

# Strategies for Effective and Efficient XML Retrieval

Norbert Gövert

University of Dortmund

- Content-oriented XML Retrieval
- Evaluation: INEX
- XML Retrieval Model – *gain high quality retrieval results*
  - Content Accessibility in hierarchically structured Documents
  - Content Augmentation
  - Results
- Implementation – *gain high quality retrieval results **fast***
  - Classification of Retrieval Algorithms
  - Tunable: quit, continue
  - Interruptible: Nosferatu\*
- Outlook

## Content-oriented XML Retrieval

XML documents explicit logical structure.

What's the benefit for users in IR applications?

**Aim:** Retrieve the *most specific* document components  
which are still *exhaustive* w. r. t. information need

Here: only content-only queries, no querying w. r. t. structure

# Evaluation: INEX

**IN**itiative for the **E**valuation of **X**ML Retrieval

Evaluation methods for standard document IR (TREC, CLEF) don't help

**INEX**: Infrastructure for the evaluation of content-oriented XML retrieval

- document collection: 12107 IEEE CS journal articles
- standard, real life tasks / topics / information needs / queries:  
content-only (CO) and content-and-structure (CAS)
- relevance assessments: topical relevance and document coverage
- evaluation metric: based on Recall and precision

## XML Retrieval Model

*Gain high quality retrieval results*

**Aim:** Retrieve the *most specific* document components  
which are still *exhaustive* w. r. t. information need

**Either:** Invent new weighting schemes for XML retrieval

**Or:** Generalize well-performing weighting schemes from traditional IR applications

## XML Retrieval Model II

**Aim:** Retrieve the *most specific* document components which are still *exhaustive* w. r. t. information need

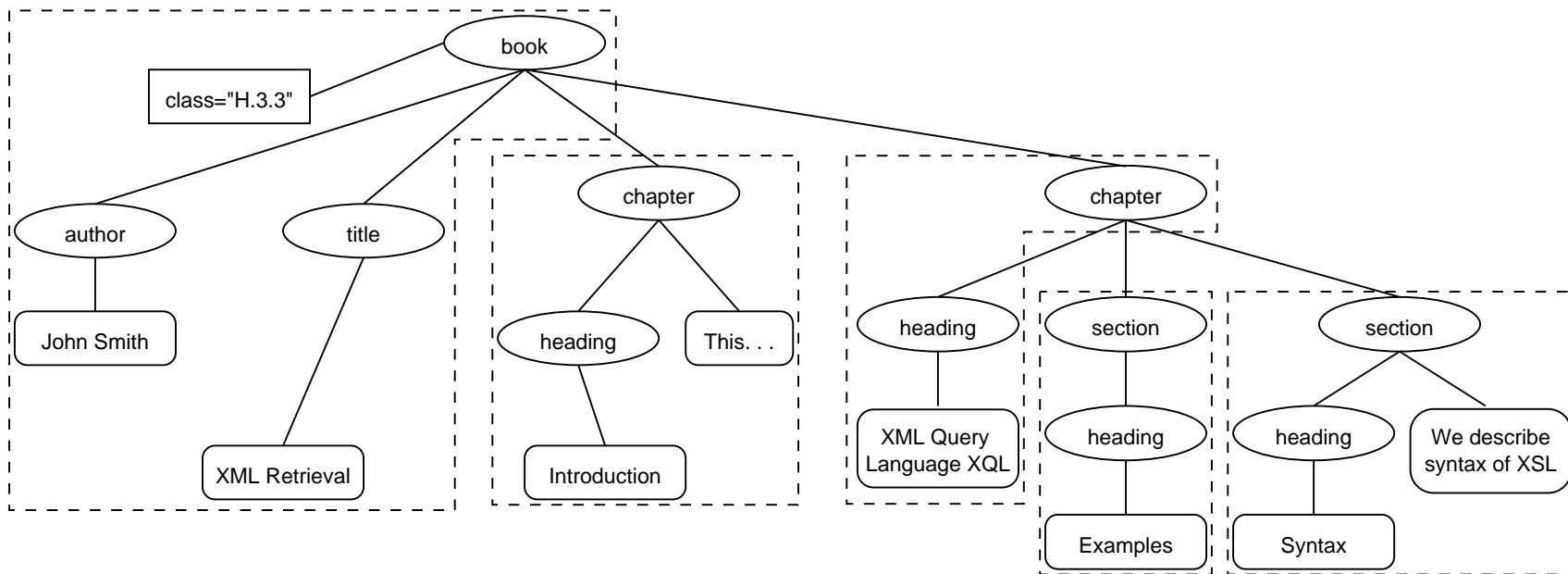
### **Approach:**

1. What are possible retrieval answers?
2. How is content of such an retrieval answer made up?  
→ *accessibility*
3. How is knowledge propagated to build up content of an answer?  
→ *augmentation*

# XML Retrieval Model III: Index Objects

Definition of atomic units in XML documents:

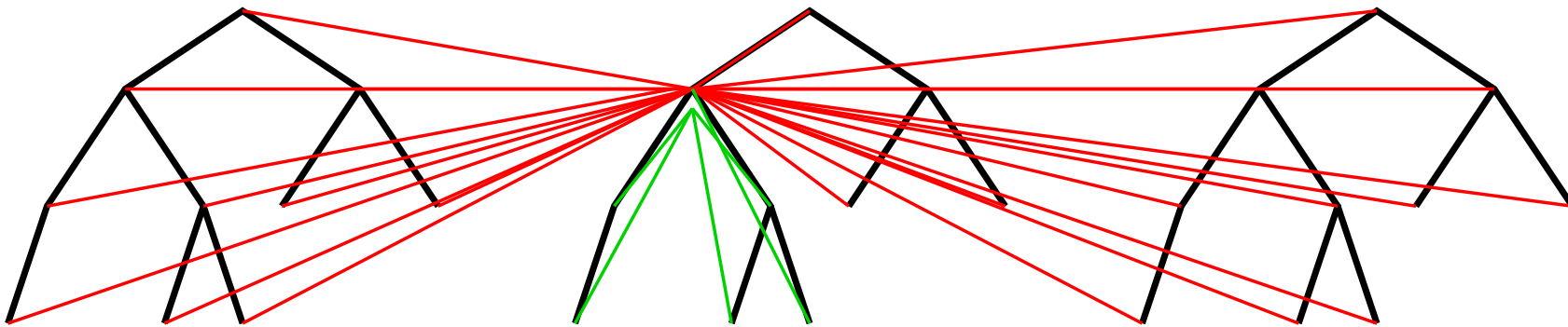
- Definition of set of retrievable answers
- Application of standard term weighting functions



## XML Retrieval Model IV: Accessibility

Content of an index object:

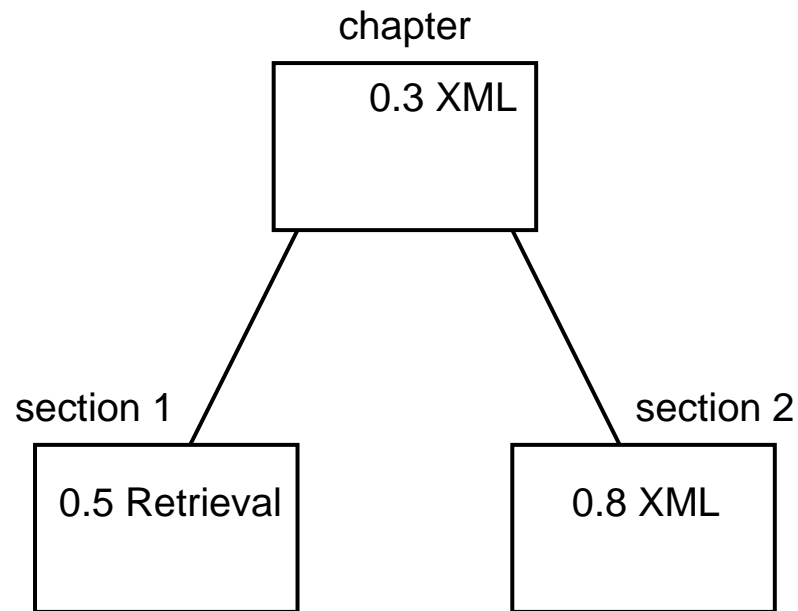
- terms contained in index object
- terms accessible from the index object



