

Accessing XML content: An information retrieval perspective

Mounia Lalmas

mounia@acm.org

Outline

- **Introduction to XML, basics and standards**
- **Document-oriented XML retrieval**
- **Evaluating XML retrieval effectiveness**

Outline

- **Introduction to XML, basics and standards**
- Document-oriented XML retrieval
- Evaluating XML retrieval effectiveness

Introduction to XML, basics and standards

- What is XML?
- Document Type Definition
- XML Schema
- Querying XML Data (as reference mainly)
 - XPath
 - XQuery

XML (eXtensible Markup Language)

- A meta-language (a language for describing other languages)
XML is able to represent a mix of structured and text (unstructured) information
- Defined by the WWW Consortium (W3C)
developed by a W3C working group, headed by James Clark.
- XML 1.0 became a W3C Recommendation on February 10, 1998
- At present XML is the *de facto* standard markup language.

XML

- XML applications: *data interchange, digital libraries, content management, complex documentation, etc.*
- XML repositories: *Library of Congress collection, SIGMOD DBLP, IEEE INEX collection, LexisNexis, ...*

(<http://www.w3.org/XML/>)

XML

- Documents have tags giving extra information about sections of the document
`<title> XML </title> <slide> Introduction ...</slide>`
- Derived from SGML (Standard Generalized Markup Language) but simpler to use
- Extensible, unlike HTML
users can add new tags, and *separately* specify how the tag should be handled for display
- Goal was (is?) to replace HTML as the language for publishing documents on the Web

XML

- The ability to specify new tags, and to create nested tag structures made XML a great way to exchange data, not just documents.

many of the use of XML has been in data exchange applications, and not just a replacement for HTML

- Tags make data self-documenting

Example of an XML document

(from database)

```
<?xml version= "1.0" encoding= "UTF-8" standalone= "yes"?>
<?xml:stylesheet type = "text/xsl" href = "staff_list.xsl"?>
<!DOCTYPE STAFFLIST SYSTEM "staff_list.dtd">
<STAFFLIST>
  <STAFF branchNo = "B005">
    <STAFFNO>SL21</STAFFNO>
    <NAME>
      <FNAME>John</FNAME><LNAME>White</LNAME>
    </NAME>
    <POSITION>Manager</POSITION>
    <DOB>1-Oct-45</DOB>
    <SALARY>30000</SALARY>
  </STAFF>
  <STAFF branchNo = "B003">
    <STAFFNO>SG37</STAFFNO>
    <NAME>
      <FNAME>Ann</FNAME><LNAME>Beech</LNAME>
    </NAME>
    <POSITION>Assistant</POSITION>
    <SALARY>12000</SALARY>
  </STAFF>
</STAFFLIST>
```

XML - Elements

- Tag: label for a section of data
- Element: section of data beginning with <tagname> and ending with matching </tagname>
- Elements must be properly nested
 - Proper nesting
 <account> ... <balance> </balance> </account>
 - Improper nesting
 <account> ... <balance> </account> </balance>
 - Formally: every start tag must have a unique matching end tag that is in the context of the same parent element.
- Every document must have a single top-level element

Example of Nested Elements

```
<bank>
  <customer>
    <customer-name> Monz </customer-name>
    <customer-street> Mile End </customer-street>
    <customer-city> London </customer-city>
    <account>
      <account-number> A-102 </account-number>
      <branch-name> QMUL </branch-name>
      <balance> 400 </balance>
    </account>
    <account>
      ...
    </account>
  </customer>
  .
  .
</bank>
```

XML - Elements

■ Mixture of text with sub-elements:

```
<account>
```

This account is seldom used any more.

```
<account-number> A-102</account-number>
```

```
<branch-name> QMUL</branch-name>
```

```
<balance>400 </balance>
```

```
</account>
```

- Useful for document markup but discouraged for data representation

XML - Attributes

- **Elements can have attributes**

```
<account acct-type = "checking" >  
  <account-number> A-102 </account-number>  
  <branch-name>QMUL </branch-name>  
  <balance> 400 </balance>  
</account>
```

- **Attributes are specified by name=value pairs inside the starting tag of an element**

- **An element may have several attributes, but each attribute name can only occur once**

```
<account acct-type = "checking" monthly-fee="5">
```

XML - Attributes Vs. Elements

- **In the context of documents, attributes are part of markup, while element contents are part of the basic document contents**
- **In the context of data representation, the difference is unclear and may be confusing**
 - `<account account-number = "A-101"> </account>`
 - `<account>`
 `<account-number>A-101</account-number> ...`
 `</account>`
- **Suggestion: use attributes for identifiers of elements, and use elements for contents**

XML – Other Syntax

- **Elements without sub-elements or text content can be abbreviated by ending the start tag with a `</>` and deleting the end tag**

`<account number="A-101" branch="QMUL" balance="200 />`

- **Comments: enclosed in `<!--` and `-->` tags.**
- **CDATA sections: instructs XML processor to ignore markup characters and pass enclosed text directly to application.**

`<![CDATA[<account> ... </account>]]>`

XML – Ordering

- In XML, elements are ordered.
- In contrast, in XML attributes are unordered.

Document Type Definition (DTD)

- **Type of an XML document can be specified using a DTD**
- **DTD constraints structure of XML data**
 - What elements can occur?
 - What attributes can/must an element have?
 - What sub-elements can/must occur inside each element, and how many times?
- **DTD does not constrain data types**
 - All values represented as strings in XML
- **DTD syntax**
 - <!ELEMENT element-name (subelements-specification) >
 - <!ATTLIST element-name (attributes) >

Element Specification in DTD

■ Sub-elements can be specified as

- names of elements
- #PCDATA (parsed character data), i.e., character strings
- EMPTY (no sub-elements) or ANY (anything can be a sub-element)

■ Example

```
<! ELEMENT depositor (customer-name account-number)>  
<! ELEMENT customer-name (#PCDATA)>  
<! ELEMENT account-number (#PCDATA)>
```

■ Sub-element specification may have regular expressions

```
<!ELEMENT bank ( ( account | customer | depositor)+)>
```

- “|” - alternatives
- “+” - 1 or more occurrences
- “*” - 0 or more occurrences
- “?” - 0 or 1 occurrence

Attribute Specification in DTD

- For each attribute
 - Name
 - Type of attribute
 - CDATA
 - ID (identifier) or IDREF (ID reference) or IDREFS (multiple IDREFs)
 - Whether
 - mandatory (#REQUIRED)
 - has a default value (value),
 - or neither (#IMPLIED)
- Examples
 - <!ATTLIST account acct-type CDATA “checking”>
 - <!ATTLIST customer
 - customer-id ID # REQUIRED
 - accounts IDREFS # REQUIRED >

DTD Example

mail.dtd

```
<!ELEMENT message  
  (urgent?, subject,  
   body)>  
<!ELEMENT subject  
  (#PCDATA)>  
<!ELEMENT body  
  (ref|#PCDATA)*>  
<!ELEMENT ref  
  (#PCDATA)>  
<!ELEMENT urgent  
  EMPTY>  
<!ATTLIST message  
  date DATE #IMPLIED  
  sender CDATA #REQUIRED  
  receiver CDATA #REQUIRED  
  mtype (TXT|MM) `TXT`>
```

Non-XML Language

Elements

Structure

Sequence

Nesting

Attributes

Namespaces

- XML data has to be exchanged between organizations
- Same tag name may have different meaning in different organizations, causing confusion on exchanged documents
- Specifying a unique string as an element name avoids confusion
- Better solution: use unique-name:element-name
- Avoid using long unique names all over document by using XML Namespaces

```
<bank xmlns:FB='http://www.FirstBank.com'>
  ...
  <FB:branch>
    <FB:branchname>Downtown</FB:branchname>
    <FB:branchcity> Brooklyn </FB:branchcity>
  </FB:branch>
  ...
</bank>
```

XML Schema

- Database schemas constrain what information can be stored, and the data types of stored values
- XML documents are not required to have an associated schema
- However, schemas are very important for XML data exchange
 - otherwise, a site cannot automatically interpret data received from another site
- Two mechanisms for specifying schema language
 - Document Type Definition (DTD)
 - Widely used
 - XML Schema
 - Newer, increasing use

XML Schema

- XML Schema is a more sophisticated schema language which addresses the drawbacks of DTDs.
 - Typing of values
 - E.g. integer, string, etc
 - Also, constraints on min/max values
 - User defined types
 - Is itself specified in XML syntax, unlike DTDs
 - Is integrated with namespaces
 - Many more features
 - List types, uniqueness and foreign key constraints, inheritance ..
- BUT: significantly more complicated than DTDs, not yet as widely used.

XML Schema -Example

(from database)

```

<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema>
  <xsd:element name="bank" type="BankType"/>
  <xsd:element name="account">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="account-number" type="xsd:string"/>
        <xsd:element name="branch-name" type="xsd:string"/>
        <xsd:element name="balance" type="xsd:decimal"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  .... definitions of customer and depositor ....
  <xsd:complexType name="BankType">
    <xsd:sequence>
      <xsd:element ref="account" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="customer" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="depositor" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>

```


Querying and Transforming XML Data

- Translation of information from one XML schema to another
- Querying on XML data
- Standard XML querying/translation languages
 - XSLT
 - Simple language designed for translation from XML to XML and XML to HTML
 - XPath
 - Simple language consisting of path expressions
 - XQuery
 - An XML query language with a rich set of features
- Wide variety of other languages have been proposed, and some served as basis for the XQuery standard (XML-QL, Quilt, XQL, ...)

Tree Model of XML Data

- Query and transformation languages based on tree model of XML data
- An XML document is modeled as a tree, with nodes corresponding to elements and attributes
 - Element nodes have children nodes, which can be attributes or sub-elements
 - Text in an element is modeled as a text node child of the element
 - Children of a node are ordered according to their order in the XML document
 - Element and attribute nodes (except root node) have a single parent, which is an element node
 - Root node has single child = root element of the document
- Terminology: node, children, parent, sibling, ancestor, descendant.

XPath

- XPath used to select document parts using path expressions
- Path expression = sequence of steps separated by “/”
- Result of path expression: set of values that along with their containing elements/attributes match the specified path

- Examples

- ☐ /bank/customer/customer-name
 - <customer-name>Joe</customer-name>
 - <customer-name>Mary</customer-name>
- ☐ bank/customer/customer-name/text()
 - returns the same names, but without the enclosing tags

XPath - Examples

- `/bank/account[balance > 400]`
returns account elements with a balance value greater than 400
- `/bank/account[balance]`
returns account elements containing a balance sub-element
- `/bank/account[balance > 400]/@account-number`
returns the account numbers of those accounts with balance > 400
- `/bank/account[customer/count() > 2]`
returns accounts with > 2 customers

XPath

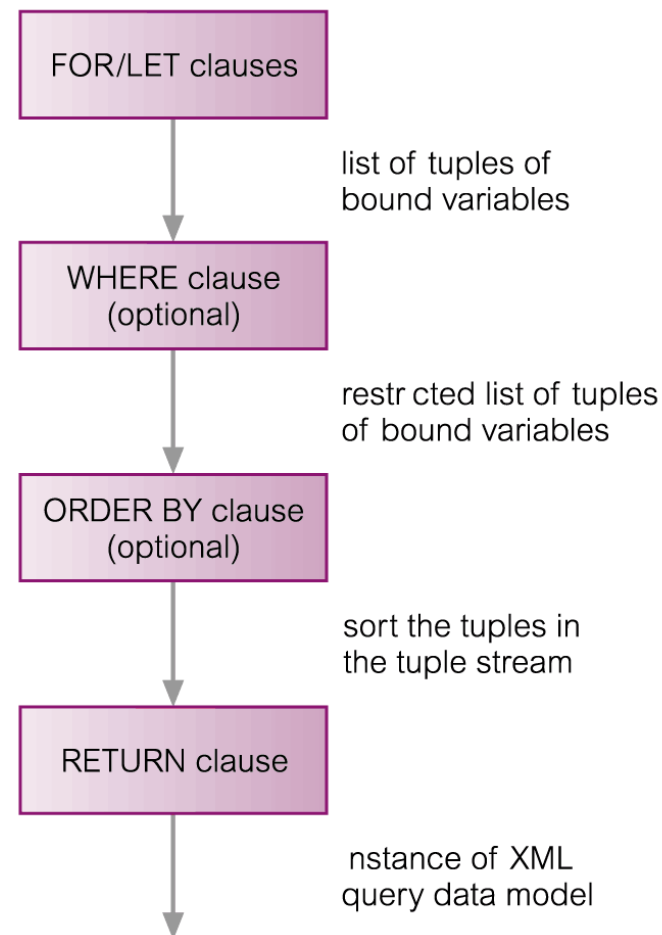
Location path	Meaning
.	Selects the context node
..	Selects the parent of the context node
/	Selects the root node, or a separator between steps in a path
//	Selects descendants of the current node
/child::STAFF (or just /STAFF)	Selects all the STAFF elements that are children of the root
child::STAFF (or just STAFF)	Selects the STAFF element children of the context node
attribute::branchNo (or just @branchNo)	Selects the branchNo attribute of the context node
attribute::* (or just @*)	Selects all the attributes of the context node
child::STAFF[3]	Selects the third STAFF element that is a child of the context node
/child::STAFF[@branchNo = "B005"]	Selects all the STAFF elements that have an attribute with a branchNo value of B005
/child::STAFF[@branchNo = "B005"][position()=1]	Selects first STAFF element that has an attribute with a branchNo value of B005

XQuery

- General purpose query language for XML data
- Currently being standardized by World Wide Web Consortium (W3C)
- Derived from the Quilt query language, itself based on features from XPath, XML-QL, SQL, OQL, Lorel, XQL, and YATL.

XQuery – FLWOR Expressions

- **FLWOR (“flower”)** expression is constructed from
 - **FOR,**
 - **LET,**
 - **WHERE,**
 - **ORDER BY,**
 - **RETURN**clauses.



Example - FLWOR Expressions

List staff at branch B005 with salary > £15,000.

```
FOR $S IN doc("staff_list.xml")//STAFF
WHERE $S/SALARY > 15000 AND
      $S/@branchNo = "B005"
RETURN $S/STAFFNO
```


Example - FLWOR Expressions

List all staff in descending order of staff number.

```
FOR $S IN doc("staff_list.xml")//STAFF  
ORDER BY $S/STAFFNO DESCENDING  
RETURN $S/STAFFNO
```

Example - FLWOR Expressions

List each branch office and average salary at branch.

```
FOR $B IN distinct-values(doc("staff_list.xml")//@branchNo))
LET $avgSalary :=
  avg(doc("staff_list.xml")//STAFF[@branchNo = $B]/SALARY)
RETURN
  <BRANCH>
    <BRANCHNO>{ $B/text() }</BRANCHNO>,
    <AVGSALARY>$avgSalary</AVGSALARY>
  </BRANCH>
```

Example - FLWOR Expressions

List branches that have more than 20 staff.

```
<LARGEBRANCHES>
```

```
  FOR $B IN
```

```
    distinct-values(doc("staff_list.xml")//@branchNo)
```

```
    LET $S:= doc("staff_list.xml")//STAFF/[ @branchNo = $B]
```

```
    WHERE count($S) > 20
```

```
  RETURN
```

```
    <BRANCHNO>{ $B/text() }</BRANCHNO>
```

```
</LARGEBRANCHES>
```

Example – Joining Two Documents

List staff along with details of their next of kin.

```
FOR $S IN doc("staff_list.xml")//STAFF,  
    $NOK IN doc("nok.xml")//NOK  
WHERE $S/STAFFNO = $NOK/STAFFNO  
RETURN  
    <STAFFNO>{ $S, $NOK/NAME }</STAFFNO>
```

Example – Joining Two Documents

List all staff along with details of their next of kin.

```
FOR $S IN doc("staff_list.xml")//STAFF
RETURN
  <STAFFNOK>
    { $S }
    FOR $NOK IN doc("nok.xml")//NOK
    WHERE $S/STAFFNO = $NOK/STAFFNO
    RETURN $NOK/NAME
  </STAFFNOK>
```

Storing XML documents in databases

- Data centric and document centric XML documents
- Different ways to store XML documents
 - Flat files
 - BLOBs
 - Object-Relational databases
 - Native XML databases

<http://www.rpbouret.com/xml/XMLAndDatabases.htm>

Outline

- Introduction to XML, basics and standards
- **Document-oriented XML retrieval**
- Evaluating XML retrieval effectiveness

Document-oriented XML retrieval

- Document vs. data- centric XML retrieval
- Focused retrieval
- Structured documents
- Structured document (text) retrieval
- XML query languages
- XML element retrieval
- (A bit about) user aspects

Data-Centric and Document-Centric XML

- Data with partial structure is called semi-structured
- XML documents are considered to be semi-structured
- XML documents classified as:
 - Data centric
 - Document centric
- Nowadays border between data and document centric XML documents is not always clear

Data-centric XML documents

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE CLASS SYSTEM "class.dtd">
<CLASS name="DCS317" num_of_std="100">
  <LECTURER lecid="111">Thomas</LECTURER>
  <STUDENT marks="70" origin="Oversea">
    <NAME>Mounia</NAME>
  </STUDENT>
  <STUDENT marks="30" origin="EU">
    <NAME>Tony</NAME>
  </STUDENT>
</CLASS>
```

Document-centric XML documents

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CLASS name="DCS317" num_of_std="100">
  <LECTURER lecid="111">Mounia</LECTURER>
  <STUDENT studid="007" >
    <NAME>James Bond</NAME> is the best student in the
    class. He scored <INTERM>95</INTERM> points out of
    <MAX>100</MAX>. His presentation of <ARTICLE>Using
    Materialized Views in Data Warehouse</ARTICLE> was
    brilliant.
  </STUDENT>
  <STUDENT stuid="131">
    <NAME>Donald Duck</NAME> is not a very good
    student. He scored <INTERM>20</INTERM> points...
  </STUDENT>
</CLASS>
```

Database and information retrieval view

■ Data-centric view

- XML as exchange format for structured data
- Used for messaging between enterprise applications
- Mainly a recasting of relational data

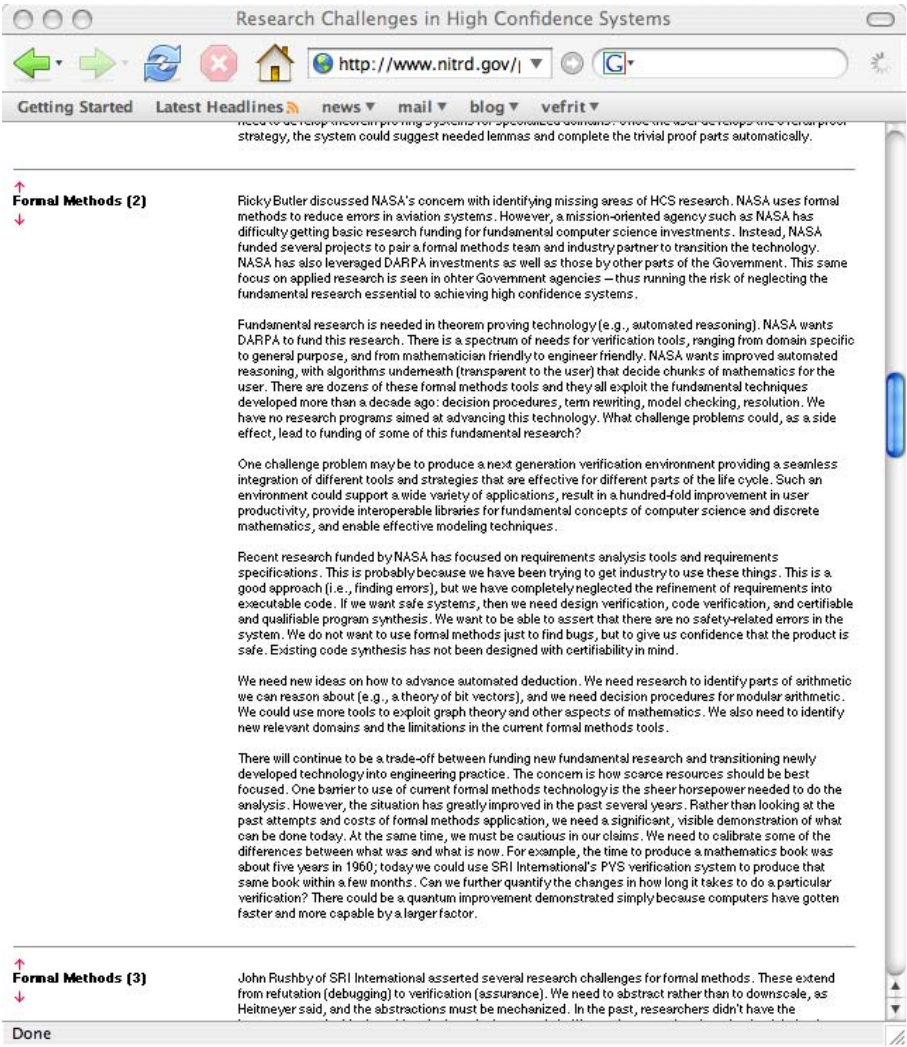
■ Document-centric view

- XML as format for representing the logical structure of documents
- Rich in text
- Demands good integration of text retrieval functionality

■ Now increasingly both views (DB+IR)

Focused retrieval: Scientific Collection

- Query
 - model checking
 - aviation systems
- Answer
 - one section in a
 - workshop report



