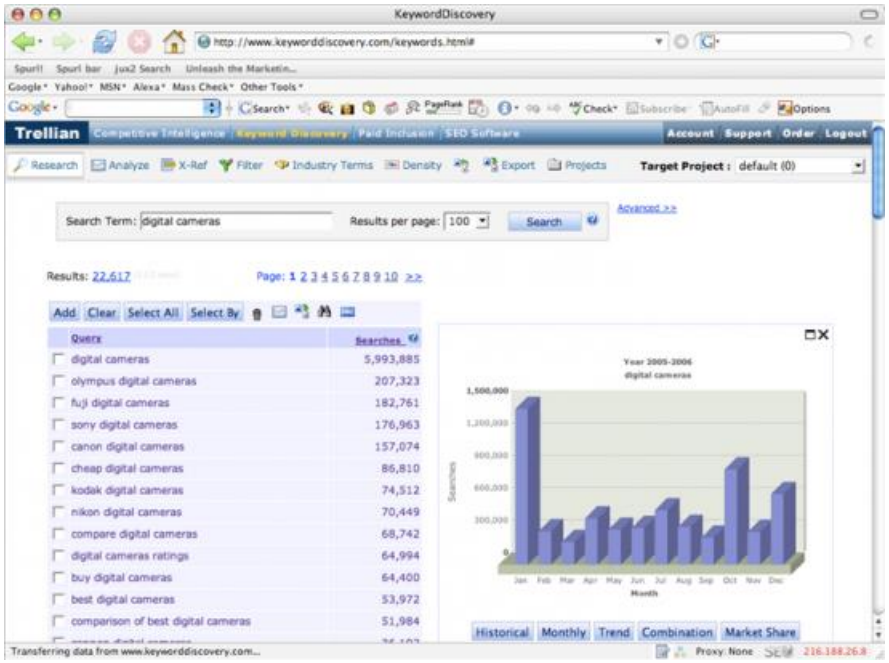
- Refers to the process of "optimizing" both the on-page and off-page ranking factors in order to achieve high search engine rankings for targeted search terms.
- Refers to the "industry" that has been created by keyword searching in order to increase relevant traffic to a website

6.50. PPC vs. "Natural" SEO

Pay-Per-Click	„Natural" SEO
results in 1-2 days	results take 2 weeks to 4 months
easier for a novice or one little knowledge of SEO	requires ongoing learning and experience to achieve results
ability to turn on and off at any moment	very difficult to control flow of traffic
generally more costly per visitor and per conversion	generally more cost-effective, does not penalize for more traffic
fewer impressions and exposure	SERPs are more popular than sponsored ads
easier to compete in highly competitive market space (but it will cost you)	very difficult to compete in highly competitive market space
ability to target „local" markets	more difficult to target local markets
better for short-term and high-margin campaigns	better for long-term and lower margin campaigns

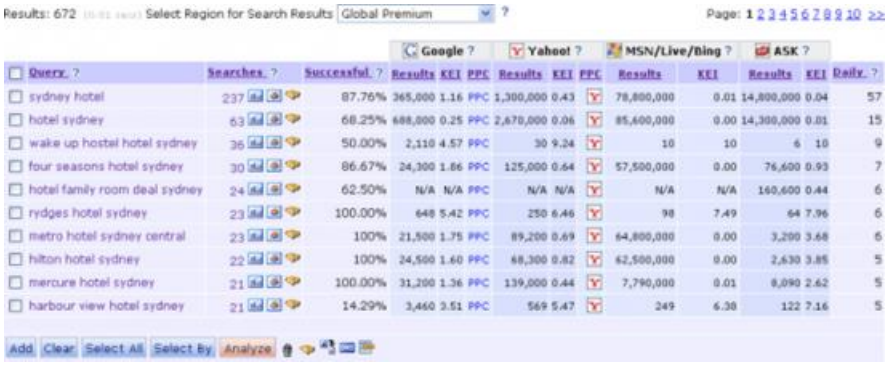
6.51. What Are Searchers Looking For?

- Keyword Research
  - "Target the wrong keywords and all your efforts will be in vain."
- The "right" keywords are. . .
  - relevant to your business
  - popular with searchers
- Competition for that keyword should also be considered
  - Calculate KEI Score (Keyword Effectiveness Indicator) = ratio of searches over number of page in search results
  - The higher the KEI Score, the more attractive the keyword is to target (assuming it's relevant to your business)



Keyword Popularity - According to KeywordDiscovery

6.52. KEI values at different search engines



6.53. SEO techniques: White Hat vs. Black Hat

White Hat

- Quality content creation
- Light keyword optimization
- Sound website building tactics
- Everything out in the open Nothing designed to trick the search engine or the user

Black Hat

- Hidden text or tiny text
- Cloaking
- Keyword stacking or stuffing
- Use non-related keywords
- Duplicate pages/ duplicate content
- Domain spam / cybersquatting
- Re-directs
- Page swapping

6.54. Black Hat (Spamming)

- Pagejacking
- Competitor names in meta tags
- Doorway pages
- Submitting to FFA ("Free For All") sites and link farms
- Buying up expired domains with high PageRanks
- Splogging (spam blogging)

6.55. White Hat: The Seven Steps to Higher Rankings

- Get Your Site Fully Indexed
- Get Your Pages Visible
- Build Links and PageRank
- Leverage Your PageRank
- Encourage Clickthrough
- Track the Right Metrics
- Avoid Worst Practices

6.56. 1 Get Your Site Fully Indexed

- Search engines are wary of "dynamic" pages - they fear "spider traps".
- Avoid stop characters (?, $and, =$) 'cgi-bin', session IDs and unnecessary variables in your URLs; frames; redirects; pop-ups; navigation in Flash/Java/Javascript/pulldown boxes.
- The better your PageRank, the deeper and more often your site will be spidered.

[fragile]

6.57. 2 Get Your Pages Visible

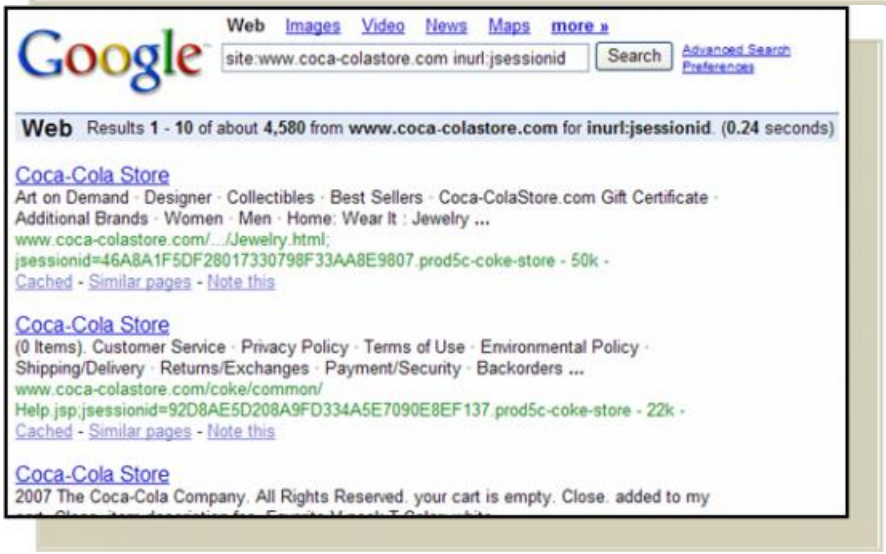
- "Title tag" is the most important content on the page
- Home page is the most important page of a site
- Incorporate keywords into title tags, hyperlink text, headings ($H1 and H2$ tags), alt tags, and high up in the page
- "Meta tags" are not a magic bullet

```
<meta name="description" content="Free Web tutorials on HTML, CSS, XML, and XHTML">
```

- Different description meta tag
- Have text for navigation, not graphics

6.58. Dynamic Site Leaves Session IDs in URLs:

Titles not specific to page ("jewelry")



Main text on page just nav labels -

Little text/keyword content

[fragile]

6.59. Optimize the use of Images

- Use brief, but descriptive filenames and alt text
- Supply alt text when using images as links
- Store images in a directory of their own
- Use commonly supported file types

Alt tags: use for graphics

```
<IMG src="star.gif" alt=,,star logo">
```

6.60. 3 Build Links and PageRank

- "Link popularity" affects search engine rankings
- PageRankTM - Links from "important" sites have more impact on your Google rankings(weighted link popularity)
- Scores range from $0 - 10$ on a logarithmic scale
- Live Search and Yahoo have similar measures to PageRankTM

6.61. Ideal internal site linking hierarchies:

Homepages often will be highest-ranking site pages since they typically have most inbound links. Good link trees inform search engines about which site pages are most important.