# XML Retrieval Model V: Augmentation

At retrieval time:

- index weight of most specific index objects are given directly
- combine weights for retrieval of higher level index objects

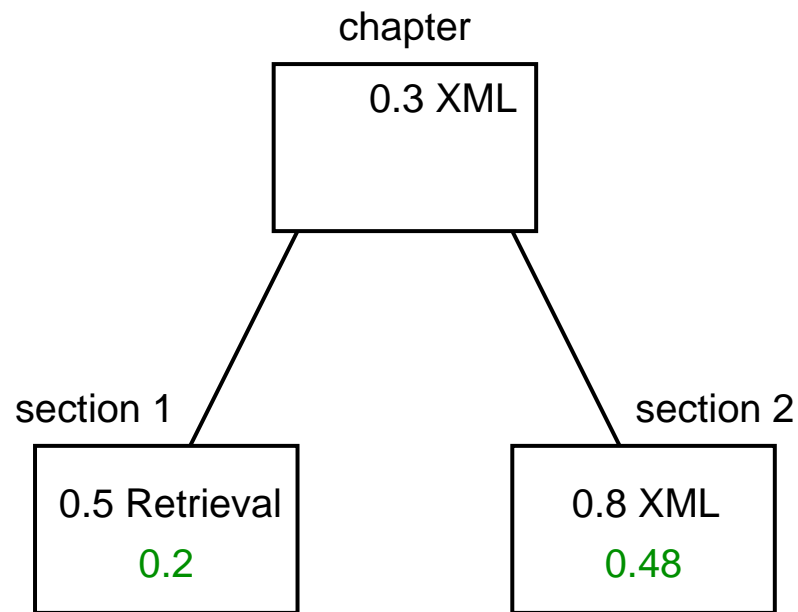


Query: 0.4 Retrieval + 0.6 XML

# XML Retrieval Model V: Augmentation

At retrieval time:

- index weight of most specific index objects are given directly
- combine weights for retrieval of higher level index objects

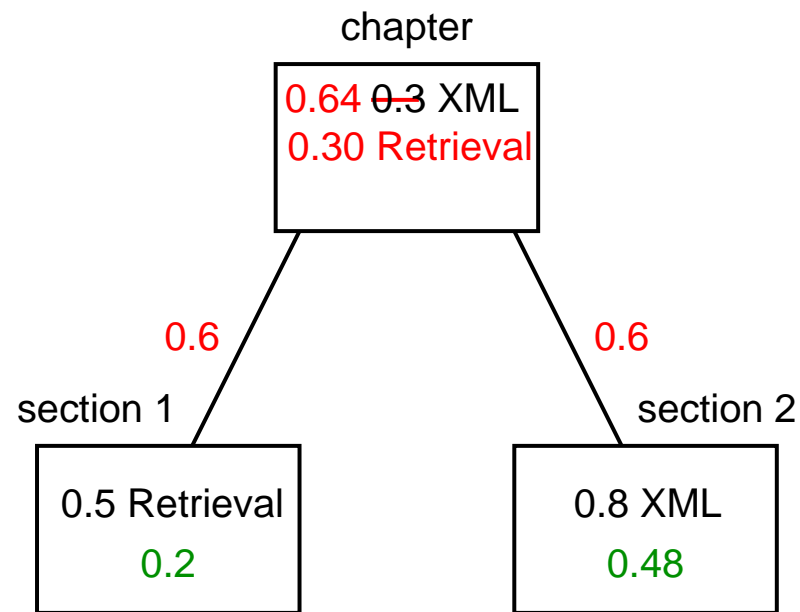


Query: 0.4 Retrieval + 0.6 XML

# XML Retrieval Model V: Augmentation

At retrieval time:

- index weight of most specific index objects are given directly
- combine weights for retrieval of higher level index objects

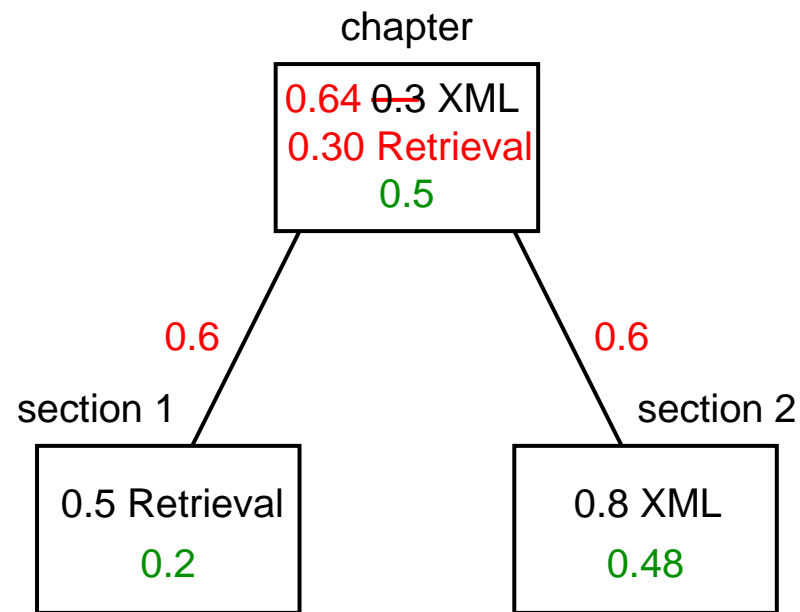


Query: 0.4 Retrieval + 0.6 XML

# XML Retrieval Model V: Augmentation

At retrieval time:

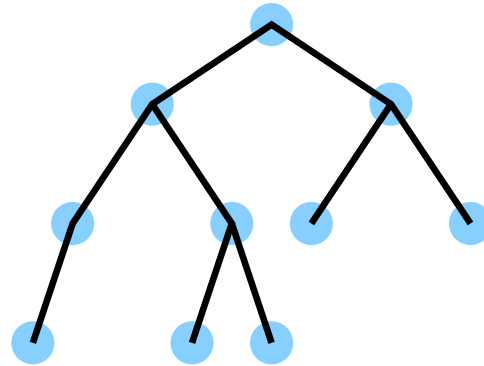
- index weight of most specific index objects are given directly
- combine weights for retrieval of higher level index objects



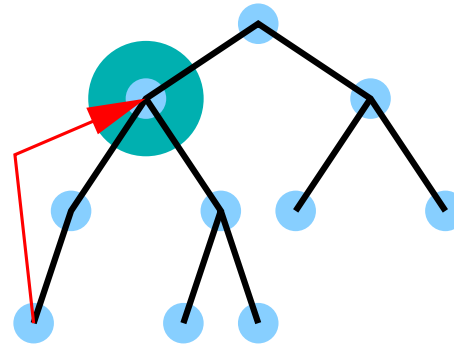
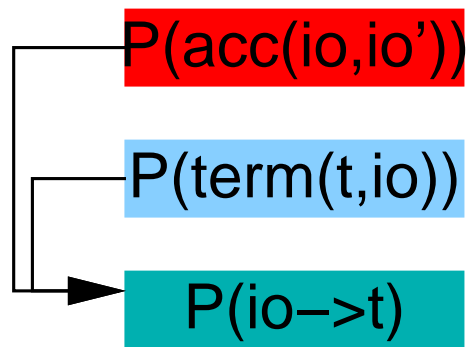
Query: 0.4 Retrieval + 0.6 XML