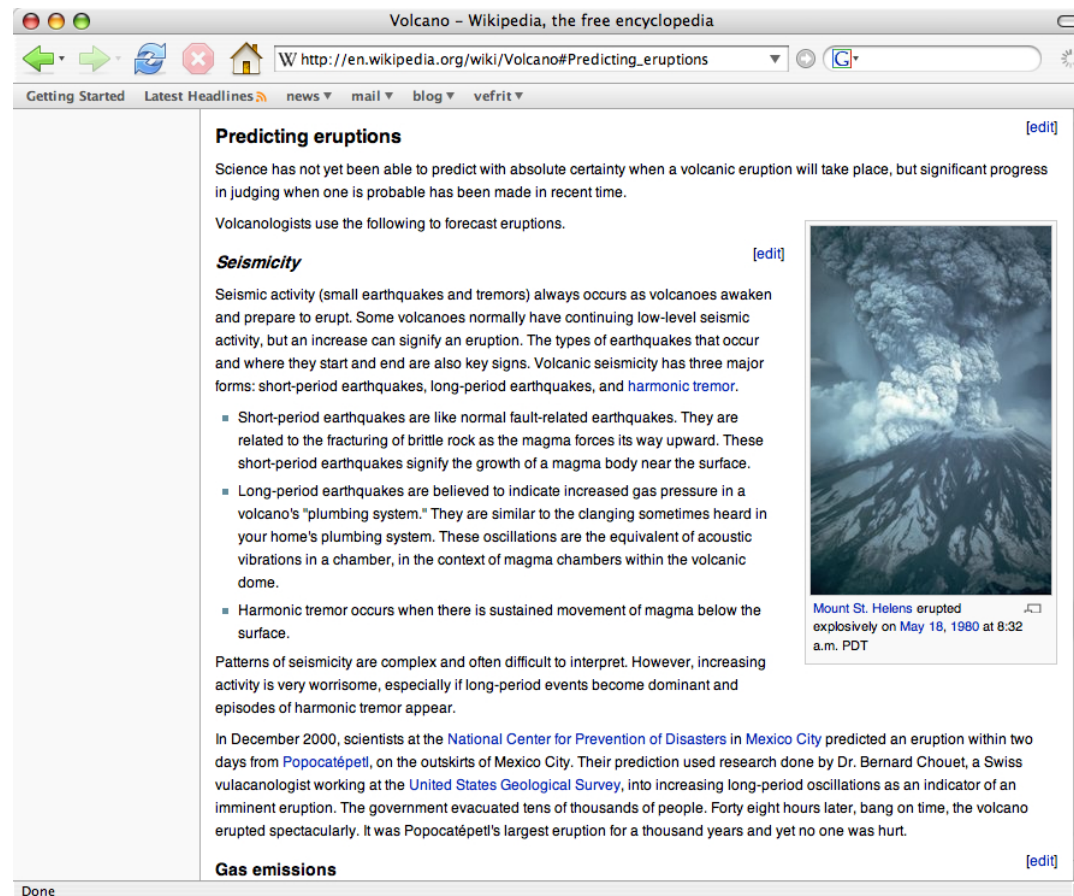
Focused Retrieval: Encyclopedia

Information need

volcanic eruption prediction

Answer

relatively small portion of the volcano topic



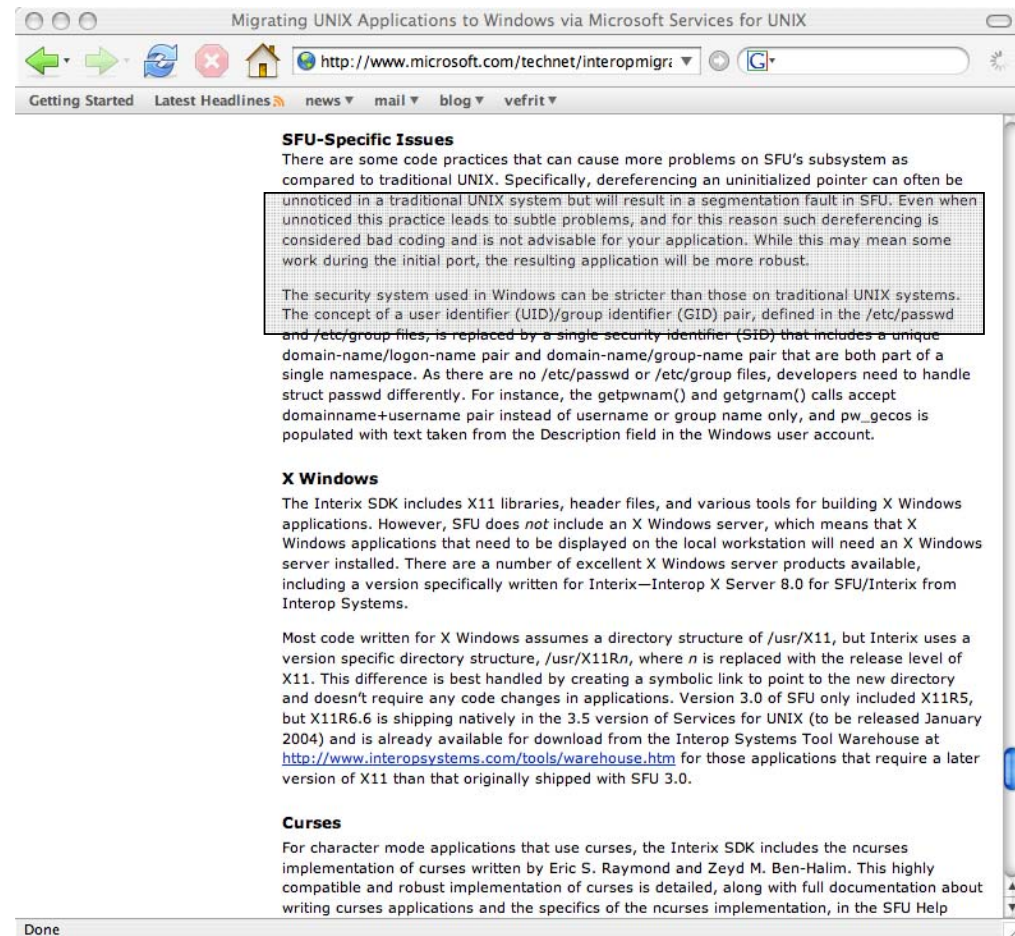
Focused retrieval: Technical Manual

■ Query

segmentation fault
windows services
for unix

■ Answer

only a single
paragraph in a long
manual

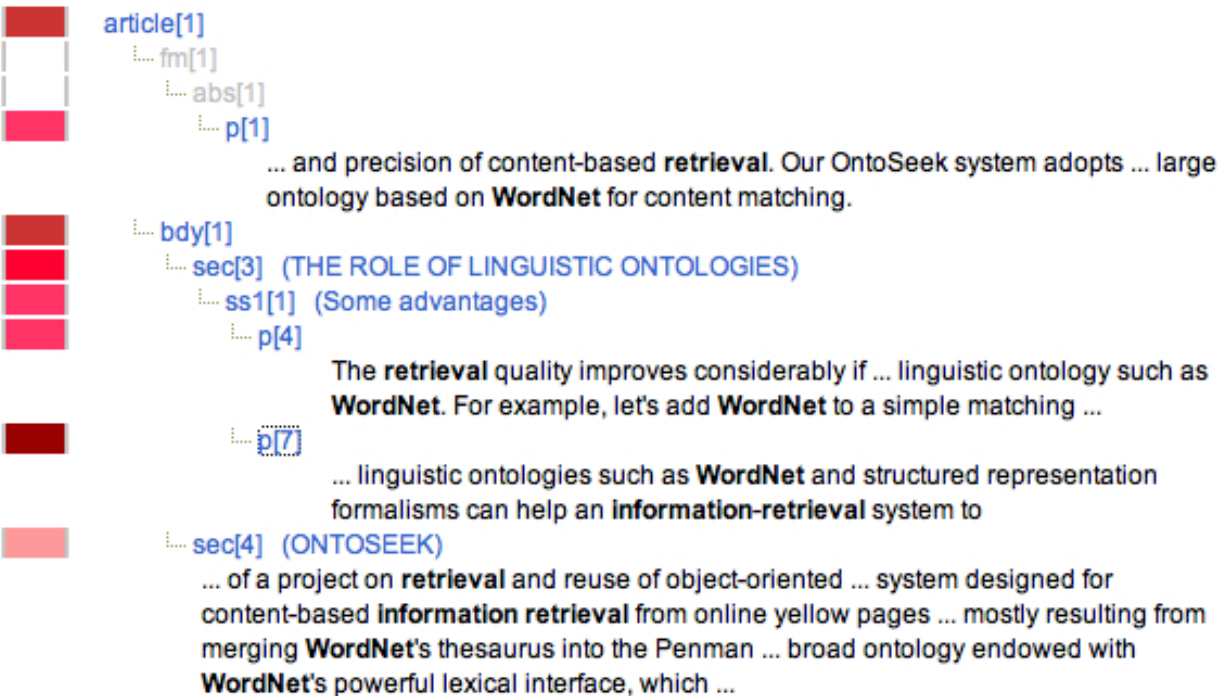


Focused retrieval: Right level of granularity

Query: wordnet information retrieval

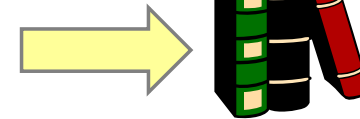
OntoSeek: Content-Based Access to the Web

Nicola Guarino, Claudio Masolo, Guido Vetere

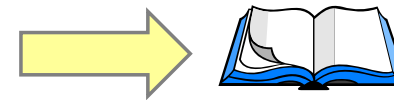


Structured Document Retrieval (SDR)

- Traditional IR is about finding relevant documents to a user's information need, e.g. entire book.



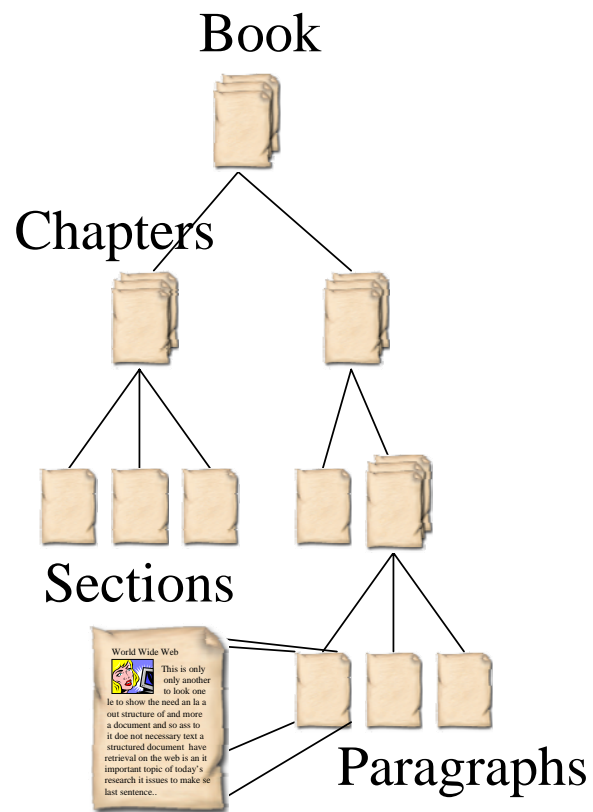
- SDR allows users to retrieve document components that are more focussed to their information needs, e.g. a chapter of a book instead of an entire book.



- The structure of documents is exploited to identify which document components to retrieve.

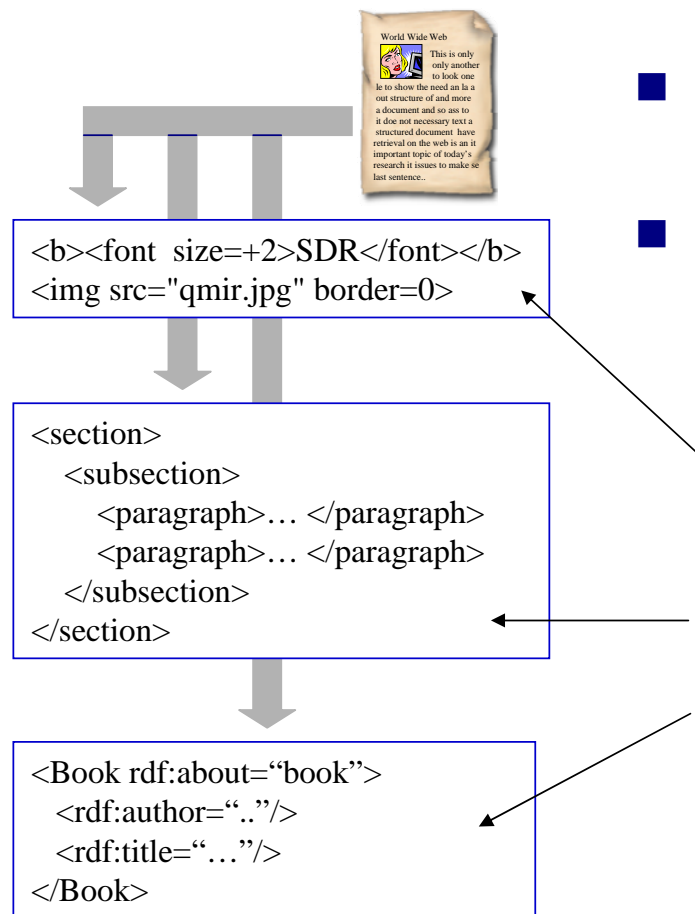
- Structure improves precision
- Exploit visual memory

Structured Documents



- ◆ In general, any document can be considered structured according to one or more structure-type
 - **Linear order of words, sentences, paragraphs ...**
 - **Hierarchy or logical structure of a book's chapters, sections ...**
 - Links (hyperlink), cross-references, citations ...
 - Temporal and spatial relationships in multimedia documents

Structured Documents



- The structure can be implicit or explicit
- Explicit structure is formalised through document representation standards (Mark-up Languages)
 - Layout
 - LaTeX (publishing), HTML (Web publishing)
 - Structure
 - SGML, XML (Web publishing, engineering), MPEG-7 (broadcasting)
 - Content/Semantic
 - RDF (ontology)

Microformats



■ Community data formats

- ☐ Personal Data: [hCard](#) (vCard)
- ☐ Calendar and Events: [hCal](#) (iCal)
- ☐ Social Networking: [XFN](#)
- ☐ Reviews: [hReview](#)
- ☐ Licenses: [rel-license](#)
- ☐ Folksonomies: [rel-tag](#)

■ Embedded in XHTML pages and RSS feeds

- ☐ Also RSS Extensions (iTunes, Yahoo! Media, Geo, Google Base, 20+ more in use)

Example: hCal

<strong class="summary">Fashion Expo in
Paris, France:
<abbr class="dtstart" title="2006-10-20">Oct 20</abbr>
to <abbr class="dtend" title="2006-10-23">22</abbr>

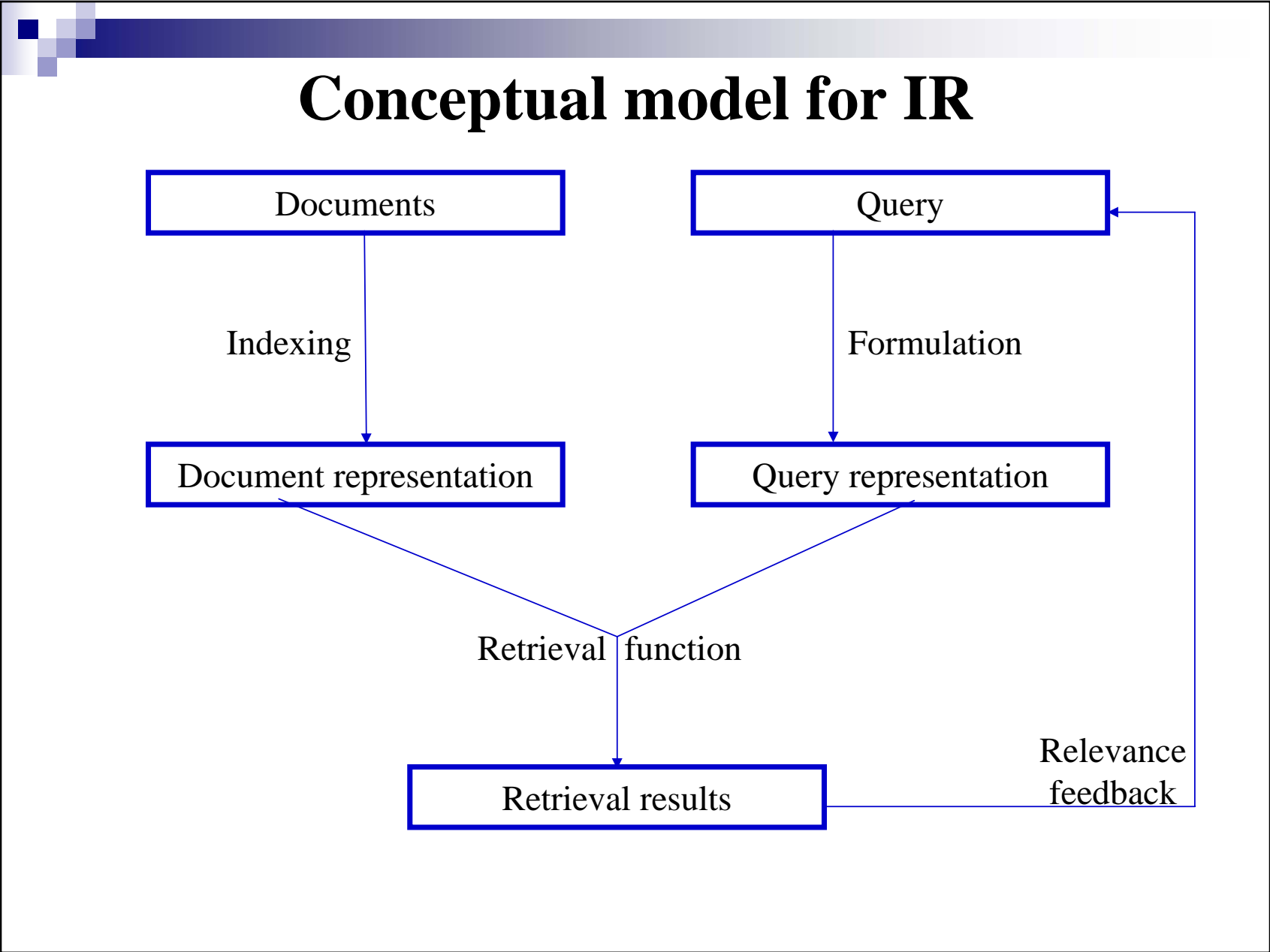
- Large and growing list of websites
 - [Eventful.com](#)
 - [LinkedIn](#)
 - [Yedda](#)
 - [upcoming.yahoo.com](#)
 - [Yahoo! Local](#), [Yahoo! Tech Reviews](#)
- Benefit from shared tools, practices (hCalendar creator, iCal Extraction)

Queries in SDR

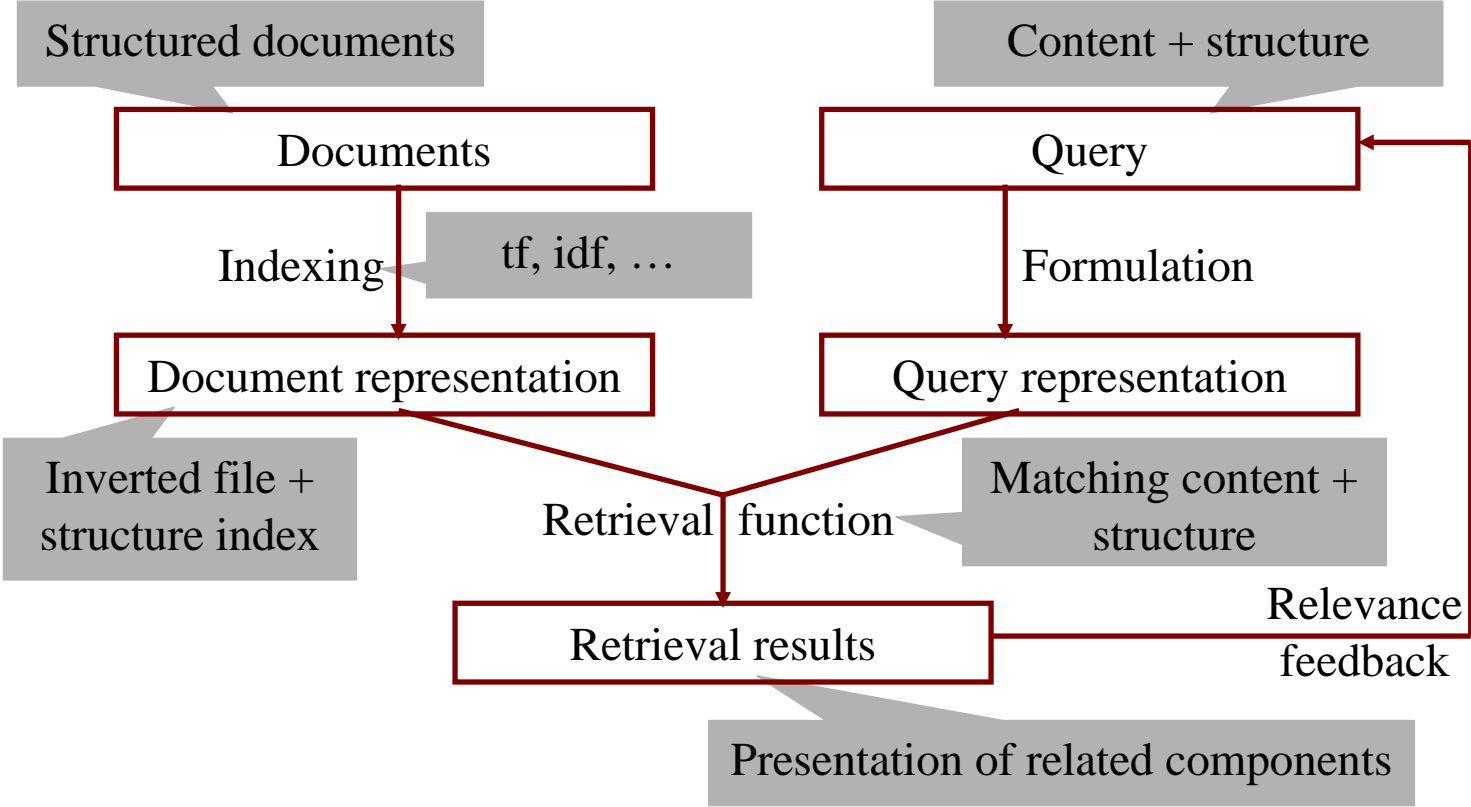
- Three types of queries:
 - Content-only (CO) queries
 - Standard IR queries but here we are retrieving document components
 - “London tube strikes”
 - Structure-only queries
 - Usually not that useful from an IR perspective
 - “Paragraph containing a diagram next to a table”

