

XML RETRIEVAL

Introduction to
Information Retrieval
CS 150
Donald J. Patterson

Content adapted from Manning, Raghavan, and Schütze
<http://www.informationretrieval.org>

XML RETRIEVAL

OVERVIEW

- Introduction
- Basic XML Concepts
- Challenges in XML IR
- Vector Space Model for XML IR
- Evaluation of XML IR

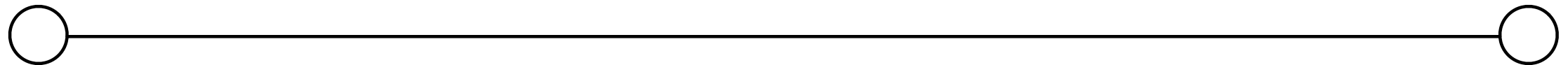


XML RETRIEVAL

A SPECTRUM

Information
Retrieval

Relational
Databases



- Information Retrieval
 - Source is raw text
 - Structure is “the document”
 - No mark-up

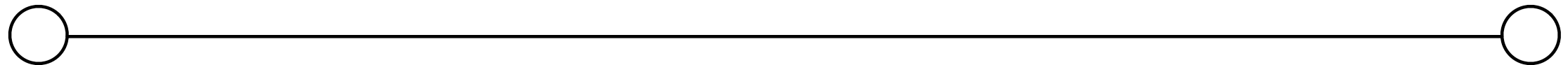


XML RETRIEVAL

A SPECTRUM

Information
Retrieval

Relational
Databases



- Relational Databases
 - Source is highly structured
 - Tables, Columns, Rows
 - Many values predefined
 - Relationships defined

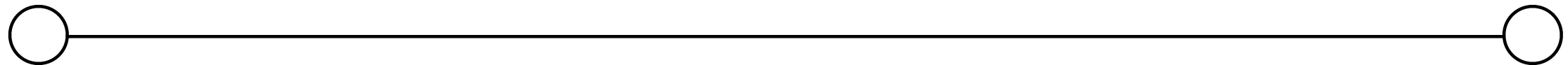


XML RETRIEVAL

A SPECTRUM

Information
Retrieval

Relational
Databases



	Unstructured IR	Relational DB
objects	unstructured documents	records
main data structure	inverted index	table
model	vector space (and others)	relational model
queries	free text	SQL

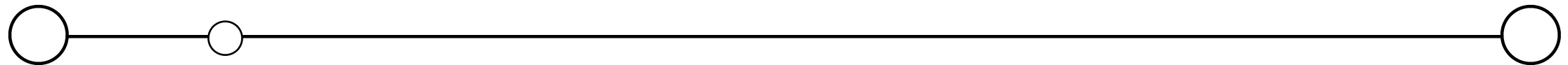


XML RETRIEVAL

A SPECTRUM

Information
Retrieval

Relational
Databases



- Zone/ Parametric Search
 - Documents have some structure
 - zones
 - “abstract”
 - “conclusion”
 - “author”
- Raw text still dominates

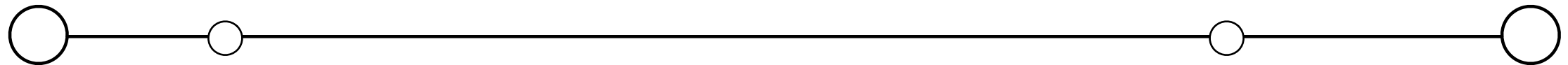


XML RETRIEVAL

A SPECTRUM

Information
Retrieval

Relational
Databases



- NoSQL
 - Documents have structure
 - Usually confined to a record
 - records aren't related
 - records can be missing data

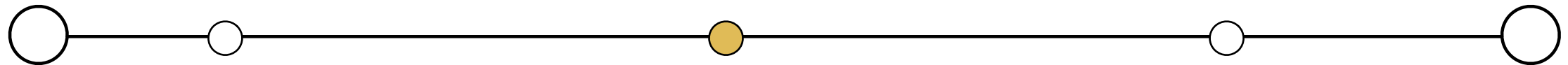


XML RETRIEVAL

A SPECTRUM

Information
Retrieval

Relational
Databases



- XML Retrieval
 - aka “Structured Retrieval”
 - documents are strongly structured with mark-up
 - queries can (optionally) contain mark-up as well



XML RETRIEVAL

APPLICATIONS OF STRUCTURED RETRIEVAL

- Digital libraries
- patent databases
- blogs
- tagged text with entities like persons and locations
 - (aka New York Times)
 - (named entity tagging)



XML RETRIEVAL

EXAMPLES OF STRUCTURED RETRIEVAL

- Digital libraries:
 - give me a full-length article on fast fourier transforms
- Patents:
 - give me patents whose claims mention RSA public key encryption and that cite US patent 4,405,829
- Entity-tagged text:
 - give me articles about sightseeing tours of the Vatican and the Coliseum



XML RETRIEVAL

WHY WE CAN'T USE SQL

- “give me articles about sightseeing tours of the Vatican and the Coliseum”
- Problem 1
 - An unranked system (DB) would return a potentially large number of articles that mention the Vatican, the Coliseum and sightseeing tours without ranking them by relevance to query



XML RETRIEVAL

WHY WE CAN'T USE SQL

- “give me articles about sightseeing tours of the Vatican and the Coliseum”
- Problem 2
 - Difficult for users to precisely state structural constraints
 - may not know which structured elements are supported by the system.
 - tours AND (COUNTRY: Vatican OR LANDMARK: Coliseum)?
 - tours AND (STATE: Vatican OR BUILDING: Coliseum)?



XML RETRIEVAL

WHY WE CAN'T USE SQL

- “give me articles about sightseeing tours of the Vatican and the Coliseum”
- Problem 3
 - Users may be completely unfamiliar with structured search and advanced search interfaces or unwilling to use them.



XML RETRIEVAL

WHY WE CAN'T USE SQL

- “give me articles about sightseeing tours of the Vatican and the Coliseum”
- Solution
 - Adapt ranked retrieval to structured documents to address these problems.



