CS3245

Information Retrieval

Lecture 9: Evaluation and XML Retrieval

9

Last Time

The VSM Reloaded

… optimized for your pleasure!

Heuristics to make search faster:

1. Don't compute what you don't need
2. Approximate things that take a lot of time

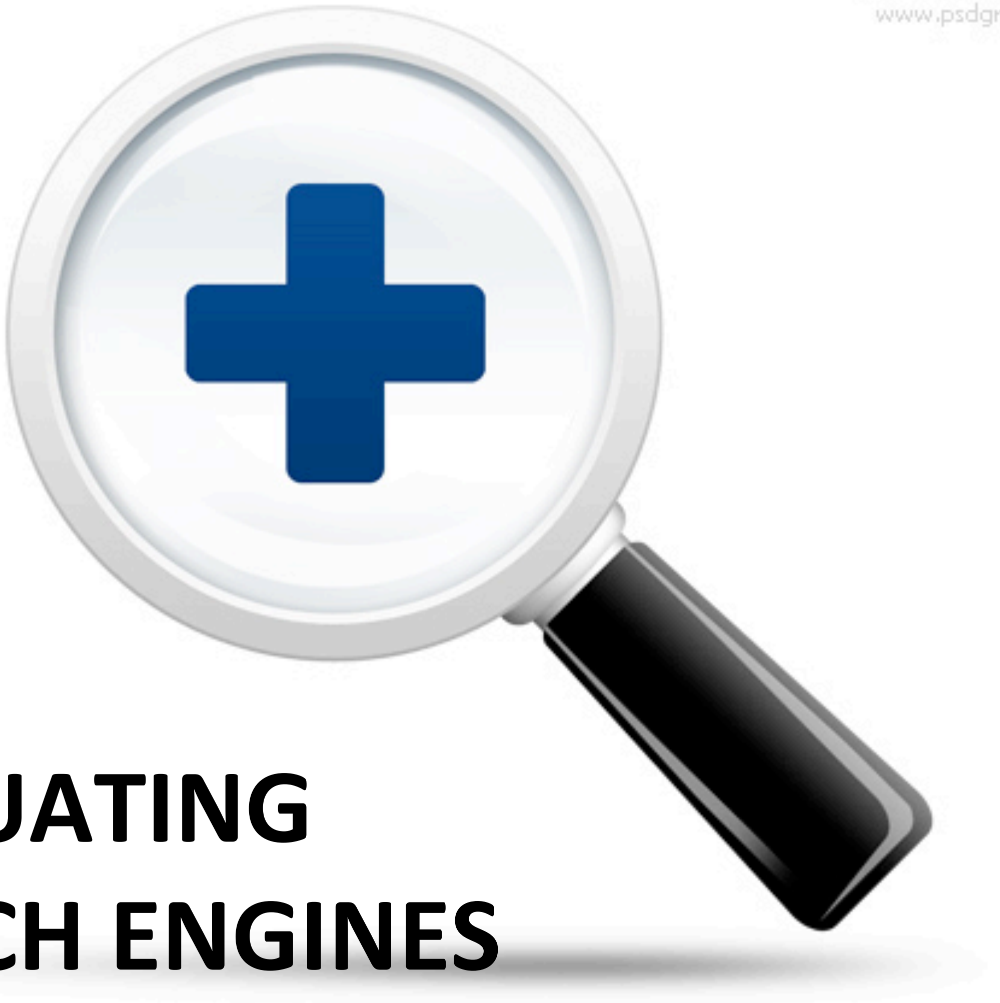
Today:

Evaluation

- How do we know if our results are any good?
  - Evaluating a search engine
    - Benchmarks
    - Precision and Recall; Composite measures
- A/B Testing (not in textbook)

XML

- Basic XML concepts
- Challenges in XML IR
- Vector space model for XML IR
- Evaluation of XML IR



EVALUATING SEARCH ENGINES

The measure of a IR engine

- How fast does it index?
  - Number of documents/hour
  - (Average document size)
- How fast does it search?
  - Latency as a function of index size
- Expressiveness of query language?
  - Ability to express complex information needs
  - Speed on complex queries
- User Interface?
- Is it free?