$*$ Sitemaps can also be used to tell SEs about pages, and to define relative priority.

6.62. 4 Leverage Your PageRank

- Your home page's PageRank gets distributed to your deep pages by virtue of your hierarchical internal linking structure (e.g. breadcrumb nav)
- Pay attention to the text used within the hyperlink ("Google bombing")
- Don't link to "bad neighborhoods"

6.63. Avoid PageRank dilution

- Canonicalization (www.domain.com vs. domain.com)
- Duplicate pages: (session IDs, tracking codes, superfluous parameters)

6.64. Write better anchor text

- Choose descriptive text
- Write concise text
- Think about anchor text for internal links too

6.65. 5) Encourage Clickthrough

- Zip's Law applies - you need to be at the top of page 1 of the search results.
- Synergistic effect of being at the top of the natural results and paid results.
- Entice the user with a compelling call-to-action and value proposition in your descriptions.

6.66. 6) Track the Right Metrics

- Indexation: ♯ of pages indexed, % of site indexed, % of product inventory indexed, ♯ of "fresh pages"
- Link popularity: ♯ of links, PageRank score
- Rankings: by keyword, "filtered" (penalized) rankings
- Keyword popularity: ♯ of searches, competition, KEI (Keyword Effectiveness Indicator) scores
- Cost/ROI: sales by keyword and by engine, cost of the lead

6.67. 7) Avoid Worst Practices

- Target relevant keywords
- Don't stuff keywords or replicate pages
- Create deep, useful content
- Don't conceal, manipulate, or over-optimize content
- Links should be relevant (no scheming!)
- Observe copyright/trademark law and Google's guidelines

6.68. Not Spam, But Bad for Rankings

- Content-less home page, Flash intros
- Title tags the same across the site
- Error pages in the search results (eg "Session expired")
- "Click here" links
- Superfluous text like "Welcome to" at beginning of titles
- Spreading site across multiple domains (usually for load balancing)
- Content too many levels deep

7. 7 Search improvement

7.1. Search improvement

- Improvement by user feedback
- Improvement without user feedback
  - Metasearch
  - Re-ranking method based on inter-document distances

7.2. Metasearch (1)

Answer for typical problems (access of limited part of the Web, different answer lists for same query): metasearch

- IR service:
  - Send queries to search engines, Web catalogues,
  - Gathers and aggregates the answers (Information fusion)
- Aim: better recall, precision, effectiveness

7.3. Metasearch (2)

- Phases of the process:
  - Choosing sources
    - Based on topic, statistics in the past, . . .
  - Document choosing
    - Quantitative aspects
  - Fusion (merging) algorithms
    - rank pozition, Retrieval Status Value

Metasearch	URL	number of sources
MetaCrawler	www.metacrawler.com	13
Dogpile	www.dogpile.com	25
SavvySearch	www.search.com	¿ 1000

7.4. Merging algorithms:

Based on Retrieval Status Value (RSV):

- CombSUM
- CombMED
- CombMIN
- CombMAX
- CombANZ
- CombMNZ

7.5. Re-ranking method based on inter-document distances

- Denote the documents by: $d_1, d_2, \ldots d_n$.

- Related weights of relevance: $r_1, r_2, \ldots r_n$.
- Distances between documents: $D(d_i, d_j) = d_{ij}$.
- Ideal r vector: r^* .
- Ideal document: d^* .

7.6. Introduce a function g

- For every d_1, d_2, where $D(d_1, d^*) = D(d_2, d^*) : g(D(d_1, d^*)) = g(D(d_2, d^*))$
- For every d_1, d_2, where $D(d_1, d^*) < D(d_2, d^*) : g(D(d_1, d^*)) > g(D(d_2, d^*))$
- For every $d \neq d^* : g(D(d^*, d^*)) > g(D(d, d^*))$

7.7. Goal of the estimation

- Find an estimated r , where:

$$\left\| \hat{\mathbf{r}} - \mathbf{r}^* \right\| \leq \left\| \mathbf{r} - \mathbf{r}^* \right\|$$

- Introduce a distance vector c :
- $c = [D(d_1, d^*), D(d_2, d^*), \ldots D(d_n, d^*)]$
- Ideal c^* for ideal r^* :

$$\left\| \hat{\boldsymbol{c}} - \boldsymbol{c}^* \right\| \leq \left\| \boldsymbol{c} - \boldsymbol{c}^* \right\|$$

7.8. Relationships

- At case of linear function $r_i = g(c_i) = r_{\max} - \alpha^* c_i$

7.9. Changing matrix

$$z_{i,j} = \begin{cases} \left| c_i - d_{i,j} \right| - c_j \ \textit{if} & c_j < \left| c_i - d_{i,j} \right| \\ c_i + d_{i,j} - c_j \ \textit{if} & c_j > c_i + d_{i,j} \\ 0 \quad \textit{if} \ \left| c_i - d_{i,j} \right| \leq c_j \leq c_i + d_{i,j} \end{cases}$$

7.10. Document Weight Improving (DWI)

- Distance matrix
- Original relevance values
- Calculation c vector
- Changing matrix
- Estimation of new c vector
- Calculation final relevance values

7.11. Example

- Q : african big-cat species
- Set of words: {lion, lynx, panther, african, species, big-cat, cheetah}
- q vector: $[0, 0, 0, 1, 1, 1, 0]$

$$
\begin{aligned}
d_1 &= [0, 0, 0, 2, 1, 2, 0], \\
d_2 &= [2, 2, 2, 4, 4, 4, 2], \\
d_3 &= [0, 1, 2, 1, 2, 2, 1], \\
d_4 &= [0, 1, 0, 0, 0, 1, 2].
\end{aligned}
$$

7.12. Original relevance values

$$r_1 = \text{sim}(d_1, q) = 0.96,$$
$$r_2 = \text{sim}(d_2, q) = 0.87,$$
$$r_3 = \text{sim}(d_3, q) = 0.75,$$
$$r_4 = \text{sim}(d_4, q) = 0.24,$$

$$\mathbf{D} = \begin{bmatrix} 0 & 15 & 6 & 7 \\ 15 & 0 & 11 & 16 \\ 6 & 11 & 0 & 7 \\ 7 & 16 & 7 & 0 \end{bmatrix}$$

7.13. Calculations

- $r_i = 1 - 0.03^* c_i$
- $c_i = (1 - r_i)/0.03$
- $c = [1.33; 4.33; 8.33; 25.33]$

$$\mathbf{Z} = \begin{bmatrix} 0 & 9.34 & -1 & -17 \\ 9.34 & 0 & 0 & -5 \\ 1 & 0 & 0 & -10 \\ 17 & 5 & 10 & 0 \end{bmatrix}$$

7.14. Final results

- $c_{new} = [10.25; 4.75; 9.08; 16.08]$
- $r_{new} = [0.69; 0.86; 0.73; 0.52]$
- Set of words: lion, lynx, panther, african, species, big-cat, cheetah

$$\begin{aligned} d_1 &= [0, 0, 0, 2, 1, 2, 0], \\ d_2 &= [2, 2, 2, 4, 4, 4, 2], \\ d_3 &= [0, 1, 2, 1, 2, 2, 1], \\ d_4 &= [0, 1, 0, 0, 0, 1, 2]. \end{aligned}$$

7.15. Multimedia retrieval

7.16. Multimedia retrieval: Description vs. Content-based retrieval

- Description-based retrieval systems
  - Build indices and perform object retrieval based on image descriptions, such as keywords, captions, size, and time of creation
  - Labor-intensive if performed manually

  - Results have typically poor quality if automated
- Content-based retrieval systems
  - Support retrieval based on the image content, such as color histogram, texture, shape, objects, and wavelet transforms

7.17. Queries in Content-Based Retrieval Systems

- Image sample-based queries:
  - Find all of the images that are similar to the given image sample
  - Compare the feature vector (signature) extracted from the sample with the feature vectors of images that have already been extracted and indexed in the image database
- Image feature specification queries:
  - Specify or sketch image features like color, texture, or shape
  - Features are translated into a feature vector. Match the feature vector with the vectors of the images in the DB.