XML RETRIEVAL

STRUCTURED RETRIEVAL

	Unstructured IR	Structured IR	Relational DB
objects	unstructured documents	Trees with Text at Leaves	records
main data structure	inverted index	?	table
model	vector space (and others)	?	relational model
queries	free text	?	SQL





XML RETRIEVAL

OVERVIEW

- Introduction
- Basic XML Concepts
- Challenges in XML IR
- Vector Space Model for XML IR
- Evaluation of XML IR



XML RETRIEVAL

XML DOCUMENT

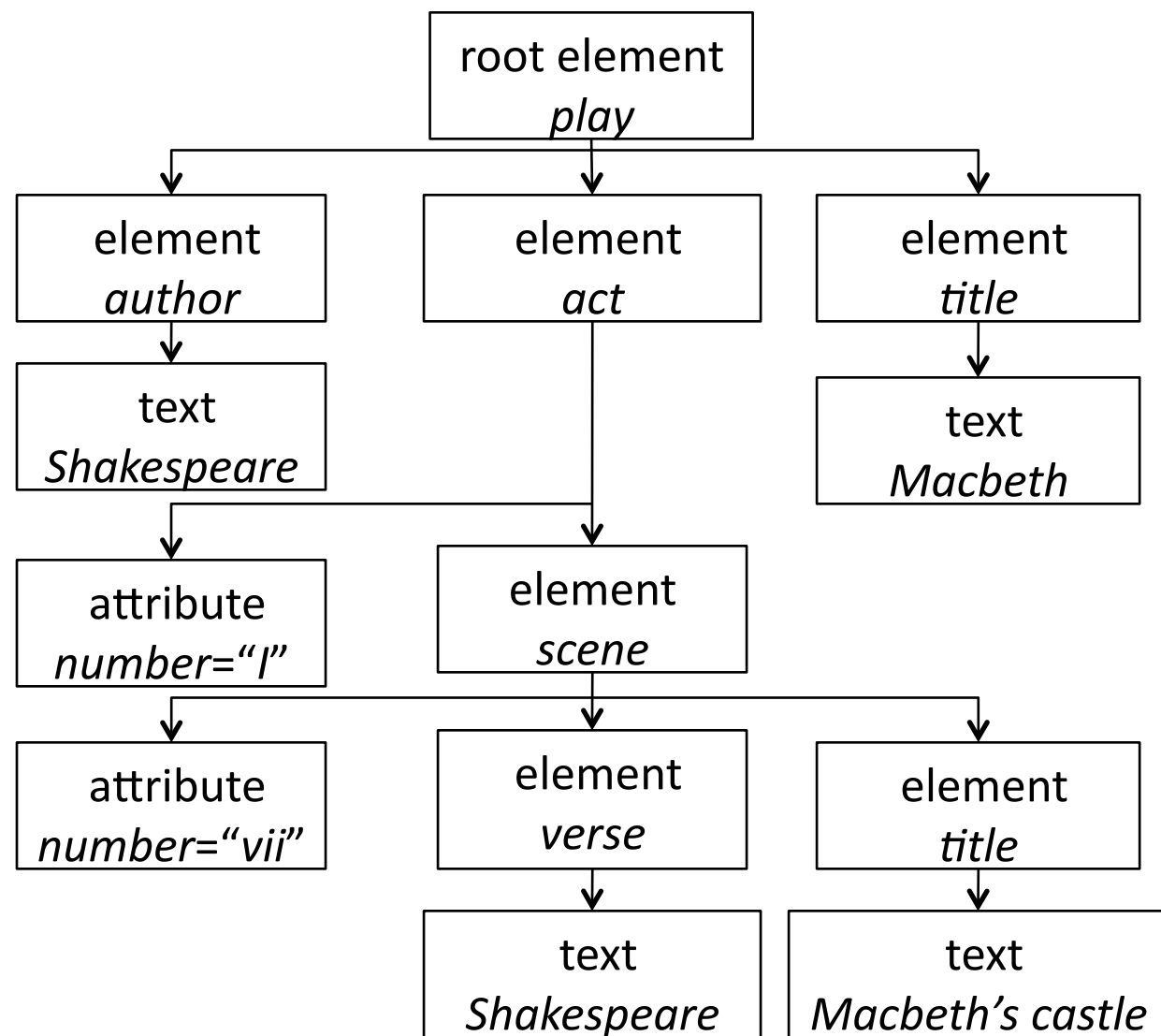
- Ordered, labeled tree
- Each node of the tree is an XML element, written with an opening and closing XML tag (e.g. <title...>, </title...>)
- An element can have one or more XML attributes (e.g. number)
- Attributes can have values (e.g. vii)
- Attributes can have child elements (e.g. title, verse)

```
<?xml version="1.0" encoding="UTF-8"?>
<play>
  <author>Shakespeare</author>
  <title>Macbeth</title>
  <act number="I">
    <scene number="vii">
      <title>Macbeth's castle</title>
      <verse>Will I with wine
        ...</verse>
    </scene>
  </act>
</play>
```