## XML Retrieval Model VI: Summary

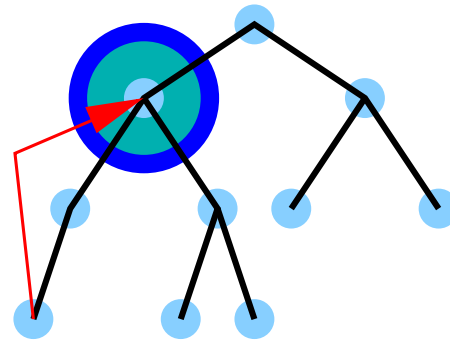
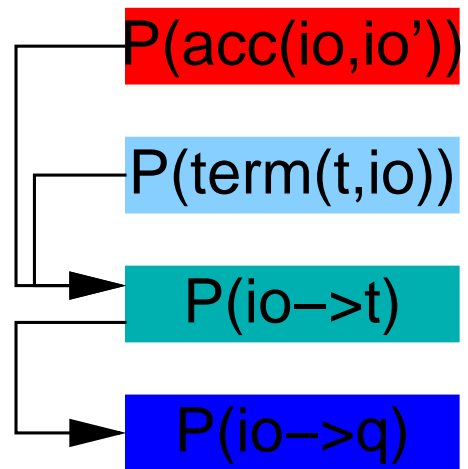
$P(\text{term}(t,io))$