7.18. Approaches Based on Image Signature

- Color histogram-based signature
  - The signature includes only color histograms based on color composition of an image regardless of its scale or orientation
- Multifeature composed signature
  - The signature includes a composition of multiple features: color histogram, shape, location, and texture

7.19. Mining Multimedia Databases

7.20. Multidimensional Analysis of Multimedia Data

Multimedia data cube

- Design is similar to traditional data cubes
- The database does not store images but their descriptors
- Metadata and plus dimensions for multimedia info
  - Feature descriptor: a set of vectors for each visual characteristic
    - Color vector: contains the color histogram
    - MFC (Most Frequent Color) vector: five color centroids
  - Layout descriptor: contains a color layout vector and an edge layout vector
    - MFO (Most Frequent Orientation) vector: five edge orientation centroids

7.21. Mining Associations in Multimedia Data

Special features:

- Need occurrences besides Boolean existence, e.g.,
  - "Two red square and one blue circle" implies a given theme.
- Need spatial relationships
  - Blue on top of white squared object is associated with brown bottom
- Need multi-resolution and progressive refinement mining
  - It is expensive to explore detailed associations among objects at high resolution
  - It is crucial to ensure the completeness of search at multi-resolution space

7.22. Mining Multimedia Databases

7.23. Mining Multimedia Databases

- From Coarse to FineResolution Mining

7.24. Preludes and application areas of CBIR

- Preludes of CBIR
  - Statistical analysis
  - Signal processing
  - Pattern recognition
  - OCR: Optical Character Recognition
- Application areas:
  - Images
  - Photo archive
  - Medical image process
  - Crime prevention
  - Army
  - Finding picture in Web

7.25. Content-based Image Retrieval (CBIR)

- Preliminaries
- Application areas

Query:

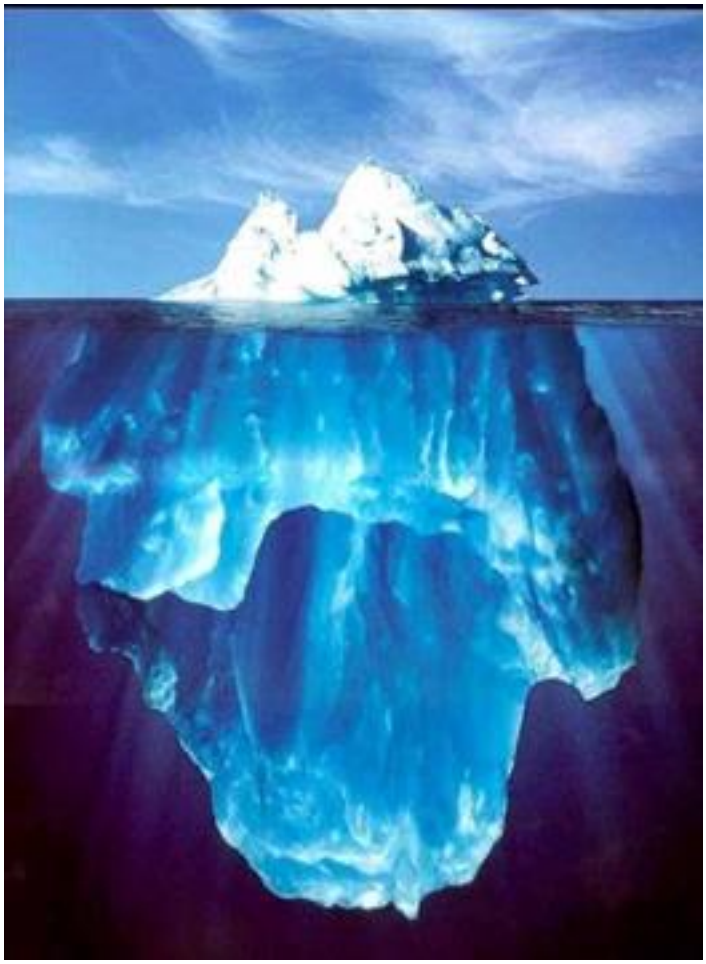
- Query By Example
  - Existing picture
  - Drawing
- Query by attributes/features:
  - Color
  - Texture
  - Shape

7.26. Specification of picture attributes

In Query and in comparing:

- Color
  - Histogram
  - Histograms of picture segments
  - Layout of picture segments
- Texture
  - Visual patterns
  - Silky or rough
- Shape
  - Forms (circle, square)
  - Segmentation, edge detection

7.27. Comparison of images



7.28. Feature extraction for images

Low-levels:

- Edge detection
- Corner detection
- Blob detection
- Ridge detection
- Interest point detection: e.g. Scale-invariant feature transform: SIFT

Curvature:

- Edge direction, changing intensity, autocorrelation.

Image motion:

- Motion detection. Area based, differential approach.

7.29. Shape Based methods for images

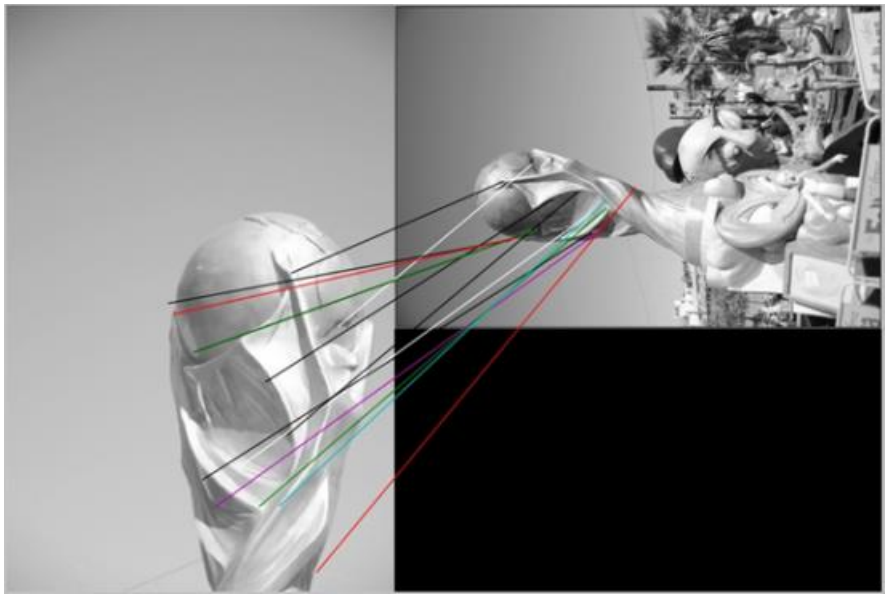
- Thresholding
- Blob extraction

- Template matching
- Hough transform
  - Lines
  - Circles/Ellipse
  - Arbitrary shapes (Generalized Hough Transform)

7.30. Scale-invariant feature transform (SIFT)

- Scale-invariant feature detection
- Feature matching and indexing (Best-bin-first search method)
- Cluster identification by Hough transform voting
- Outlier detection