XML RETRIEVAL

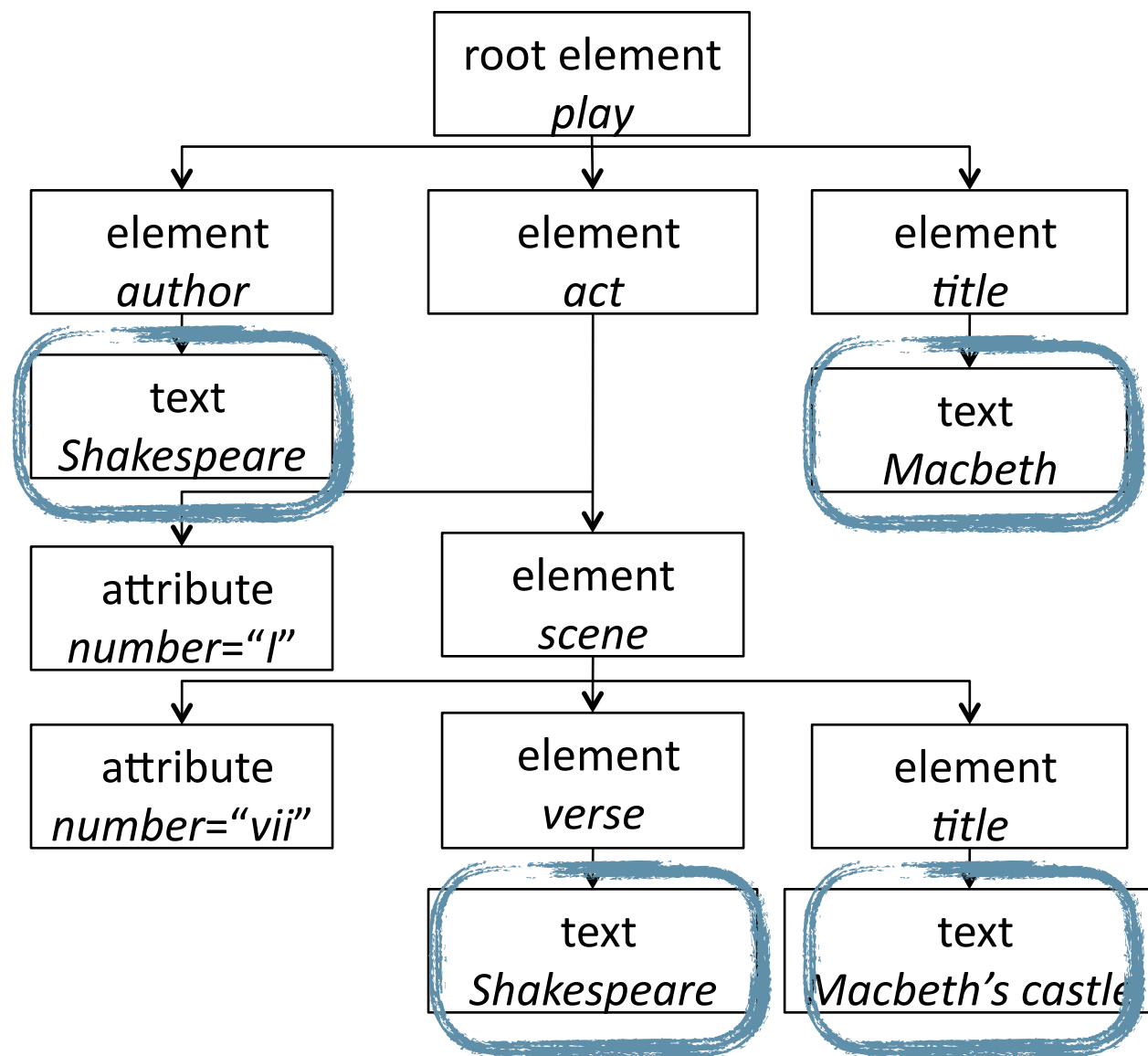
XML DOCUMENT AS DOM



```
<?xml version="1.0" encoding="UTF-8"?>
<play>
  <author>Shakespeare</author>
  <title>Macbeth</title>
  <act number="I">
    <scene number="vii">
      <title>Macbeth's castle</title>
      <verse>Will I with wine
        ...</verse>
    </scene>
  </act>
</play>
```

XML RETRIEVAL

XML DOCUMENT AS DOM



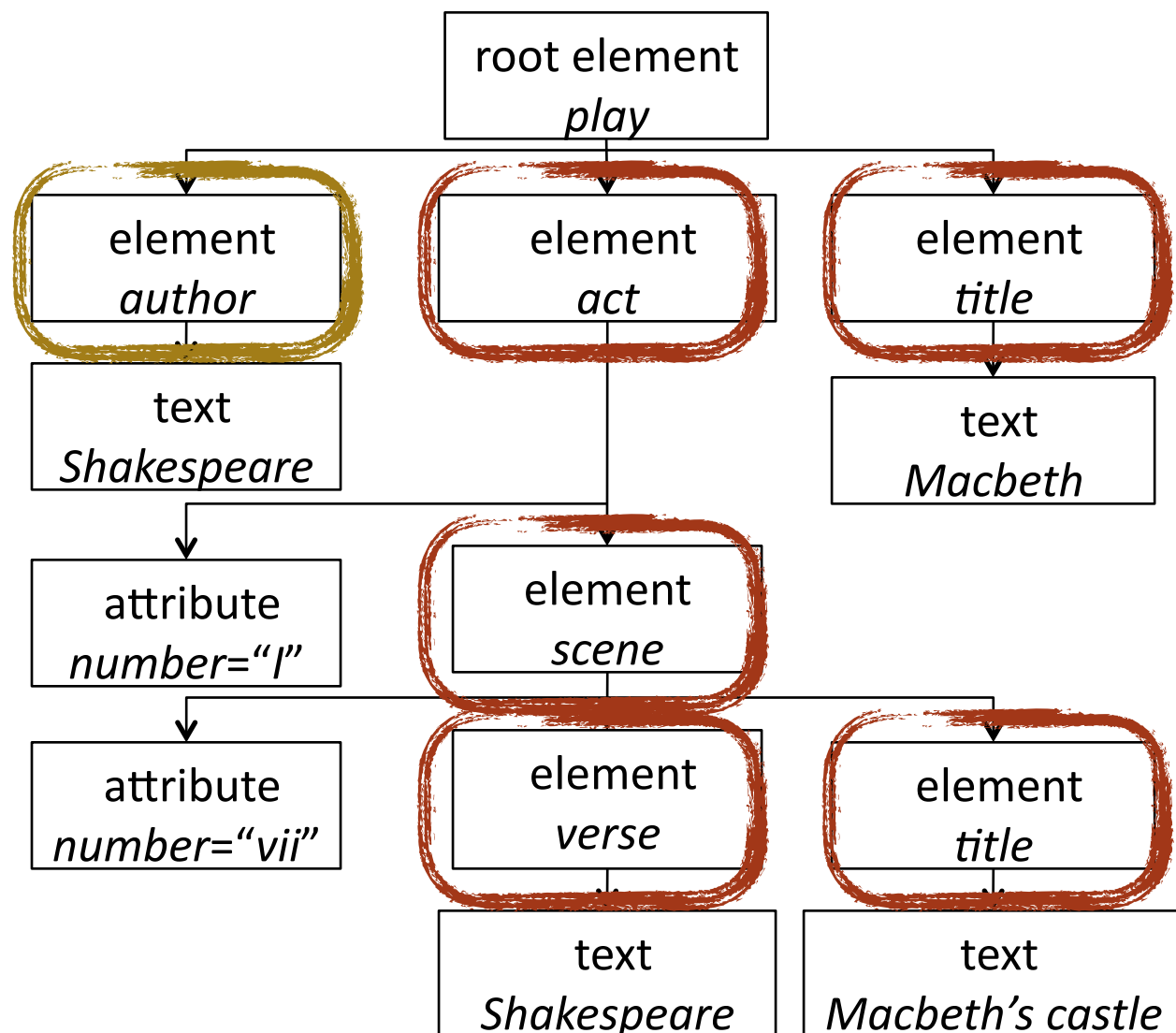
```
<?xml version="1.0" encoding="UTF-8"?>
<play>
  <author>Shakespeare</author>
  <title>Macbeth</title>
  <act number="I">
    <scene number="vii">
      <title>Macbeth's castle</title>
      <verse>Will I with wine
        ...</verse>
    </scene>
  </act>
</play>
```



- Leaf nodes consist of text

XML RETRIEVAL

XML DOCUMENT AS DOM



```
<?xml version="1.0" encoding="UTF-8"?>
<play>
  <author>Shakespeare</author>
  <title>Macbeth</title>
  <act number="I">
    <scene number="vii">
      <title>Macbeth's castle</title>
      <verse>Will I with wine
        ...</verse>
    </scene>
  </act>
</play>
```

10

- Internal nodes are
 - document structure
 - meta data

XML RETRIEVAL

XML DOCUMENT OBJECT MODEL (DOM)

- standard for accessing and processing XML documents
- The DOM represents elements, attributes and text within elements as nodes in a tree.
- With a DOM API, we can process an XML documents by starting at the root element and then descending down the tree from parents to children.



