6

UNIT - VI

Advanced Information Retrieval

Syllabus

XML Retrieval : Basic XML concepts, Challenges in XML retrieval, Vector space model for XML retrieval, Evaluation of XML retrieval, Text-Centric vs. Data-Centric XML retrieval.

Recommendation system : Collaborative Filtering and Content Based Recommendation of Documents and Products. Introduction to Semantic Web. (6·14, 6·12, 6·19 - 6·20)

6.1 Introduction

- Information retrieval system are designed for storing and searching the textual data which is in raw format. Data retrieval eg. Relational databases has a standard structure in which the data is saved in the system. Data storage and retrieval is easier in such relational database systems. Queries like:
- "Select, from, Where" defines the way in which user can specify the attributes which he want to get from the database based on some constraints. In case of information retrieval system where only textual data is handled, the inverted files, suffix trees or signature files are the choices for storing the information.
- In some cases, the text present in the collection has a typical structure. The information is stored in this structure in the system. For such information, the retrieval system needs to be modified which can retrieve the relevant documents which satisfy the conditions in terms of text as well as the structure. This search proves is called as **Structural retrieval.**
- The example queries of structural retrieval are "Find the short paper containing the word cryptography", Search for the copyright mentioning the term information retrieval".

Table 6.1.1 : Relational database (RDB) search, unstructured information retrieval and structured information retrieval

	RBD search	Unstructured reterival	Structured retrieval
Objects	Records	Unstructured documents	Trees with text at leaves
Model	Relation model	Vaectro space and orther	?
Main data structure	Tabel	Inverted index	?
Queries	SQL	Free text queries	?

- For above queries, unranked document retrieval such as Boolean model is not suitable. In terms of Boolean model, user is totally unaware about the structure of the document. Table 6.1.1 shows the comparison between relational database search, unstructured information retrieval and structured information retrieval.

- Another type of IR problem is between structured IR and unstructured IR which is called as Parametric and Zone search. In this search, there are parametric fields such as date or file size. The zones contains the long text such as book name or document name. The query for such retrieval defines the constraints on parameters and zone fields.

6.2 Basic XML Concepts

Following are the basic concepts of XML .

(i) Extensible Markup Language (XML) : XML is the Standard for encoding the structured documents. XML is the labelled ordered tree.

- Each node in the tree is an XML element and represented with an opening and closing tag.
- Elements has one or more attributes.
- Fig. 6.2.1 shows the example of XML document. The XML elements present in this document are "title", "act", and "scene". The "scene" element is presented with tags <scene> .... </scene>. Scene element has the attribute "number" whose value is "vii".

```
<play>
<author>Shakespeare</author>
<title>Macbeth</title>
<act number="I">
<scene number="vii">
<title>Macbeth's castle</title>
<verse>Will I with wine and wassail ...</verse>
</scene>
</act>
</play>
```

Fig. 6.2.1 : An example of XML document

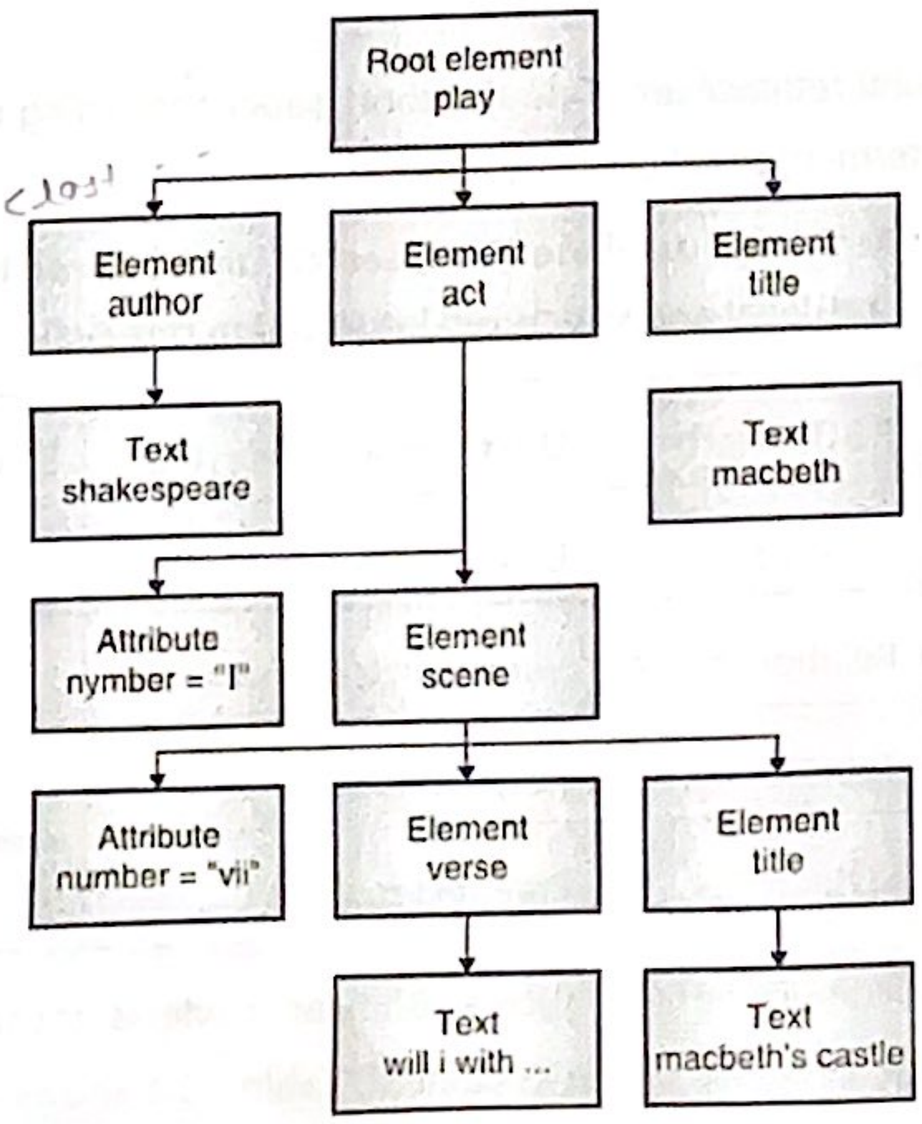


Fig. 6.2.2 : Simplified DOM object representation of the XML document shown in Fig. 6.2.1

- Fig. 6.2.2 shows the tree structure of the document. Here the leaf nodes contain the text such as Shakespeare, Macbeth, and Macbeth's castle. The internal nodes of the tree encode either the structure of the document (*title*, *act*, and *scene*) or metadata functions (*author*).
- XML Document Object Model (DOM) is the standard for accessing and processing the XML documents. It is the hierarchical structure in which the text, attributes and elements are represented as the nodes. DOM API can process an XML document by starting at the root element and then descending down the tree from parents to children.

(ii) ***Xpath*** **:** Xpath is a standard for enumerating paths in an XML document collection. The paths are also called as XML contexts. Only a small subset of XPath is needed for our purposes. The XPath expression node selects all nodes of that name. Successive elements of a path are separated by slashes, so act/scene selects all *scene* elements whose parent is an *act* element. Double slashes indicate that an arbitrary number of elements can intervene on a path: play//scene selects all *scene* elements occurring in a *play* element.

For Fig. 6.2.2, the set consists of a single *scene* element, which is accessible via the path *play*, *act*, *scene* from the top. An initial slash starts the path at the root element. /play/title selects the play's title in Fig. 6.2.1, /play//title selects a set with two members (the play's title and the scene's title), and /scene/title selects no elements. The final element in the path is the vocabulary term. While selecting it for a specific element the vocabulary term can be mentioned with the element separated by #. For example : title# "Macbeth" selects all titles containing the term Macbeth.

(iii) **Schema :** Schema defines the structural constraints of XML documents for a particular application. For example, user may specify the schema for Shakespeare's plays. In this, scenes can only occur as children of acts and that only acts and scenes have the *number* attribute.

There are 2 standards for schemas :

- XML Document Type Definition (DTD)
- XMl Schema

```
//article
[.//yr = 2001 or .//yr = 2002]
//section
[about(.,summer holidays)]
```

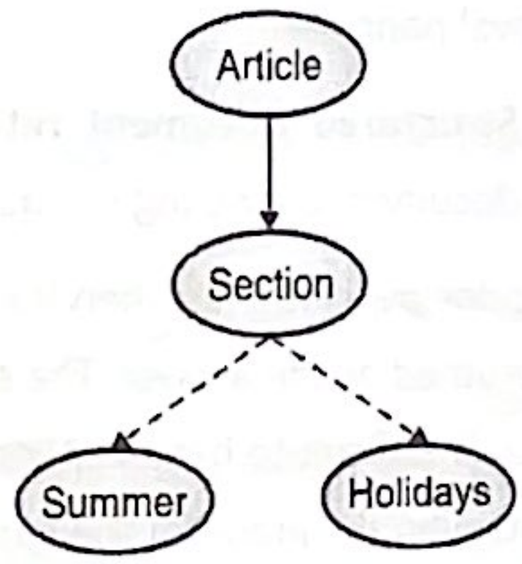


Fig. 6.2.3 : NEXI query and its tree structure representation

(iv) **Narrowed NEXI Extended XPath I(NEXI) :** XML queries can be specified in standard format which is called as NEXI. Fig. 6.2.3 shows the example of NEXI query and its tree representation. The query statement is a single line query in which the conditions for different elements are defined. For example, the query specified in Fig. 6.2.3 is a search for sections about the "summer holidays" that are part of articles from 2001 or 2002. As in XPath double slashes indicate that an arbitrary number of elements can intervene on a path. The dot in a clause in square brackets refers to the element the clause modifies. The clause [.//yr = 2001 or .//yr = 2002] modifies //article. Thus, the dot refers to //article in this case. Similarly, the dot in [about(., summer holidays)] refers to the section that the clause modifies. The about clause is a ranking constraint : Sections that occur in the right type of article are to be ranked according to how relevant they are to the topic summer holidays.