Measures for a search engine

- All of the preceding criteria are *measurable*: we can quantify speed/size
  - we can make expressiveness precise
- The key measure: user happiness
  - What is this?
  - Speed of response/size of index are factors
  - But blindingly fast, useless answers won't make a user happy
- Need a way of quantifying user happiness

Measuring user happiness

- Question: who is the user we are trying to make happy?
  - Answer: Depends on the setting

- <u>Web engine</u>:
  - User finds what they want and return to the engine
    - Can measure rate of return users
  - User completes their task – search as a means, not end

- <u>eCommerce site</u>: user finds what they want and buy
  - Is it the end-user, or the eCommerce site, whose happiness we measure?
  - Measure time to purchase, or fraction of searchers who become buyers?

Measuring user happiness

- Enterprise (company/govt/academic): Care about "user productivity"
  - How much time do my users save when looking for information?
  - Many other criteria having to do with breadth of access, secure access, etc.

Happiness: elusive to measure

- Most common proxy: *relevance* of search results
- But how do you measure relevance?

We'll examine one method and the issues around it

- Relevance measurement requires **3** elements:
  1. A set document collection
  2. A set suite of queries
  3. A usually binary assessment of either Relevant or Non-relevant for each query and each document
     - Some work on graded relevance, but not the standard

Evaluating an IR system

- Note: the **information need** is translated into a **query**
- Relevance is assessed relative to the **information need** *not* the **query**

- E.g., Information need: *I'm looking for information on whether drinking red wine is more effective at reducing your risk of heart attacks than white wine.*
- Query: ***wine red white heart attack effective***

i.e., we evaluate whether the doc addresses the information need, not whether it has these words

Slide courtesy Google Inc.

Why it's important: Example Think-Aloud Session

00:12 [actor most oscars]

00:10	So this is celebrity with most Oscars…
00:11	Actor… ah… most…
00:13	I'm just going to try that…most Oscars… don't know…
00:19	(reading) "News results for 'actors most Oscars' … " huh..
00:25	Oh, then that would be currently "Brokeback"… "prior voices"… "truth in Oscar's relevance"…
00:32	…now I know…
00:35	… you get a lot of weird things..hold on…
00:38	"Are Filipinos ready for gay flicks?"
00:40	How does that have to do with what I just….did...?
00:43	Ummm…
00:44	So that's where you can get surprised… you're like, where is this… how does this relate…umm…
00:45	Bond…I would think…
00:46	So I don't know, it's interesting…
01:08	**Dan**: Did you realize you were in the News section?
01:09	Oh, no I didn't. How did I get that? …
01:10	Oooh… no I didn't.

1:15 [actor most oscars Academy]

6

Unranked retrieval evaluation: Precision and Recall

- **Precision**: fraction of retrieved docs that are relevant = P(relevant|retrieved)
- **Recall**: fraction of relevant docs that are retrieved = P(retrieved|relevant)

	Relevant	Non-relevant
Retrieved	true positive	false positive
Not Retrieved	false negative	true negative

Precision $P = tp/(tp + fp)$

Recall $R = tp/(tp + fn)$

Should we use accuracy for evaluation instead?

- Given a query, a Boolean engine classifies each doc as Relevant or Non-Relevant
- The **accuracy** of an engine: the fraction of these classifications that are correct
  - (tp + tn) / (tp + fp + fn + tn)

- **Accuracy** is a commonly used evaluation measure in classification (e.g. HW1)

Quick Question: Why is this not a very useful evaluation measure in IR?