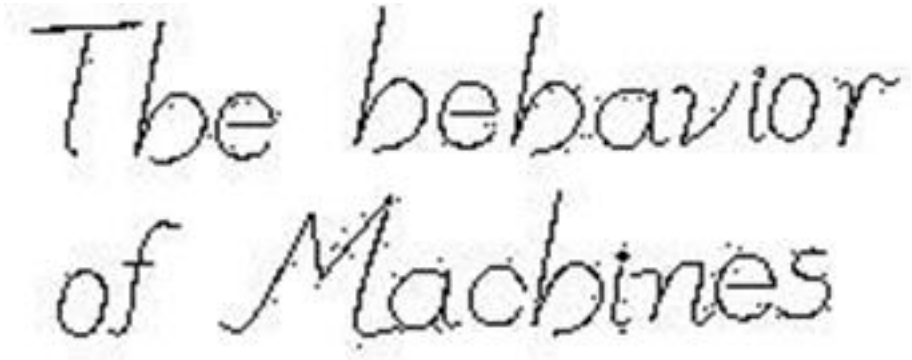
7.31. Matching by SIFT

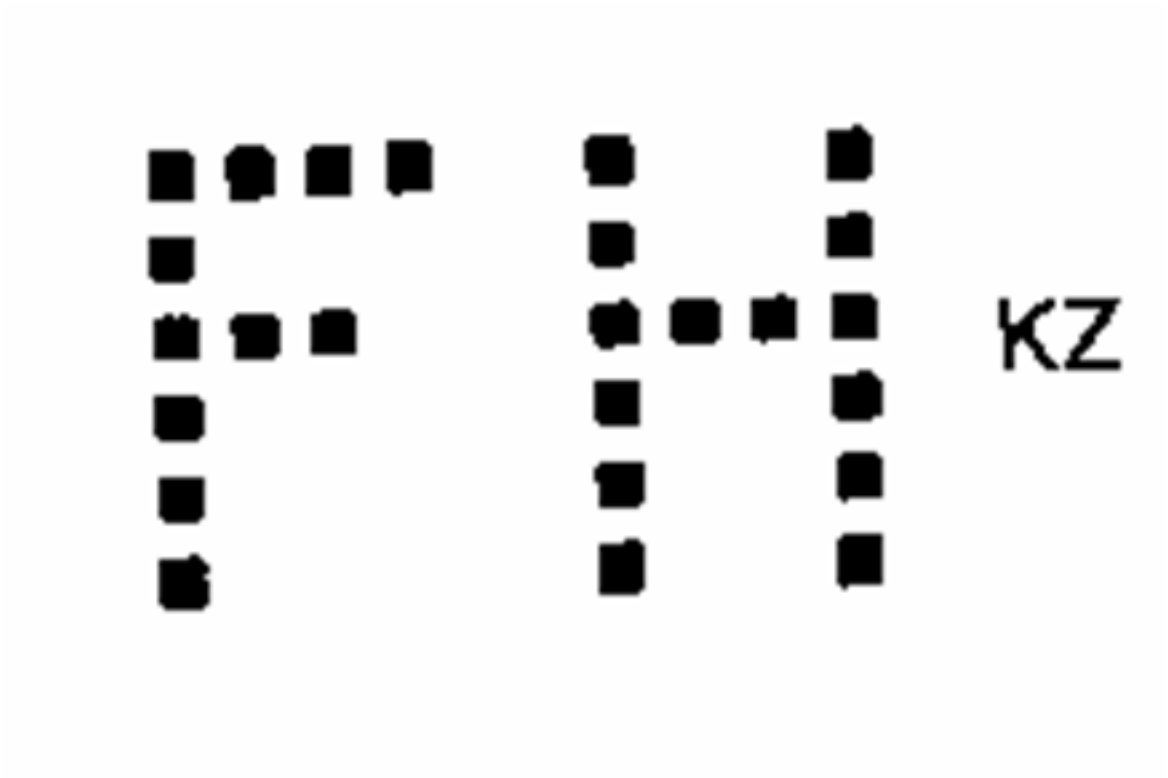


7.32. Optical Character Recognition (OCR)

- Why is accurate the result of OCR?
  - Number of characters is small
  - Limited font set
  - Known printers
- Why can not be direct used the OCR?
  - Context is essential element of the semantic recognition

7.33. Importance of context





7.34. Role of the context

- Further example:
  - I Have Fever, I Am Ill.
  - I have not seen Alexander III.
- Conclusion:
  - Wide context
  - Details

7.35. Text retrieval vs. Image retrieval

- Retrieval for words, phrases
- Very good results at general text as well.
- Retrieval for concepts
- Weak results at general areas.
- Better results at specified areas: OCR, medical diagnostics.

7.36. Experiment of concept search in Text retrieval

We find a story about a dog named Lucy

- Query: dog Lucy

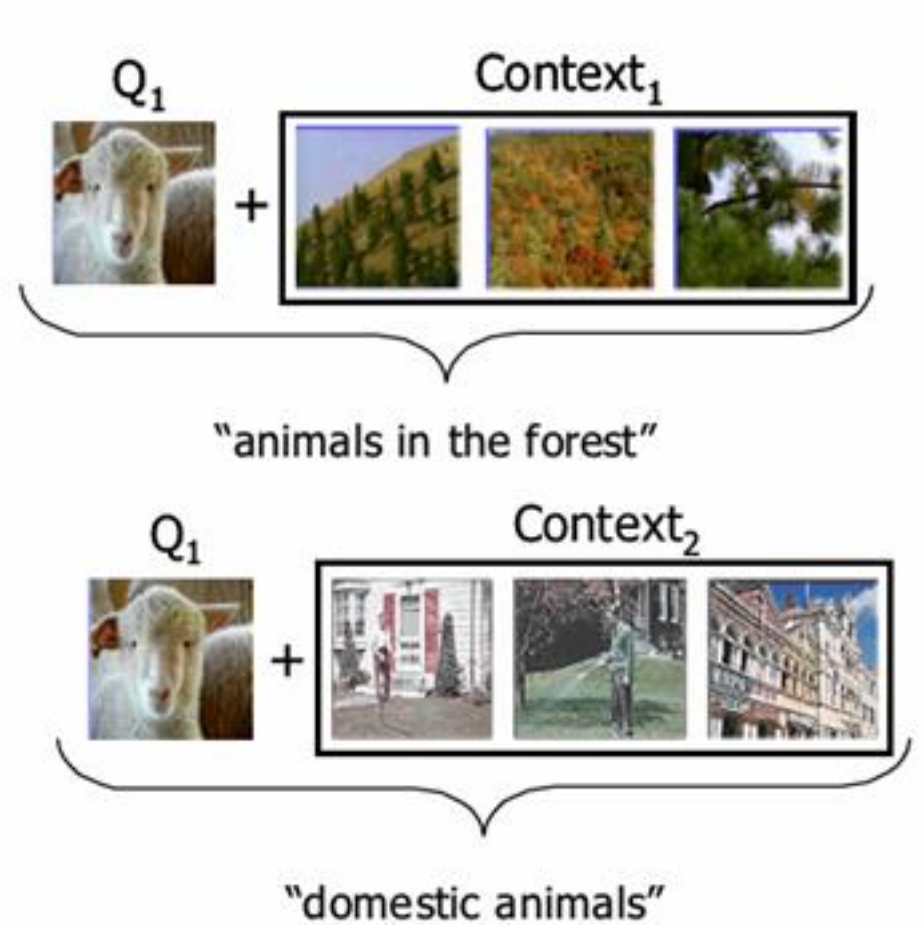
  - Many false answers, like Lucy has seen a dog.
- Query : dog named Lucy
  - False answers, like Lucy has seen a bird. We have a dog named Nero.
- Query : "dog named Lucy"
  - There is no false answers, but we will not get positive answers: Lucy has run. This dog . . .