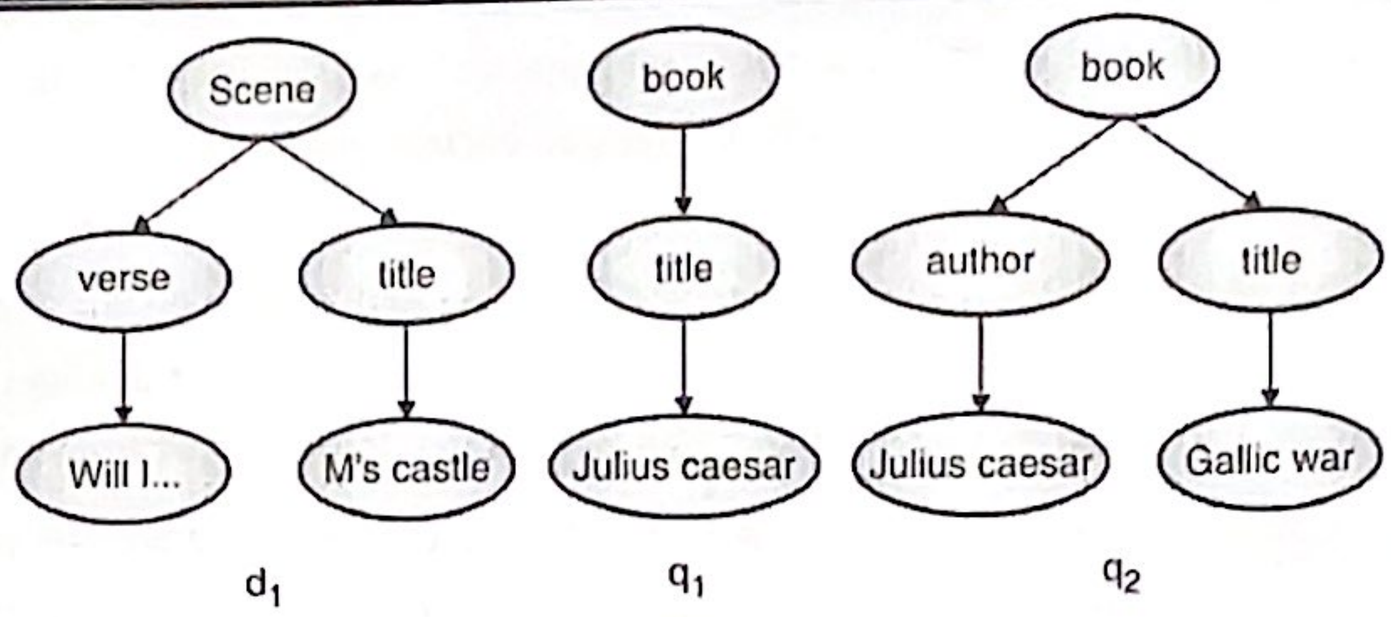


Fig. 6.2.4 : Tree representation of XML Documents and queries

- Fig. 6.2.4 shows the tree representations of XML documents in which the attribute filed is not considered, for example, "number". Such trees are called as **element-node trees.**
- The queries can also be represented in terms of trees. The approach is called as **query by example** approach. q_1 is a search for books whose titles score highly for the keywords Julius Caesar. q_2 is a search for books whose author elements score highly for Julius Caesar and whose title elements score highly for Gallic war.

6.3 Challenges in XML Retrieval

In structural retrieval, the collection consists of structured documents and queries are either structured or unstructured. Following are challenges related to XML retrieval :

Users want parts of documents (i.e., XML elements), not entire documents as IR systems :

- In structural retrieval, users are interested in the details of specific XML elements. For examples : scenes of Shakespeare. One criterion for selecting the most appropriate part of a document is the structured document retrieval principle :
  - **Structured document retrieval principle :** A system should always retrieve the most specific part of a document answering the query.
- The principle says that when the query is specified, the smallest unit which satisfies the query constraints needs to be returned as the answer. The search should not go below that level. But, deciding which level of the text to stop search is difficult to handle. Align to this, it is necessary to define which part of the unit needs to be indexed while maintaining the index for the structured document.

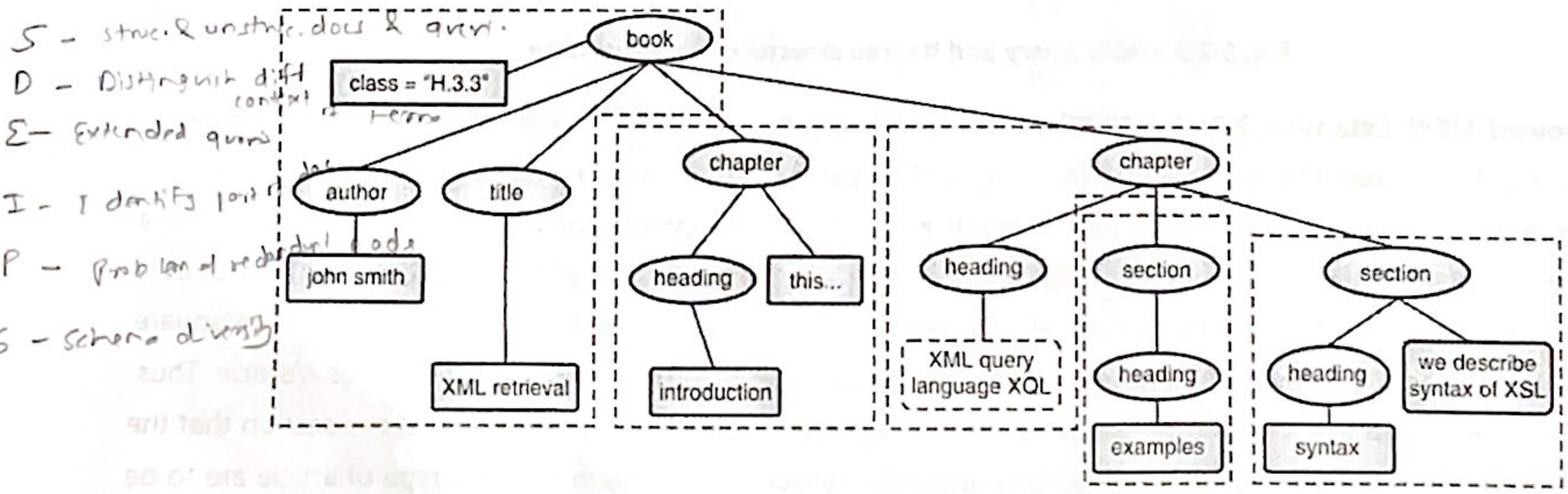


Fig. 6.3.1 : Partitioning an XML document into non-overlapping indexing units

- One solution to this indexing problem is representing the document in terms of non-overlapping indexing units. Fig. 6.3.1 shows the example of partitioning documents into non-overlapping indexing units. books, chapters and sections are the indexing units which are non-overlapping. But, the issue related to such indexing is it does not make sense from user's perspective.
- Another way of handling the problem is to make largest element as the indexing unit. For example, we can consider "book" as the indexing unit. The search process will work at this level. As the part of post-process, search may consider the elements which are part of the indexing unit. For example, after searching a specific book, as the second step, the search will work on specific section. Thus, the search will become two stage process. Unfortunately, this two stage retrieval process fails to return the best sub-element for many queries because the relevance of a whole book is often not a good predictor of the relevance of small sub-elements within it.
- The third way of search is do the search process at each level. At each level, select the element which is most relevant to the query. For the query Macbeth's castle in Fig. 6.2.1, we would retrieve the title *Macbeth's castle* in the first pass and then decide in a post processing step whether to return the title, the scene, the act or the play. This approach also has the disadvantage that the leaf element is often not a good predictor of the relevance of elements it is contained in.

Problem of redundant nodes as the part of query result :

- The least restrictive approach is to index all elements. In this case the tree will have maximum levels as every element becomes the node. But this approach is also problematic as few XML elements are defined for formatting and does not really helpful for the search purpose. Thus, it will increase unnecessary comparison steps during search process. Also, in some cases, the redundant nodes will be returned as the part of search. These redundant nodes are called as the nested nodes. Returning such redundant nodes in the result is not good practice. Because of the redundancy caused by nested elements it is common to restrict the set of elements that are eligible to be returned. Restriction strategies include :
  - Discard all small elements.
  - Discard all element types that users do not look at (this requires a working XML retrieval system that logs this information).
  - Discard all element types that assessors generally do not judge to be relevant (if relevance assessments are available).
  - Only keep element types that a system designer or librarian has deemed to be useful search results.
- In most of these approaches, result sets will still contain nested elements. Thus, we may want to remove some elements in a post-processing step to reduce redundancy.
- If the user knows the schema of the XML document and specifies the query accordingly, then the search process becomes specific and very less chances of redundant nodes in the result. But, most of times user does not have idea about the structure of the document.

Distinguish different contexts of a term :

- Relating to nesting, XML retrieval also face the challenge to distinguish between the different contexts of the term. For example, Consider the word "Gate". When the word "Gates" is used in singular form it can be part of Author name. But when the word is used in plural form "Gates" has other meaning and which can not be part of author name. Thus, we may need to distinguish different contexts of a term when we compute term statistics for ranking, in particular Inverse Document Frequency (idf) statistics.

- One solution is to compute idf for XML-context/term pairs, e.g., to compute different idf weights for author# "Gates" and section# "Gates". A compromise is only to consider the parent node *x* of the term and not the rest of the path from the root to *x* to distinguish contexts. There are still conflations of contexts that are harmful in this scheme. For instance, we do not distinguish names of authors and names of corporations if both have the parent node *name*. But most important distinctions, like the example contrast author#"Gates" vs. section#"Gates", will be respected.