Precision/Recall

- You can get high recall (but low precision) by retrieving all docs for all queries!
- Recall is a non-decreasing function of the number of docs retrieved

- In a good system, precision decreases as either the number of docs retrieved or recall increases
  - This is not a theorem, but a result with strong empirical confirmation

Difficulties in using precision/recall

- Should average over large document collection/ query ensembles
- Need human relevance assessments
  - But people are subjective; they aren't reliable assessors
- Assessments have to be binary
  - Can we give graded assessments?
- Heavily skewed by collection/queries pairing
  - Results may not translate from one collection to another

We'll return to this point later

A combined measure: *F*

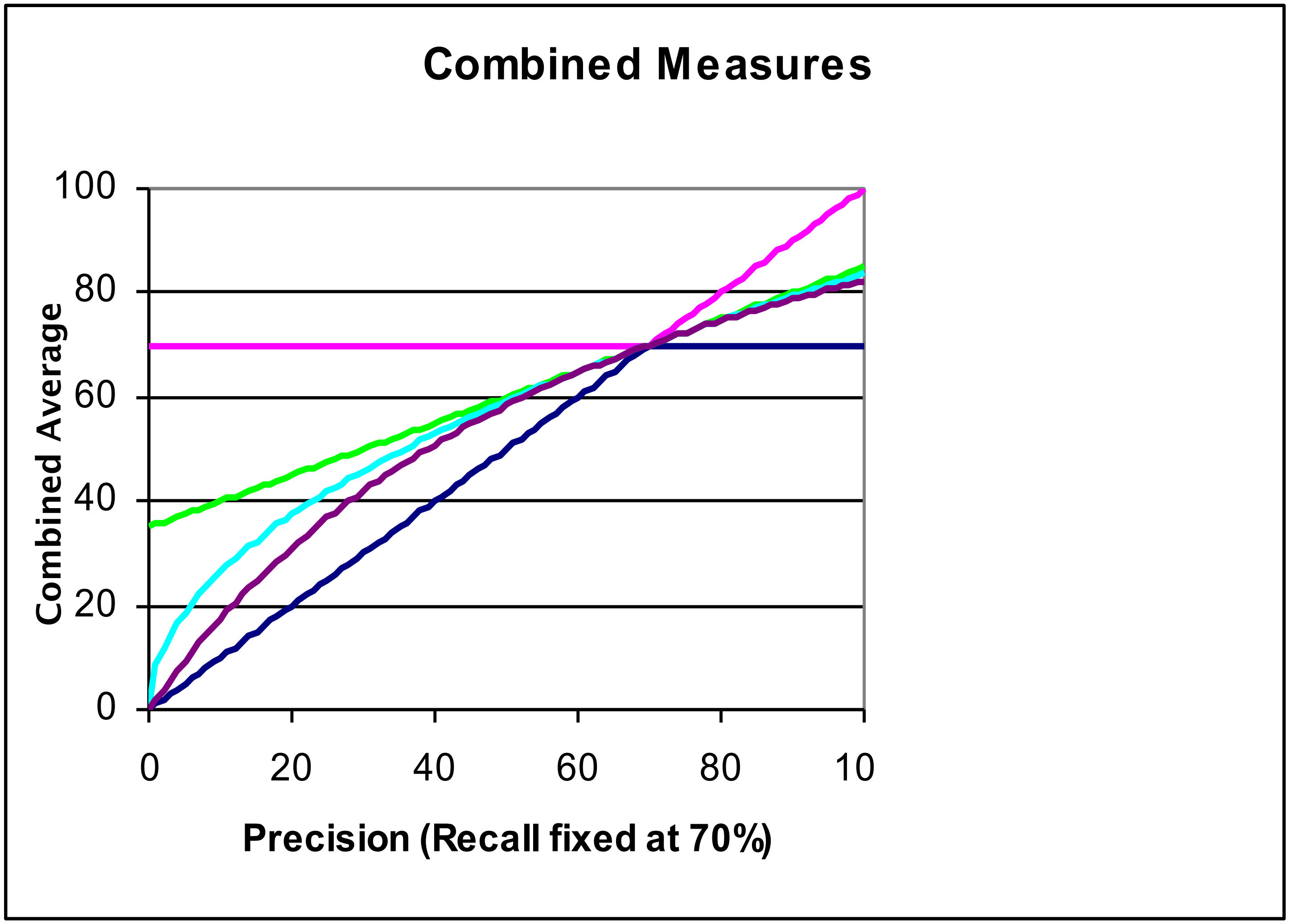
- Combined measure that assesses precision/recall tradeoff is **F measure** (weighted harmonic mean):