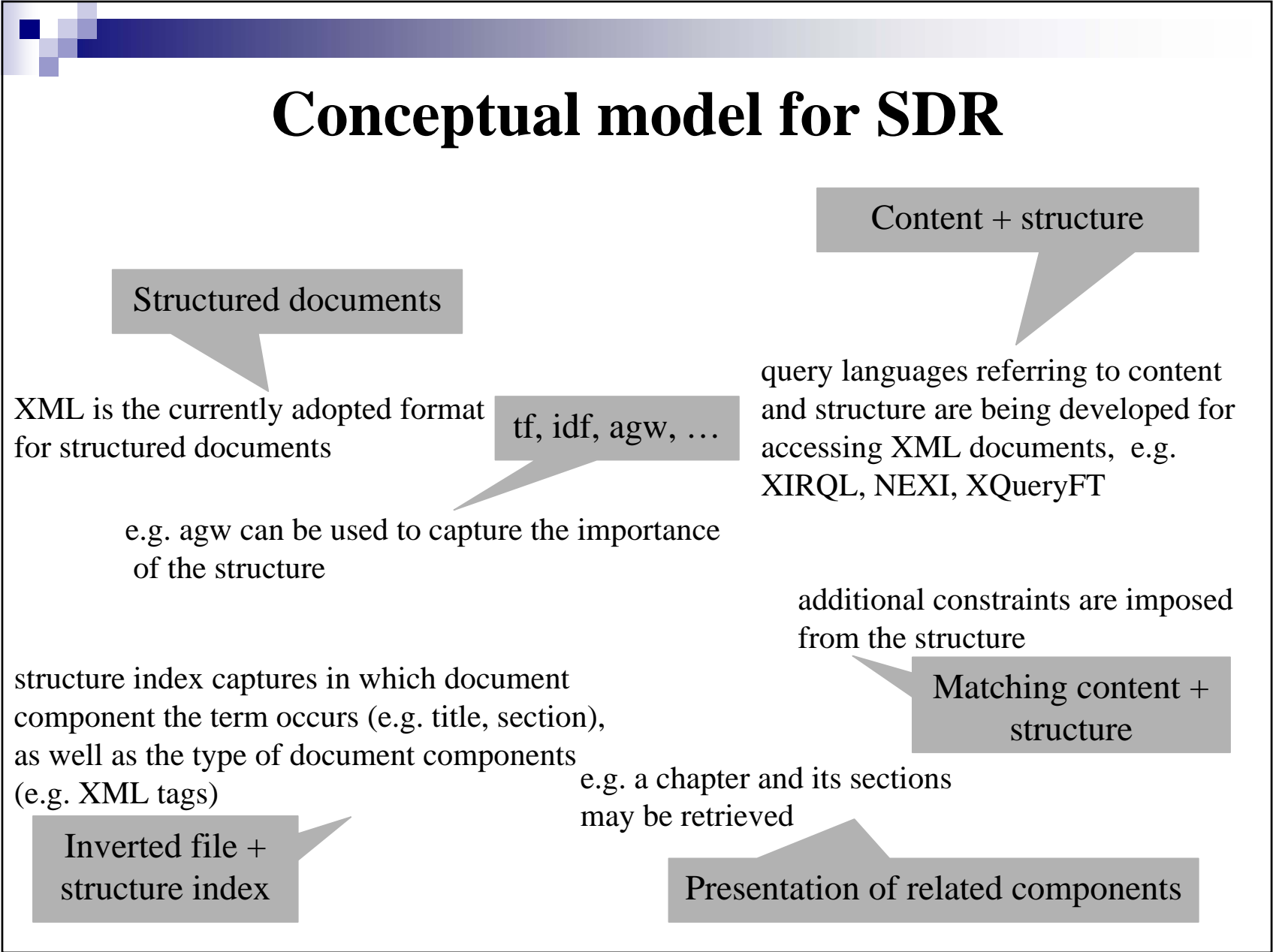
Queries in SDR

- Three types of queries:
 - Content-and-structure (CAS) queries
 - Put on constraints on which types of components are to be retrieved
 - E.g. “Sections of an article in the Times about congestion charges”
 - E.g. Articles that contain sections about congestion charges in London, and that contain a picture of Ken Livingstone, and return titles of these articles”
 - Inner constraints (support elements), target elements



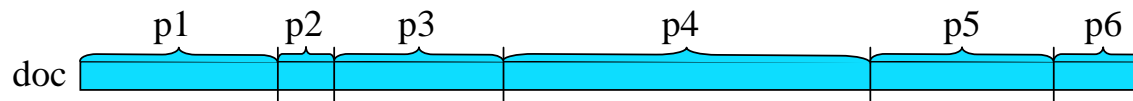
Conceptual model for SDR





Passage retrieval

- Passage: continuous part of a document,
Document: set of passages

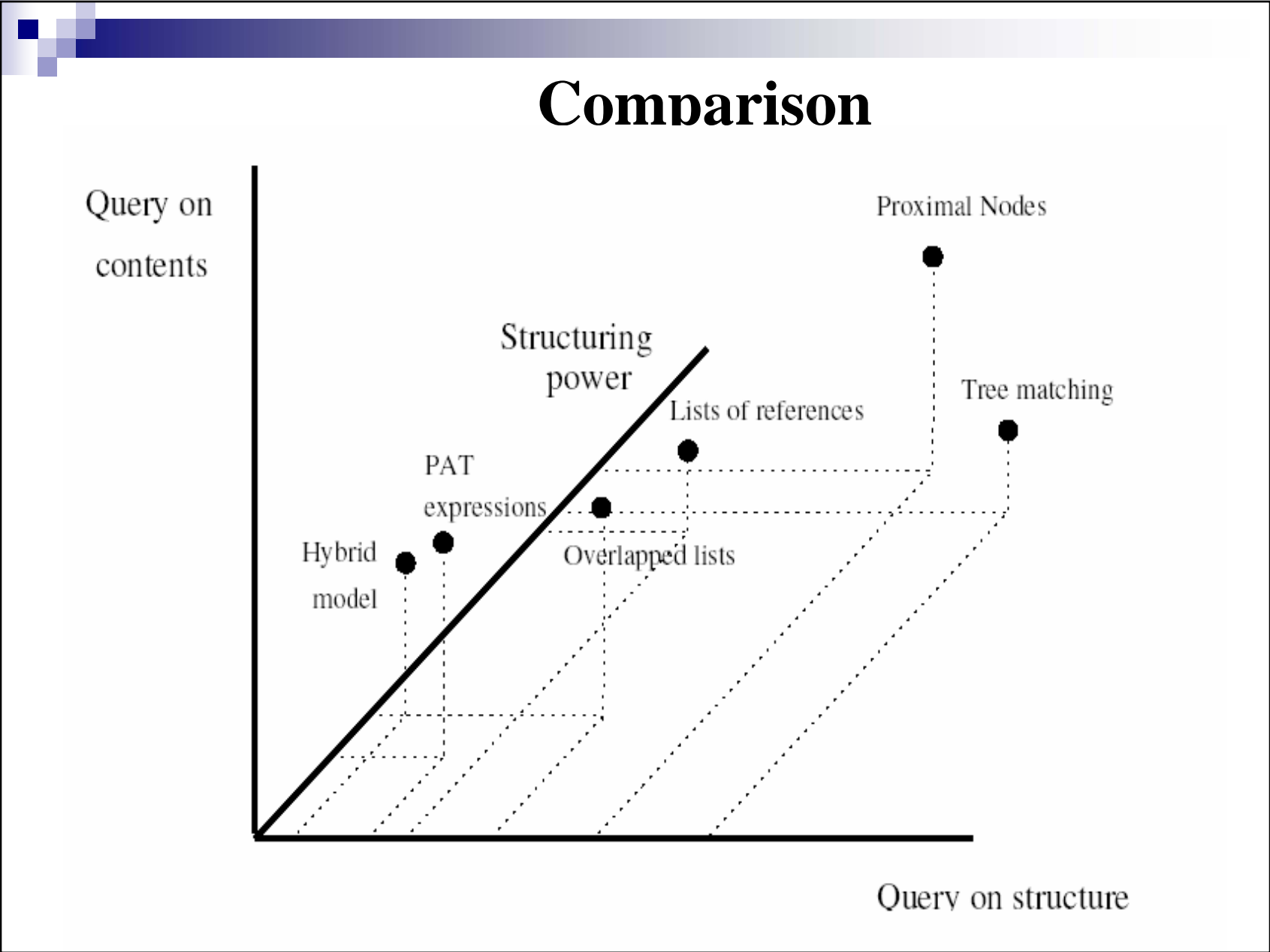


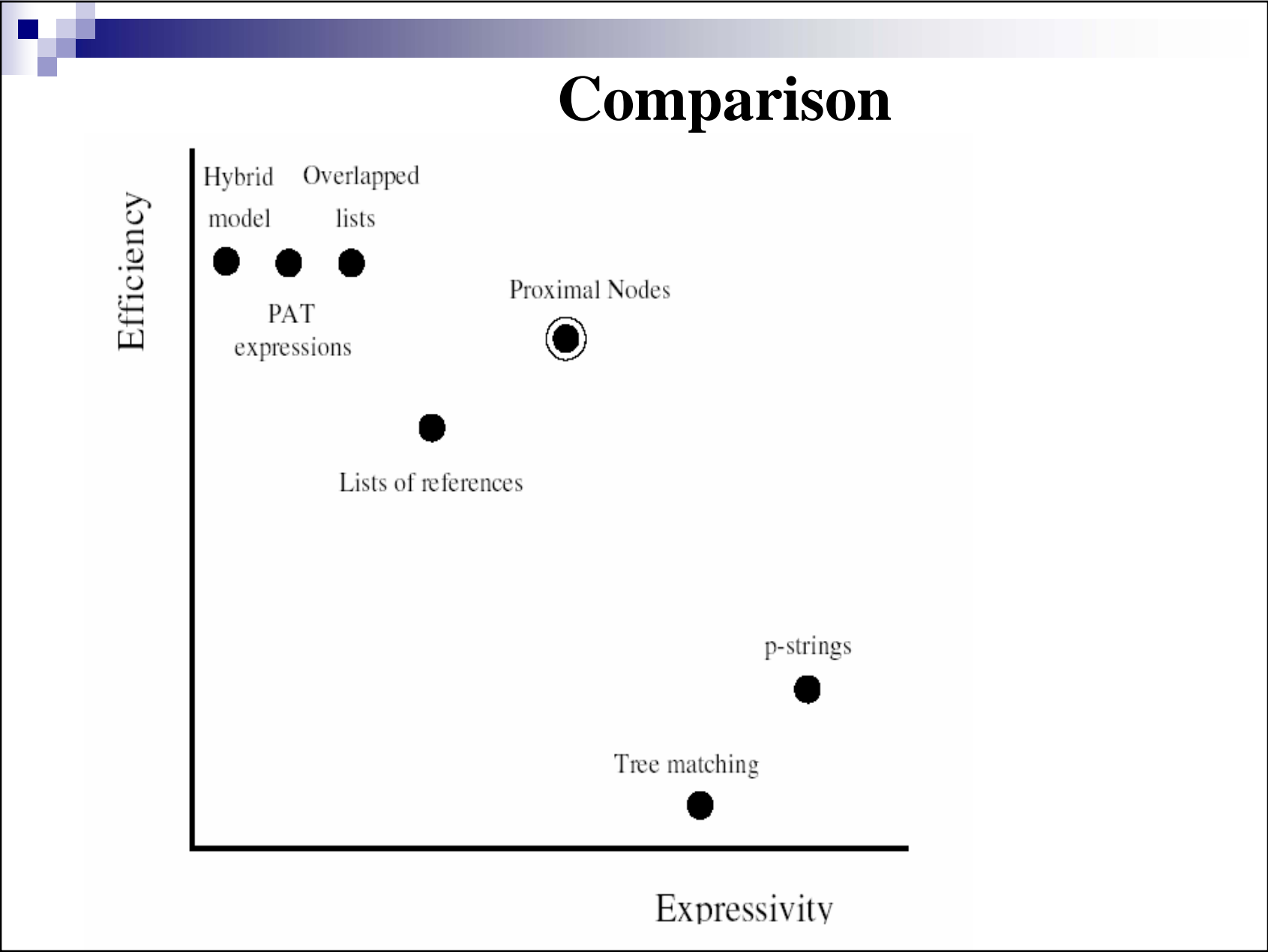
- A passage can be defined in several ways:
 - Fixed-length e.g. (300-word windows, overlapping)
 - Discourse (e.g. sentence, paragraph) ← e.g. according to logical structure but fixed (e.g. passage = sentence, or passage = paragraph)
 - Semantic (TextTiling based on sub-topics)
- Apply IR techniques to passages
 - Retrieve passage or document based on highest ranking passage or sum of ranking scores for all passages
 - Deal principally with content-only queries

(Callan, SIGIR 1994; Wilkinson, SIGIR 1994; Salton et al, SIGIR 1993; Hearst & Plaunt, SIGIR 1993; ...)

Structured document (text) retrieval

- Trade-off: expressiveness vs. efficiency
- Models (1989-1995)
 - Hybrid model (flat fields)
 - PAT expressions
 - Overlapped lists
 - Reference lists
 - Proximal nodes
 - Region algebra
 - Proposed as Algebra for XML-IR-DB Sandwich
 - p-strings
 - Tree matching





Example: Proximal Nodes

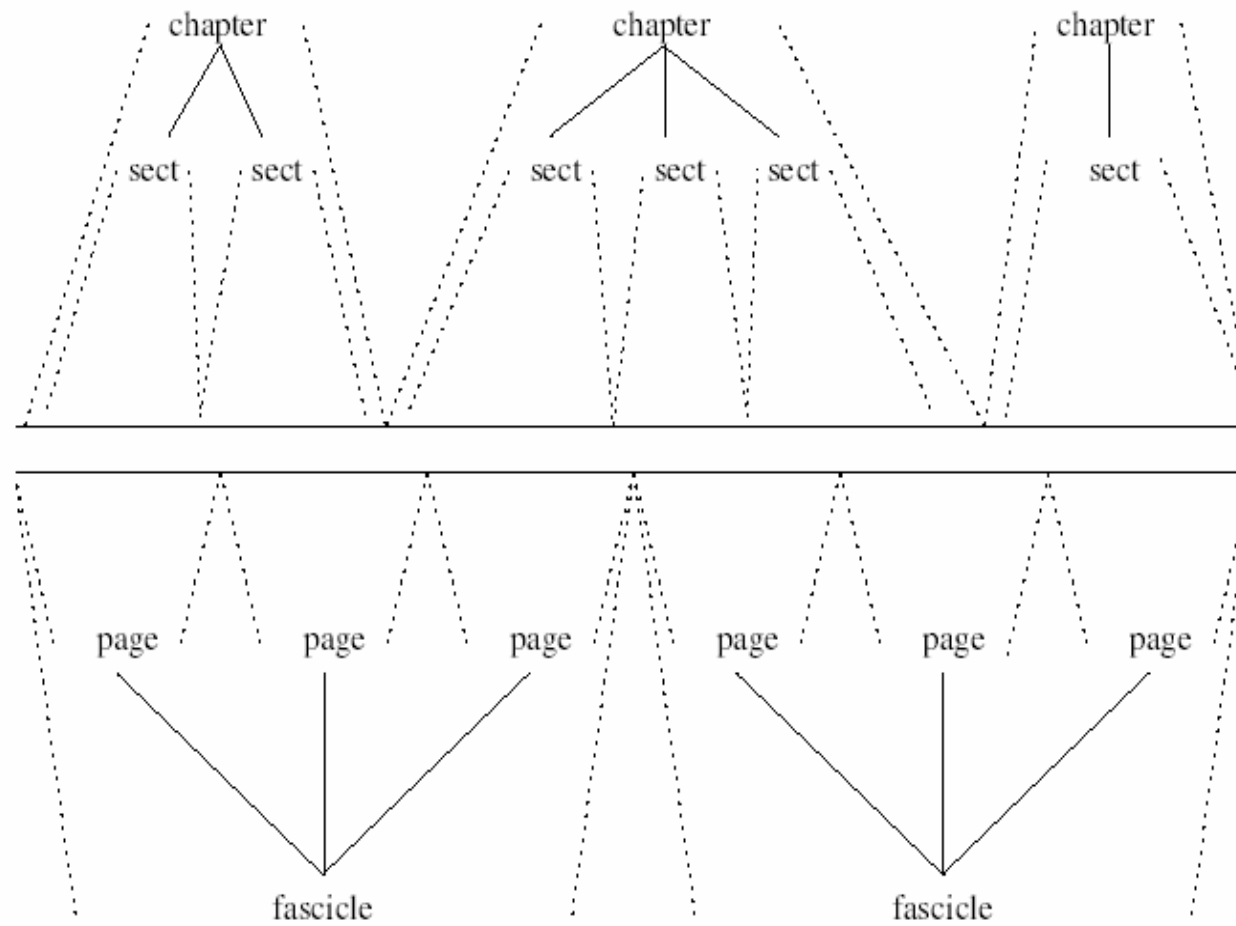
- Hierarchical structure
- Set-oriented language
- Avoid traversing the whole database
- Bottom-up strategy
- Solve leaves with indexes
- Operators work with near-by nodes
- Operators cannot use the text contents
- Most XPath and XQuery expressions can be solved using this model

(Navarro & Baeza-Yates, 1995)

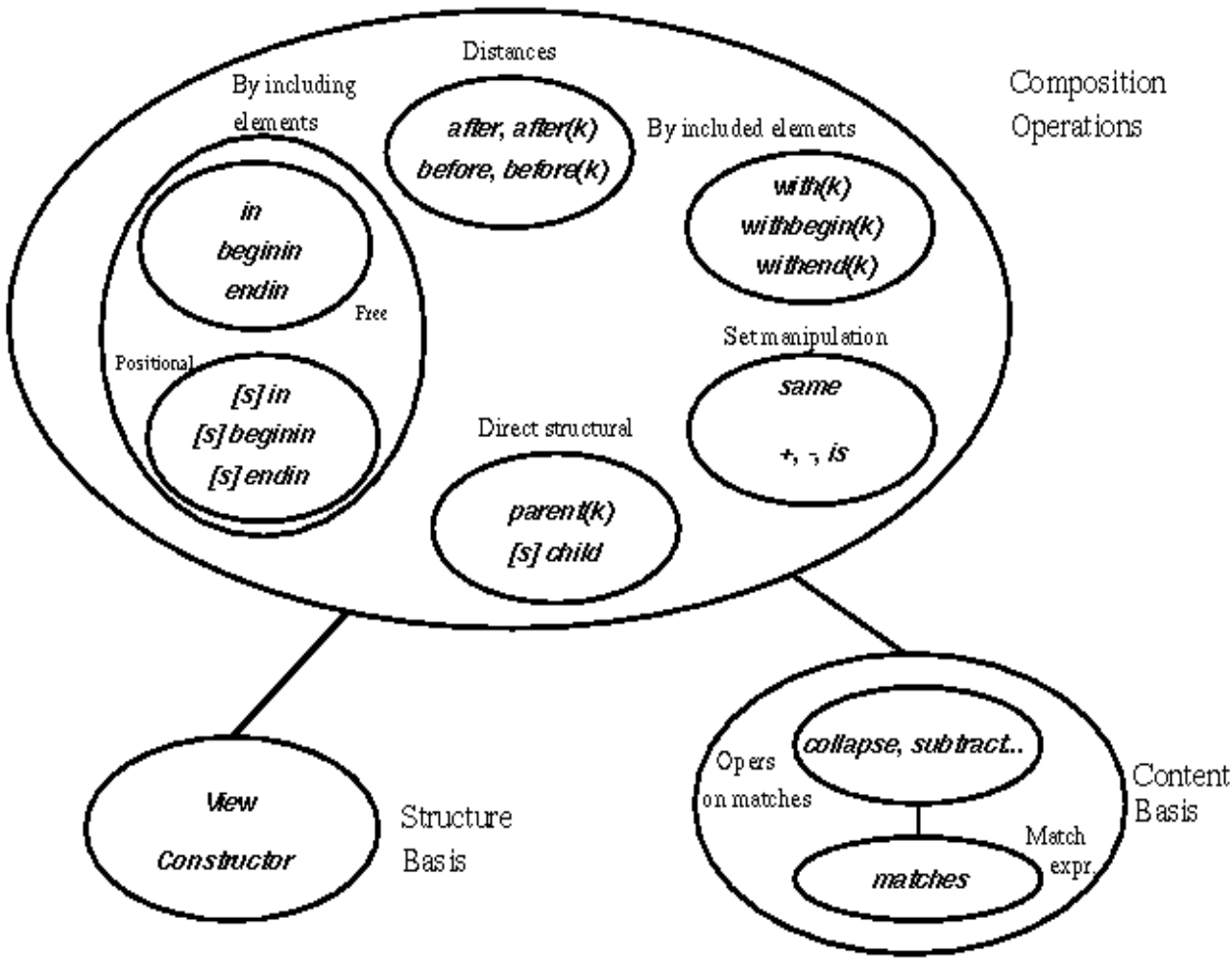
Proximal Nodes: Data Model

- Text = sequence of symbols (filtered)
- Structure = set of independent and disjoint hierarchies or “views”
- Node = Constructor + Segment
- Segment of node \supseteq segment of children
- Text view, to modelize pattern-matching queries
- Query result = subset of some view