7.37. Trend of CBIR

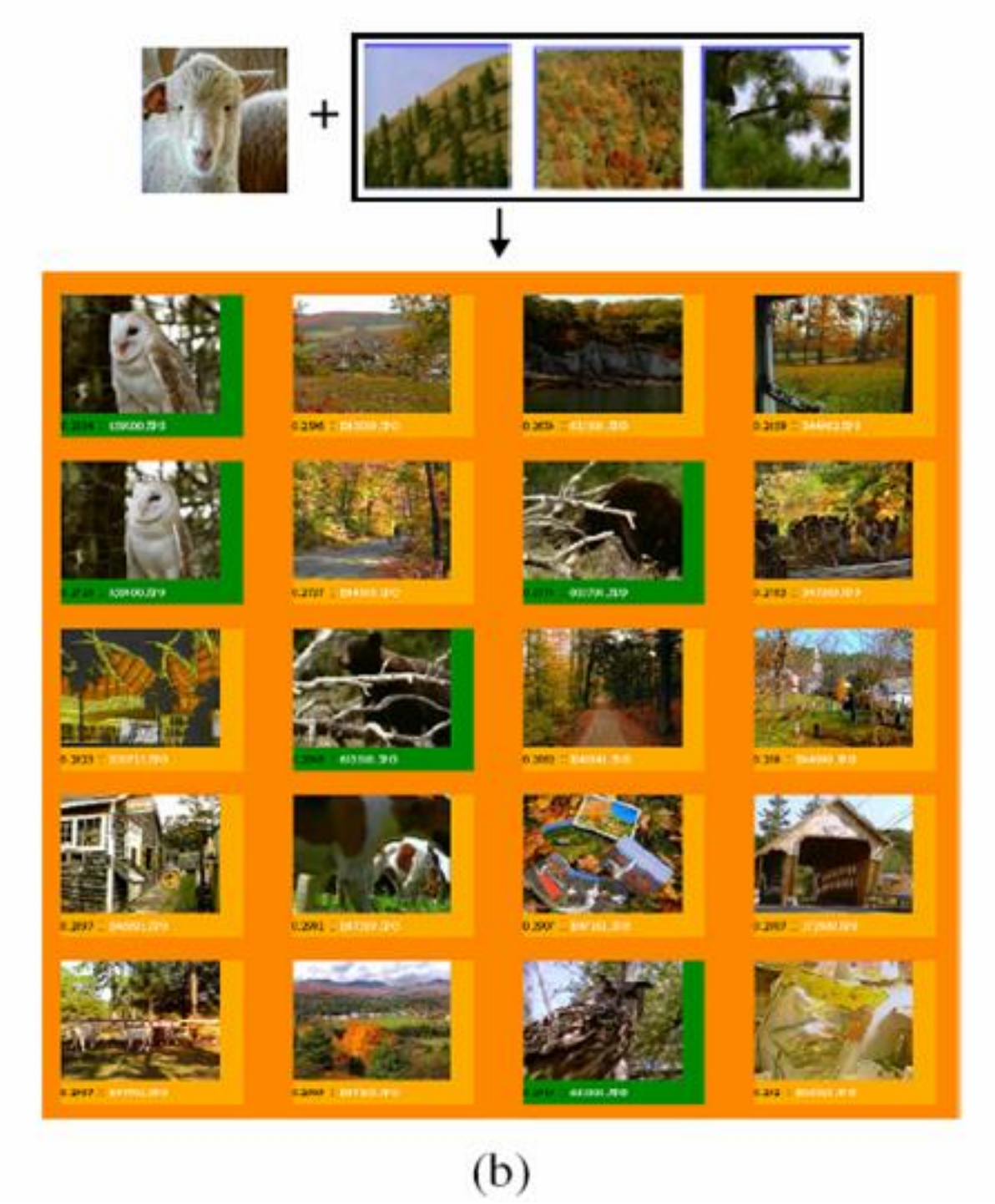
Image features vs. metadata

- Content information (colour, shape, etc.)
- Metadata (title, keywords, etc.)

7.38. New in CBIR: context



(a)



7.39. Video Retrieval

TRECVID http://trecvid.nist.gov/

Video search engines:

- Free, e.g. Picsearch Video Search, VideoSurf
- Bound, e.g. AOL Video, Google Video, Yahoo Video Search

Ranking:

- Related words to the video
- Ratings
- Number of links
- Users evaluations

7.40. Multimedia Information Retrieval

- Mixed media types
- Multimodal index
- Query:
  - Example: part of image, video
  - Text based
  - Place and time
  - Combination of them
- Standards: MPEG-7

7.41. Dimension reduction

Feature selection methods:

- Information gain
- Chi-square distribution

Dimension reduction with feature extraction:

- Principal components analysis
- Singular value decomposition
- Latent semantic analysis

7.42. Information Gain (IG) for Text Mining

Documents

- Classes: c_j
- Information Gain (IG) measures the occurance of term t_k vs. lack of it.
- Terms with low values will be deleted.

$$IG = \sum_{c \in c_j, \bar{c}_j} \sum_{t \in t_k, \bar{t}_k} P(t,c) \log \frac{P(t,c)}{P(t) \cdot P(c)}$$

7.43. X^2 Chi-square distribution for Text Mining

$$\chi^2(t_k, c_j) = \frac{N \cdot (n(t_k, c_j) \cdot n(\bar{t}_k, \bar{c}_j) - n(t_k, \bar{c}_j) \cdot n(\bar{t}_k, c_j))^2}{n(c_j) \cdot n(\bar{c}_j) \cdot n(t_k) \cdot n(\bar{t}_k)}$$

Measures the independency

- $n(c_j)$ number of documents in class c_j
- $n(t_k)$ number of documents containing term t_k
- $n(t_k, c_j)$ number of documents in class c_j containing term t_k

7.44. Principal components analysis

- This techniques removes the redundancy from a set of correlated variables.
- Few derived variables (principal components) are introduced.
- The task is to define these principal components (that are relatively independent of one another).

7.45. SVD

- singular value decomposition (SVD) is a factorization of a real or complex matrix.
- $D = U^* A^* V^T$
- U is unitary matrix,
- A is diagonal matrix
- V^* (conjugate transpose of V) is unitary matrix

7.46. Latent Semantic Indexing for Text Mining

- Index by larger units, "concepts" $\oplus$ sets of terms used together
- Retrieve a document that share concepts with a relevant one (even if it does not contain query terms)
- Group index terms together (map into lower dimensional space). So some terms are equivalent.
  - Not exactly, but this is the idea
  - Eliminates unimportant details

- Depends on a parameter (what details are unimportant?)

7.47. Latent Semantic Indexing (LSI)

- Basic idea
  - Similar documents have similar word frequencies
  - Difficulty: the size of the term frequency matrix is very large
  - Use a singular value decomposition (SVD) techniques to reduce the size of frequency table
  - Retain the K most significant rows of the frequency table
- Method
  - Create a term frequency matrix, $freq_matrix$
  - SVD construction: Compute the singular valued decomposition of $freq_matrix$ by splitting it into 3 matrices, U, A, V
  - Vector identification: For each document d, replace its original document vector by a new excluding the eliminated terms
  - Index creation: Store the set of all vectors, indexed by one of a number of techniques

7.48. LSI

- Elimination in matrix A (zero and almost zero elements)
- Reduced D matrix : D_{red} ($P' \leq P$ rank)
- $D_{red} = U^*_{red} A^*_{red} V^T_{red}$
- where $U_{red} \in R^{MxP'}, V_{red} \in R^{NxP'}$
- $d_{new} = d^{T*} U^*_{red} A^{-1}_{red}$

7.49. Text Index Building

7.50. Bag of words

- This is a simple representation of document.
- Document: special set of words, where an instance can occur in more times.
- E.g. for vector model the Bag of words representation is required
- E.g.:
  - Peter gives an apple to Bob.
  - Bob gives an apple to Peter.