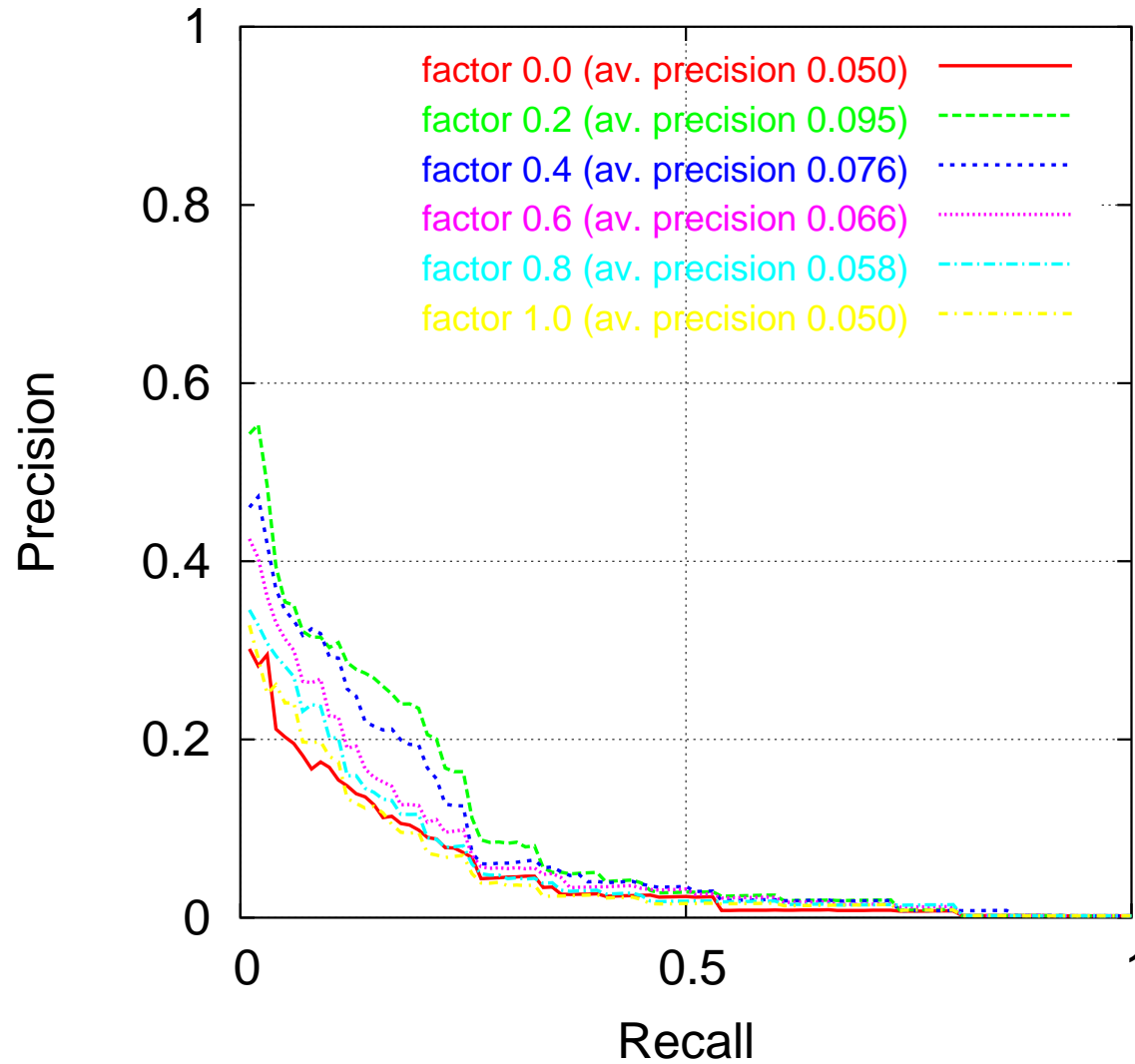
## XML Retrieval Model VI: Summary



## XML Retrieval Model VI: Summary



# XML Retrieval Model VII: Results



# Implementation: Classification of Retrieval Algorithms

*gain high quality retrieval results **fast***

- Assume linear retrieval function (as usual)
  - Assume inverted list (IL) access path
  - Entries in ILs: <index object id, weight>
- Retrieval time depends linearly on no of postings read from ILs

Approaches:

**Full computation:** use all IL entries for computation of retrieval result

**Partial computation:** restrict number of IL entries regarded at retrieval time

→ *tunable*

**Full computation, early result presentation** use all IL entries, but provide preliminary results at any time the user requests it

→ *interruptible*

## Implementation II: Tunable

restrict number of IL entries to be regarded at retrieval time:  
ignore some query terms during retrieval process

**Quit** algorithm:

- order query terms by decreasing query term weight
- only regard ILs of top-ranked query terms

**Continue** algorithm:

- like *quit* algorithm, but:
- use ILs of down-ranked query terms to refine ranking:
  - skip IL entries which have not occurred in ILs of top-ranked query terms
- IL implementation with random access facility (skip lists) needed

## Implementation III: Tunable (Results)

term ratio, savings, and average precision given in %

term ratio	quit		continue	
	saving	avg prec	saving	avg prec
100	0	9.7	0	9.7
90	29	9.8	25	9.7
80	43	10.3	40	9.8
70	57	10.2	52	9.8
60	66	10.0	61	10.0
50	73	8.7	68	10.0
40	83	8.7	78	9.7
30	88	9.0	83	9.9
20	93	8.8	88	9.6
10	98	5.9	95	7.5

## Implementation IV: Interruptible

Full computation of IL entries, but allow users to request next element of result at any time

(issue an interrupt)

**Nosferatu\*** algorithm:

- sort ILs by decreasing index weights of entries
- select next IL entry to process by decreasing impact on retrieval result  
(query term weight and index weight of IL entries)
- interrupt: return index object with highest weight so far

## Implementation V: Interruptible (Results)

- precision in %
- interrupts every 1 second

	avg prec	precision at x documents				
		5	10	20	50	100
baseline <sub>1</sub>	9.8	27.0	22.2	18.0	13.1	9.8
Nosferatu*	7.5	26.1	21.3	16.5	12.1	8.9

## Conclusions

- Concepts of accessibility and augmentation allow for *effective* XML retrieval
- Tunable and interruptible retrieval algorithms allow for *efficient* retrieval while retaining high quality results