Proximal Nodes: Hierarchies

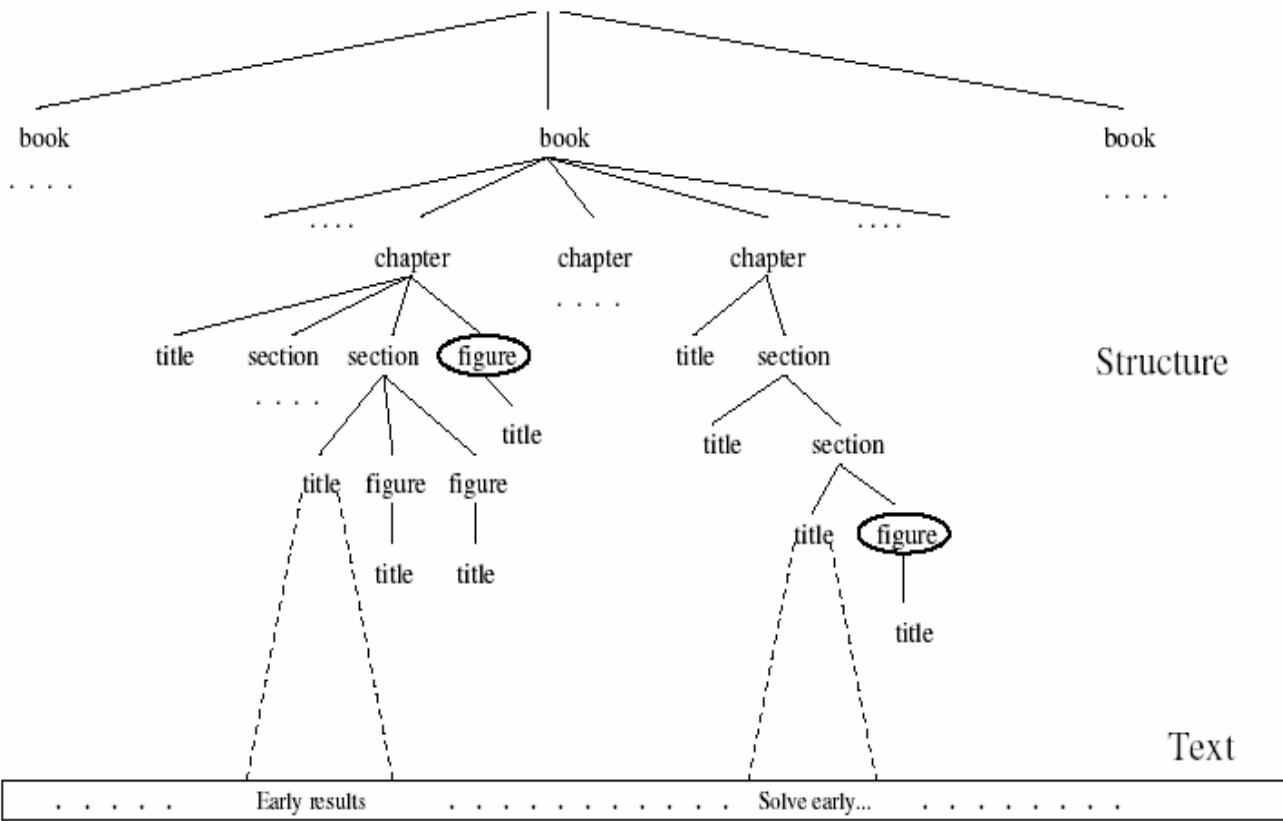


Proximal Nodes: Operations



Proximal Nodes: Query Example

[last] figure in (chapter with (section with (title with "early")))



Query languages for XML

- Four “levels” of expressiveness
 - Keyword search (CO Queries)
 - “xml”
 - Tag + Keyword search
 - book: xml
 - Path Expression + Keyword search (CAS Queries)
 - /book[./title about “xml db”]
 - XQuery + Complex full-text search
 - for \$b in /book
let score \$s := \$b ftcontains “xml” && “db”
distance 5

Query languages for XML

- Keyword search (CO Queries)
 - “xml”
- Tag + Keyword search
 - book: xml
- Path Expression + Keyword search (CAS Queries)
 - /book[./title about “xml db”]
- XQuery + Complex full-text search
 - for \$b in /book
let score \$s := \$b ftcontains “xml” && “db”
distance 5

XRank

```

<workshop date="28 July 2000">
  <title> XML and Information Retrieval: A SIGIR 2000 Workshop </title>
  <editors> David Carmel, Yoelle Maarek, Aya Soffer </editors>
  <proceedings>
    <paper id="1">
      <title> XQL and Proximal Nodes </title>
      <author> Ricardo Baeza-Yates </author>
      <author> Gonzalo Navarro </author>
      <abstract> We consider the recently proposed language ... </abstract>
      <section name="Introduction">
        Searching on structured text is becoming more important with XML ...
        <subsection name="Related Work">
          The XQL language ...
        </subsection>
      </section>
      ...
      <cite xmlns:xlink="http://www.acm.org/www8/paper/xmlql" ... </cite>
    </paper>
    ...
  
```

(Guo etal, SIGMOD 2003)

XRank

```

<workshop date="28 July 2000">
  <title> XML and Information Retrieval: A SIGIR 2000 Workshop </title>
  <editors> David Carmel, Yoelle Maarek, Aya Soffer </editors>
  <proceedings>
    <paper id="1">
      <title> XQL and Proximal Nodes </title>
      <author> Ricardo Baeza-Yates </author>
      <author> Gonzalo Navarro </author>
      <abstract> We consider the recently proposed language ... </abstract>
      <section name="Introduction">
        Searching on structured text is becoming more important with XML ...
        <subsection name="Related Work">
          The XQL language ...
        </subsection>
      </section>
      ...
      <cite xmlns:xlink="http://www.acm.org/www8/paper/xmlql"> ... </cite>
    </paper>
    ...
  
```

XIRQL

<workshop date="28 July 2000">

<title> XML and Information Retrieval: A SIGIR 2000 Workshop </title>

<editors> David Carmel, Yoelle Maarek, Aya Soffer </editors>

<proceedings>

<paper id="1">

<title> XQL and Proximal Nodes </title>

<author> Ricardo Baeza-Yates </author>

<author> Gonzalo Navarro </author>

<abstract> We consider the recently proposed language ... </abstract>

<section name="Introduction">

index nodes Searching on structured text is becoming more important with XML ...

 The XQL language

</section>

...

<cite xmlns:xlink="http://www.acm.org/www8/paper/xmlql" ... </cite>

</paper>

...

(Fuhr & Großjohann, SIGIR 2001)

Query languages for XML

- Keyword search (CO Queries)
 - “xml”
- Tag + Keyword search
 - book: xml
- Path Expression + Keyword search (CAS Queries)
 - /book[./title about “xml db”]
- XQuery + Complex full-text search
 - for \$b in /book
let score \$s := \$b ftcontains “xml” && “db”
distance 5

XSearch

Not a meaningful result

```
<workshop date="28 July 2000">
  <title> XML and Information Retrieval: A SIGIR 2000 Workshop </title>
  <editors> David Carmel, Yoelle Maarek, Aya Soffer </editors>
  <proceedings>
    <paper id="1">
      <title> XQL and Proximal Nodes </title>
      <author> Ricardo Baeza-Yates </author>
      <author> Gonzalo Navarro </author>
      <abstract> We consider the recently proposed language ... </abstract>
      <section name="Introduction">
        Searching on structured text is becoming more important with XML ...
      </section>
    </paper>
    <paper id="2">
      <title> XML Indexing </title>
      ...
    </paper id="2">
```

(Cohen etal, VLDB 2003)

Query languages for XML

- Keyword search (CO Queries)
 - “xml”
- Tag + Keyword search
 - book: xml
- Path Expression + Keyword search (CAS Queries)
 - /book[./title about “xml db”]
- XQuery + Complex full-text search
 - for \$b in /book
let score \$s := \$b ftcontains “xml” && “db”
distance 5

XPath

- `fn:contains($e, string)` returns true iff `$e` contains `string`

`//section[fn:contains(./title, “XML Indexing”)]`

(W3C 2005)

XIRQL

- Weighted extension to XQL (precursor to XPath)

$$//section[0.6 \cdot ./* \$cw\$ \text{“XQL”} + 0.4 \cdot .//section \$cw\$ \text{“syntax”}]$$

(Fuhr & Großjohann, SIGIR 2001)

XXL

- Introduces similarity operator ~

Select Z

From <http://www.myzoos.edu/zoos.html>

Where zoos.#.zoo As Z and
Z.animals.(animal)?.specimen as A and
A.species ~ “lion” and
A.birthplace.#.country as B and
A.region ~ B.content

(Theobald & Weikum, EDBT 2002)

NEXI

- Narrowed Extended XPath I
- INEX Content-and-Structure (CAS) Queries
- Specifically targeted for content-oriented XML search (i.e. “aboutness”)

`//article[about(./title, apple) and
about(./sec, computer)]`

(Trotman & Sigurbjornsson, INEX 2004)

Query languages for XML

- Keyword search (CO Queries)
 - “xml”
- Tag + Keyword search
 - book: xml
- Path Expression + Keyword search (CAS Queries)
 - /book[./title about “xml db”]
- XQuery + Complex full-text search
 - for \$b in /book
let score \$s := \$b ftcontains “xml” && “db”
distance 5

Schema-Free XQuery

- Meaningful least common ancestor (mlcas)

```
for $a in doc("bib.xml")//author
  $b in doc("bib.xml")//title
  $c in doc("bib.xml")//year
where $a/text() = "Mary" and
      exists mlcas($a,$b,$c)
return <result> {$b,$c} </result>
```

(Li et al, VLDB 2003)

XQuery Full-Text

■ Two new XQuery constructs

1) FTContainsExpr

- Expresses “Boolean” full-text search predicates
- Seamlessly composes with other XQuery expressions

2) FTScoreClause

- Extension to FLWOR expression
- Can score FTContainsExpr *and* other expressions

(W3C 2005)

FTContainsExpr

`//book ftcontains “Usability” && “testing” distance 5`

`//book[./content ftcontains “Usability” with stems]/title`

`//book ftcontains /article[author=“Dawkins”]/title`

FTScore Clause

In any
order {

```
FOR $v [SCORE $s]? IN [FUZZY] Expr
LET ...
WHERE ...
ORDER BY ...
RETURN
```

Example

```
FOR $b SCORE $s in
    /pub/book[. ftcontains "Usability" && "testing"
    and ./price < 10.00]

ORDER BY $s
RETURN $b
```

FTScore Clause

In any
order {

```

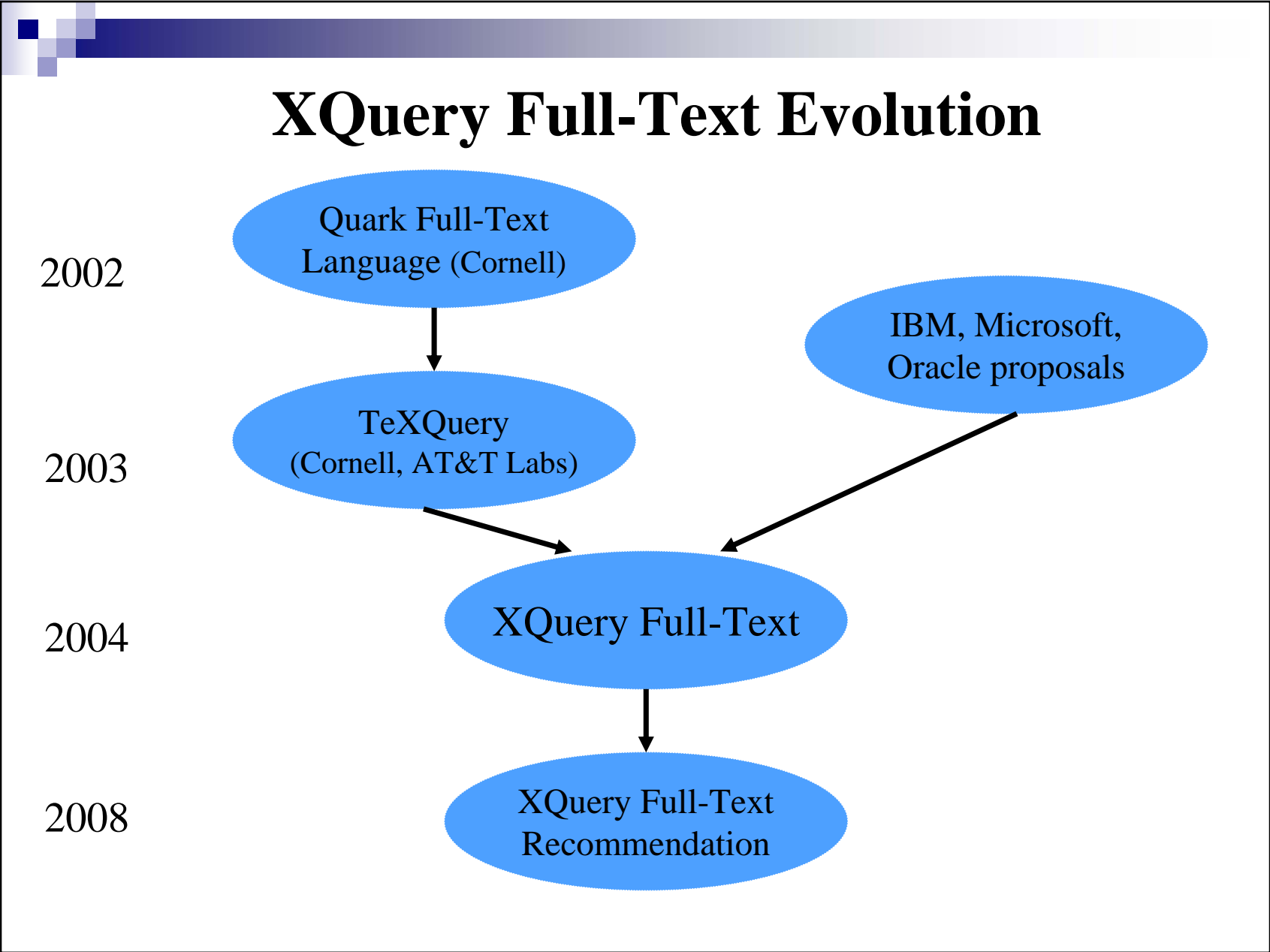
FOR $v [SCORE $s]? IN [FUZZY] Expr
LET ...
WHERE ...
ORDER BY ...
RETURN
  
```

Example

```

FOR $b SCORE $s in FUZZY
    /pub/book[. ftcontains "Usability" && "testing"]

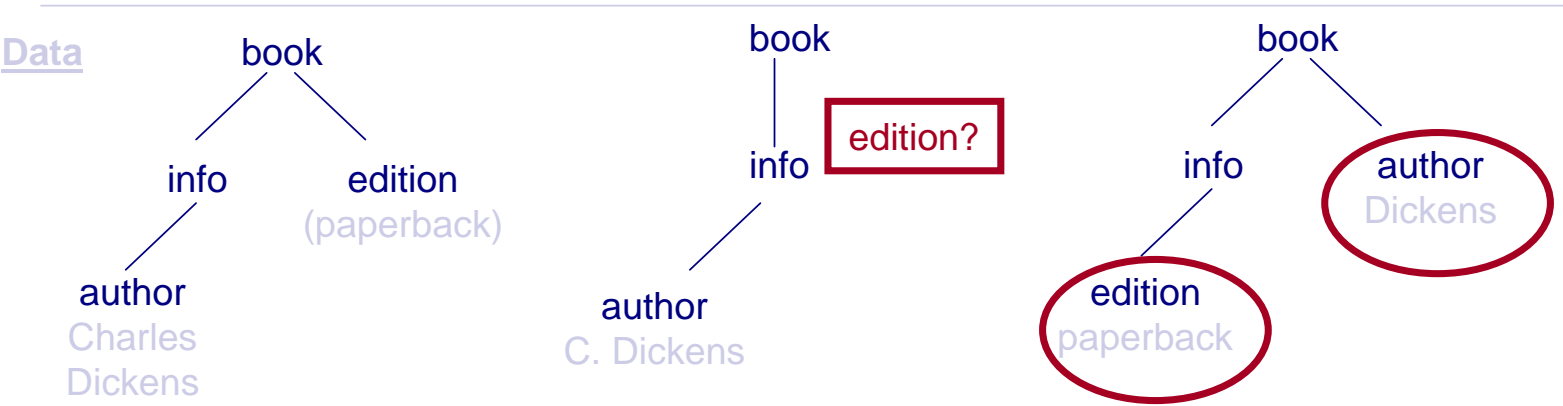
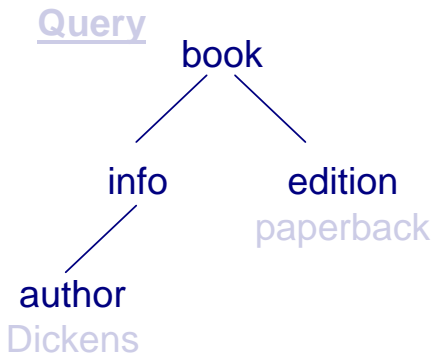
ORDER BY $s
RETURN $b
  
```



XML Query Relaxation (FlexPath)

where DB and IR meet

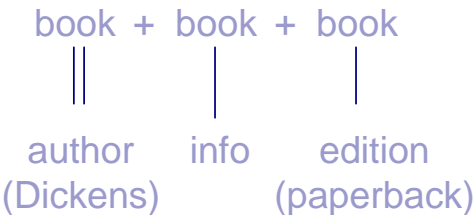
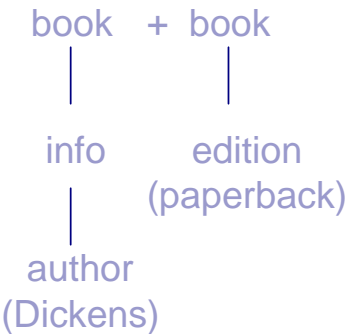
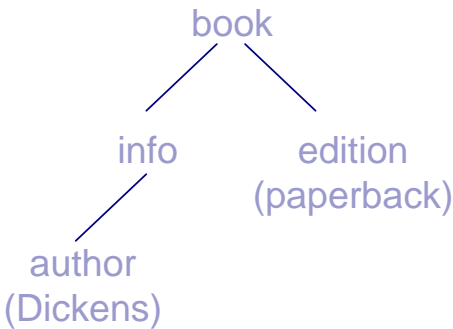
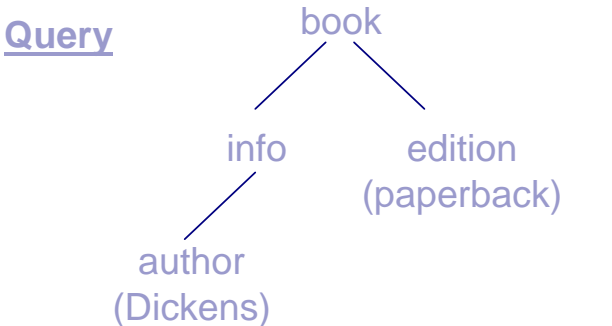
- Tree pattern relaxations:
 - Leaf node deletion
 - Edge generalization
 - Subtree promotion



(Amer-Yahia, SIGMOD 2004) (Schlieder, EDBT 2002)
(Delobel & Rousset, 2002) (Amer-Yahia et al, VLDB 2005)

A Family of XML Scoring Methods

- **Twig scoring**
 - High quality
 - Expensive computation
- **Path scoring**
- **Binary scoring**
 - Low quality
 - Fast computation



(Amer-Yahia, VLDB 2005)

Query languages for XML - Recap

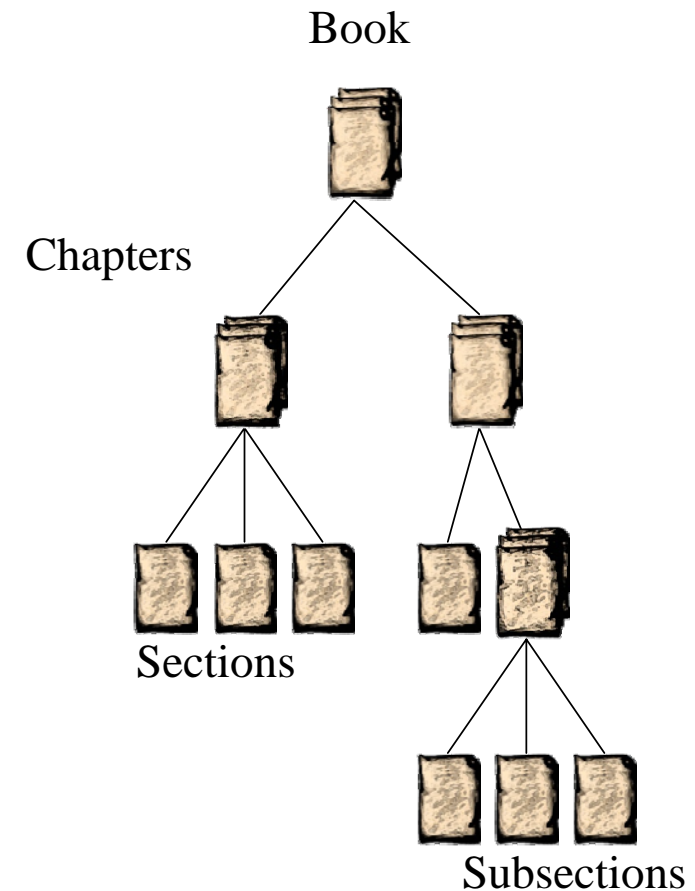
- Virtues and setbacks of XML query languages
 - Expressive query languages
 - But, too complex for many applications
 - Different interpretations

Element retrieval

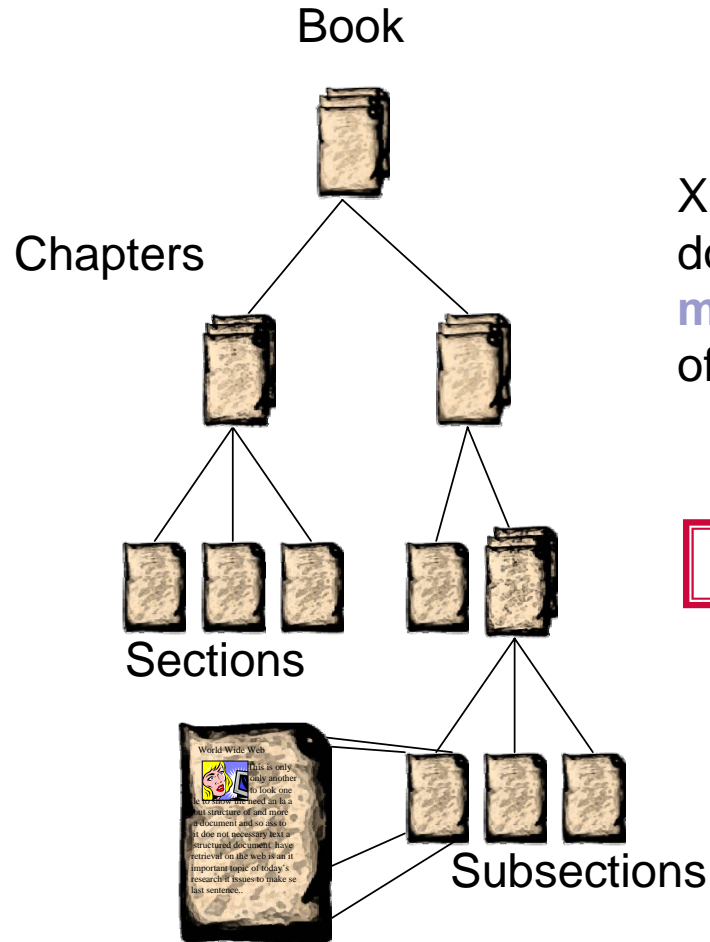
- XML retrieval vs. document retrieval
- XML retrieval = **Focused** retrieval
- Challenges
 1. Term statistics
 2. Relationship statistics
 3. Structure statistics
 4. Overlapping elements
 5. Interpretations of structural constraints
- Ranking
 1. Retrieval units
 2. Combination of evidence
 3. Post-processing

XML retrieval vs. document retrieval

- No predefined unit of retrieval
- Dependency of retrieval units
- Aims of XML retrieval:
 - Not only to find relevant elements
 - But those at the appropriate level of granularity



Content-oriented XML retrieval = Focused Retrieval



XML retrieval allows users to retrieve document components that are **more focused**, e.g. a subsection of a book instead of an entire book.