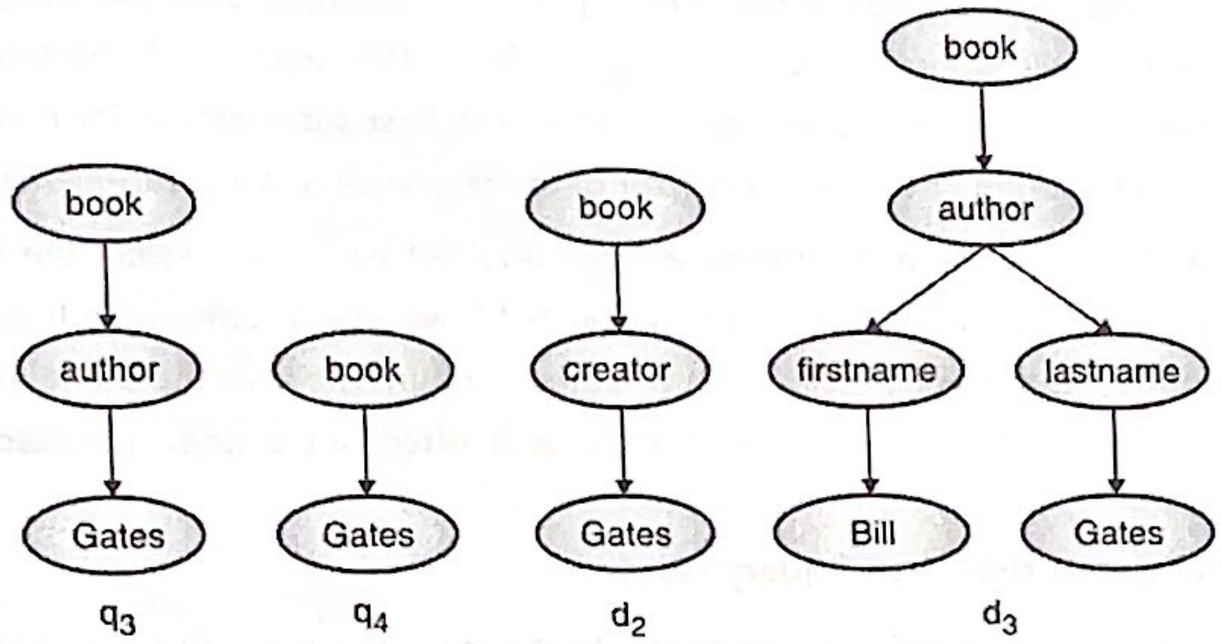


Fig. 6.3.2 : Schema heterogeneity : intervening nodes and mismatched names

1. **Schema Heterogeneity (schema diversity) :** In most of the times, different schema are present for the documents in the collection. This phenomenon is called as *Schema Heterogeneity (schema diversity)*. Thus the term present in done document may have different context in another document. Fig. 6.3.2 shows the schema heterogeneity challenge. In this case, in document d_2, "Gates" term is present as creator, whereas in document d3, "Gates" term is present as the last name.

- In some cases, the structure of the schema are different. For example, in one schema full author name can be a single entity whereas in another schema the author name may be further divided in "first name" and "last name". In these cases, if we employ a strict schema matching, then it may possible that few documents which are actually relevant will not be listed in the answer list.

- This poses a challenge for interface design in XML retrieval. Ideally, the user interface should expose the tree structure of the collection and allow users to specify the elements they are querying. If we take this approach, then designing the query interface in structured retrieval is more complex than a search box for keyword queries in unstructured retrieval.

2. **Extended queries :** We can also support the user by interpreting all parent-child relationships in queries as descendant relationships with any number of intervening nodes allowed. We call such queries ***extended queries***. For example, consider the query q_4 in Fig. 6.3.2. Here we define the query to search book which contains the "Gates". The dashed arrow shows that the term "Gates" can be present in the descent tree of the node "book".

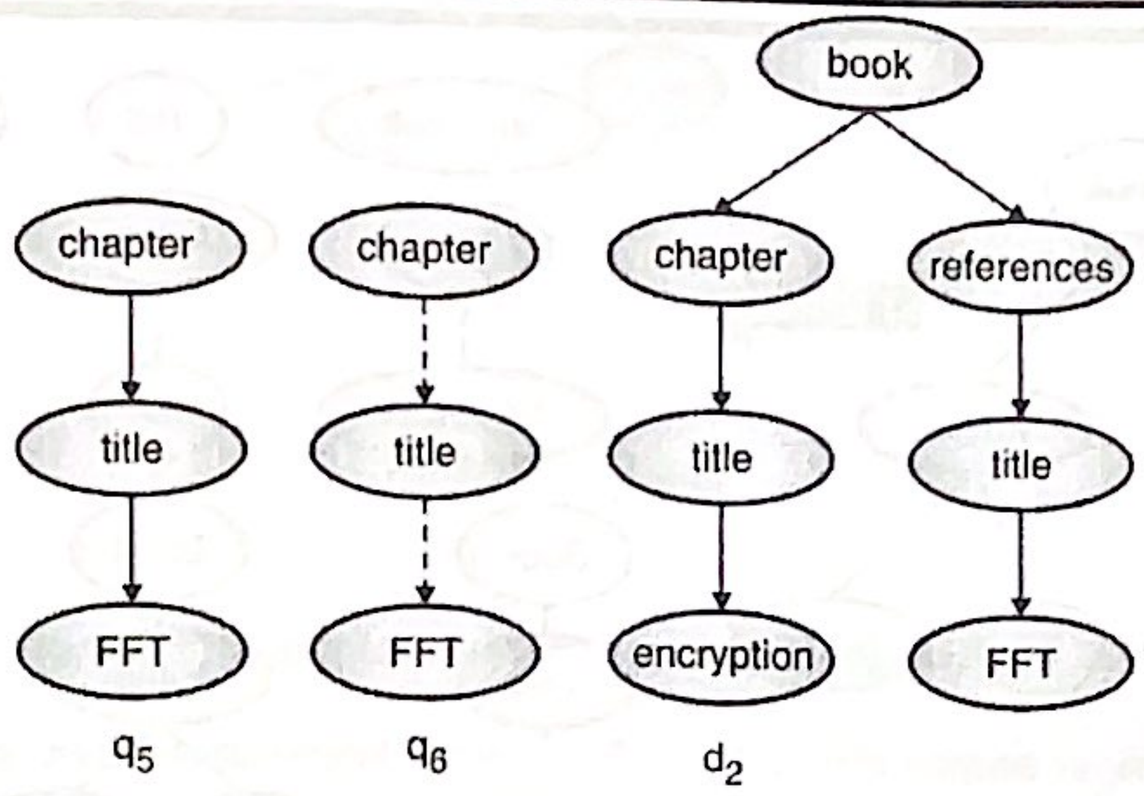


Fig. 6.3.3 : A structural mismatch between two queries and a document

- Consider the situation in which user is interested in the book with title "FFT". Fig. 6.3.3 (q_5) represents the query. If we apply the extended search as shown in (q_6) or basic search as shown in (d_4) of Fig. 6.3.3 both will not return us any result. The situation is there is no chapter which has the title "FFT". But in document d_4, as the part of references, the book having title "FFT" is present.
- In such cases, the IR system needs to relax the strict rules for showing the exact match result. In such case, the document d_4 can be part of query result. Even though the document does not have the title "FFT", but the user may get its reference from the "references" elements. Thus, even though few documents are not exactly relevant to user queries, they may be part of the results. Elements that do not meet structural constraints perfectly should be ranked lower, but they should not be omitted from search results.

6.4 A Vector Space Model for XML Retrieval

The vector model defined for unstructured queries are not directly suitable for structured queries.

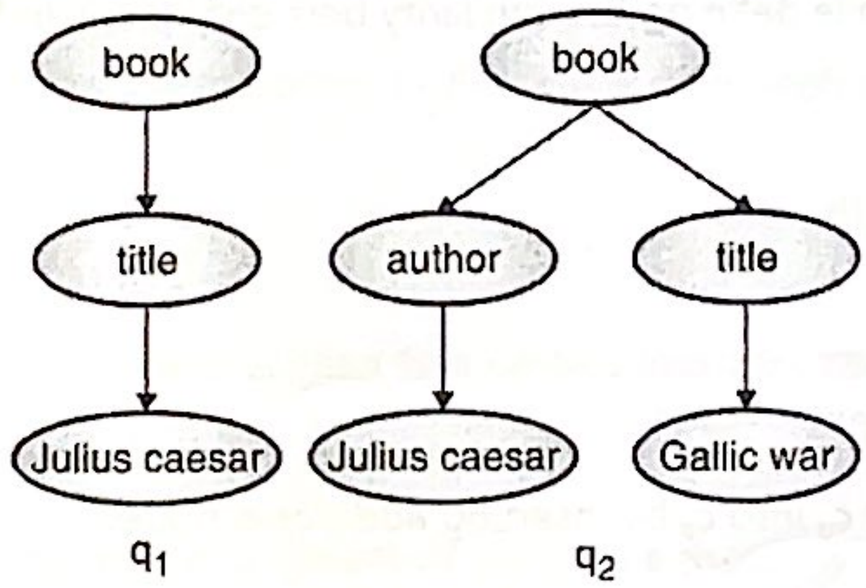


Fig. 6.4.1 : Structured queries

- Consider that user is interested in the book "Julius Caesar". In this case, the query specified in Fig. 6.4.1 (q_1) matches with the user's query. There should not be match with the query Fig. 6.4.1 (q_2). When we considered the unstructured query, there is no discrimination between the fields in which the term is present. Thus, In XML retrieval, we must separate the title word Caesar from the author name Caesar. One way of doing this is to have each dimension of the vector space encode a word together with its position within the XML tree.