7.51. Query

- Which plays of Shakespeare contain the words Brutus AND Caesar but NOT Calpurnia?
- Could grep all of Shakespeare's plays for Brutus and Caesar, then strip out lines containing Calpurnia?
  - Slow (for large corpora)
  - NOT Calpurnia is non-trivial
  - Other operations (e.g., find the phrase Romans and countrymen) not feasible

7.52. Term-document incidence

- 1 if play contains word, 0 otherwise

7.53. Incidence vectors

- So we have a $0/1$ vector for each term.
- To answer query: take the vectors for Brutus, Caesar and Calpurnia (complemented) bitwise AND.
- 110100 AND 110111 AND $101111 = 100100$.

7.54. Answers to query

- Antony and Cleopatra, Act III, Scene iiAgrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,When Antony found Julius Caesar dead,He cried almost to roaring; and he weptWhen at Philippi he found Brutus slain.
- Hamlet, Act III, Scene iiLord Polonius: I did enact Julius Caesar I was killed i' theCapitol; Brutus killed me.

7.55. Bigger corpora

- Consider $n = 1M$ documents, each with about $1K$ terms.
- Avg 6 bytes/term incl spaces/punctuation
  - $6GB$ of data in the documents.
- Say there are $m = 500K$ distinct terms among these.

7.56. Can't build the matrix

- $500Kx1M$ matrix has half-a-trillion 0's and 1's.
- But it has no more than one billion 1's.
  - matrix is extremely sparse.
- What's a better representation?
  - We only record the 1 positions.

7.57. Inverted index

- For each term T, must store a list of all documents that contain T.
- Do we use an array or a list for this?
- Brutus
- Calpurnia
- Caesar
- 13
- 16
- What happens if the word Caesar is added to document 14?

7.58. Inverted index

- Linked lists generally preferred to arrays
  - Dynamic space allocation
  - Insertion of terms into documents easy
  - Space overhead of pointers
- 2
- 4
- 8
- 16
- 32
- 64
- 128
- 1
- 2
- 3
- 5
- 8
- 13
- 21
- 34
- 13
- 16

- Sorted by docID (more later on why).

7.59. Inverted index construction

- Documents to be indexed.
- Friends, Romans, countrymen.

7.60. Indexer steps

- Sequence of (Modified token, Document ID) pairs.
- Doc 1
- I did enact JuliusCaesar I was killedi' the Capitol;Brutus killed me.
- Doc 2
- So let it be withCaesar. The nobleBrutus hath told youCaesar was ambitious
- Sort by terms.

Core indexing step.

- Multiple term entries in a single document are merged.
- Frequency information is added.

Why frequency?

- The result is split into a Dictionary file and a Postings file.

7.61. The index we just built

- How do we process a query?
  - What kinds of queries can we process?
- Which terms in a doc do we index?
  - All words or only "important" ones?
- Stopword list: terms that are so common that they're ignored for indexing.
  - e.g., the, a, an, of, to . . .
  - language-specific.

7.62. Query processing

- Consider processing the query:Brutus AND Caesar
  - Locate Brutus in the Dictionary;
    - Retrieve its postings.
  - Locate Caesar in the Dictionary;

    - Retrieve its postings.
  - "Merge" the two postings:
- 128
- 34
- Brutus
- Caesar

7.63. The merge

- Walk through the two postings simultaneously, in time linear in the total number of postings entries
- 128
- 34
- 2

If the list lengths are m and n, the merge takes $O(m+n)$

operations.

Crucial: postings sorted by docID.

7.64. Boolean queries: Exact match

- Queries using AND, OR and NOT together with query terms
  - Views each document as a set of words
  - Is precise: document matches condition or not.
- Primary commercial retrieval tool for 3 decades.
- Professional searchers (e.g., Lawyers) still like Boolean queries:
  - You know exactly what you're getting.

7.65. More general merges

- Exercise: Adapt the merge for the queries:Brutus AND NOT CaesarBrutus OR NOT Caesar

Can we still run through the merge in time

$O(m+n)$?

7.66. Merging

What about an arbitrary Boolean formula?

(Brutus OR Caesar) AND NOT

(Antony OR Cleopatra)

- Can we always merge in "linear" time?
- Can we do better?

7.67. Query optimization

- What is the best order for query processing?
- Consider a query that is an AND of t terms.
- For each of the t terms, get its postings, then AND together.
- Brutus
- Calpurnia
- Caesar
- 13
- 16

Query: Brutus AND Calpurnia AND Caesar

7.68. Query optimization example

- Process in order of increasing freq:
  - start with smallest set, then keep cutting further.

This is why we kept

freq in dictionary

Execute the query as (Caesar AND Brutus) AND Calpurnia.

7.69. More general optimization

- e.g., (Hamlet OR son) AND (king OR father)
- Get frequencies for all terms.
- Estimate the size of each OR by the sum of its frequencies (conservative).
- Process in increasing order of OR sizes.

7.70. Exercise

Recommend a query processing order for

(apple OR banana) AND(boy OR girl) AND(buy OR eat)

Term	Frequency
apple	187
banana	25
boy	215
buy	48
girl	129
eat	21

7.71. Query processing exercises

- If the query is friends AND romans AND (NOT countrymen), how could we use the freq of countrymen?
- Exercise: Extend the merge to an arbitrary Boolean query. Can we always guarantee execution in time linear in the total postings size?
- Hint: Begin with the case of a Boolean formula query: each query term appears only once in the query.

7.72. Skip pointers for faster postings

7.73. Recall basic merge

- Walk through the two postings simultaneously, in time linear in the total number of postings entries
- 128
- 31
- 2

If the list lengths are m and n, the merge takes $O(m+n)$ operations.

Can we do better?

Yes, if index isn't changing too fast.

7.74. Augment postings with skip pointers (at indexing time)

- Why?
- To skip postings that will not figure in the search results.
- How?
- Where do we place skip pointers?
- 31
- 8
- 16
- 128

7.75. Query processing with skip pointers

- 31
- 8
- 16
- 128

Suppose we've stepped through the lists until we process 8 on each list.

When we get to 16 on the top list, we see that its successor is 32.

But the skip successor of 8 on the lower list is 31, so we can skip ahead past the intervening postings.

7.76. Where do we place skips?

- Tradeoff:
  - More skips $\rightarrow$ shorter skip spans $\Rightarrow$ more likely to skip. But lots of comparisons to skip pointers.
  - Fewer skips $\rightarrow$ few pointer comparison, but then long skip spans $\Rightarrow$ few successful skips.

7.77. Placing skips

- Simple heuristic: for postings of length L, use $\sqrt{L}$ evenly-spaced skip pointers.
- This ignores the distribution of query terms.
- Easy if the index is relatively static; harder if L keeps changing because of updates.

7.78. Problems and questions about term search

7.79. Detour: problems and questions

- What if a doc consisted of components
  - Each component has its own access control list.
- Your search should get a doc only if your query meets one of its components that you have access to.
- More generally: doc assembled from computations on components
  - e.g., in Lotus databases or in content management systems
- Welcome to the real world $\ldots$ more later.

7.80. Beyond term search

- What about phrases?
- Proximity: Find Gates NEAR Microsoft.
  - Need index to capture position information in docs.

- Zones in documents: Find documents with (author = Ullman) AND (text contains automata).

7.81. Evidence accumulation

- 1 vs. 0 occurrence of a search term
  - 2 vs. 1 occurrence
  - 3 vs. 2 occurrences, etc.
- Need term frequency information in docs

7.82. Ranking search results

- Boolean queries give inclusion or exclusion of docs.
- Need to measure proximity from query to each doc.
- Whether docs presented to user are singletons, or a group of docs covering various aspects of the query.

7.83. Structured vs unstructured data

- Structured data tends to refer to information in "tables"

Employee	Manager	Salary
Smith	Jones	50000
Chang	Smith	60000
Ivy	Smith	50000

Typically allows numerical range and exact match

(for text) queries, e.g.,

Salary $<$ 60000 AND Manager $=$ Smith.