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

- People usually use balanced F_1 measure
  - i.e., with $\beta = 1$ or $\alpha = ½$
- Harmonic mean is a conservative average

Blanks on slides, you may want to fill in

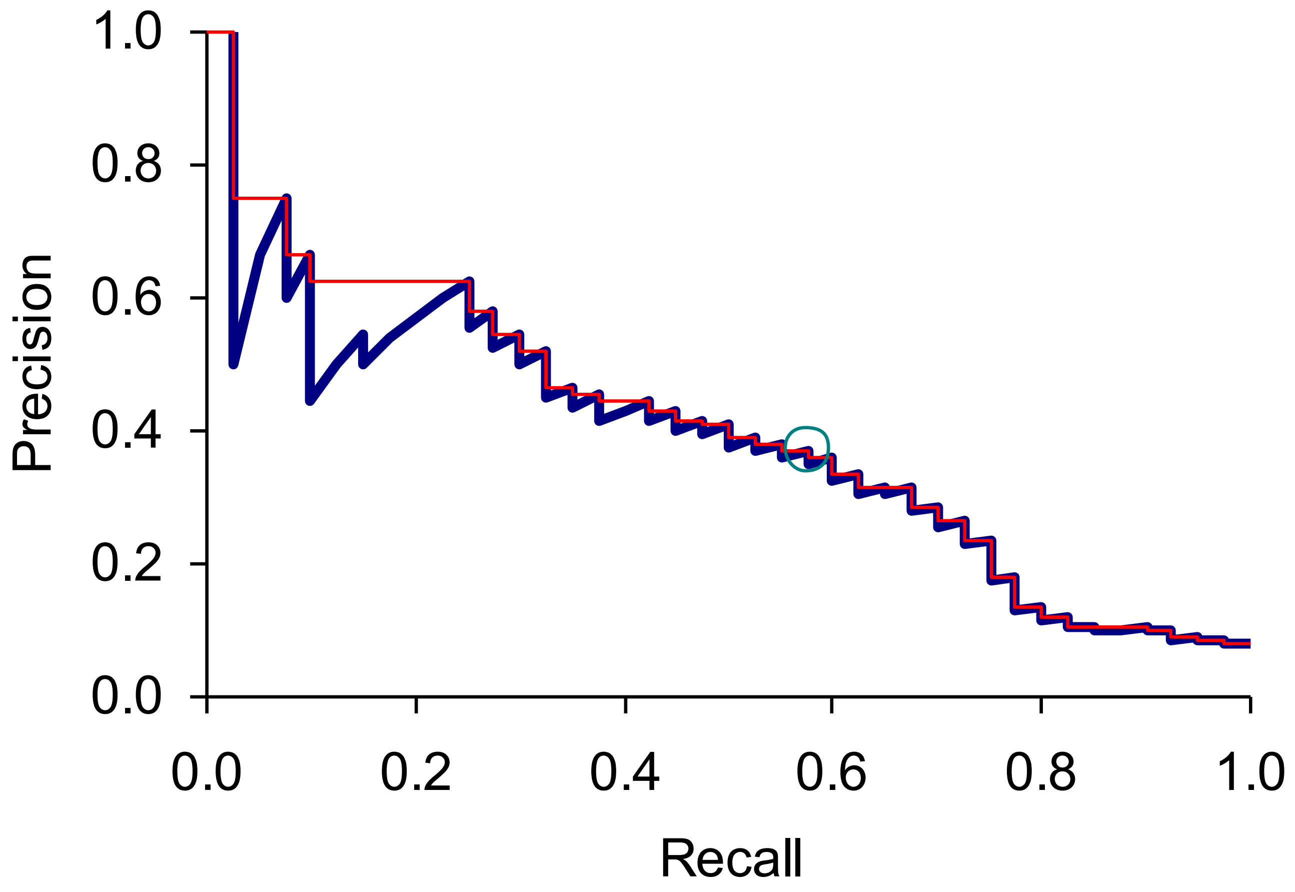
F_1 and other averages



Evaluating ranked results

- Evaluation of ranked results:
  - The system can return any number of results
  - By taking various numbers of the top returned documents (levels of recall), we can produce a *precision-recall curve*