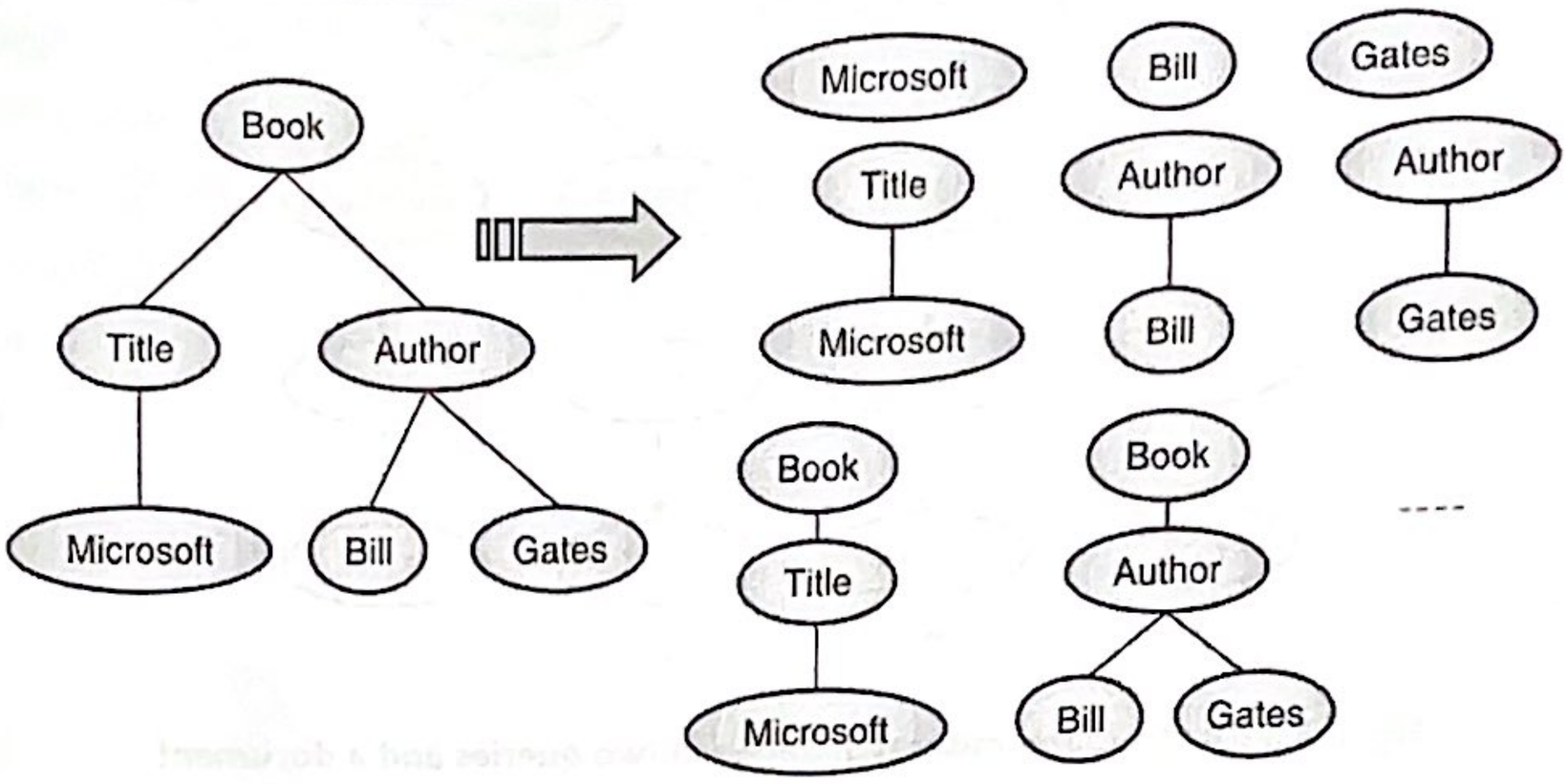


Fig. 6.4.2 : A mapping of an XML document to a set of lexicalized subtrees

- Consider the XML document whose tree structure is represented in Fig. 6.4.2.
- Take each text node and break it in multiple nodes, one node for each word.
- Then consider all possible lexicalized subtrees which contains the at least one vocabulary term (word). Few such subtrees are shown in Fig. 6.4.2. These lexicalized subtrees define the dimensions of the vector space. We can now represent queries and documents as vectors in this space of lexicalized subtrees and compute matches between them.
- While defining the vector space model for structured queries, we consider that the IR system will give the result which is exact match with the query. Also, the documents which are near to the query (has very low relevance) are also listed in the result.
- Consider following parameters while defining the similarity between query and the document for structured IR :

  C_q: Path in a query

  $|C_q|$: Number of nodes in query path

  C_d: A path in a document

  $|C_d|$: Number of nodes in a document

  c_q matches c_d iff we can transform c_q into c_d by inserting additional nodes.

- **Context resemblance (C_R) :** It is the similarity of a path c_q in a query and a path c_d in a document and computed as :

$$CR(c_q, c_d) = \begin{cases} \frac{1 + |c_q|}{1 + |c_d|} & \text{if } c_q \text{ matches } c_d \\ 0 & \text{if } c_q \text{ does not match } c_d \end{cases}$$

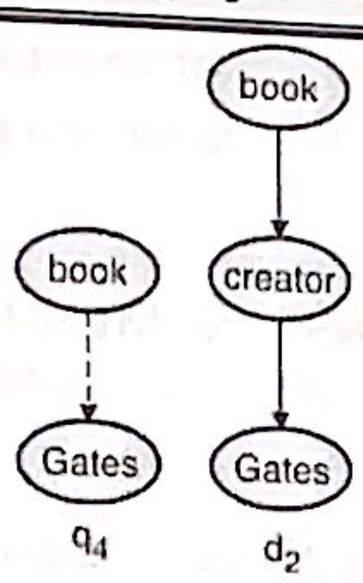


Fig. 6.4.3 : Example 1 to compute Context resemblance

Example 1 : Consider the query and the document shown in Fig. 6.4.3. The context resemblance for query q_4 and the document d_2 is:

$$CR(c_{q4}, c_{d2}) = \frac{3}{4} = 0.75$$

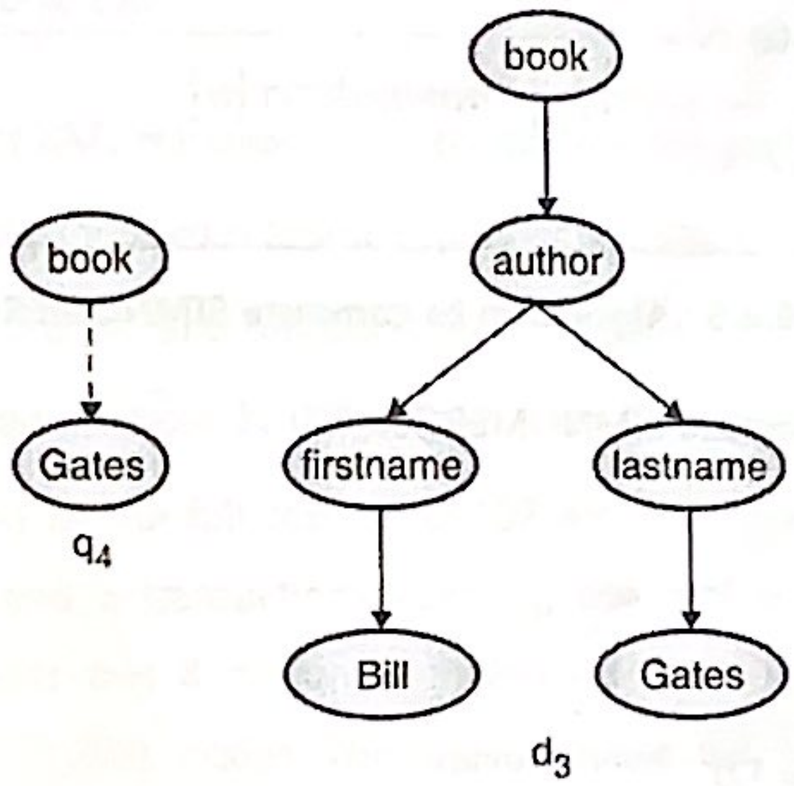


Fig. 6.4.4 : Example 2 to compute Context resemblance

Example 2 : Consider the query and the document shown in Fig. 6.4.3. The context resemblance for query q_4 and the document d_3 is :

$$CR(cq_4, cd_3) = \frac{3}{5} = 0.6$$

Example 3 : If the query and the document are identical, then their context resemblance value is 1.0.

Computing final score of a document with respect to query :

The final score for a document is computed as a variant of the cosine measure which is called as SIMNOMERGE and is computed as :

$$\text{Simnomerge}(q, d) = \sum_{c_k \in B} \sum_{c_1 \in B} C_R(c_k, c_1) \sum_{t \in V} \text{weight}(q, t, c_k) \frac{\text{weight}(d,t,c_1)}{\sqrt{\sum c \in B, t \in V \text{ weight}^2(d,t,c)}}$$

Where, V : The vocabulary of non-structural terms

B : It is the set of all XML contexts

Weight(q, t, c) and weight(d, t, c) are the weights of term t in XML context c in query q and document d, respectively. These weights are computed based on tf-idf factor such as $idf_t \cdot w_{ft,d}$. The inverse document frequency idf_t depends on which elements we use to compute df_t

```
SCOREDOCUMENTSWITHSIMNOMERGE(q, B, V, N, normalizer)
 1  for n ← 1 to N
 2  do score[n] ← 0
 3  for each ⟨c_q, t⟩ ∈ q
 4  do w_q ← WEIGHT(q, t, c_q)
 5     for each c ∈ B
 6     do if CR(c_q, c) > 0
 7           then postings ← GETPOSTINGS(⟨c, t⟩)
 8                for each posting ∈ postings
 9                do x ← CR(c_q, c) * w_q * weight(posting)
10                   score[docID(posting)] += x
11  for n ← 1 to N
12  do score[n] ← score[n]/normalizer[n]
13  return score
```

Fig. 6.4.5 : Algorithm to compute SIMNOMERGE

Fig. 6.4.5 shows the algorithm to compute SIMNOMERGE.

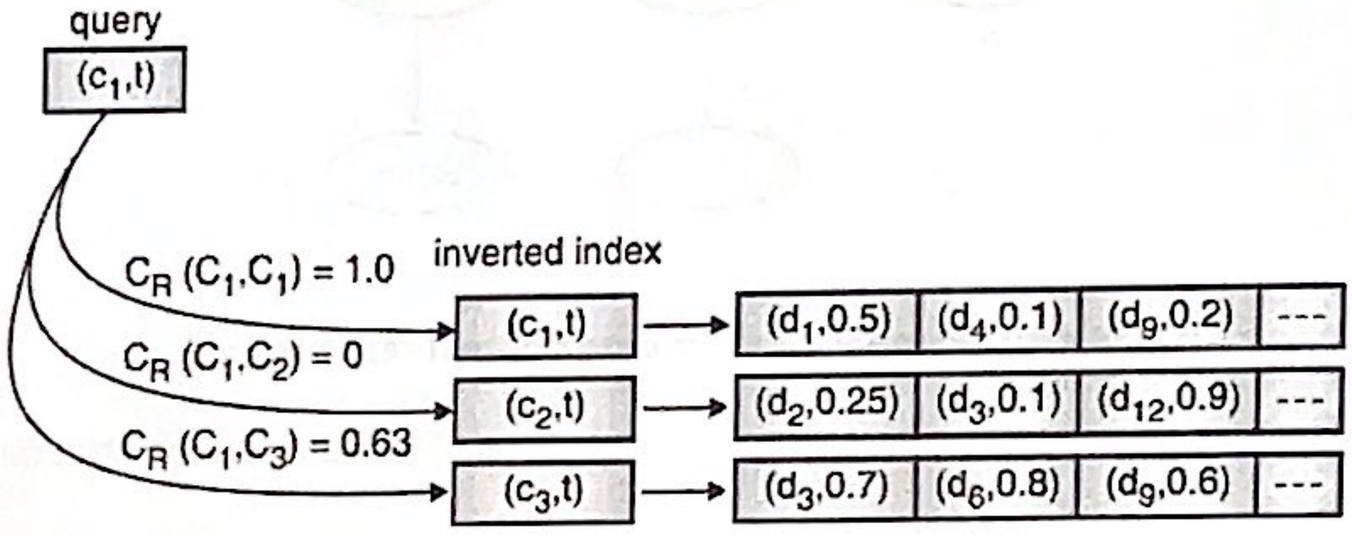


Fig. 6.4.6 : Scoring of a query with one structural term in SIMNOMERGE

Ex. 6.4.1 : Compute the query-document similarities in Fig. 6.4.6.