SEARCHING = QUERYING + BROWSING

Note:

Here, document component = XML element

Focused Retrieval for XML: Principle

- A XML retrieval system should always retrieve the most specific part of a document answering a query.
- Example query: **football**
- Document

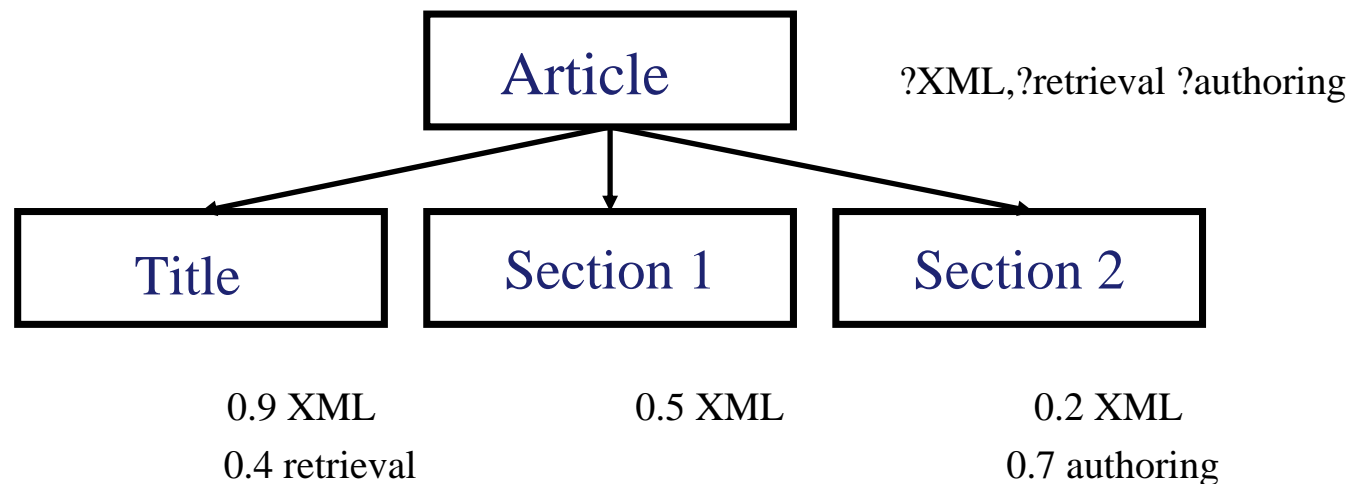
```
<chapter> 0.3 football
  <section> 0.5 history </section>
  <section> 0.8 football 0.7 regulation </section>
</chapter>
```
- Return <section>, not <chapter>

Content-oriented XML retrieval = Focused Retrieval

Return document components of **varying granularity** (e.g. a book, a chapter, a section, a paragraph, a table, a figure, etc), relevant to the user's information need both with regards to **content** and **structure**.

SEARCHING = QUERYING + BROWSING

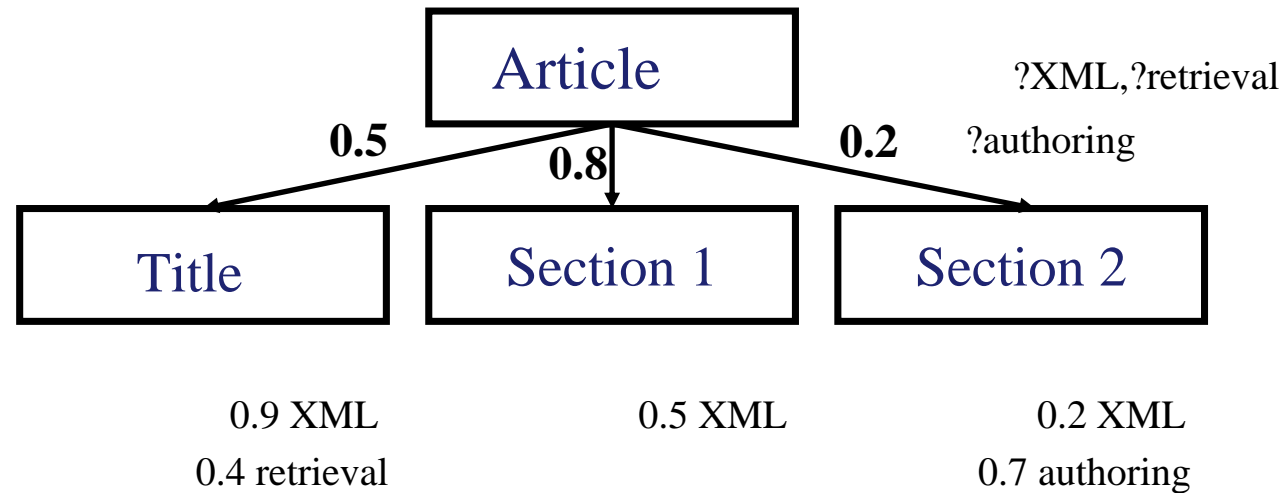
Challenge 1: Term statistics



No fixed retrieval unit + nested document components:

- ☐ how to obtain element and collection statistics (e.g. tf, idf)?
- ☐ which aggregation formalism to use?
- ☐ inner or outer aggregation?

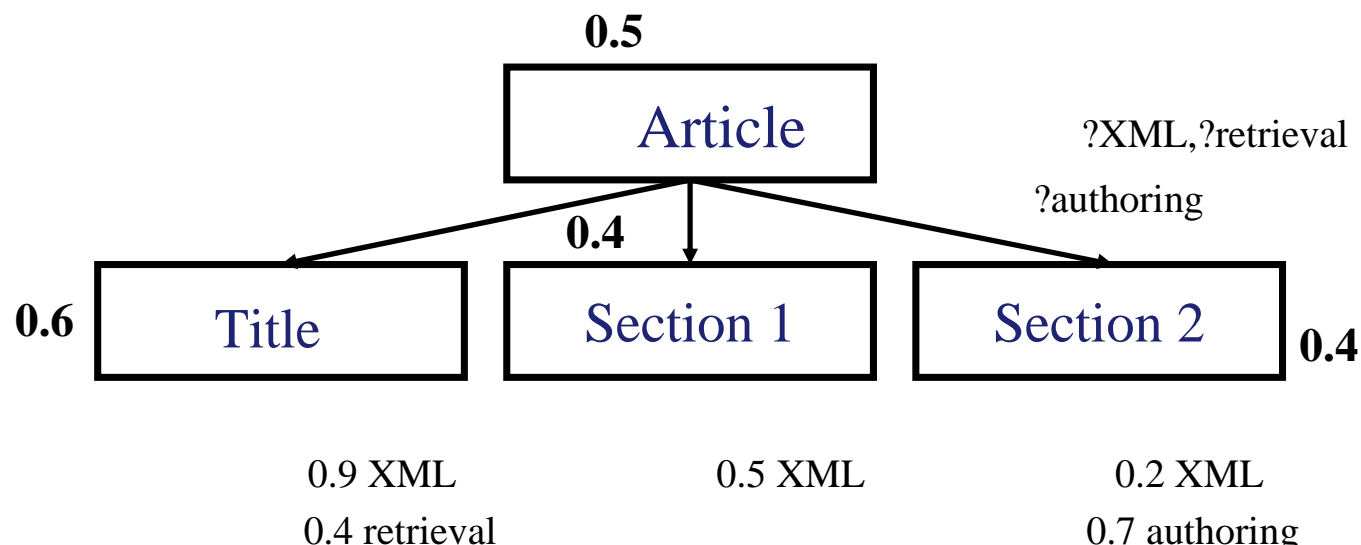
Challenge 2: Relationship statistics



Relationship between elements:

- ☐ which sub-element(s) contribute best to content of its parent element and vice versa?
- ☐ how to estimate (or learn) relationship statistics (e.g. size, number of children, depth, distance)?
- ☐ how to aggregate term and/or relationship statistics?

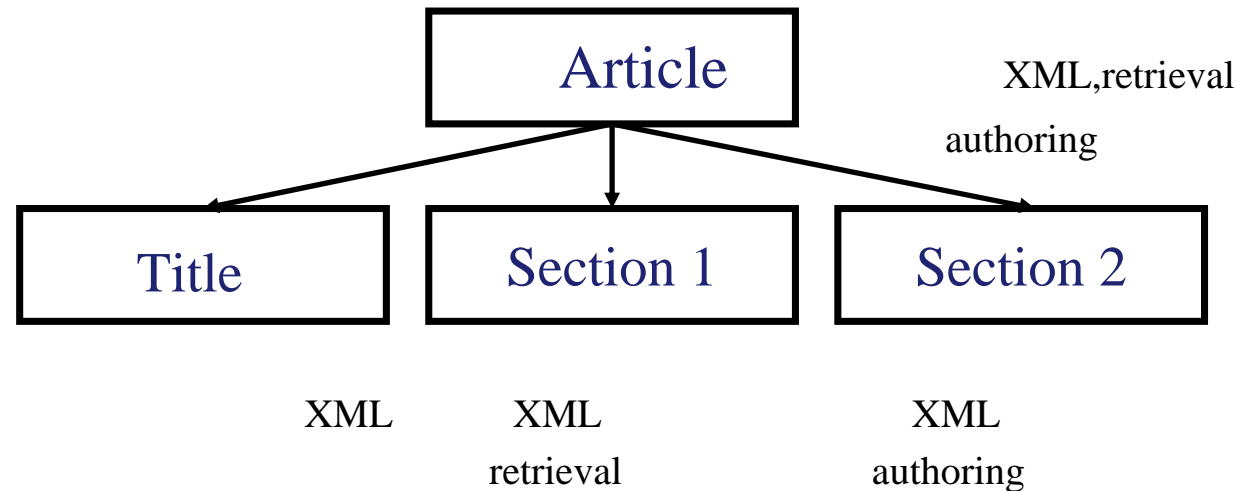
Challenge 3: Structure statistics



Different types of elements:

- ☐ which element is a good retrieval unit?
- ☐ is element size an issue?
- ☐ how to estimate (or learn) structure statistics (frequency, user studies, size, depth)?
- ☐ how to aggregate term, relationship and/or structure statistics?

Challenge 4: Overlapping elements

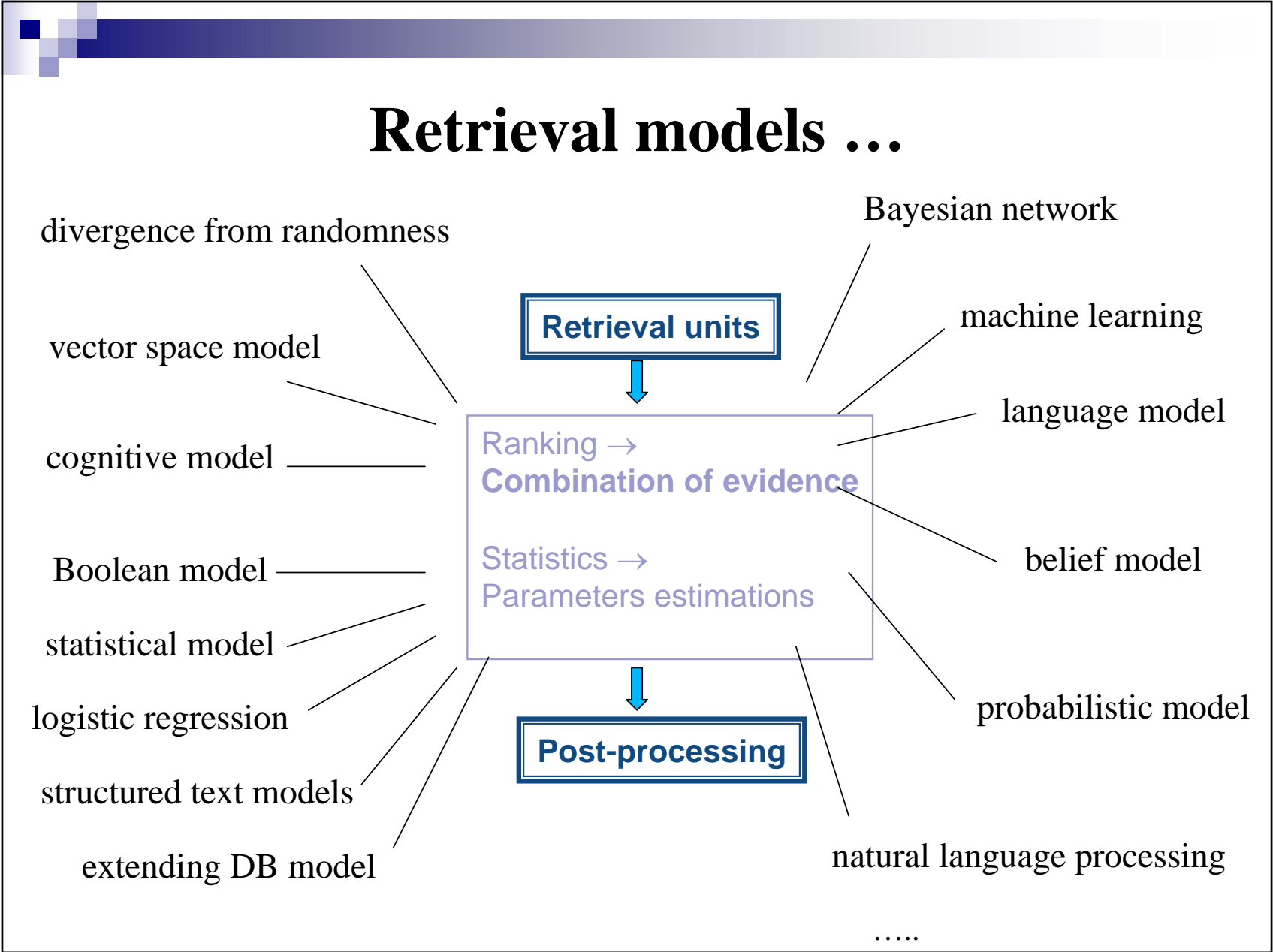


Nested (overlapping) elements:

- ☐ section 1 and article are both relevant to "XML retrieval"
- ☐ which one to return so that to reduce overlap?
- ☐ should the decision be based on user studies, size, types, etc?

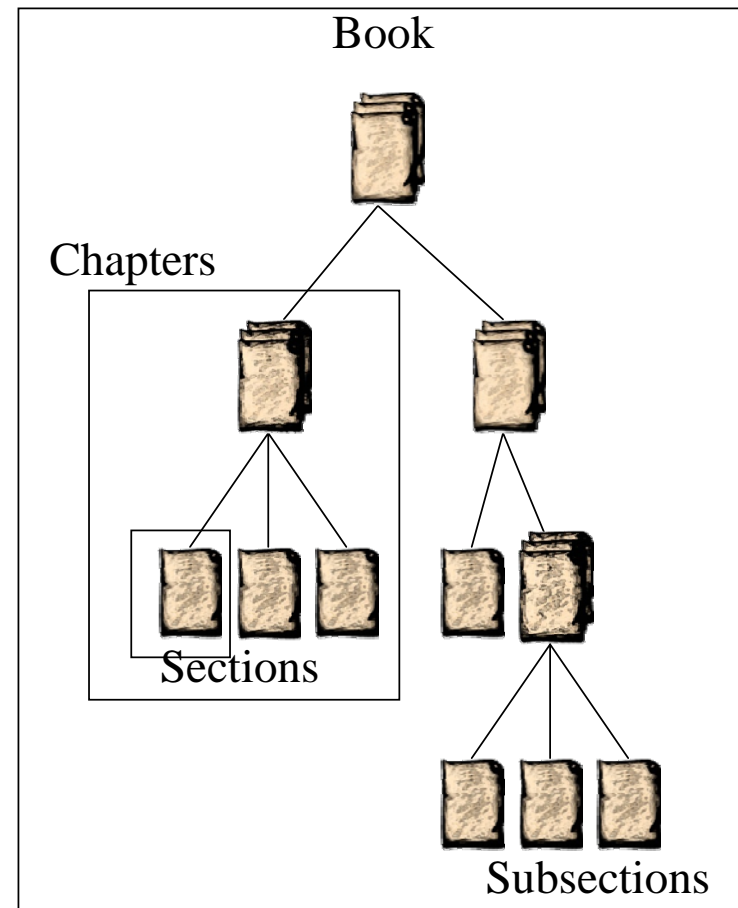
Challenge 5: Expressing and interpreting structural constraints

- Ideally:
 - There is one DTD/schema
 - User understands DTD/schema
- In practice: rare
 - Many DTs/schemas
 - DTDs/Schema not known in advance
 - DTDs/Schema change
 - Users do not understand DTDs/schema
- Need to identify “similar/synonym” elements/tags
- Importance (weight) of tags
- Strict or vague interpretation of the structure
- Relevance feedback/blind feedback?



Retrieval units: What to Index?

- XML documents are trees
 - hierarchical structure of nested elements (sub-trees)
- What should we put in the index?
 - there is no fixed unit of retrieval



Retrieval units: XML sub-trees

Assume a document like

```
<article>
  <title>XXX</title>
  <abstract>YYY</abstract>
  <body>
    <sec>ZZZ</sec>
    <sec>ZZZ</sec>
  </body>
</article>
```

Index separately

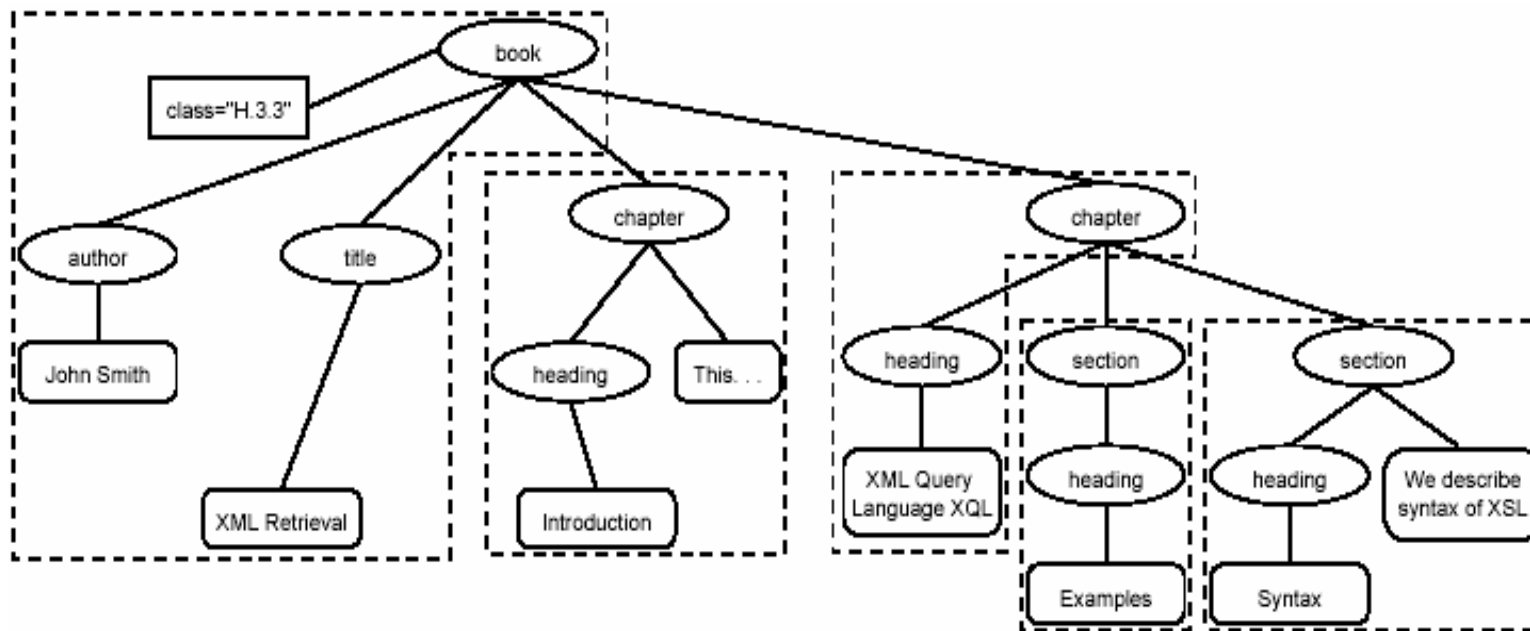
- <article>XXX YYY ZZZ ZZZ </article>
- <title>XXX</title>
- <abstract>YYY</abstract>
- <body>ZZZ ZZZ</body>
- <sec>ZZZ</sec>
- <sec>ZZZ</sec>

Retrieval units: XML sub-trees

- Indexing sub-trees is closest to traditional IR
 - each XML elements is bag of words of itself and its descendants
 - and can be scored as ordinary plain text document
- Advantage: well-understood problem
- Negative:
 - redundancy in index
 - terms statistics

 - Led to the notion of indexing nodes
 - Problem: how to select them?
 - manually, frequency, relevance data

(XIRQL) Indexing nodes



(Fuhr & Großjohann, SIGIR 2001)

Retrieval units: Disjoint elements

Assume a document like

```
<article>
  <title>XXX</title>
  <abstract>YYY</abstract>
  <body>
    <sec>ZZZ</sec>
    <sec>ZZZ</sec>
  </body>
</article>
```

Index separately

- <title>XXX</title>
- <abstract>YYY</abstract>
- <sec>ZZZ</sec>
- <sec>ZZZ</sec>

Note that <body> and <article> have not been indexed

Retrieval units 2: Disjoint elements

- Main advantage and main problem
 - (most) article text is not indexed under /article
 - avoids redundancy in the index

- But how to score higher level (non-leaf) elements?
 - Propagation/Augmentation approach
 - Element specific language models

(Geva, INEX 2004, INEX 2005)

Propagation - GPX model

Leaf elements score

$$L = N^{n-1} \sum_{i=1}^n \frac{t_i}{f_i}$$

n : the number of unique query terms
 N : a small integer ($N=5$, but any $10 > N > 2$ works)
 t_i : the frequency of the term in the leaf element
 f_i : the frequency of the term in the collection

Branch elements score

$$RSV = D(n) \sum_{i=1}^n L_i$$

n : the number of children elements
 $D(n) = 0.49$ if $n = 1$
 0.99 Otherwise

$D(n)$ = relationship statistics

L_i : child element score
 scores are recursively propagated up the tree

Element specific language model (simplified)

Assume a document

<bdy>

<sec>cat...</sec>

<sec>dog...</sec>

</bdy>

Query: cat dog
(Ogilvie & Callan, INEX 2004)

- Assume

- $P(\text{dog}|\text{bdy/sec}[1])=0.7$
- $P(\text{cat}|\text{bdy/sec}[1])=0.3$
- $P(\text{dog}|\text{bdy/sec}[2])=0.3$
- $P(\text{cat}|\text{bdy/sec}[2])=0.7$

- Mixture $P(w|e) = \sum \lambda_i P(w|e_i)$

- With uniform weights ($\lambda=0.5$)
- λ = relationship statistics
- $P(\text{cat}|\text{bdy})=0.5$
- $P(\text{dog}|\text{bdy})=0.5$
- So /bdy will be returned

Retrieval units: Distributed

- Index separately particular types of elements
- E.g., create separate indexes for

A blue rectangular box containing a list of XML element types: articles, abstracts, sections, subsections, subsubsections, and paragraphs ...