A precision-recall curve



Averaging over queries

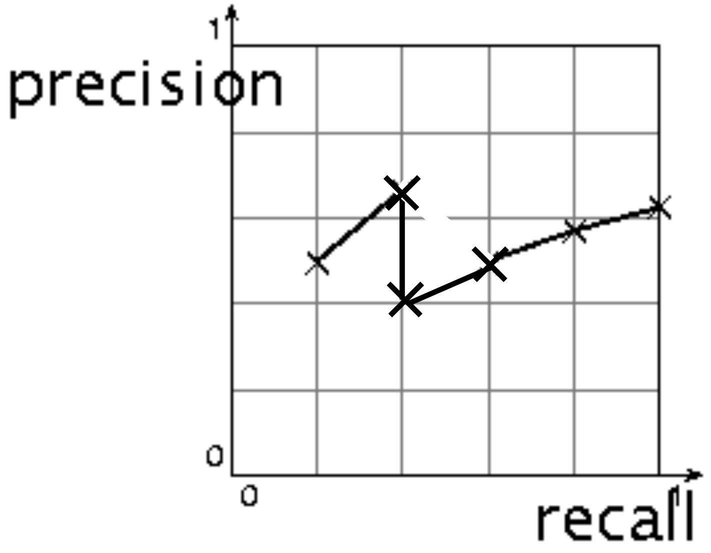
- A precision-recall graph for one query isn't a very sensible thing to look at
- Instead, average performance over a query collection.

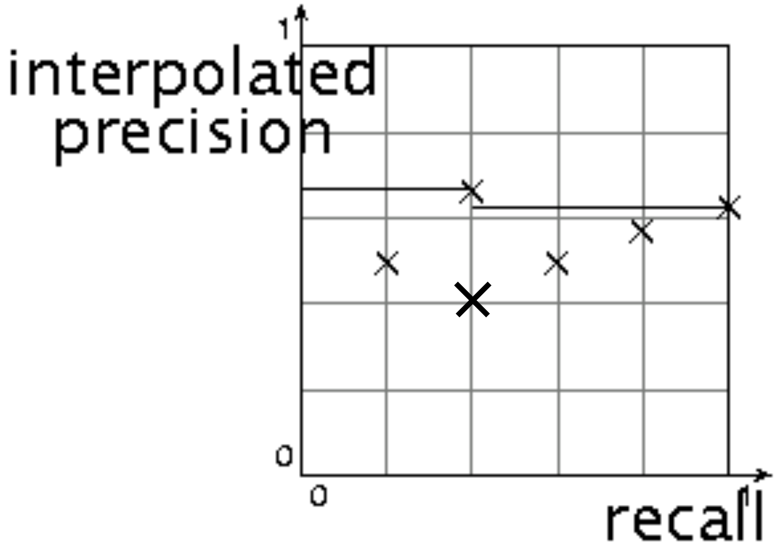
But there's a technical issue:

- Precision-recall calculations place some points on the graph
- How do you determine a value (interpolate) between the points?

Interpolated precision

- Idea: If locally precision increases with increasing recall, then you should get to count that…
- So you take the max of precisions to the right of the value





Evaluation

- Graphs are good, but often we want a summary measure!
  - Precision at fixed retrieval level
    - Precision-at-*k*: Precision of top *k* results
    - Perhaps appropriate for most of web search: all people want are good matches on the first one or two results pages
    - But: averages badly and has an arbitrary parameter of *k*
  - 11-point interpolated average precision
    - The standard measure in the early TREC competitions: you take the precision at 11 levels of recall varying from 0 to 1 by tenths of the documents, using interpolation (the value for 0 is always interpolated!), and average them
    - Evaluates performance at all recall levels

Yet more evaluation measures…

- Mean average precision (MAP)
  - Average of the precision value obtained for the top *k* documents, each time a relevant doc is retrieved
  - Avoids interpolation, use of fixed recall levels
  - MAP for query collection is arithmetic ave.
    - Macro-averaging: each query counts equally
- R-precision
  - If have known (though perhaps incomplete) set of relevant documents of size *Rel,* then calculate precision of top *Rel* docs returned
  - Perfect system could score 1.0.