Soln. :

<$c1$, t> is one of the structural terms in the query.

We successively retrieve all postings lists for structural terms <c', t>i with the same vocabulary term t. Three example postings lists are shown.

- For the first one, we have $C_R(c_1, c_1) = 1.0$ since the two contexts are identical.
- The next context has no context resemblance with c_1 : $CR(c_1, c_2) = 0$. The corresponding postings list is ignored.
- The context match of c_1 with c_3 is 0.63>0 and it will be processed.

In this example, the highest ranking document is d_9 with a similarity of $1.0 \times 0.2 + 0.63 \times 0.6 = 0.578$. To simplify the figure, the query weight of hc_1, ti is assumed to be 1.0.

- **Alternative similarity function (SIMMERGE) :** SIMMERGE relaxes the matching conditions of query and document further in the following three ways.
- We collect the statistics used for computing weight (*q*, *t*, *c*) and weight (*d*, *t*, *c*) from *all* contexts that have a non-zero resemblance to *c* (as opposed to just from *c* as in SIMNOMERGE). For instance, for computing the document frequency of the structural term atl#"recognition", we can also count occurrences of recognition in XML contexts fm/atl, article//atl etc.
- Modify Equation of SIMNOMERGE by merging all structural terms in the document that have a non-zero context resemblance to a given query structural term. For example, the contexts /play/act/scene/title and /play/title in the document will be merged when matching against the query term /play/title#"Macbeth".
- The context resemblance function is further relaxed : Contexts have a nonzero resemblance in many cases where the definition of C_R in Equation of $C_R(c_q, c_d)$ returns 0.

6.5 Evaluation of XML Retrieval

- The INitiative for the Evaluation of XML retrieval (INEX) program is the premier venue for XML retrieval.
- It is a collaborative effort that has produced reference collections, sets of queries, and relevance judgments.
- A yearly INEX meeting is held to present and discuss research results.
- The INEX 2002 collection consisted of about 12,000 articles from IEEE journals.
- By 2004, the collection consisted of the full text of 12,107 articles marked up in XML from 12. Journal of IEEE Computer Society publications and 6 transactions covering the period from 1995. In 2002, the total size was 494 MB and the number of items was 8 million. Included collection Academic papers of various lengths. On average, the article contains 1,532 XML nodes. The knot depth is 6.9. In 2005, the collection was expanded to include more IEEE publications. Computer society. A total of 4,712 new articles were added between 2002 and 2004, for a total. 16,819 articles with a total size of 764MB and 11 million items.
- INEX 2006 uses another document collection consisting of English documents from Wikipedia. This collection consists of the full text of 659,388 articles from the Wikipedia project, tagged with XML. 113,483 Category 4 tiers totaling over 60 GB (4.6 GB without images) and 30 million elements. The structure of the collection is similar to the IEEE collection, but It is a larger set of tags (1,241 unique tags compared to 176 in the IEEE collection) and contains a large number of tags. Links between documents (represented as XLinks).
- There are two types of information required by INEX : content-only or CO issues, and Content And Structure (CAS) issues. CO topics are regular keyword queries, just as they are for unstructured information gathering. CAS topics have structural constraints in addition to keywords. Relevance assessment is more complex than unstructured search because CAS queries have both structural and content-related criteria.
- INEX 2002 defines component coverage and topic relevance as orthogonal dimensions of relevance. The component coverage dimension evaluates whether the retrieved item is "structurally" correct, that is, it can never be too low or too high in the tree. Distinguish between four cases.
  i) Accurate coverage (E) : The information sought is the main topic of the component, and the component is a meaningful unit of information.

ii) Too small (S) : The information sought is the main subject of the component, but the component is not meaningful (self-contained) information.

iii) It's too big (L) : The information you're looking for is in the component, but not the main topic.

iv) No cover (N) : The information you are looking for is not a component topic.

- The topical relevance dimension also has four levels : highly relevant (3), fairly relevant (2), marginally relevant (1) and nonrelevant (0). Components are judged on both dimensions and the judgments are then combined into a digit-letter code. 2S is a fairly relevant component that is too small and 3E is a highly relevant component that has exact coverage. In theory, there are 16 combinations of coverage and relevance, but many cannot occur. For example, a nonrelevant component cannot have exact coverage, so the combination 3N is not possible.

The combination of relevance and coverage is quantified as follows :

$$Q(rel, cov) = \begin{cases} 1.00 & \text{if } (rel, cov) = 3E \\ 0.75 & \text{if } (rel, cov) \in \{2E, 3L\} \\ 0.50 & \text{if } (rel, cov) \in \{1E, 2L, 2S\} \\ 0.25 & \text{if } (rel, cov) \in \{1S, 1L\} \\ 0.00 & \text{if } (rel, cov) = 0N \end{cases}$$

- This evaluation scheme takes account of the fact that binary relevance judgments, A 2S component provides incomplete information and may be difficult to interpret without more context, but it does answer the query partially. The quantization function Q does not impose a binary choice relevant non relevant and instead allows us to grade the component as partially relevant.
- The number of relevant components in a retrieved set *A* of components can then be computed as :

$$\#\text{relevant items retrieved} = \sum_{c \in A} Q(rel(c), cov(c))$$

- One flaw of measuring relevance this way is that overlap is not accounted for. This problem is worse in XML retrieval because of the problem of multiple nested elements occurring in a search result.

6.6 Text-Centric vs. data-centric XML Retrieval

Text centric	Data centric
XML document retrieval is characterized by (i) long text fields (e.g., sections of a document), (ii) inexact matching, and (iii) relevance-ranked results. Relational databases do not deal well with this use case	*Data-centric XML* mainly encodes numerical and non-text attribute value data. When querying data-centric XML, we want to impose exact match conditions in most cases. This puts the emphasis on the structural aspects of XML documents and queries. An example is: *Find employees whose salary is the same this month as it was 12 months ago.* This query requires no ranking. It is purely structural and an exact matching of the salaries in the two time periods is probably sufficient to meet the user's information need.

Text centric	Data centric
Text-centric approaches are appropriate for data that are essentially text documents, marked up as XML to capture document structure. This is becoming a de facto standard for publishing text databases since most text documents have some form of interesting structure – paragraphs, sections, footnotes etc. Examples include assembly manuals, issues of journals, Shakespeare's collected works and newswire articles	Data-centric approaches are commonly used for data collections with complex structures that mainly contain non-text data.
A text-centric retrieval engine will have a hard time with proteomic data in bioinformatics or with the representation of a city map that (together with street names and other textual descriptions) forms a navigational database.	
Text or document centric XML Examples: • XHTML webpages • Shakespeare XML • Large collections of (annotated text) • ACM, IEEE and Elsevier collection • Google books • National libraries • Parliamentary Proceedings Counterexample : MySQL dump.	Data centric XML Examples • RDF data • RSS feed • almost all examples you see in theoretical XML papers (books or cars "database") • CIA Factbook, XML-Mondial • XMark benchmark and all other XML benchmarks • MySQL database dump Counterexample : Your homepage in XHTML.
Use Case Variable, free-form data, with some fixed embedded structures.	**Use Case** XML schema-based data, with little variation and little structural change over time.
Typical Data Technical article, with author, date and title fields.	**Typical Data** Employee record
Storage Model Binary XML	**Storage Model** Object-Relational (structured)
Indexing • **XMLIndex** index with structured and unstructured components. • XML search indedx	**Indexing** B-tree index

6.7 Recommendation System : Introduction

University Question

Q. Define Recommender System ? **SPPU : Dec. 16, May 17, May 19, 4 Marks**

- "Which song is closest to the song I like the most?"
- "Which movie should I see next?"
- "What is a good place to travel for a middle class family?"
- Such subjective questions will have weird results from a 'typical' search engine.
- Search engines are generic and they will return the same results for all its users.
- So, the new systems "Recommender Systems" were proposed.
- These are intelligent systems which based on background the user or situation or sequence choices recommend certain new items.
- **Example :** A person, fan of a HERO, who has watched more than 10-15 films of the same HERO, will most probably like new film of same HERO. So, recommender system will suggest this person new film with same HERO.
- **Definition :** A system that predicts the rating or value a user will assign to an item based on pattern of user's previous choices or selections is known as recommender system.

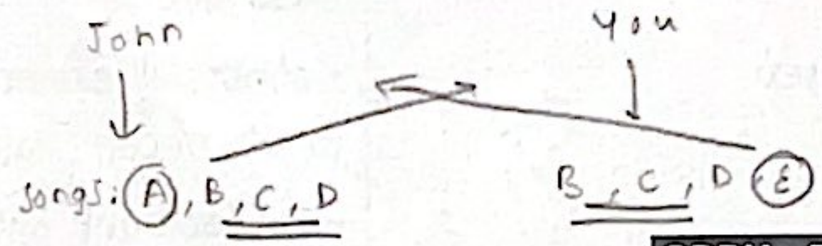
6.8 Collaborative Filtering

University Questions

Q. What is Collaborative Filtering ? SPPU : May 12, May 16, 3 Marks

Q. Describe Collaborative Filtering. SPPU : Dec. 14, 8 Marks

Q. Explain in brief collaborative filtering. SPPU : Dec. 16, May 17, May 19, 4 Marks



- Collaborative Filtering (CF) is the process of filtering information or we can call patterns using different techniques. And these techniques involve collaboration among multiple agents, viewpoints, data sources, etc.
- Collaborative Filtering (CF) is also defined as the common web technique for generating personalized recommendations.
- Examples of its use include Amazon, iTunes, Netflix, LastFM, StumbleUpon, and Delicious.
- The applications using collaborative filtering typically involve very large data sets.
- Collaborative filtering methods are used in many applications which include sensing and monitoring data, financial data, user data i.e. used in electronic commerce and web 2.0 applications.