XML RETRIEVAL

XPATH

- standard for enumerating path in an XML document collection.
- We will also refer to paths as XML contexts or simply contexts



XML RETRIEVAL

SCHEMA

- puts constraints on the structure of allowable XML documents.
- E.g. a schema for Shakespeare's plays: scenes can occur as children of acts.
- Two standards for schemas for XML documents are:
 - XML DTD (document type definition)
 - XML Schema



XML RETRIEVAL

OVERVIEW

- Introduction
- Basic XML Concepts
- Challenges in XML IR
- Vector Space Model for XML IR
- Evaluation of XML IR



XML RETRIEVAL

FIRST CHALLENGE

- Which document parts to retrieve
- users want us to return parts of documents (i.e., XML elements), not entire documents as IR systems usually do in unstructured retrieval.



XML RETRIEVAL

FIRST CHALLENGE

- Example
 - If we query Shakespeare's plays for Macbeth's castle, should we return the scene, the act or the entire play?

```
<?xml version="1.0" encoding="UTF-8"?>
<play>
  <author>Shakespeare</author>
  <title>Macbeth</title>
  <act number="I">
    <scene number="vii">
      <title>Macbeth's castle</title>
      <verse>Will I with wine
      ...</verse>
    </scene>
  </act>
</play>
```



XML RETRIEVAL

FIRST CHALLENGE

- Example
 - If we query Shakespeare's plays for Macbeth's castle, should we return the scene, the act or the entire play?
 - In this case, the user is probably looking for the scene.
 - However, an otherwise unspecified search for Macbeth should return the play of this name, not a subunit.



XML RETRIEVAL

FIRST CHALLENGE

- Example
 - If we query Shakespeare's plays for Macbeth's castle, should we return the scene, the act or the entire play?
 - In this case, the user is probably looking for the scene.
 - However, an otherwise unspecified search for Macbeth should return the play of this name, not a subunit.
- Solution
 - structured document retrieval principle



XML RETRIEVAL

STRUCTURED DOCUMENT RETRIEVAL PRINCIPLE

- One criterion for selecting the most appropriate part of a document:

A system should always retrieve the most specific part of a document answering the query.

- Motivates a retrieval strategy that returns the smallest unit that contains the information sought, but does not go below this level.



XML RETRIEVAL

STRUCTURED DOCUMENT RETRIEVAL PRINCIPLE

- Hard to implement this principle algorithmically.
- E.g. query: title:Macbeth
 - can match both the title of the tragedy, Macbeth, and the title of Act I, Scene vii, Macbeth's castle.
- But in this case, the title of the tragedy (higher node) is preferred.
- Difficult to decide which level of the tree satisfies the query.



XML RETRIEVAL

SECOND CHALLENGE

- What document parts to index?
- Central notion for indexing and ranking in IR: documents unit or **indexing unit**.
- In unstructured retrieval, usually straightforward: files on your desktop, email messages, web pages on the web etc.



XML RETRIEVAL

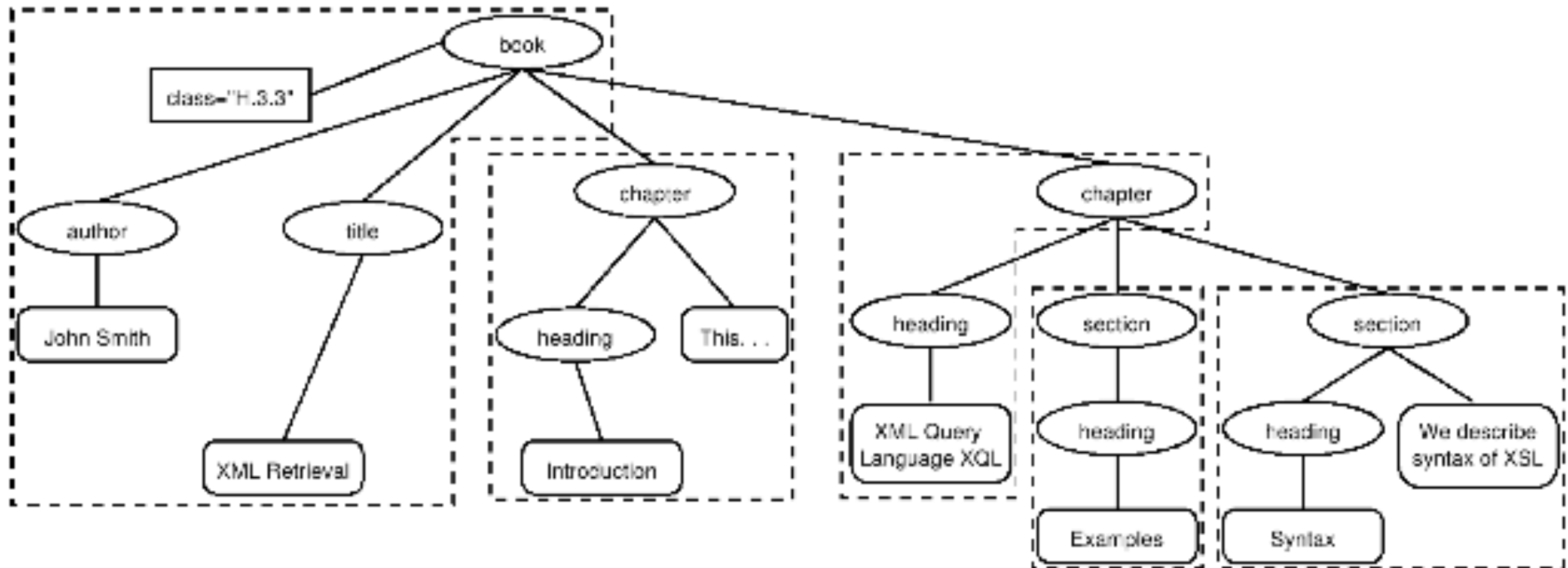
SECOND CHALLENGE

- What document parts to index?
- Central notion for indexing and ranking in IR: documents unit or **indexing unit**.
- In structured retrieval, there are four main different approaches to defining the indexing unit
 - non-overlapping pseudodocuments
 - top down
 - bottom up
 - all



XML RETRIEVAL

APPROACH 1: PSUEDODODOCUMENTS



- Indexing units: books, chapters, section, but without overlap.
- Disadvantage: pseudodocuments may not make sense to the user

XML RETRIEVAL

APPROACH 2: TOP DOWN

- Top down (2-stage process):
 - Start with one of the latest elements as the indexing unit, e.g. the book element in a collection of books
 - Then, postprocess search results to find for each book the subelement that is the best hit.
 - This two-stage retrieval process often fails to return the best subelement because the relevance of a whole book is often not a good predictor of the relevance of small subelements within it.