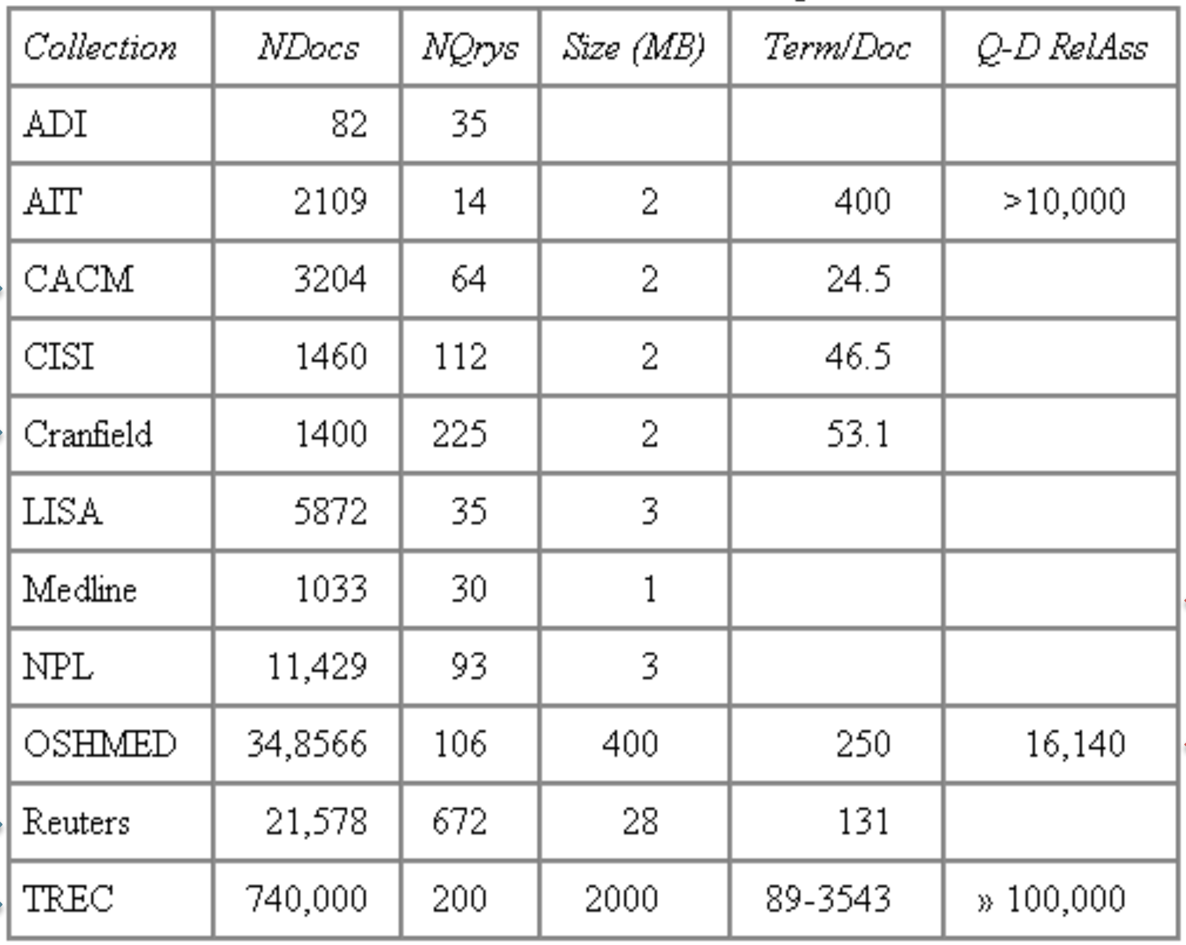
Variance

- For a test collection, it is usual that a system does poorly on some information needs (e.g., MAP = 0.1) and excellent on others (e.g., MAP = 0.7)
- Indeed, it is usually the case that the variance in performance of the same system across queries is much greater than the variance of different systems on the same query.
- That is, there are easy information needs and hard ones!