6.8.1 Methodology

Collaborative filtering systems can have different methodologies out of that following are as shown in Fig. 6.8.1.

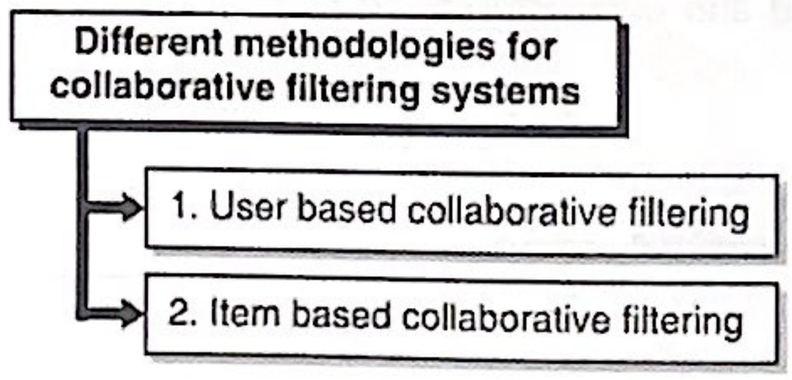


Fig. 6.8.1 : Methodologies for collaborative filtering system

1. User based collaborative filtering

- Find out all the users who shares same rating patterns with the active user i.e. the user whom the prediction is for.
- Use that ratings i.e. the rating from like-minded users to calculate the prediction for the active user.
- Specific application of this is the user-based nearest neighbour algorithm.

2. Item based collaborative filtering

- Invented by Amazon.com and proceeded in item-centric manner.
- In this by determining the relationship between pair of items an item-item matrix is build.
- Then this matrix is used to infer the data on the current user.
- That means item based collaborative filtering estimates a user's preference towards an item by looking his/her preferences towards similar item. Here similarity means similarity of rating behaviour and not the similarity of content.
- Example of this is slope one item-based collaborative filtering family.

6.8.2 Types of Collaborative Filtering

Collaborative Filtering (CF) technique is one of the most successful techniques in recommender systems. Two types of algorithms for collaborative filtering have been researched.

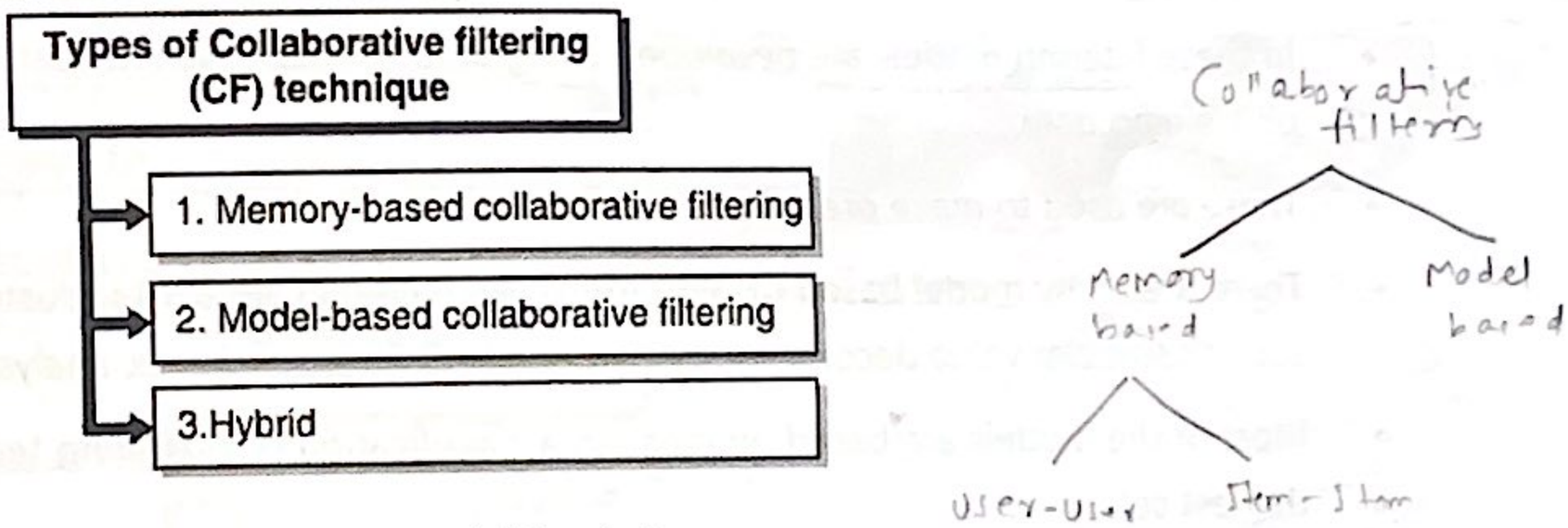


Fig. 6.8.2 : Types of CF technique

1. Memory-based collaborative filtering

- Identify the similarity between two users by comparing their ratings on a set of items.
- This is used for making recommendations.
- Typical examples of this mechanism are neighbourhood based CF and item-based or user-based top-N recommendations.
- The neighbourhood-based algorithm calculates the similarity between two users or items and then produces a prediction for the user taking the weighted average of all the ratings. Multiple mechanisms such as Pearson correlation and vector cosine based similarity are used for this.
- The user based top-N recommendation algorithm identifies the k users which are similar to an active user using similarity based vector model. After the k most similar users are found, their corresponding user-item matrices are aggregated to identify the set of items to be recommended. The method to find the similar users is the Locality sensitive hashing and this method implements the nearest neighbour mechanism in linear time.

Advantages

- The qualities of predictions are rather good.
- The explain ability of the result.
- This is a relatively simple algorithm and easy to implement.
- New data can be added easily and incrementally.
- It is very easy to update the database, since it uses the entire database every time it makes a prediction.

Disadvantages

- It uses the entire database every time it makes a prediction, so it needs to be in memory it is very, very slow.
- Even when in memory, it uses the entire database every time it makes a prediction, so it is very slow.
- It can sometimes not make a prediction for certain active users/items. This can occur if the active user has no items in common with all people who have rated the target item.
- It depends on human ratings
- Its performance decreases when data gets sparse, which is frequent with web related items.
- It cannot handle new users or new items.
- Not always as fast and scalable as we would like them to be, especially in the context of actual systems that generate real-time recommendations on the basis of very large datasets.

2. Model-based collaborative filtering

- In these filtering models are developed using data mining, machine learning algorithms to find patterns based on training data.
- These are used to make predictions for real data.
- There are many model based CF algorithms like Bayesian Networks, clustering models, latent semantic models such as singular value decomposition, probabilistic latent semantic analysis.
- Most of the models are based on creating a classification or clustering technique to identify the user based on the test set.
- Model-based collaborative filtering algorithms provide item recommendation by first developing a model of user ratings.

Advantages

- It imparts scalability to the overall system. As most of the models resulting from model-based algorithms are much smaller than the actual dataset, even for very large datasets, the model ends up being small enough to be used efficiently.
- The qualities of predictions are rather good.
- Model-based systems are faster as the time required to query the model is smaller than that required to query the whole dataset.

Disadvantages

As building a model is time- and resource-consuming process, it is more difficult to add data to model-based systems. So it is making them inflexible.

Predictions are not that much accurate as we are not using complete dataset available to us.

Model building is expensive. One needs to have a tradeoffs between prediction performance and scalability.

3. Hybrid

- To overcome the limitations of native collaborative filtering most of the applications combine the memory-based and the model-based CF algorithms.
- It improves the prediction performance.
- It overcomes the CF problems such as sparsity and loss of information.
- But they have increased complexity and are expensive to implement.

6.8.3 Advantages of Collaborative Filtering

University Question

Q. Discuss advantages of Collaborative Filtering. **SPPU : May 12, May 16, 3 Marks**

- Completely domain independent that means domain knowledge is not needed.
- Collaborative filtering has proven to be very effective at thinking out-of the box.
- The quality of its results improves over time and implicit user feedback sufficient.

6.8.4 Disadvantages of Collaborative Filtering

University Question

Q. Discuss disadvantages of Collaborative Filtering. **SPPU : May 12, May 16, 3 Marks**

As quality is dependent on large historical data set it causes two problems.

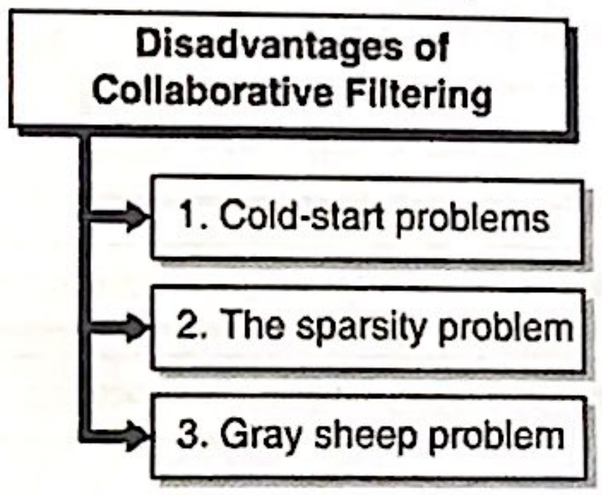


Fig. 6.8.3 : Disadvantages of collaborative filtering

1. Cold-start problems

- When a new user arrives at the system, there is no sufficient rating information to sketch user's preferences and there may be a lack of information about the user itself. Both situations must be tackled by the recommender system.
- Every time a new object is created the recommender system must recommend this new item and make to any of the users of the system that might be interested on it.

2. The sparsity problem

This problem typically occurs in systems that contains large number of items. And out of that item plenty of items are rated only by few users, and many of the users rated only few items.

3. Gray sheep problem

This problem is related with the new user problem. Some users don't rate items or they may not provide any means to extract their taste form their social interactions. Therefore, the system hasn't got enough information about them, such in the case of new users.