XML RETRIEVAL

APPROACH 3: BOTTOM UP

- Search all leaves, select the most relevant ones and then extend them to larger units in postprocessing.
- Similar problem as top down: the relevance of a leaf element is often not a good predictor of the relevance of elements it is contained in.



INDEX ALL



THE XML

XML RETRIEVAL

APPROACH 4: ALL ELEMENTS

- Example:
 - For the query Macbeth's castle we would return all of the play, act, scene and title elements on the path between the root node and Macbeth's castle. The leaf node would then occur 4 times in the result set: 1 directly and 3 as part of other elements.



XML RETRIEVAL

APPROACH 4: ALL ELEMENTS

- Example:
 - We call elements that are contained within each other **nested elements**.
 - Returning redundant nested elements in a list of returned hits is not very user-friendly.



XML RETRIEVAL

THIRD CHALLENGE

- What to do about nested elements?
- Because of the redundancy caused by the nested elements it is common to restrict the set of elements eligible for retrieval.



XML RETRIEVAL

THIRD CHALLENGE: NESTED ELEMENTS

- Restriction strategies include:
 - discard all small elements
 - discard all element types that users do not look at (analyzing XML retrieval system logs)
 - 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.



XML RETRIEVAL

THIRD CHALLENGE: NESTED ELEMENTS

- Other options:
 - remove nested elements in a postprocessing step to reduce redundancy.
 - collapse several nested elements in the results list and use highlighting of query terms to draw the user's attention to the relevant passages.



XML RETRIEVAL

THIRD CHALLENGE: NESTED ELEMENTS

- Why might highlighting work?
- Gain 1:
 - enables users to scan medium-sized elements (e.g., a section); thus, if the section and the paragraph both occur in the results list, it is sufficient to show the section.
- Gain 2:
 - paragraphs are presented in-context (i.e., their embedding section). This context may be helpful in interpreting the paragraph.



XML RETRIEVAL

THIRD CHALLENGE: NESTED ELEMENTS

- Nesting makes statistics hard
- We may need to distinguish different contexts of a term when we compute term statistics for ranking, in particular inverse document frequency (idf).
- Example:
 - The term **Gates** under the node **author** is unrelated to an occurrence under a content node like **section** if used to refer to the plural of “gate”. It makes little sense to compute a single document frequency for Gates in this example.



XML RETRIEVAL

THIRD CHALLENGE: NESTED ELEMENTS

- Nesting makes statistics hard
- Solution
 - compute idf for XML-context term pairs.
 - sparse data problems (many XML-context pairs occur too rarely to reliably estimate df)
 - compromise:
 - consider the parent node x of the term and not the rest of the path from the root to x to distinguish contexts.



XML RETRIEVAL

OVERVIEW

- Introduction
- Basic XML Concepts
- Challenges in XML IR
- Vector Space Model for XML IR
- Evaluation of XML IR



XML RETRIEVAL

LEXICALIZED SUBTREES

- Goal
 - to have each dimension of the vector space encode a word together with its position within the XML tree.
- How
 - Map XML documents to lexicalized subtrees.



REMEMBER OUR TERM DOCUMENT MATRIX



XML RETRIEVAL

TERMS WITH STRUCTURE

- There is a tradeoff between the dimensionality of the space and the accuracy of query results.
- If we restrict dimensions to vocabulary terms, then we have a standard vector space retrieval system that will retrieve many documents that do not match the structure of the query (e.g., Gates in the title as opposed to the author element).
- If we create a separate dimension for each lexicalized subtree occurring in the collection, the dimensionality of the space becomes too large.



XML RETRIEVAL

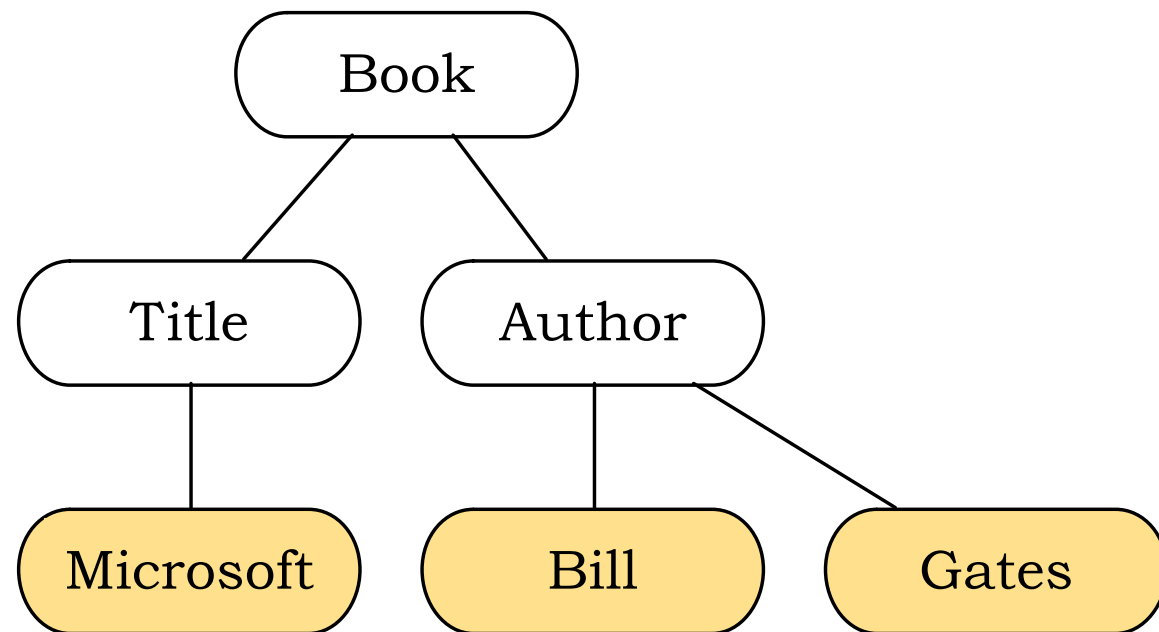
TERMS WITH STRUCTURE

- Compromise:
 - index all paths that end in a single vocabulary term, in other words all XML-context term pairs. We call such an XML-context term pair a structural term and denote it by $\langle c, t \rangle$: a pair of XML-context c and vocabulary term t .