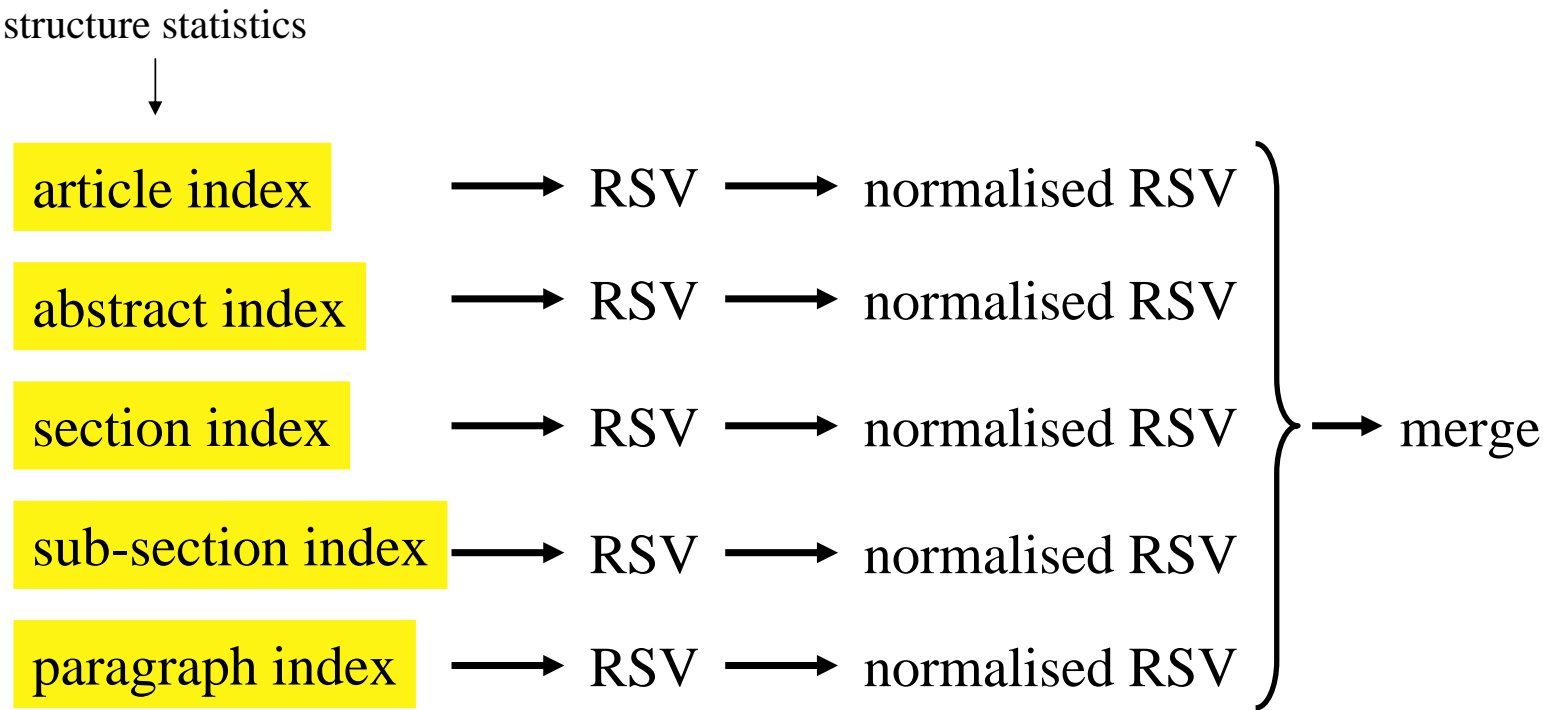
articles
abstracts
sections
subsections
subsubsections
paragraphs ...

A right-facing curly brace grouping the list of element types in the blue box to the text "structure statistics".

structure statistics

- Each index provides statistics tailored to particular types of elements
 - language statistics may deviate significantly
 - queries issued to all indexes
 - results of each index are combined (after score normalization)

Distributed: Vector space model



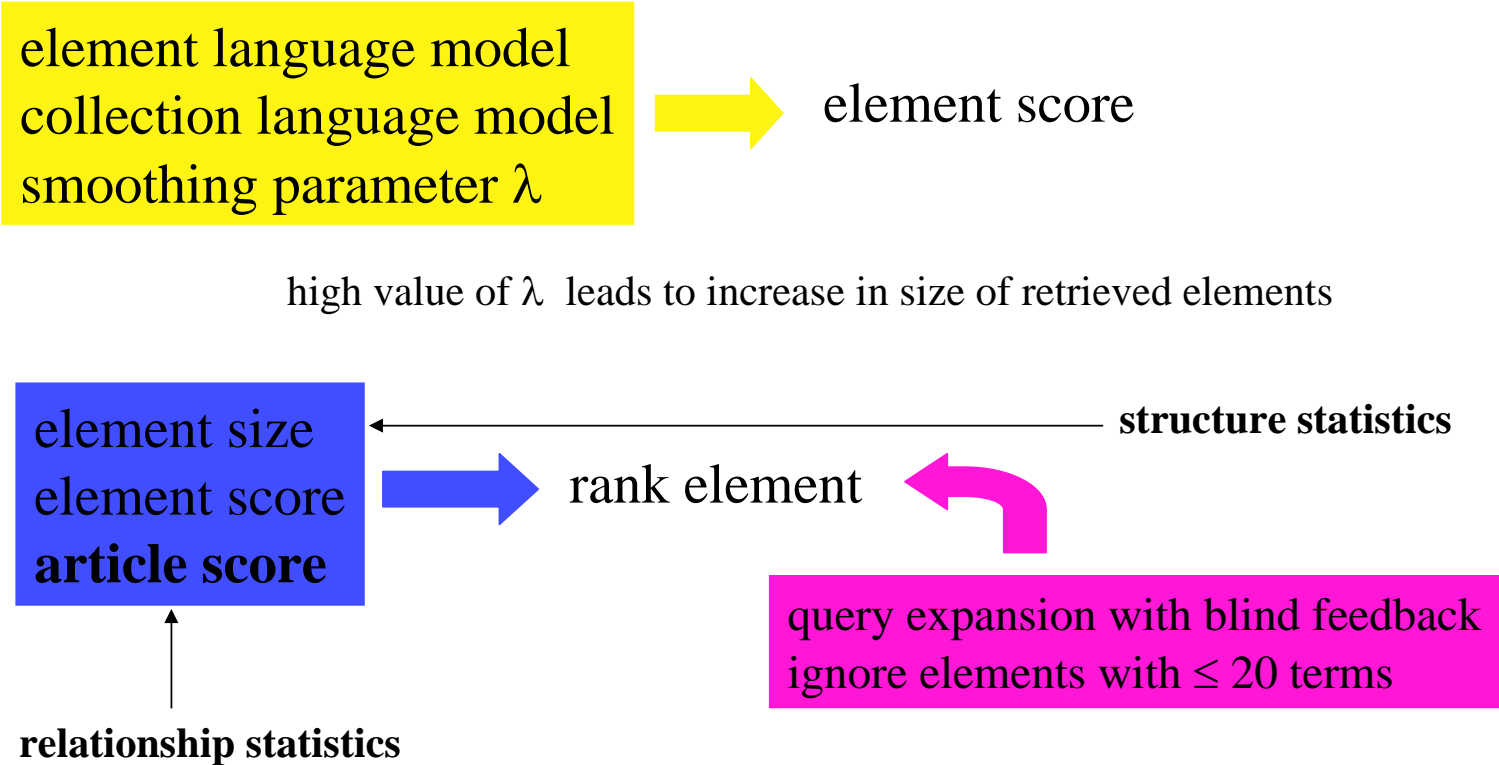
tf and idf as for fixed and non-nested retrieval units

(Mass & Mandelbrod, INEX 2004)

Retrieval units: Distributed

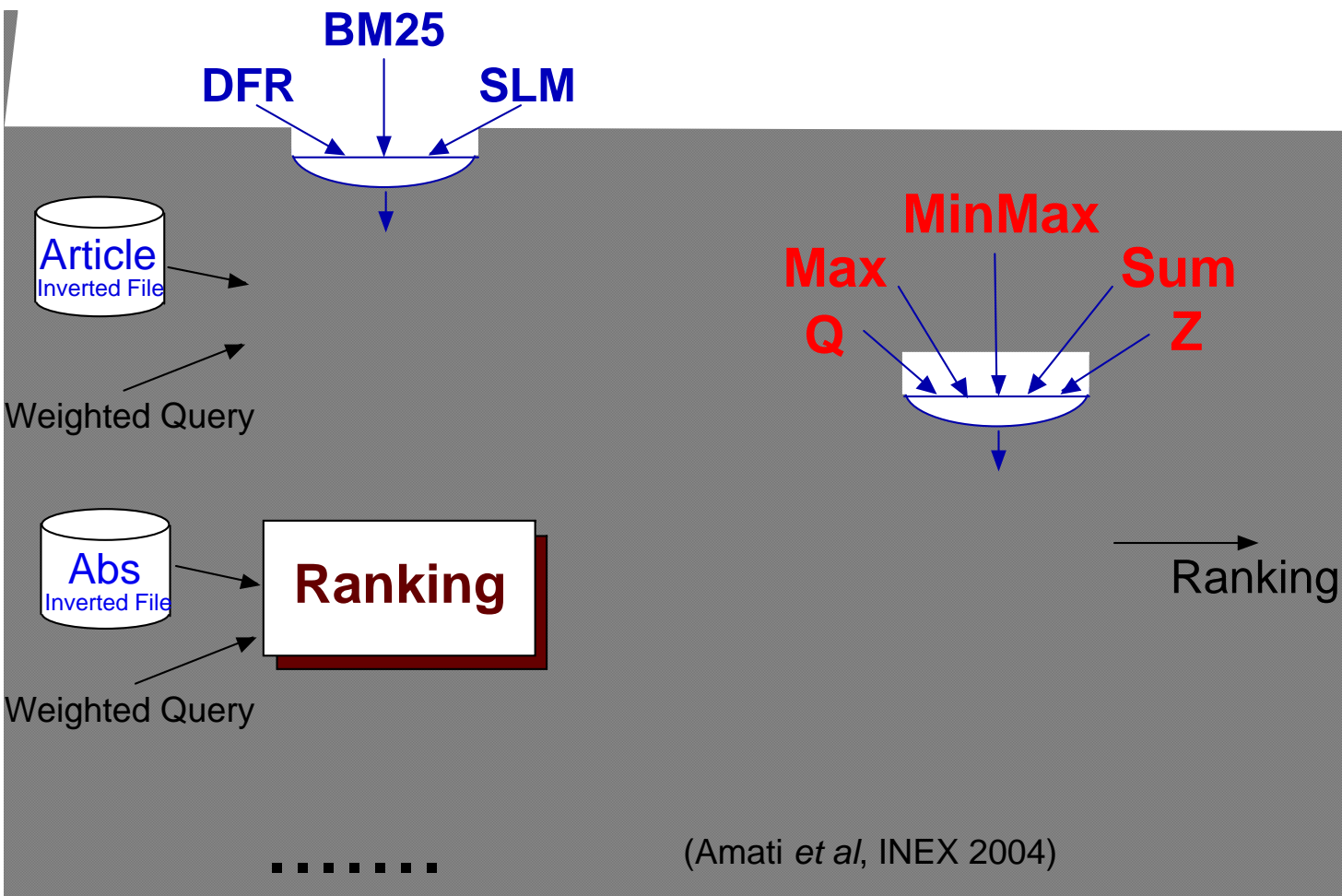
- Only part of the structure is used
 - Element size
 - Relevance assessment
 - Others
- Main advantages compared to disjoint element strategy:
 - avoids score propagation which is expensive at run-time
 - index redundancy is basically pre-computing propagation
 - XML specific propagation requires nontrivial parameters to train
- Indexing methods and retrieval models are “standard” IR
 - although issue of merging - normalization

Combination: Language model



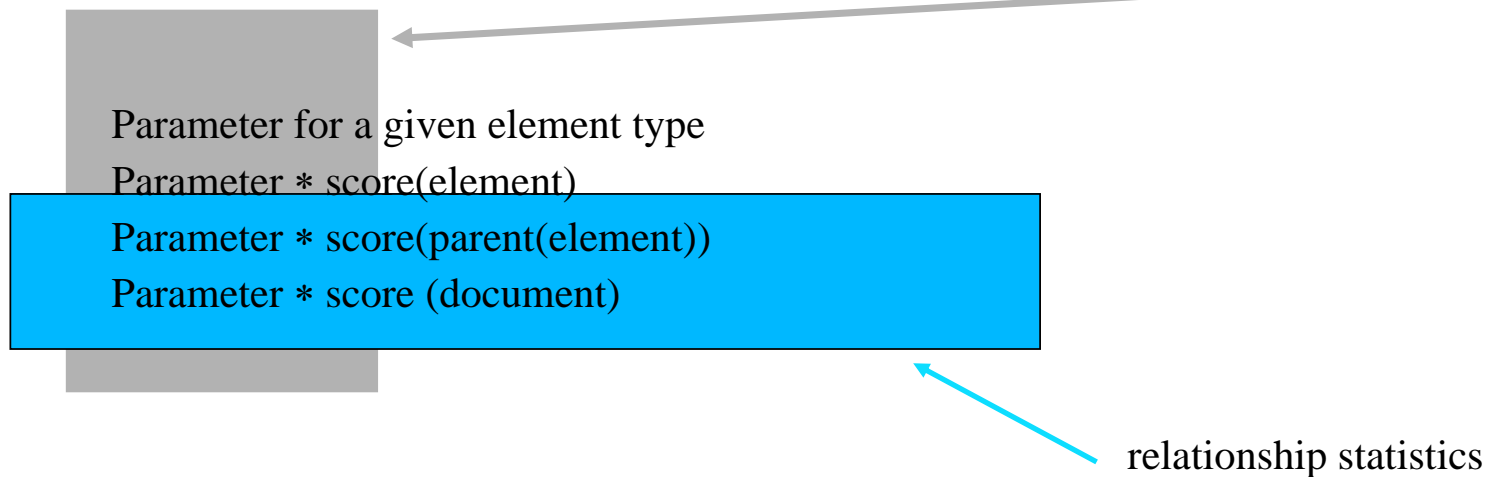
(Sigurbjörnsson etal, INEX 2003, INEX 2004)

Combination: Normalization



Combination: Machine learning

- Use of standard machine learning to train a function that combines



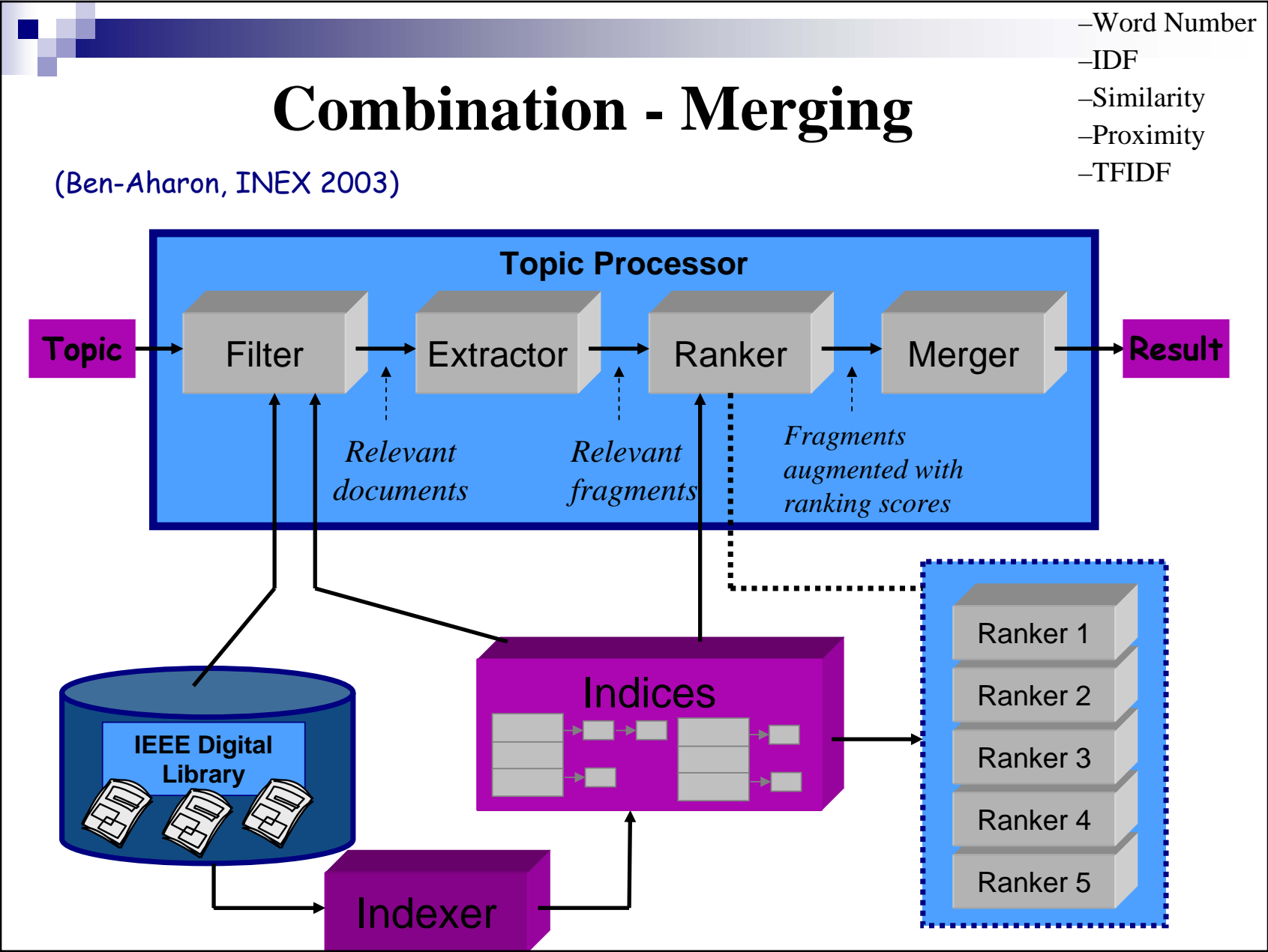
- Training done on relevance data (previous years)
- Scoring done using OKAPI

(Vittaut & Gallinari, ECIR 2006)

Combination: Contextualization

- Basic ranking by adding weight value of all query terms in element.
- Re-weighting is based on the idea of using the ancestors of an element as a context.
 - Root: combination of the weight of an element its $1.5 * \text{root}$.
 - Parent: average of the weights of the element and its parent.
 - Tower: average of the weights of an element and all its ancestors.
 - Root + Tower: as above but with $2 * \text{root}$.
- Here root is the document

(Arvola et al, CIKM 2005, INEX 2005)



Post-processing: Displaying XML Retrieval Results

- XML element retrieval is a core task
 - how to estimate the relevance of individual elements
- However, it may not be the end task
 - Simply returning a ranked list of elements results seems insufficient
 - **may have overlapping elements**
 - elements from the same article may be scattered
- This may be dealt with in special XML retrieval interfaces
 - Cluster results, provide heatmap, best entry point, ...

New retrieval tasks (at INEX)

- INEX 2005-7 addressed two new retrieval tasks
 - Thorough is 'pure' XML element retrieval as before
 - **Focused does not allow for overlapping elements to be returned**
 - **Fetch and Browse requires results to be clustered per article**
 - Various variants
- New tasks require post-processing of 'pure' XML element runs
 - geared toward displaying them in a particular interface

Post-processing: Controlling Overlap

What most approaches are doing:

- Given a ranked list of elements:
 1. select element with the highest score within a path
 2. discard all ancestors and descendants
 3. go to step 1 until all elements have been dealt with
- (Also referred to as brute-force filtering)

“Post”-Processing: Removing overlap

- Sometimes with some “prior” processing to affect ranking:

- ☐ Use of a utility function that captures the amount of useful information in an element

Element score * Element size * Amount of relevant information

- ☐ Used as a prior probability
- ☐ Then apply “brute-force” overlap removal

(Mihajlovic et al, INEX 2005; Ramirez et al, FQAS 2006))

Post-processing: Controlling Overlap

- Start with a component ranking, elements are re-ranked to control overlap.
- Retrieval status values of those components containing or contained within higher ranking components are iteratively adjusted
- (depends on amount of overlap “allowed”)
 1. Select the highest ranking component.
 2. Adjust the retrieval status value of the other components.
 3. Repeat steps 1 and 2 until the top m components have been selected.