6.9 Content based Recommendation of Documents and Products

University Questions

Q. Explain in detail content based recommendation of documents. **SPPU : Dec. 16, 8 Marks**

Q. What is content based recommendation? **SPPU : May 17, 8 Marks**

- In contrast to the above Collaborative filtering method, Content Based method focuses on a single use or single item profile.
- In Content Based (CB) recommendation method contents of user profile or the item characteristics are given more importance.
- This requires minimum settings at starting and system grows spirally for recommendations.
- It is based on properties of the item.
- It matches item properties with user profile.
- It uses discrete values of attributes in contrast to continuous values of ratings like in case of collaborative filtering.
- Interactions of the user with the system will enhance the results spirally. Interactions will make the adjustment of the parameter values as per the taste of the user.
- This is not dependent on any other user in the system.
- Two major parts of the system are as shown in Fig. 6.9.1.

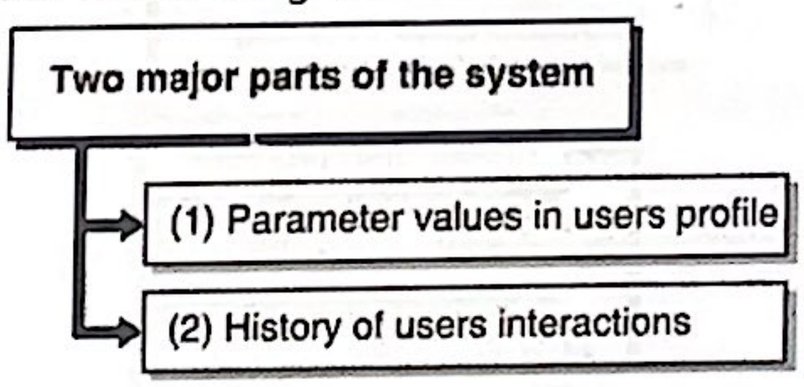


Fig. 6.9.1 : Parts of system

1. Parameter values in users profile

The parameters are fixed for the system. Each user initially is given a default set of parameter values. With his likes and dislikes parameters are fine tuned.

Example : A user preferring action movies more will have a "Action films" parameter set with higher value than "Emotional Drama" parameter.

2. History of users interactions

The value of the parameters in the user profile solely depends on the history of the user's interaction with the system. If there is minimal interaction then the system may not give best of the recommendations. For a regular user systems recommendations will be much better.

Advantages

- Easy to start the start the system.
- Can run with just one user and few products in the system.
- User profile will be adjusted as the interactions of user increase.
- Fine tuned attributes will provide efficient results and recommendations.

Disadvantages

- System is single dimensional and will recommend user similar items which user is already using. Variegated recommendations are not possible, which is possible in collaborative systems.
- Limited by the number of attributes defined in the system is a serious bottle neck of the system.
- Solely depends on how the parameters are set in the system and users number of interactions with the system.
- Users associations with other users are not taken in consideration.

6.10 Introduction to Semantic Web : Semantic Search Systems

- Semantic search systems are developed to search the relevant documents based on the user's query and the context of the query.
- Semantic search improves the quality of search with the help of context of search, variation of the query words, its synonyms, generalized and specialized queries, natural language queries. It gives more accurate results and increases the user's satisfaction.
- Semantic search is the set of techniques for retrieving knowledge from richly structured data sources like ontologies and XML as found on the Semantic Web.

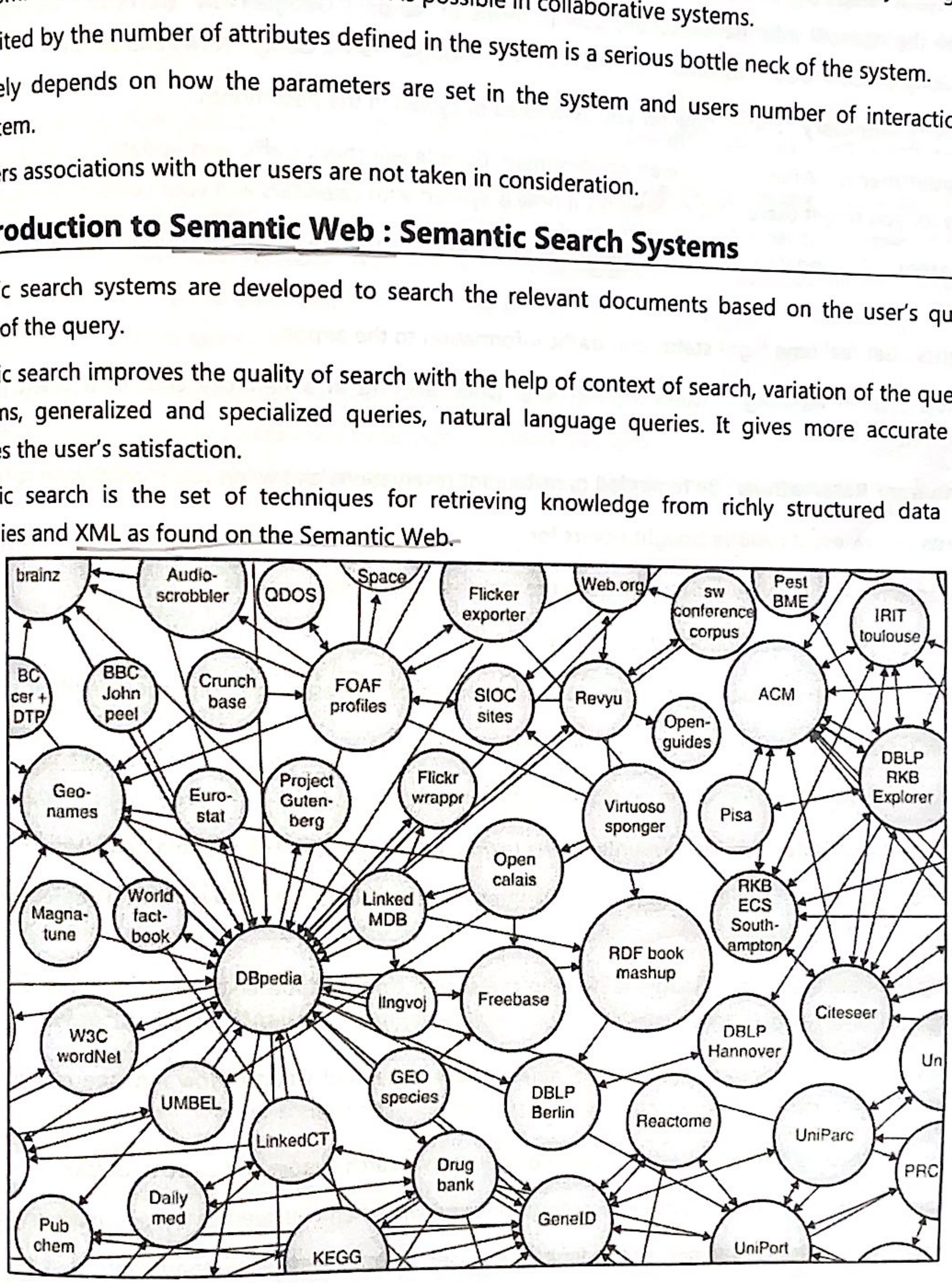


Fig. 6.10.1

6.11 Google Semantic Web

- Google has made several steps towards the semantic web. Google has developed various tools which helps user in day to day life and gives the relevant information when and where needed.
- Google has developed "Google Now" which is available on Android and iOS operating systems. It operates as an intelligent personal assistant- Predicting and conveying the information based on user's interest and current condition.
- "Google Now" keeps track of user's habits, interests, actions, places the user visits and uses the Knowledge Graph to display the relevant information to the user in terms of "cards". "Google Now" works in synchronization with various Google facilities such as Gmail, Google calendar, Google Places, Google News and so on.
  - **Activity Summary** : Shows how far you've walked or cycled in the past month.
  - **Appointments** : When you have an appointment, Google will check traffic and update you on how long it will take for you to get there. This only works if now is synced with calendars and your current location.
  - **Weather :** Get updated on the weather for where you currently are or where you will be traveling in the near future.
  - **Flights** : Get real time flight status and traffic information to the airport.
  - **Hotels** : Learn to navigate toward your hotel when arriving in a new city and learn when you need to checkout.
  - **Restaurant Reservations** : Be reminded of restaurant reservations and when you should leave to make them.
  - **Events :** Show events you've bought tickets for.
  - **Packages** : Manage online orders and track the status of packages.
  - **Birthdays** : Never forget important birthdays.
  - **Places** : Recommends places, bars, restaurants, and places of interests based on your current location.
  - **Public Transit :** Learn of public transit schedules and locations.
  - **Stocks :** Get real time information on stocks that you track.
  - **Sports** : Keep updated on your favourite sports teams or buy tickets to the next big game.
  - **Research Topics/News** : Get suggestions to interesting web pages and news articles based on your past searches.
- Considering the benefits of the "Google Now" and the strong Google Knowledge Graph, most of the businesses are syncing their business with "Google Now".
- Partnerships with Zillow, Fandango and some airlines have expanded Google Now into the categories of Real Estate and Movies.
  - **Zillow** : See nearby real estate listings based on your past search history and current location.
  - **Fandango :** See movie tickets you've recently purchased and when you need to leave to make the show time.
  - **Boarding Pass** : United Airlines and a limited number of other airlines support flight and boarding pass integration.

6.12 Google Knowledge Graph

- Consider the scenario where you are asking the spoken questions to Google assistant and it is giving you the suggestions or the answers. It is with the help of the knowledge base available with the Google which is known as Google Knowledge Graph.
- Google has the strong knowledge base which is designed with the help of various sources and it enhances the results of the Google. The information based on the user's interest is presented to the user in an info box next to the search results.
- Knowledge Graph infoboxes were added to Google's search engine in May 2012, starting in the United States, with international expansion by the end of the year. The information covered by the Knowledge Graph grew significantly after launch, tripling its size within seven months (covering 570 million entities and 18 billion facts) and answering "roughly one-third" of the 100 billion monthly searches Google processed in May 2016. The Knowledge Graph has been criticized for providing answers without source attribution or citation.
- Information from the Knowledge Graph is presented as a box, which Google has referred to as the "knowledge panel", to the right (top on mobile) of search results. According to Google, this information is retrieved from many sources, including the *CIA World Factbook*, Wikidata, and Wikipedia. In October 2016, Google announced that the Knowledge Graph held over 70 billion facts. There is no official documentation on the technology used for the Knowledge Graph implementation.
- Goggle is in process of building the largest knowledge with the help of its resources. Wikipedia, *CIA World Factbook* and Freebase are all sources used by Google to gather data about people, events, animals, events, history and other topics. Users can also become the contributor to add the data in Google Knowledge base.