XML RETRIEVAL

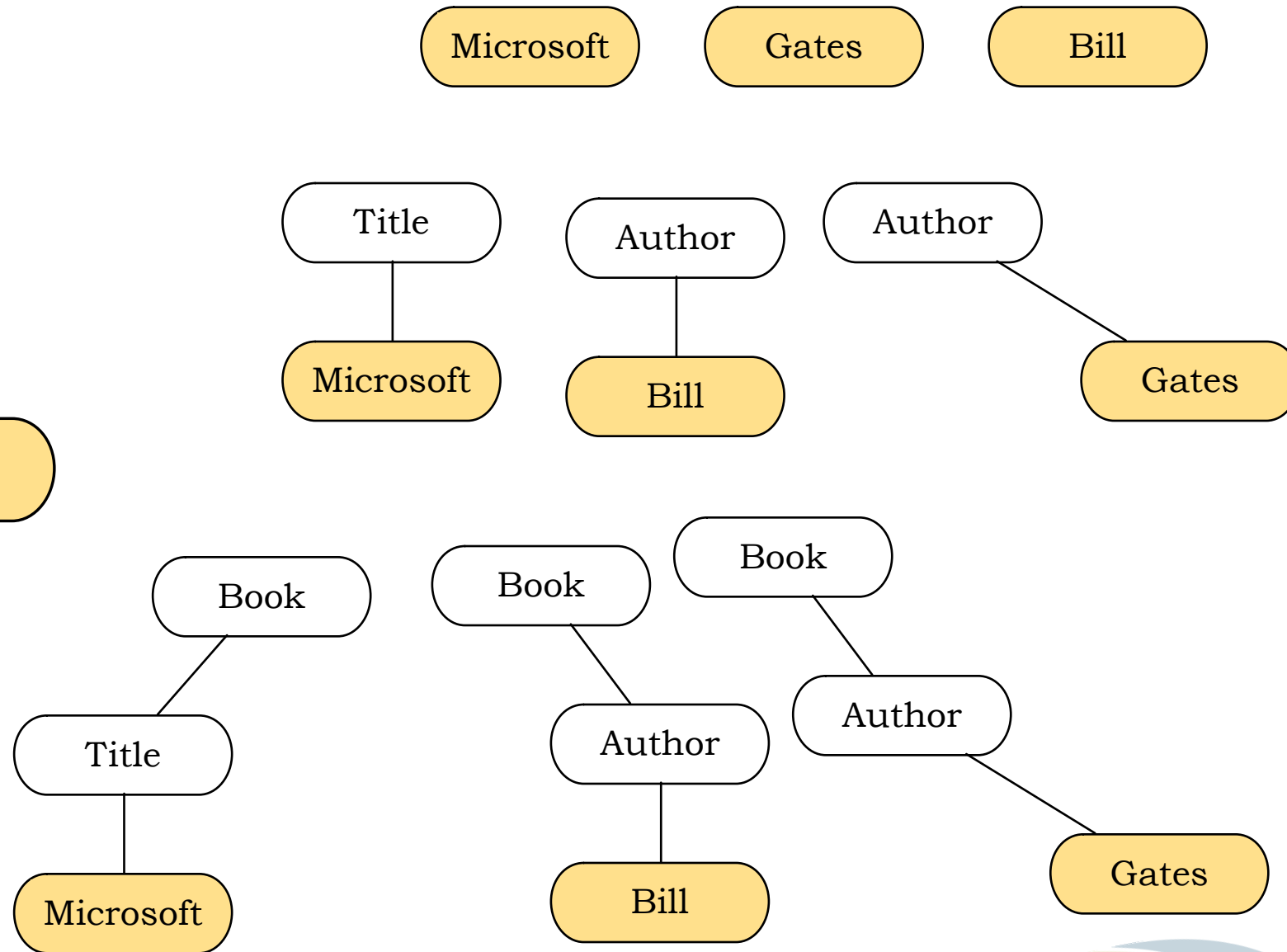
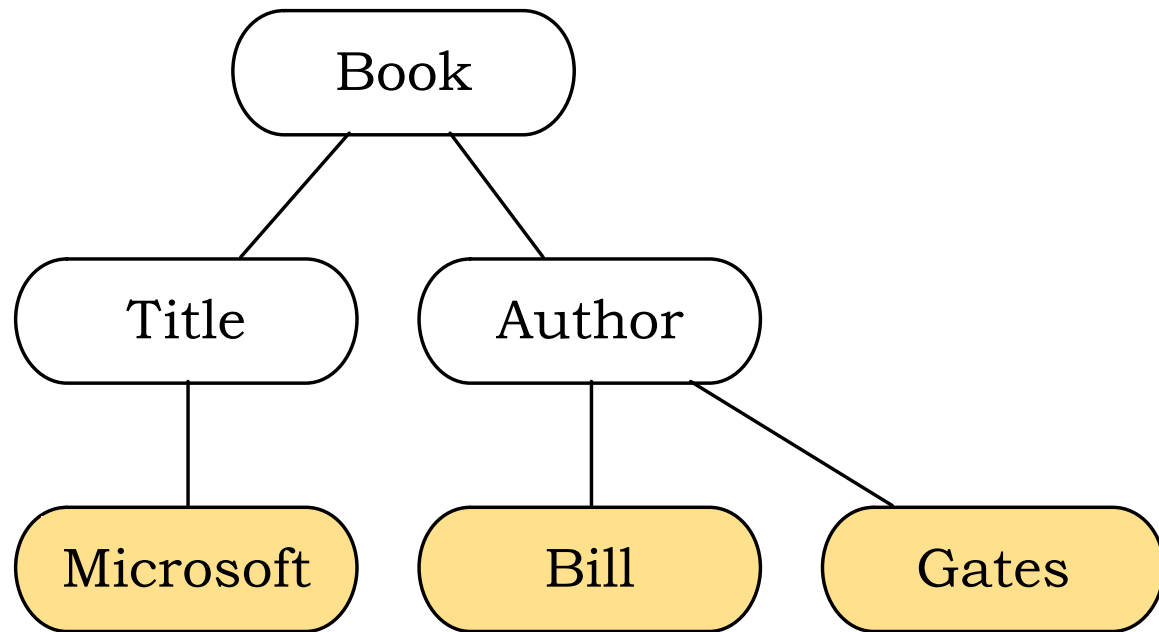
LEXICALIZED SUBTREES



- Take each text node (leaf) and break it into multiple nodes, one for each word. E.g. split Bill Gates into Bill and Gates
- Define the dimensions of the vector space to be lexicalized subtrees of documents – subtrees that contain at least one vocabulary term.