7.84. Unstructured data

- Typically refers to free text
- Allows
  - Keyword queries including operators
  - More sophisticated "concept" queries e.g.,
    - find all web pages dealing with drug abuse
- Classic model for searching text documents

7.85. Semi-structured data

- But in fact almost no data is "unstructured"
- E.g., this slide has distinctly identified zones such as the Title and Bullets

- Facilitates "semi-structured" search such as
  - Title contains data AND Bullets contain search

7.86. More sophisticated semi-structured search

- Title is about Object Oriented Programming AND Author something like stro * rup
- where * is the wild-card operator
- Issues:
  - how do you process "about"?
  - how do you rank results?
- The focus of XML search.

7.87. Tokenization

7.88. Tokenization

- Input: "Friends, Romans and Countrymen"
- Output: Tokens
  - Friends
  - Romans
  - Countrymen
- Each such token is now a candidate for an index entry, after further processing
  - Described below
- But what are valid tokens to emit?

7.89. Parsing a document

- What format is it in?
  - pdf/word/excel/html?
- What language is it in?
- What character set is in use?

Each of these is a classification problem.

But there are complications . . .

7.90. Format/language stripping

- Documents being indexed can include docs from many different languages

  - A single index may have to contain terms of several languages.
- Sometimes a document or its components can contain multiple languages/formats
  - French email with a Portuguese pdf attachment.
- What is a unit document?
  - An email?
  - With attachments?
  - An email with a zip containing documents?

7.91. Tokenization

- Issues in tokenization:
  - Finland's capital → Finland? Finlands? Finland's?
  - Hewlett-Packard → Hewlett and Packard as two tokens?
  - San Francisco: one token or two? How do you decide it is one token?

7.92. Language issues

- Accents: résumé vs. resume.
- L'ensemble → one token or two?
  - L ? L' ? Le ?
- How are your users like to write their queries for these words?

7.93. Tokenization: language issues

- Chinese and Japanese have no spaces between words:
  - Not always guaranteed a unique tokenization
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats

End-user can express query entirely in (say) Hiragana!

7.94. Normalization

- In "right-to-left languages" like Hebrew and Arabic: you can have "left-to-right" text mixed (e.g., for dollar amounts).
- Need to "normalize" indexed text as well as query terms into the same form
- Character-level alphabet detection and conversion
  - Tokenization not separable from this.

- Sometimes ambiguous:

Morgen will ich in MIT . . .

Is this

German "mit"?

7.95. Punctuation

- Ne'er: use language-specific, handcrafted "locale" to normalize.
  - Which language?
  - Most common: detect/apply language at a pre-determined granularity: doc/paragraph.
- State-of-the-art: break up hyphenated sequence. Phrase index?
- U.S.A. vs. USA - use locale.
- a.out

7.96. Numbers

- 3/12/91
- Mar. 12, 1991
- 55 B.C.
- B-52
- My PGP key is 324a3df234cb23e
- 100.2.86.144
  - Generally, don't index as text.
  - Will often index "meta-data" separately
    - Creation date, format, etc.

7.97. Case folding

- Reduce all letters to lower case
  - exception: upper case (in mid-sentence?)
    - e.g., General Motors
    - Fed vs. fed
    - SAIL vs. sail

7.98. Lemmatization

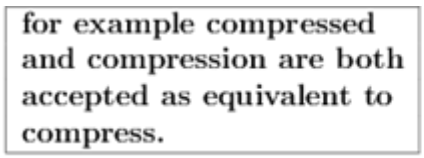
- Reduce inflectional/variant forms to base form

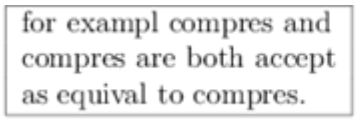
- E.g.,
  - am, are, is $\rightarrow$ be
  - car, cars, car's, cars' $\rightarrow$ car
- the boy's cars are different colors $\rightarrow$ the boy car be different color

7.99. Stemming

7.100. Stemming

- Reduce terms to their "roots" before indexing
  - language dependent
  - e.g., automate(s), automatic, automation all reduced to automat.





7.101. Types of stemmers

- Algorithmic, language specified rewriting rules (e.g. Porter, Snowball, Lovins)
- Dictionary with words and stems
- Other procedures, like statistical methods

7.102. Mistakes of stemming

- Under-stemming: There are two words with equivalent semantics. Algorithm assigns two different stems for these.
- Over-stemming: There are two words with different semantics. Algorithm assigns the same stem for these.

7.103. Measuring the stemmers

- Under-stemming Index

$$UI = 1 - (n_s/n_{all})$$

n_s : successfully reduced word pairs to a common stemn_{all} : word pairs possessing common stem

- Over-stemming Index

$$OI = 1 - (n_s/n_{all})$$

n_s : : successfully reduced word pairs to different stemsn_{all} : word pairs possessing different stems

7.104. Evaluation of stemmers

- Stemming Weight:

$$SW = OI/UI$$

  This shows the strength of reduction.

- Weak stemmer: reduces only few words to common stem.
- Strong stemmer: reduces many words to common stem.

7.105. Porter's algorithm

- Commonest algorithm for stemming English
- Conventions + 5 phases of reductions
  - phases applied sequentially
  - each phase consists of a set of commands
  - sample convention: Of the rules in a compound command, select the one that applies to the longest suffix.

7.106. Phases and typical rules in Porter

- phase : plural, -ed, -ing
- phase : simplification of double characters
- phase : cutting the end
- phase : investigation of last but one character
- phase : cutting e , replacement $II \rightarrow I$

E.g.

- sses $\rightarrow$ ss
- ies $\rightarrow$ i
- ational $\rightarrow$ ate
- tional $\rightarrow$ tion

7.107. Examples for Porter algorithm

- Generalizations1. phase
- Generalization2. phase

- Generalize3. phase
- General4. phase
- Gener
- Oscillators1. phase
- Oscillator2. phase
- Oscillate4. phase
- Oscill5. phase
- Oscil

[fragile]

7.108. Snowball

- Generalization of the Porter stemmer for other languages; stemming framework with its own string manipulator language
- Not effective for all languages because its descriptive power is restricted by the very simple two operators (cut, replace), see e.g. Hungarian
- ```
  http://snowball.tartarus.org/
  ```
- Supported languages include: French, Spanish, Portuguese, Italian, German, Russian, Finnish, Swedish, Norvegian, Danish, Armenian, Turkish, etc.

[fragile]

7.109. Other stemmers

- Other stemmers exist, e.g., Lovins stemmer

```
http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm
```

- Single-pass, longest suffix removal (about 250 rules)
- Motivated by Linguistics as well as IR
- Full morphological analysis - modest benefits for retrieval
- Hungarian stemmers: Tordai (Snowball), HunStem / Hunmorph, HelyesLEM (Morphologic)