6.13 Taxonomy and Ontology

A. Why develop an ontology ?

- The development of ontologies has been moving from the realm of Artificial-Intelligence laboratories to the desktops of domain experts.
- Ontologies have become common on the World-Wide Web. The ontologies on the Web range from large taxonomies categorizing Web sites (such as on Yahoo!) to categorizations of products for sale and their features (such as on Amazon.com).
- The WWW Consortium (W3C) is developing the Resource Description Framework, a language for encoding knowledge on Web pages to make it understandable to electronic agents searching for information.
- The Defense Advanced Research Projects Agency (DARPA), in conjunction with the W3C, is developing DARPA Agent Markup Language (DAML) by extending RDF with more expressive constructs aimed at facilitating agent interaction on the Web.
- Many disciplines now develop standardized ontologies that domain experts can use to share and annotate information in their fields. Medicine, for example, has produced large, standardized, structured vocabularies such as SNOMED and the semantic network of the Unified Medical Language System. Broad general-purpose ontologies are emerging as well.
- Ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them.

B. Why would someone want to develop ontology ?

Some of the reasons are as shown in Fig. 6.13.1.

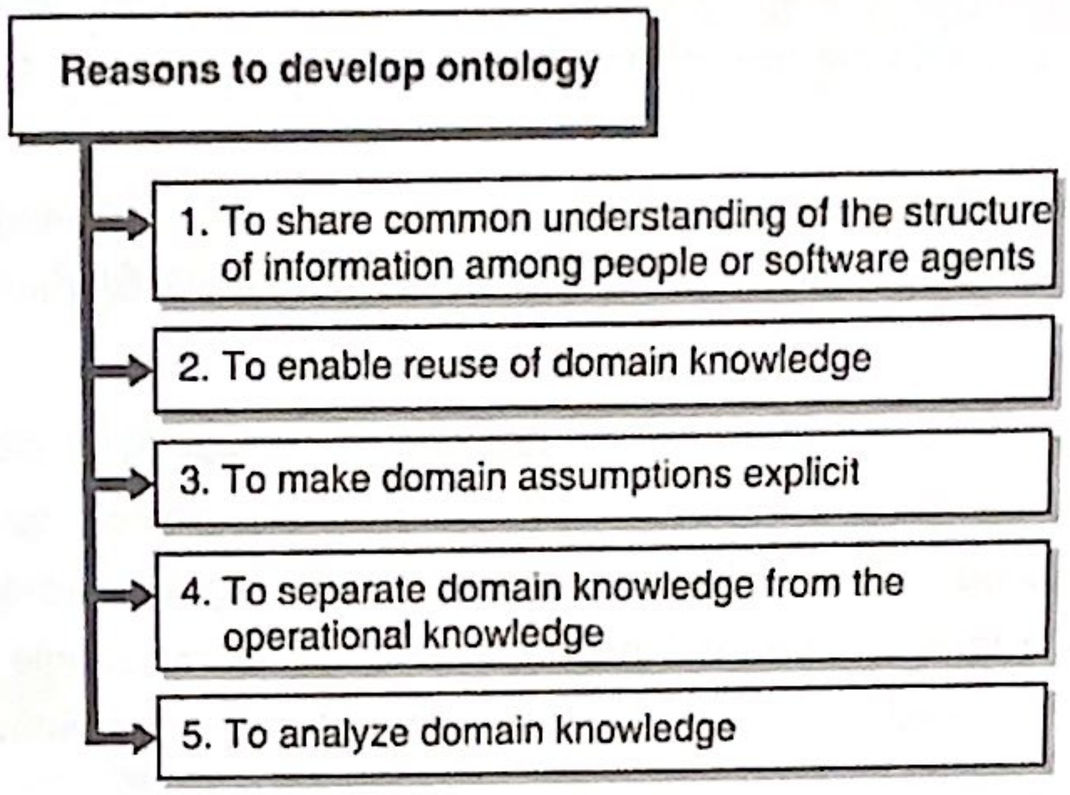


Fig. 6.13.1 : Reasons to develop ontology

1. Sharing common understanding of the structure of information among people or software agents

It is one of the more common goals in developing ontologies. For example, suppose several different Web sites contain medical information or provide medical e-commerce services. If these Web sites share and publish the same underlying ontology of the terms they all use, then computer agents can extract and aggregate information from these different sites. The agents can use this aggregated information to answer user queries or as input data to other applications.

2. Enabling reuse of domain knowledge

This was one of the driving forces behind recent surge in ontology research. For example, models for many different domains need to represent the notion of time. This representation includes the notions of time intervals, points in time, relative measures of time, and so on. If one group of researchers develops such ontology in detail, others can simply reuse it for their domains. Additionally, if we need to build a large ontology, we can integrate several existing ontologies describing portions of the large domain. We can also reuse a general ontology and extend it to describe our domain of interest.

3. Making explicit domain assumptions

Underlying an implementation makes it possible to change these assumptions easily if our knowledge about the domain changes. A hard-coding assumption about the world in programming-language code makes these assumptions not only hard to find and understand but also hard to change, in particular for someone without programming expertise. In addition, explicit specifications of domain knowledge are useful for new users who must learn what terms in the domain mean.

4. Separating the domain knowledge from the operational knowledge

It is another common use of ontologies. We can describe a task of configuring a product from its components according to a required specification and implement a program that does this configuration independent of the products and components themselves. We can then develop ontology of PC-components and characteristics and apply the algorithm to configure made-to-order PCs. We can also use the same algorithm to configure elevators if we "feed" elevator component ontology to it.

5. Analyzing domain knowledge

It is possible once a declarative specification of the terms is available. Formal analysis of terms is extremely valuable when both attempting to reuse existing ontologies and extending them.

6.13.1 Ontology

- Ontology can be defined as an explicit specification of a conceptualization. Where conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose.
- A more concrete definition of ontology is a document or file that formally defines the relations among terms.
- In few and simplest words ontology can be defined as a formal Knowledge Representation System (KRS) composed by three main elements : classes (or concepts or topics), instances (which are individuals which belongs to a class) and properties (which link classes and instances allowing to insert information's regarding the world represented into the Ontology).
- To obtain a structured representation of the information through the ontologies is one of the main objectives in order to realize the so called Semantic In the context of the Semantic Web, in fact, ontologies are expected to play an important role in helping automated processes to access information.
- In particular, ontologies are expected to be used to provide structured vocabularies that explicate the relationships between different terms, allowing intelligent agents (and humans) to interpret their meaning.
- Another important aspect regarding the role of ontologies is linked to the issue of the information overload. One of the problem of the actual World Wide Web, in fact, regards the fact that large part of the information provided to the users as output of an explicit query are un-relevant.
- Through the implementation of ontologies within dedicated information systems (e.g. search engines) this problem can be reduced or, in the future, solved completely (at least in theory) because ontologies architecture (which usually is hierarchic) should be able to design the unique path that the information required through queries must follow to arrive at the web resources containing the desired information.
- Since the 2006 RDF, RDF Schema and OWL are generally considered as standard Semantic Web languages. In particular the OWL, which is the language, is the most expressive language. It allows to augment the number (and the quality) of inferences that the software agents are able to do.

6.13.2 Taxonomy

- Taxonomy is the science of identifying and naming species, and arranging them into a classification. Taxonomies are initially designed for human consumption therefore some domain knowledge that is obvious and assumed by stakeholders is often omitted in their specifications.
- Moreover, taxonomies classifying large and complex items usually have the following characteristics :
  1. The entities being classified and the attributes upon which the classification is based, are themselves complex concepts.
  2. Multiple attributes (different concepts) might be used to classify entities at the same level.
  3. Attributes are not orthogonal and might result in overlapping concepts in low-level entities (an object can fit into multiple categories).

- There is a need for a systematic approach for annotating assumed semantics, clarifying complex concepts, and transforming them into formal representation before taxonomies can be effectively used for semantic association. Semantic depends on context and context depends on applications.
- In other words, the semantic of a standard is open depending on how they are used. To avoid a standard being bound to specific applications, the intrinsic semantic of a standard without context should include the following :

1. **Attributes** being used for classification under the general perception in the application domain and
2. **Entities** under the inheritance of the taxonomy and the attributes.

6.13.3 Ontology Development from Taxonomy

- Ontology can be defined as an explicit specification of a conceptualization. It is a description of the concepts or terms and relationships that can exist in an application domain.
- Centered on terms and relations, the transformation of taxonomy into ontology is described in the following steps.

Step 1 : Relation set identification

- The goal of this step is to identify a sufficient and necessary set of orthogonal relations for a given taxonomy/standard so that assumed domain knowledge and complex concepts can be formally specified.
- This step should be manually done by standard committees who know best about the original intended use of the standards. The set should be constructed from two types of relations : primitive and derived.

1. **Primitive relations** are those that are unambiguously understood by the general public and the relationship between concepts connected by them does not change over time.
   - Moreover, they reflect the intrinsic properties of objects or describe time and space and the intention of users when the objects are used. In addition, their definitions should include set relationship, such as instance-instance, instance-class, and class-class, to avoid ambiguity.
   - For example, part of is ambiguous since it could mean a subcomponent of an object or the membership of an object in a class. Its meaning can be identified as the first explanation if instance- instance is specified.
2. **Derived relations** are those that can be composed/modelled from primitive relations.
   - To elaborate this step, a small portion of the top three levels in Master Format taxonomy, Division 5 (D_5) Metals and Division 6 (D_6) Wood and Plastic rooted from Material, is exemplified as follows.

a. Division 5 - Metals

- 05100 Structural Metal Framing
- 05120 Structural steel
- 05140 Structural aluminium
- 05160 Metal framing systems
- 05400 Cold formed metal framing
- 05410 Load bearing metal studs
- 05420 Cold formed metal joists
- 05430 Slotted channel framing