XML RETRIEVAL

LEXICALIZED SUBTREES



XML RETRIEVAL

LEXICALIZED SUBTREES

- We can now represent queries and documents as vectors in this space of lexicalized subtrees and compute matches between them,
 - e.g. using the vector space formalism.
- Vector space formalism in unstructured VS. structured IR
 - The main difference is that the dimensions of vector space in unstructured retrieval are vocabulary terms whereas they are lexicalized subtrees in XML retrieval.




XML RETRIEVAL

COMPARING SUBTREES

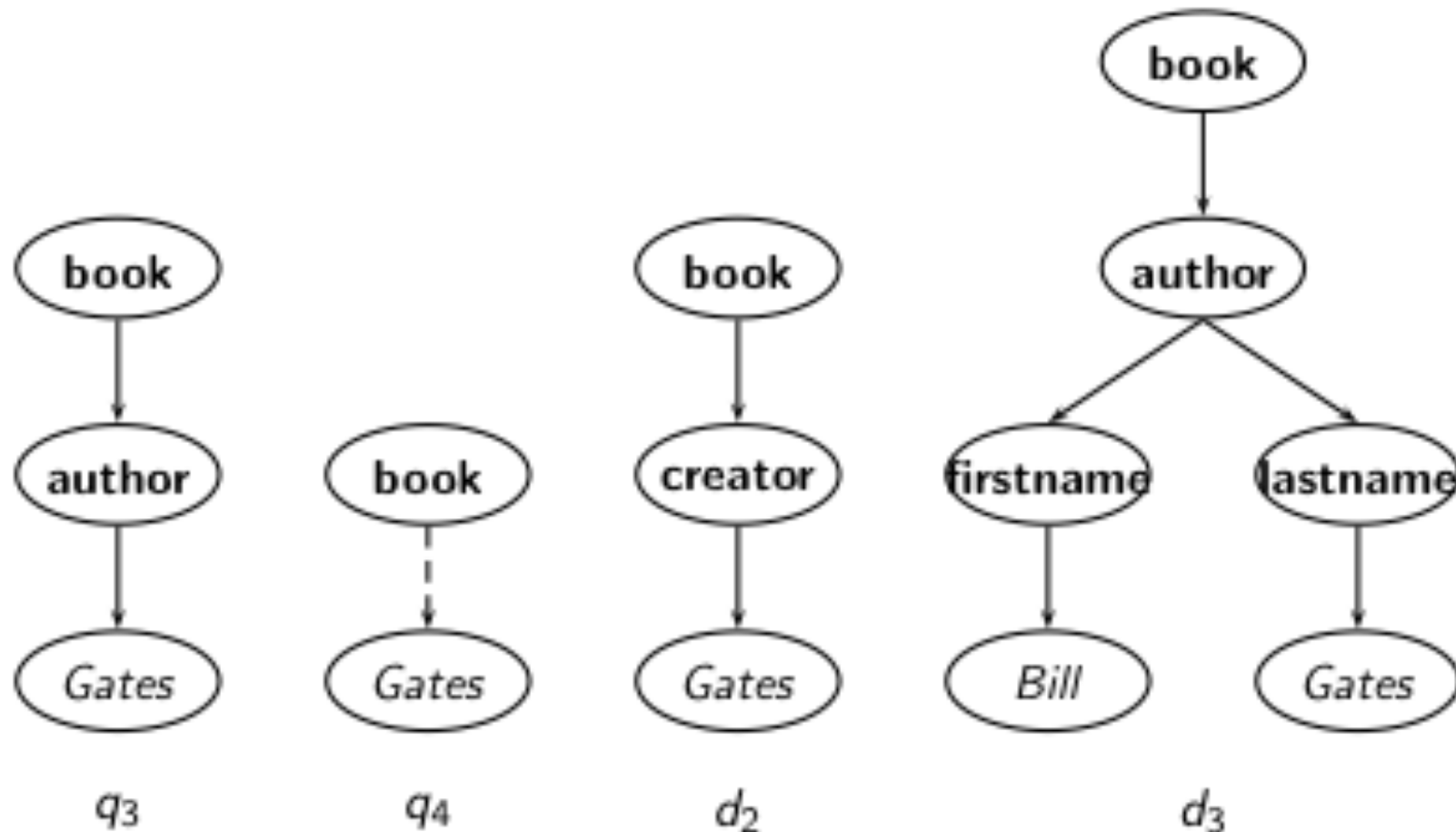
- A simple measure of the similarity of a path c_q in a query and a path c_d in a document is the following context resemblance function Cr :

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

- $|c_q|$ and $|c_d|$ are the number of nodes in the query path and document path, resp.
 - c_q matches c_d iff we can transform c_q into c_d by inserting additional nodes.
- 

XML RETRIEVAL

COMPARING SUBTREES



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


$\text{CR}(c_{q4}, c_{d2}) = 3/4 = 0.75$. The value of $\text{CR}(c_q, c_d)$ is 1.0 if q and d are identical.

XML RETRIEVAL

DOCUMENT SIMILARITY MEASURE

- The final score for a document is computed as a variant of the cosine measure, which we call SimNoMerge(q,d).

$$\sum_{c_k \in B} \sum_{c_l \in B} CR(c_k, c_l) \sum_{t \in V} \text{weight}(q, t, c_k) \frac{\text{weight}(d, t, c_l)}{\sqrt{\sum_{c \in B, t \in V} \text{weight}^2(d, t, c)}}$$

- V is the vocabulary of non-structural terms
 - B is the set of all XML contexts
 - weight (q, t, c), weight(d, t, c) are the weights of term t in XML context c in query q and document d, resp. (standard weighting e.g. $\text{idf}_t \times \text{wf}_{t,d}$, where idf_t depends on which elements we use to compute df_t .)
 - SimNoMerge(q, d) is not a true cosine measure since its value can be larger than 1.0.
- 