7.110. Exercise for stemming
```

FULL WORD	Truncate (5)
Divide	Divid
Dividing	Divid
Divided	Divid
Division	Divis
Divisor	Divis
Divine	Divin
Divination	Divin

7.111. Exercise for stemming

FULL WORD	Truncate (5)	Truncate (4)
Divide	Divid	Divi
Dividing	Divid	Divi
Divided	Divid	Divi
Division	Divis	Divi
Divisor	Divis	Divi
Divine	Divin	Divi
Divination	Divin	Divi

7.112. Dictionary entries - first cut

ensemble.french
ć™,é–“.japanese
MIT.english
mit.german
guaranteed.english
entries.english
sometimes.english
tokenization.english

These may be grouped by language. More on this in query processing.

7.113. Language detection

7.114. Language detection using n-grams

- Task: specify the language of a text
  - Should be error prone: internet texts contain many typing and orthographical errors
  - Should be able to identify all European (character based) languages
  - . . . and of course, should be effective
- Approach: N-gram classification

7.115. N-grams

- Definition: N consecutive characters of a string (character n-gram)
  - There exist other types, e.g., word n-grams
- Example: ␣ TEXT ␣
  - Bigrams: ␣ T, TE, EX, XT, T␣
  - Trigrams: ␣ TE, TEX, EXT, XT␣ , T␣ ␣
  - 4-grams: ␣ TEX, TEXT, EXT␣ , XT␣ , T␣ ␣ ␣ ,
- In general: a string of length $k(k \geq 3)$ has $k+1$ bi-, tri-, és 4-grams.

7.116. Start: Zipf's law

- Order the terms of a corpus based on frequency, on the front there will be the stop words, at the bottom so-called hapaxes.
- The k_{th} most frequent term in a natural language text has frequency in inverse proportion to k .
- The transition between frequent and rare works is continuous, linear on log-log scale.
- The same law is valid for N-grams.

7.117. Zipf's law: Reuters-21578

Reuters-21578: A corpus provided by with 21578 docs; a benchmark document collection heavily used for text mining research

Contains economical short news

7.118. Zipf's law: web 2.2 (Hungarian)

Web2.2: Hungarian corpus created by crawling the Hungarain web in 2003. Contains 1.5 billion terms.

"Hungarian" means at least 96% of all terms in a website passed the Hungarian spell checker

7.119. Method

- Training data:
  - We need training data for each language to be identified, some $10K$ suffice per language.
- Learning: for each language a language profile is created from training data, which contains the most frequent N-grams.
- Classification: we create profiles from the new texts and compare them to language profiles.

7.120. Creating profiles

- Tokenization (punctuation filtering, except apostrophe)
- N-gram generation ($N = 1..5$), position saved

- Administer frequencies using a hash table
- Order N -grams by frequency and keep the top of the list

7.121. Observations

- The top 300 frequent N -grams correlate very well with the language
- Long technical and belles-lettres texts in English have quite similar profiles, and they are very different from the profile of an arbitrary French text
- Most frequent n -grams:
  - unigrams
  - Substrings of prepositions
- From 300 start the topic specific n -grams

7.122. Comparing profiles

- Based on their rank positions
- If an n -gram is present in both profiles then the distance of their positions is taken
- If an n -gram is present in only one profile, we take a predefined max distance
- Distances are added

$$d(\mathbf{d}_i, \mathbf{c}_j) = \sum_{k=1}^{N} \begin{cases} (\text{pos}_i(k) - \text{pos}_j(k)), & \text{ha } k \in \text{pos}_j \\ N, & \text{ha } k \notin \text{pos}_j \end{cases}$$

7.123. Example

7.124. Flowchart

7.125. Results

- The classification success rate is independent from the text's length.
- Longer language profiles yield usually better classification performance.
- Seldom the longer profile yielded inferior performance, typically for mixed language texts.
- 99.8% correct classification rate in case of language profiles of length 400!
- The approach can also be for topical classification.

7.126. Phrase queries, Positional indexes

[fragile]

7.127. Phrase queries

- Want to answer queries such as stanford university - as a phrase
- Thus the sentence "I went to university at Stanford" is not a match.
- No longer suffices to store only <term : docs> entries.

7.128. A first attempt: Biword indexes

- Index every consecutive pair of terms in the text as a phrase
- For example the text "Friends, Romans and Countrymen" would generate the biwords
  - friends romans
  - romans and
  - and countrymen
- Each of these is now a dictionary term
- Two-word phrase query-processing is now immediately solved.

7.129. Longer phrase queries

- Longer phrases are processed as several biword phrases:
- stanford university palo alto can be broken into the Boolean query on biwords:

stanford university AND university palo AND palo alto

we cannot verify that the docs matching the above Boolean query do contain the phrase.

7.130. Extended biwords

- Parse the indexed text and perform part-of-speech-tagging (POST).
- Bucket the terms into (say) Nouns (N) and articles/prepositions (X).
- Now deem any string of terms of the form NX^*N to be an extended biword.
  - Each such extended biword is now made a term in the dictionary.
- Example:
  - catcher in the rye

$$N \quad X \; X \quad N$$

7.131. Query processing

- Given a query, parse it into N's and X's
  - Segment query into enhanced biwords

  - Look up index
- Issues
  - Parsing longer queries into conjunctions
  - E.g., the query tangerine trees and marmalade skies is parsed into
  - tangerine trees AND trees and marmalade AND marmalade skies

7.132. Other issues

- False positives, as noted before
- Index blowup due to bigger dictionary

7.133. Positional indexes

- Store, for each term, entries of the form:<number of docs containing term;doc1: position1, position2 . . . ;doc2: position1, position2 . . . ;etc.>

7.134. Positional index example

<be: 993427;

1: 7, 18, 33, 72, 86, 231;

2: 3, 149;

4: 17, 191, 291, 430, 434;

5: 363, 367, Which of docs 1,2,4,5

could contain "to be or not to be"?

- Can compress position values/offsets
- Nevertheless, this expands postings storage substantially

7.135. Processing a phrase query

- Extract inverted index entries for each distinct term: to, be, or, not.
- Merge their doc:position lists to enumerate all positions with "to be or not to be".
  - to:
    - $2 : 1, 17, 74, 222, 551; 4 : 8, 16, 190, 429, 433; 7 : 13, 23, 191; \ldots$
  - be:
    - $1 : 17, 19; 4 : 17, 191, 291, 430, 434; 5 : 14, 19, 101; \ldots$
- Same general method for proximity searches

7.136. Rules of thumb

- Positional index size factor of $2-4$ over non-positional index
- Positional index size $35-50\%$ of volume of original text
- Caveat: all of this holds for "English-like" languages

8. 8 Exercise

8.1. General information

- Please write a query in the most frequent used search engines!
- Query in Google, Bing, Yahoo.
- Investigate the results!

8.2.

Task 1

- What is the recall and precision at first 20 results? Please calculate the F_1 value as well!

Task 2

- Please plot recall and precision values as a function of the answer set size!

8.3.

Task 3

- Please plot precision versus recall diagram!

Task 4

- Compare the different search engines based on the previous indicators and diagrams!

8.4. Exercise

Meta search

8.5. General information

- Please give a query in the most frequent used image search engines!
- Investigate the results and identify the same images!

8.6. Task

Please calculate to the answer list with different merging algorithms:

- CombSUM

- CombMED
- CombMIN
- CombMAX
- CombANZ
- CombMNZ

8.7. Exercise

Exercise for tfxidf weight calculation

8.8. General information

Let us consider the following brief documents, as a little corpus:Â

- Document1: Film is considered to be an important art form.
- Document2: A film is also called movie in the USA.
- Document3: In Europe film is preferred.
- Document4: Movie is considered to have its own language.
- Document5: The influence of film reviews is extremely important.
- For a fast and effective search a representation of corpus needs to be built. In this representation weight values should be determined for every term in every document.

8.9.

Task 1

- Please calculate the tfxidf weights for every term in every document in the corpus!

Task 2

- What will be the answer for the query: USA film?

8.10. Exercise

Stemming

8.11. General information

- Please investigate the following simple stemmer!

FULL WORD	Truncate (5)
Divide	Divid
Dividing	Divid
Divided	Divid
Division	Divis
Divisor	Divis
Divine	Divin
Divination	Divin

8.12. Task 1

- Calculate the under stemming index!
- Calculate the over stemming index!
- How much is the stemming weight?

8.13. Comparison of two stemmers

FULL WORD	Truncate (5)	Truncate (4)
Divide	Divid	Divi
Dividing	Divid	Divi
Divided	Divid	Divi
Division	Divis	Divi
Divisor	Divis	Divi
Divine	Divin	Divi
Divination	Divin	Divi

8.14. Task 2

- Calculate the under stemming index at second stemmer!
- Calculate the over stemming index at second stemmer!
- How much is the stemming weight at second stemmer?
- Please compare these two stemmers!