(Clarke, SIGIR 2005)

Post-Processing: Removing overlap

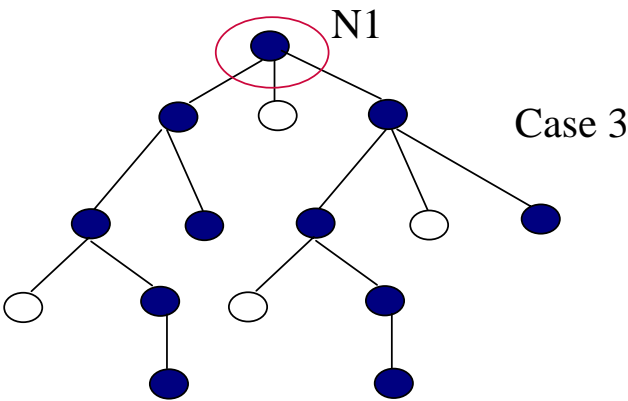
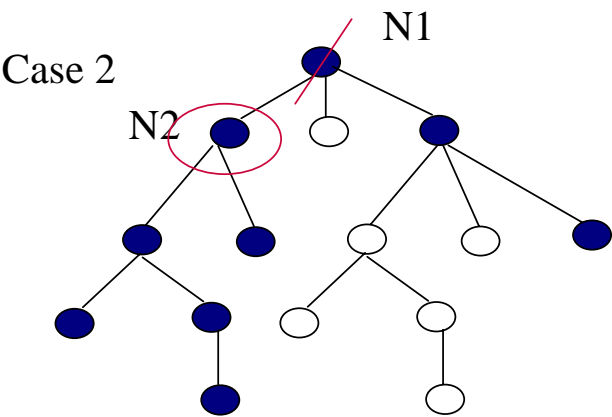
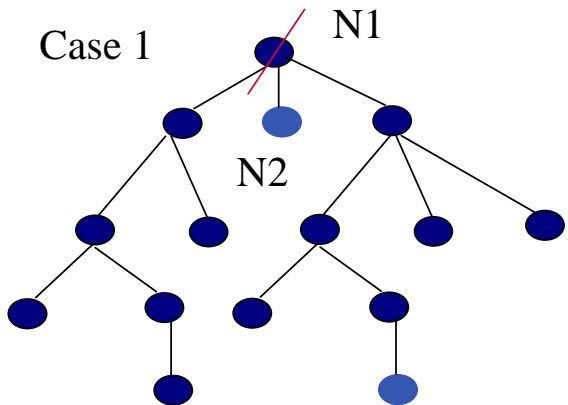
(Mass & Mandelbrod, INEX 2005)

Smart filtering

Given a list of rank elements

- group elements per article
- build a result tree 
- “score grouping”:

- for each element N1
 1. score N2 > score N1
 2. concentration of good elements
 3. even distribution of good elements



CAS query processing: sub-queries

- Sub-queries decomposition

- *//article [search engines] // sec [Internet growth] AND sec [Yahoo]*



- article [search engines]
 - sec [Internet growth]
 - sec [Yahoo]
- Run each sub-queries and then combine
- Reward structure matching (strict vs vague)

(Sauvagnat et al, INEX 2005)

Example of combination: Probabilistic algebra

// article [about(., bayesian networks)] // sec [about(., learning structure)]

$$R(\text{learning structure}) \cap \text{label}^{-1}(\text{sec}) \\ \cap \text{descendants}(R(\text{bayesian networks}) \cap \text{label}^{-1}(\text{article}))$$

- “Vague” sets
 - $R(\dots)$ defines a vague set of elements
 - $\text{label}^{-1}(\dots)$ can be defined for strict or vague interpretation
- Intersections and Unions are computed as probabilistic “and” and fuzzy-or.

(Vittaut et al, INEX 2004)

Vague structural constraints

- Define score between two tags/paths
- Boost content score with tag/path score
- Use of dictionary of equivalent tags/synonym list
 - Analysis of the collection DTD
 - Syntactic, e.g. “p” and “ip1”
 - Semantic, e.g. “capital” and “city”
 - Analysis of past relevance assessments
 - For topic on “section” element, all types of elements assessed relevant added to “section” synonym list
 - Probabilistic estimation of tag weights
- Ignore structural constraint for target, support element or both
- Relaxation techniques from DB (e.g. lowest common ancestor, etc)

XML Element retrieval - Recap

- Choice of retrieval units can affect the “type” of retrieval models
- XML retrieval can be viewed as a **combination of evidence** problem
- No “clear winner” in terms of retrieval models
 - We still miss the benchmark/baseline approach
 - Lots of heuristics
- BUT WHAT SEEM TO WORK WELL:
 - Element
 - Document
 - Size
- *Thorough investigation for all ranking models, all indexing approaches, and all evidence needed*

User aspects

- User study - INEX interactive track
- Incorporating user behaviour

Evaluation of XML retrieval: INEX

- Evaluating the effectiveness of **content-oriented** XML retrieval approaches
- Similar methodology as for TREC, but adapted to XML retrieval

(to be described later)



Interactive Track in 2004

- **Investigate behaviour of searchers when interacting with XML components**
- **Content-only Topics**
 - topic type an additional source of context
 - Background topics / Comparison topics
 - 2 topic types, 2 topics per type
 - 2004 INEX topics have added task information
- **Searchers**
 - “distributed” design, with searchers spread across participating sites

Topic Example

<title>+new +Fortran +90 +compiler**</title>**

<description> How does a Fortran 90 compiler differ from a compiler for the Fortran before it. **</description>**

<narrative> I've been asked to make my Fortran compiler compatible with Fortran 90 so I'm interested in the features Fortran 90 added to the Fortran standard before it. I'd like to know about compilers (they would have been new when they were introduced), especially compilers whose source code might be available. Discussion of people's experience with these features when they were new to them is also relevant. An element will be judged as relevant if it discusses features that Fortran 90 added to Fortran. **</narrative>**

<keywords>new Fortran 90 compiler**</keywords>**

Baseline system

dbdk_training in Baseline
System



query was: text classification naive bayes
Results **1** - **10** of **100**.
Result pages: **1** **2** **3** **4** **5** **6** **7** **8** **9** **10** [next](#)



Search Result

- 1: (0.247) **Scalable Feature Mining for Sequential Data**
Neal Lesh Mitsubishi Electric Research Lab Mohammed J. Zaki Rensselaer Polytechnic Institute Mitsunori Ogiwara University of Rochester
[Result path: /article\[1\]/bdy\[4\]/sec\[5\]](#)
- 2: (0.204) **Probability and Agents**
Marco G. Valtorta University of South Carolina mgv@cse.sc.edu Michael N. Huhns University of South Carolina huhns@sc.edu
[Result path: /article\[1\]/bdy\[4\]/sec\[3\]](#)
- 3: (0.176) **Combining Image Compression and Classification Using Vector Quantization**
Karen L. Oehler Member IEEE Robert M. Gray Fellow IEEE
[Result path: /article\[1\]/bdy\[4\]/sec\[4\]/ss1\[2\]/ss2\[4\]](#)
- 4: (0.175) **Text-Learning and Related Intelligent Agents: A Survey**
Dunja Mladenic J. Stefan Institute
[Result path: /article\[1\]/bm\[5\]/app\[4\]/sec\[5\]](#)
- 5: (0.175) **Detecting Faces in Images: A Survey**
Ming-Hsuan Yang Member IEEE David J. Kriegman Senior Member IEEE Narendra Ahuja Fellow IEEE
[Result path: /article\[1\]/bdy\[4\]/sec\[2\]/ss1\[9\]/ss2\[10\]](#)

Some results

- How far down the ranked list?
 - 83 % from rank 1-10
 - 10 % from rank 11-20
- Query operators rarely used
- 80 % of queries consisted of 2, 3, or 4 words
- Accessing components
 - ~2/3 was from the ranked list
 - ~1/3 was from the document structure (ToC)
- 1st viewed component from the ranked list
 - 40% article level, 36% section level, 22% ss1 level, 4% ss2 level
- ~ 70 % only accessed 1 component per document

Document-centric XML retrieval: Conclusions

- SDR → now mostly about XML retrieval
- Efficiency:
 - Not just documents, but all its elements
- Models
 - Units
 - Statistics
 - Combination
- User tasks
- Link to web retrieval / novelty retrieval
- Interface and visualisation
- Clustering, categorisation, summarisation

Outline

- Introduction to XML, basics and standards
- Document-oriented XML retrieval
- **Evaluating XML retrieval effectiveness**

Evaluating XML retrieval effectiveness

- Structured document retrieval and evaluation
- XML retrieval evaluation
 - Collections
 - Topics
 - Retrieval tasks
 - Relevance and assessment procedures
 - Metrics

SDR and Evaluation

■ Passage retrieval

- Test collection built for that purpose, where passages in relevant documents were assessed (Wilkinson SIGIR 1994)

■ Structured document retrieval

- Web retrieval collection (museum) (Lalmas & Moutogianni, RIAO 2000)
- Fictitious collection (Roelleke et al, ECIR 2002; Ruthven & Lalmas JDoc 1998)
- Shakespeare collection (Kazai *et al*, ECIR 2003)

■ **INEX initiative** (Kazai *et al*, JASIST 2004; INEX proceedings; SIGIR forum reports, ...)

- “Real” large test collection following TREC methodology
- Evaluation campaign
- XML

Evaluation of XML retrieval: INEX

- Evaluating the effectiveness of **content-oriented** XML retrieval approaches
- **Collaborative** effort \Rightarrow participants contribute to the development of the collection
 - queries
 - relevance assessments
 - methodology
- Similar methodology as for TREC, but adapted to XML retrieval



<http://inex.is.informatik.uni-duisburg.de/>

Document collections					
Year	number documents	number elements	size	average number elements	average element depth
2002-2004	12,107	8M	494MB	1,532	6.9
2005	16,819	11M	764MB	“	“
2006-2007	659,388	52M	60 (4.6)GB	161.35	6.72

IEEE

Wikipedia

(Denoyer & Gallinari, SIGIR Forum, June 2006)

Two types of topics

■ Content-only (CO) topics

- ☐ ignore document structure
- ☐ simulates users, who do not have any knowledge of the document structure or who choose not to use such knowledge

■ Content-and-structure (CAS) topics

- ☐ contain conditions referring both to content and structure of the sought elements
- ☐ simulate users who do have some knowledge of the structure of the searched collection

CO topics 2003-2004

<title>

"Information Exchange", +"XML", "Information Integration"

</title>

<description>

How to use XML to solve the information exchange (information integration) problem, especially in heterogeneous data sources?

</description>

<narrative>

Relevant documents/components must talk about techniques of using XML to solve information exchange (information integration) among heterogeneous data sources where the structures of participating data sources are different although they might use the same ontologies about the same content.

</narrative>

CAS topics 2003-2004

<title>

//article[(./fm//yr = '2000' OR ./fm//yr = '1999') AND about(., '"intelligent transportation system"')]/sec[about(., 'automation +vehicle')]

</title>

<description>

Automated vehicle applications in articles from 1999 or 2000 about intelligent transportation systems.

</description>

<narrative>

To be relevant, the target component must be from an article on intelligent transportation systems published in 1999 or 2000 and must include a section which discusses automated vehicle applications, proposed or implemented, in an intelligent transportation system.

</narrative>

NEXI

- Narrowed Extended XPath I
- INEX Content-and-Structure (CAS) Queries
- Specifically targeted for content-oriented XML search (i.e. “**aboutness**”)

`//article[about(.//title, apple) and
about(.//sec, computer)]`

(Trotman & Sigurbjörnsson, INEX 2004)

(Sigurbjörnsson & Trotman, INEX 2003)

CO+S topics 2005-2006

<title>markov chains in graph related algorithms</title>

<castitle>//article//sec[about(.,+"markov chains" +algorithm +graphs)] </castitle>

<description>Retrieve information about the use of markov chains in graph theory and in graphs-related algorithms.

</description>

<narrative>I have just finished my Msc. in mathematics, in the field of stochastic processes. My research was in a subject related to Markov chains. My aim is to find possible implementations of my knowledge in current research. I'm mainly interested in applications in graph theory, that is, algorithms related to graphs that use the theory of markov chains. I'm interested in at least a short specification of the nature of implementation (e.g. what is the exact theory used, and to which purpose), hence the relevant elements should be sections, paragraphs or even abstracts of documents, but in any case, should be part of the content of the document (as opposed to, say, vt, or bib).

</narrative>

Retrieval tasks

■ Ad hoc retrieval:

“a simulation of how a library might be used and involves the searching of a static set of XML documents using a new set of topics”

- ☐ Ad hoc retrieval for CO topics
- ☐ Ad hoc retrieval for CAS (+S) topics

■ Core task:

- ☐ “identify the most appropriate granularity XML elements to return to the user, with or without structural constraints”

CO retrieval task (2002 -)

■ Specification:

- ☐ make use of the CO topics
- ☐ retrieves the most specific elements and only those, which are relevant to the topic
- ☐ no structural constraints regarding the appropriate granularity
- ☐ must identify the most appropriate XML elements to return to the user

■ Two main strategies

- Thorough strategy
- Focused strategy

Thorough strategy (“2002” - 2006)

- Specification:

- ☐ “core system's task underlying most XML retrieval strategies, which is to estimate the relevance of potentially retrievable elements in the collection”
- ☐ overlap problem viewed as an interface and presentation issues
- ☐ challenge is to rank elements appropriately

- Task that most XML approaches performed up to 2004 in INEX.

Focused strategy (2005 -)

■ Specification:

“find the most exhaustive and specific element on a path within a given document containing relevant information and return to the user only this most appropriate unit of retrieval”

- ☐ no overlapping elements
- ☐ return parent (2005) / child (2006-7) if same estimated relevance between parent and child elements
- ☐ preference for specificity over exhaustivity

CAS retrieval task (2002 - 2004)

■ **Strict content-and-structure:**

- retrieve relevant elements that exactly match the structure specified in the query (2002, 2003)

■ **Vague content-and-structure:**

- retrieve relevant elements that may not be the same as the target elements, but are structurally similar (2003)
- retrieve relevant elements even if do not exactly meet the structural conditions; treat structure specification as hints as to where to look (2004)

CAS (+S) retrieval task (2005 -)

- Make use of CO+S **topics**: <castitle>
- **Structural hints**:
 - “Upon discovering that his/her <title> query returned many irrelevant elements, a user might decide to add structural hints, i.e. to write his/her initial CO query as a CAS query”

open standards for digital video in distance learning



//article//sec[about(.,open standards for digital video in distance learning)]

- Two strategies (as for CO retrieval task):
 - *Focussed strategy*
 - *Thorough strategy*

(Trotman & Lalmas, SIGIR Poster 2006)

Fetch & Browse (2005 - 2007)

- Document ranking, and in each document, element ranking or set (called Relevant in Context in 2006-7)

OntoSeek: Content-Based Access to the Web

Nicola Guarino, Claudio Masolo, Guido Vetere

article[1]

fm[1]

abs[1]

p[1]

... and precision of content-based **retrieval**. Our OntoSeek system adopts ... large ontology based on **WordNet** for content matching.

bdy[1]

sec[3] (THE ROLE OF LINGUISTIC ONTOLOGIES)

ss1[1] (Some advantages)

p[4]

The **retrieval** quality improves considerably if ... linguistic ontology such as **WordNet**. For example, let's add **WordNet** to a simple matching ...

p[7]

... linguistic ontologies such as **WordNet** and structured representation formalisms can help an **information-retrieval** system to

sec[4] (ONTOSEEK)

... of a project on **retrieval** and reuse of object-oriented ... system designed for content-based **information retrieval** from online yellow pages ... mostly resulting from merging **WordNet**'s thesaurus into the Penman ... broad ontology endowed with **WordNet**'s powerful lexical interface, which ...

(Courtesy of Sigurbjörnsson)

Best in context (2006 - 2007)

- Document ranking, and in each document, return the best entry point
 - Element from where to start reading
 - Analysis:
 - Mostly not the beginning of the document
 - Often the element that is part of the first relevant fragment

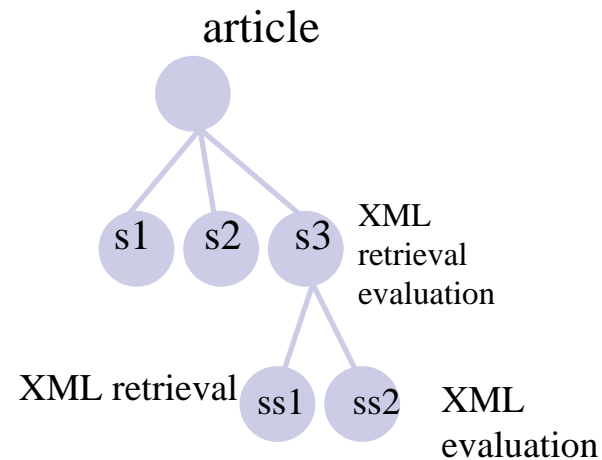
(Kamp et al, SIGIR 2007 Poster)

Relevance in XML retrieval

- A document is **relevant** if it “has significant and demonstrable bearing on the matter at hand”.
- Common assumptions in laboratory experimentation:
 - **Objectivity**
 - **Topicality**
 - **Binary nature**
 - **Independence**

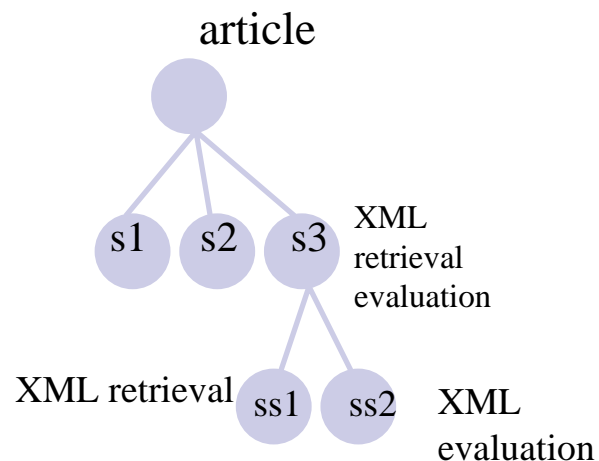
(Borlund, JASIST 2003)

(Goevert et al, JIR 2006)



(based on Chiaramella et al, FERMI fetch and browse model 1996)

Relevance in XML retrieval: INEX 2003 - 2004



- **Topicality not enough**
- **Binary nature not enough**
- **Independence is wrong**

- **Relevance = (0,0) (1,1) (1,2) (1,3) (2,1) (2,2) (2,3) (3,1) (3,2) (3,3)**

exhaustivity = how much the section discusses the query: 0, 1, 2, 3

specificity = how focused the section is on the query: 0, 1, 2, 3

- **If a subsection is relevant so must be its enclosing section, ...**

Relevance - to recap

- find smallest component (\rightarrow specificity) that is highly relevant (\rightarrow exhaustivity)
- **specificity**: extent to which a document component is focused on the information need, while being an informative unit.
- **exhaustivity**: extent to which the information contained in a document component satisfies the information need.

continuous scale defined as ratio (in characters) of the highlighted text to element size.

  \sim

The dilemma in testing embedded DRAM arises from differences in character between ASICs and commodity DRAMs. In the case of commodity DRAMs, despite huge amounts of production, manufacturers produce only a few different products at the same time. As a result, they can optimize the testing methodology for each product. In contrast, companies produce a large variety of ASIC products, but the production volume of each product is small. Also, ASICs require a very short turnaround time. Therefore, customizing the test methodology for each product is difficult. ASICs require a common test environment that covers all product variations.



Exhaustivity dimension

Scale reduced to 3+1:

- ☐ Highly exhaustive (2): the element discussed most or all aspects of the query.
- ☐ Partly exhaustive (1): the element discussed only few aspects of the query.
- ☐ Not exhaustive (0): the element did not discuss the query.
- ☐ Too Small (?): the element contains relevant material but is too small to be relevant on it own.