XML RETRIEVAL

SIMNOMERGE ALGORITHM

SCOREDOCUMENTSWITHSIMNOMERGE($q, B, V, N, \text{normalizer}$)

```
1  for  $n \leftarrow 1$  to  $N$ 
2  do  $\text{score}[n] \leftarrow 0$ 
3  for each  $\langle c_q, t \rangle \in q$ 
4  do  $w_q \leftarrow \text{WEIGHT}(q, t, c_q)$ 
5      for each  $c \in B$ 
6      do if  $\text{CR}(c_q, c) > 0$ 
7          then  $\text{postings} \leftarrow \text{GETPOSTINGS}(\langle c, t \rangle)$ 
8              for each  $\text{posting} \in \text{postings}$ 
9                  do  $x \leftarrow \text{CR}(c_q, c) * w_q * \text{weight}(\text{posting})$ 
10                      $\text{score}[\text{docID}(\text{posting})] + = x$ 
11 for  $n \leftarrow 1$  to  $N$ 
12 do  $\text{score}[n] \leftarrow \text{score}[n] / \text{normalizer}[n]$ 
13 return  $\text{score}$ 
```



XML RETRIEVAL

OVERVIEW

- Introduction
- Basic XML Concepts
- Challenges in XML IR
- Vector Space Model for XML IR
- Evaluation of XML IR



XML RETRIEVAL

INITIATIVE FOR THE EVALUATION OF XML RETRIEVAL (INEX)

INEX 2002 collection statistics

12,107	number of documents
494 MB	size
1995—2002	time of publication of articles
1,532	average number of XML nodes per document
6.9	average depth of a node
30	number of CAS topics
30	number of CO topics



XML RETRIEVAL

INEX TOPICS

- Two types:
 - content-only or CO topics: regular keyword queries as in unstructured information retrieval
 - content-and-structure or CAS topics: have structural constraints in addition to keywords
- Since CAS queries have both structural and content criteria, relevance assessments are more complicated than in unstructured retrieval



XML RETRIEVAL

INEX RELEVANCE ASSESSMENTS

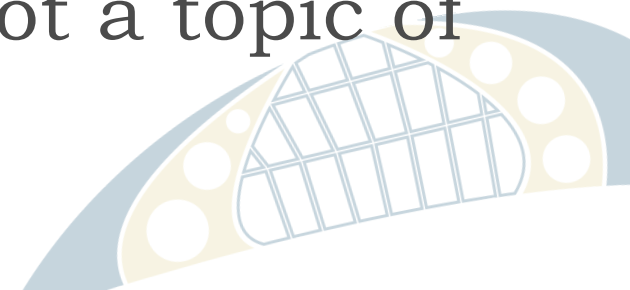
- INEX 2002 defined component coverage and topical relevance as orthogonal dimensions of relevance.
- **Component coverage**: Evaluates whether the element retrieved is “structurally” correct, i.e., neither too low nor too high in the tree.



XML RETRIEVAL

INEX RELEVANCE ASSESSMENTS

- **Component coverage:**
 - Exact coverage (E): The information sought is the main topic of the component and the component is a meaningful unit of information.
 - Too small (S): The information sought is the main topic of the component, but the component is not a meaningful (self-contained) unit of information.
 - Too large (L): The information sought is present in the component, but is not the main topic.
 - No coverage (N): The information sought is not a topic of the component.



XML RETRIEVAL

INEX RELEVANCE ASSESSMENTS

- **Topical Relevance**
 - highly relevant (3),
 - fairly relevant (2),
 - marginally relevant (1)
 - nonrelevant (0).



XML RETRIEVAL

INEX RELEVANCE ASSESSMENTS

- Combined

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

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



XML RETRIEVAL

INEX RELEVANCE ASSESSMENTS

- As an approximation, the standard definitions of precision and recall can be applied to this modified definition of relevant items retrieved, with some subtleties because we sum graded as opposed to binary relevance assessments.
- Overlap is not accounted for: BOO

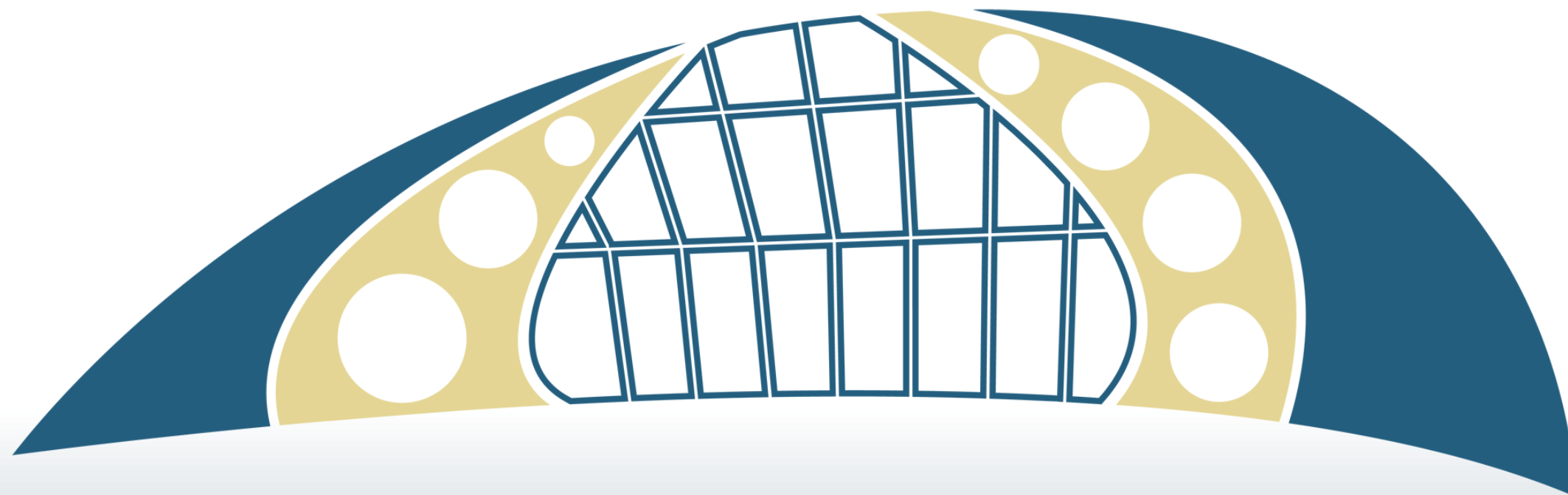


XML RETRIEVAL

SUMMARY

- Structured or XML IR:
 - effort to port unstructured (standard) IR know-how onto a scenario that uses structured (DB-like) data
- Specialized applications (e.g. patents, digital libraries)
- A decade old, unsolved problem
- <http://www.is.informatik.uni-duisburg.de/projects/inex/index.html.en>





WESTMONT **INSPIRED**
— COMPUTING LAB —