New assessment procedure led to better quality assessments
(Piwowarski et al, 2007)

Further simplification

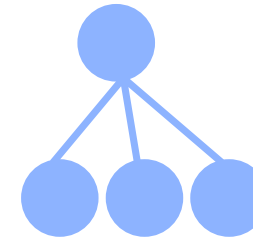
- Statistical analysis on the INEX 2005 data:
 - The exhaustivity 3+1 scale is not needed in most scenarios to compare XML retrieval approaches
 - *The two small maybe simulated by some threshold length*
- INEX 2006-7 use only the specificity dimension to “measure” relevance
 - The same highlighting approach is used
 - *Some investigation being done regarding the two small elements*

(Ogilvie & Lalmas, 2006)

Measuring effectiveness: Metrics

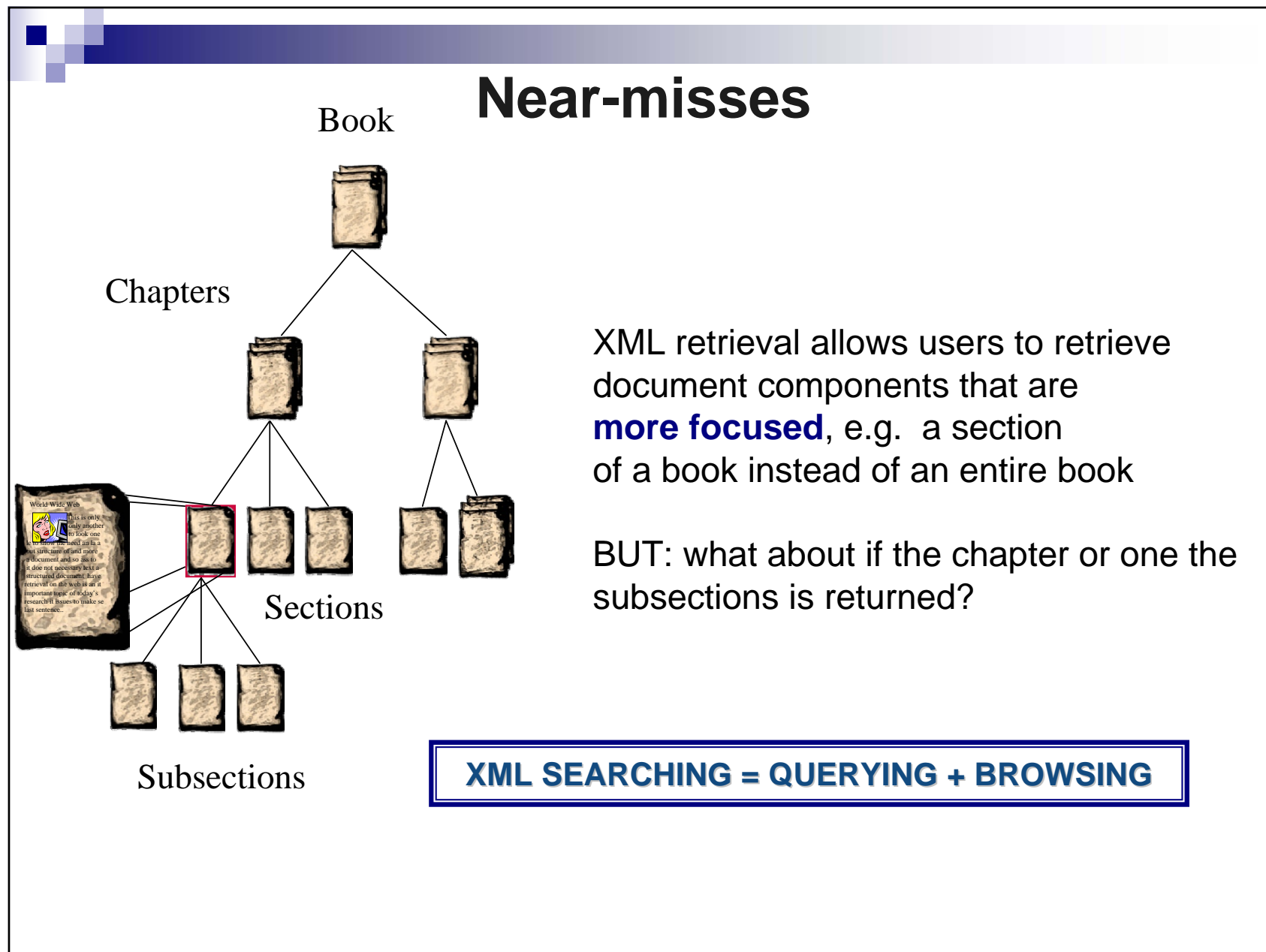
■ Need to consider:

- Multi-graded dimensions of relevance
- Near-misses

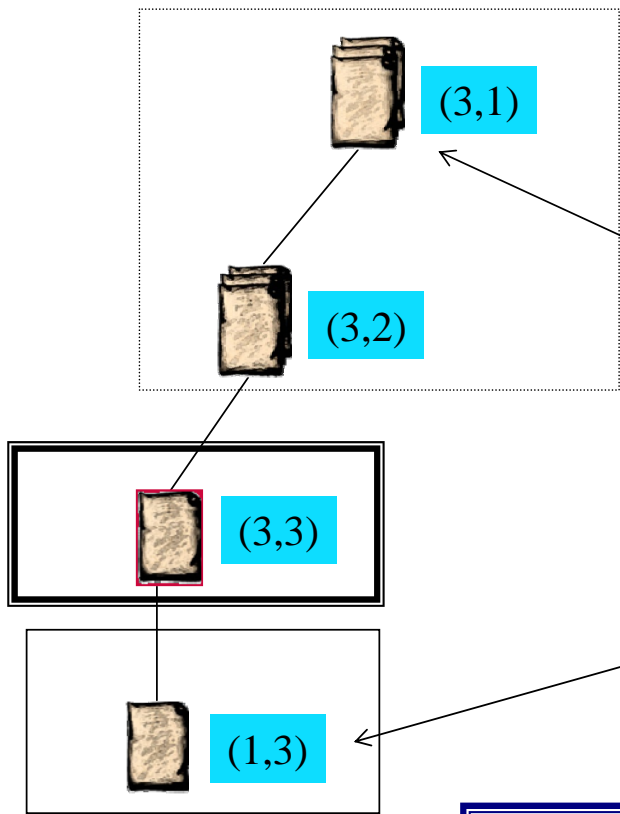


■ Metrics

- `inex_eval` (also known as `inex2002`) (Goevert & Kazai, INEX 2002)
official INEX metric 2002-2004
- `inex_eval_ng` (also known as `inex2003`) (Goevert et al, JIR 2006)
- ERR (expected ratio of relevant units) (Piwowarski & Gallinari, INEX 2003)
- xCG (XML cumulative gain) (Kazai & Lalmas, TOIS 2006)
official INEX metric 2005-2006
- `t2i` (tolerance to irrelevance) (de Vries *et al*, RIAO 2004)
- EPRUM (Expected Precision Recall with User Modelling) (Piwowarski & Dupret, SIGIR 2006)
- HiXEval (Highlighting XML Retrieval Evaluation) (Pehcevski & Thom, INEX 2005)
 - Variant of it is now official INEX metric 2007- (Kamps *et al*, INEX 2007)
- ...



Near-misses (2004 scale)



XML retrieval allows users to retrieve document components that are **more focused**, e.g. a section of a book instead of an entire book

BUT: what about if the chapter or one the subsections is returned?

XML SEARCHING = QUERYING + BROWSING

(exhaustivity, specificity)

Retrieve the best XML elements according to content and structure criteria (2004 scale):

- Most exhaustive and the most specific = (3,3)

- Near misses = (3,3) + (2,3) (1,3) ← specific

- Near misses = (3, 3) + (3,2) (3,1) ← exhaustive

- Near misses = (3, 3) + (2,3) (1,3) (3,2) (3,1) (1,2) ...

near-misses

Two multi-graded dimensions of relevance

■ Several “user models”

- **Expert and impatient**: only reward retrieval of highly exhaustive and specific elements (3,3) → *no near-misses*
- Expert and patient: only reward retrieval of highly specific elements (3,3), (2,3) (1,3) → (2,3) and (1,3) are *near-misses*
- ...
- **Naïve and has lots of time**: reward - to a different extent - the retrieval of any relevant elements; i.e. everything apart (0,0) → *everything apart (3,3) is a near-miss*

■ Use a quantisation function for each “user model”

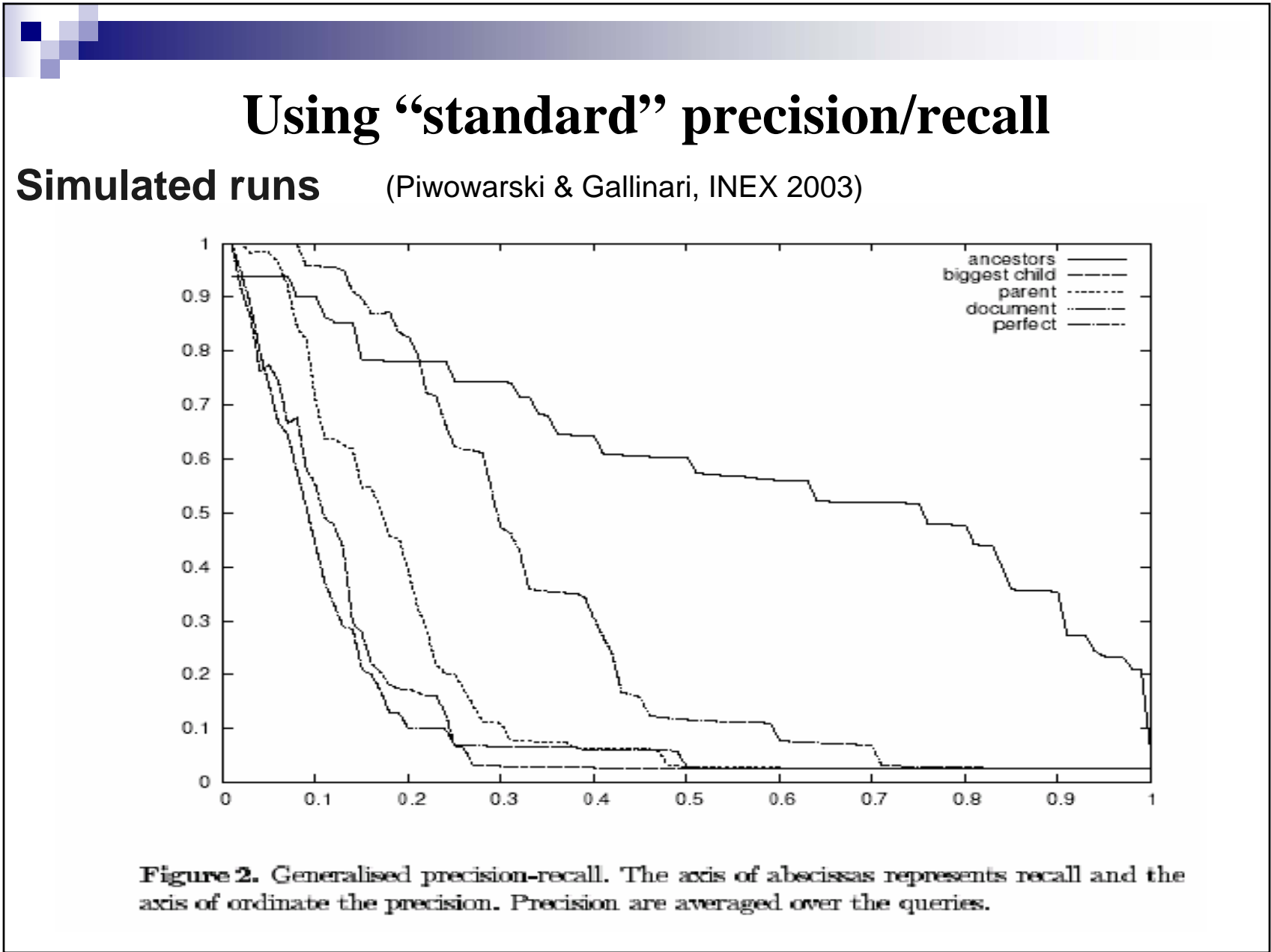
Examples of quantization functions

Expert and impatient

$$quant_{strict}(e,s) = \begin{cases} 1 & \text{if } (e,s) = (3,3) \\ 0 & \text{otherwise} \end{cases}$$

Naïve and has a lot of time

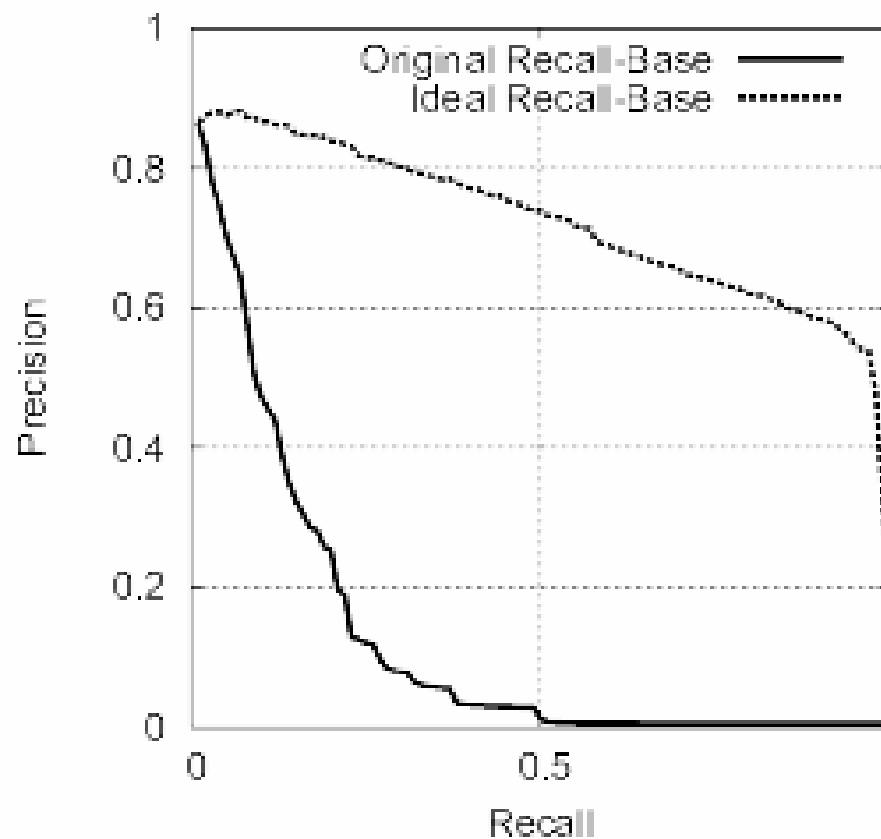
$$quant_{\text{gen}}(e,s) = \begin{cases} 1.00 & \text{if } (e,s) = (3,3) \\ 0.75 & \text{if } (e,s) \in \{(2,3), (3,2), (3,1)\} \\ 0.50 & \text{if } (e,s) \in \{(1,3), (2,2), (2,1)\} \\ 0.25 & \text{if } (e,s) \in \{(1,1), (1,2)\} \\ 0.00 & \text{if } (e,s) = (0,0) \end{cases}$$



Overlap in results			
Official INEX 2004 Results for CO topics			
Rank	Systems (runs)	Avg Prec	Overlap
1.	IBM Haifa Research Lab (CO-0.5-LAREFIENMENT)	0.1437	80.89
2.	IBM Haifa Research Lab (CO-0.5)	0.1340	81.46
3.	University of Waterloo (Waterloo-Baseline)	0.1267	76.32
4.	University of Amsterdam (UAMS-CO-T-FBack)	0.1174	81.85
5.	University of Waterloo (Waterloo-Expanded)	0.1173	75.62
6.	Queensland University of Technology (CO_PS_Stop50K)	0.1073	75.89
7.	Queensland University of Technology (CO_PS_099_049)	0.1072	76.81
8.	IBM Haifa Research Lab (CO-0.5-Clustering)	0.1043	81.10
9.	University of Amsterdam (UAMS-CO-T)	0.1030	71.96
10.	LIP6 (simple)	0.0921	64.29

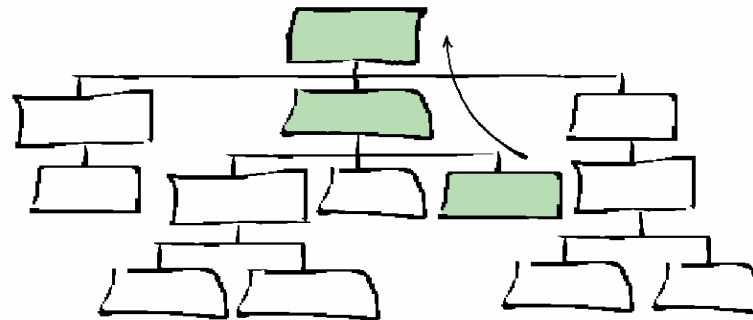
Overlap in recall-base

100% recall only if all relevant elements returned including overlapping elements



(Kazai et al, SIGIR 2004)

Relevance propagates up!



- ~26,000 relevant elements on
~14,000 relevant paths
- **Propagated assessments: ~45%**
- **Increase in size of recall-base: ~182%**
(INEX 2004 data)

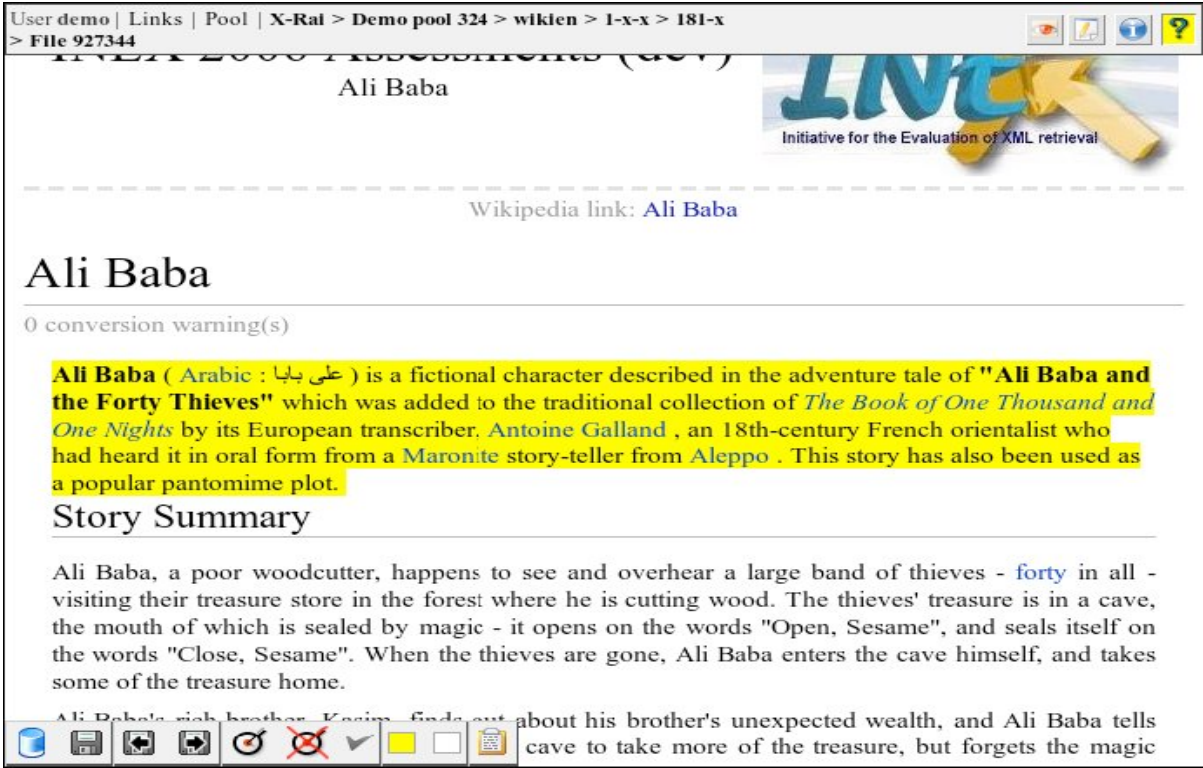
(Kazai et al, SIGIR 2004)

XCG: XML cumulated gain measures

- Based on cumulated gain measure for IR (Kekäläinen and Järvelin, TOIS 2002)
- Accumulate gain obtained by retrieving elements up to a given rank; thus not based on precision and recall → **user-oriented measures**
- Extended to include a precision/recall behaviour → **system-oriented measures**
- Require the construction of
 - an ideal recall-base to **separate what should be retrieved and what are near-misses**
 - an associated ideal run, which **contains what should be retrieved**
- with which retrieval runs are compared, which include what is being retrieved, **including near-misses**.

(Kazai & Lalmas, TOIS 2006)

HiXEval - Generalized precision and recall based on amount of highlighted content



For each element, we derive:
rsize: number of highlighted characters
size: number of characters

For each topic, we derive:
Trel: number of highlighted characters
in collection

HiXEval - Generalized precision and recall based on amount of highlighted content

- Precision at rank r
$$P(r) = \frac{\sum_{i=1}^r \text{rsize}(e_i)}{\sum_{i=1}^r \text{size}(e_i)}$$
- Recall at rank r
$$R(r) = \frac{1}{T_{\text{rel}}} \cdot \sum_{i=1}^r \text{rsize}(e_i)$$
- F-measure at rank r , average precision, MAP, etc

(Pehcevski & Thom, INEX 2005; Kamps *et al*, INEX 2007)

Evaluation and INEX - Recap

- Larger and more realistic collection with Wikipedia
- Better understanding of information needs and retrieval scenarios
- Better understanding of how to measure effectiveness
 - Near-misses and overlaps
 - Application to other IR problems
- Who are the real users?
 - Larsen et al, SIGIR 2006 poster; Betsi et al, SIGIR 2006 poster; Pharo & Trotman, SIGIR Forum 2007.

Conclusions

- XML Retrieval is still under development
- Technology is also changing
- Major advances in XML search (ranking) approaches made possible with INEX
- Evaluating XML retrieval effectiveness itself a research problem
- Many open problems for research

Areas for Open Problems

■ DB and IR

- Interaction between traditional DB query optimization (query rewriting) and ranking

■ “Old” vs. new IR models

- Combination of evidence problem
- What evidence to use?

■ Simple/succinct vs. complex/verbose QL

- Define an XQuery core?

Areas for Open Problems

- Indexing & searching
 - Efficient algorithms
- INEX test collection and effectiveness
 - Too complex?
 - What constitutes a retrieval baseline?
 - Generalisation of the results on other data sets
- Quality evaluation (Web, XML)
 - Who are the users?
 - What are their information needs?
 - What are the requirements?

Beyond XML retrieval

- Focused retrieval
- Aggregated results
- Structural context summarization
- Beyond the logical structure

Acknowledgements

- These slides are based on a number of presentations from the presenters at other events and from other researchers.
 - M. Consens, R. A. Baeza-Yates, M. Lalmas, S. Amer-Yahia: XML retrieval: DB/IR in theory, web in practice. VLDB 2007
 - S. Amer-Yahia, R. Baeza-Yates, M. Consens and M. Lalmas. XML Retrieval: Integrated IR-DB Challenges and Solutions. SIGIR 2007.
 - S. Amer-Yahia and M. Lalmas. Accessing XML Content: From DB and IR Perspectives, CIKM 2005.
 - R. Baeza-Yates and N. Fuhr. XML Retrieval, SIGIR 2004
 - R. Baeza-Yates and M. Consens. The Continued Saga of DB-IR Integration, SIGIR 2005
 - M. Lalmas. Structure/XML retrieval. ESSIR 2005, ESSIR 2007
 - M. de Rijke, J. Kamps and M. Marx. Retrieving Content and Structure, ESSLLI 2005
 - B. Sigurbjörnsson, Element Retrieval in Action, QMUL Seminar 2005
 - R. Baeza-Yates and M. Lalmas, XML Information Retrieval, SIGIR 2006