CREATING TEST COLLECTIONS FOR EVALUATION

Test Collections

TABLE 4.3 Common Test Corpora



Collection	*NDocs*	*NQrys*	*Size (MB)*	*Term/Doc*	*Q-D RelAss*
ADI	82	35			
AIT	2109	14	2	400	>10,000
CACM	3204	64	2	24.5	
CISI	1460	112	2	46.5	
Cranfield	1400	225	2	53.1	
LISA	5872	35	3		
Medline	1033	30	1		
NPL	11,429	93	3		
OSHMED	34,8566	106	400	250	16,140
Reuters	21,578	672	28	131	
TREC	740,000	200	2000	89-3543	» 100,000

Scientific papers

Scientific papers

Medical

Medical

News

News

From document collections to test collections

- Still need the other **2** things
  - Test queries
  - Relevance assessments
- Test queries
  - Must be relevant to docs available
  - Best designed by domain experts
  - Random query terms generally not a good idea
- Relevance assessments
  - Human judges, time-consuming
  - Are human panels perfect?

Kappa measure for inter-judge (dis)agreement

- Kappa measure
  - Agreement measure among judges
  - Designed for categorical judgments
  - Corrects for chance agreement

- Kappa = [P(A) – P(E)] / [1 – P(E)]
- P(A) – proportion of time judges agree
- P(E) – what agreement would be by chance

- Gives 0 for chance agreement, 1 for total agreement.

P(A)? P(E)?

Kappa Measure: Example

# of docs matching judgment type	Judge 1	Judge 2
300	Relevant	Relevant
70	Non-relevant	Non-relevant
20	Relevant	Non-relevant
10	Non-relevant	Relevant

Kappa Example

P(A) = 370/400 = 0.925

P(nonrelevant) = (10+20+70+70)/800 = 0.2125

P(relevant) = (10+20+300+300)/800 = 0.7878

P(E) = $0.2125^2 + 0.7878^2$ = 0.665

Kappa = (0.925 – 0.665)/(1-0.665) = 0.776

- Kappa > 0.8 ➔ good agreement
- 0.67 < Kappa < 0.8 ➔ "tentative conclusions"

- Depends on purpose of study
- For >2 judges: average pairwise kappas

TREC

- TREC's Ad Hoc task from first 8 TRECs was the standard IR task
  - 50 detailed information needs a year
  - Human evaluation of **pooled** results returned
  - More recently other related things: Web, Hard, QA, interactive track

NUS' strength in recent years

- A query from TREC 5 (1996)

```
<top>
<num>225</num>
<desc>What is the main function of the Federal
  Emergency Management Agency (FEMA) and the
  funding level provided to meet emergencies?
  Also, what resources are available to FEMA such
  as people, equipment, facilities?</desc>
</top>
```

Interjudge Agreement: TREC 3

information need	number of docs judged	disagreements	NR	R
51	211	6	4	2
62	400	157	149	8
67	400	68	37	31
95	400	110	108	2
127	400	106	12	94

Shows that there are queries that are easier than others

Critique of pure relevance

- Relevance versus **Marginal Relevance**
  - A document can be redundant even if it is highly relevant
  - Duplicates
  - The same information from different sources
  - Marginal relevance is a better measure of utility for the user.
- Using facts/entities as evaluation units more directly measures true relevance.
- But often harder to create evaluation set

Can we avoid human judgment?

Unfortunately, no

- Makes experimental work hard
  - Especially on a large scale
  - Can be tedious, expensive to calculate
  - Recently, use crowdsourcing methods to collect data
- In some very specific settings, can use proxies
  - E.g.: for approximate vector space retrieval, we can compare the cosine distance closeness of the closest docs to those found by an approximate retrieval algorithm
- But once we have test collections, we can reuse them

Evaluation at large search engines

- Search engines have test collections of queries and hand-ranked results
- Recall is difficult to measure on the web
- Search engines often use precision at top *k* (e.g., *k* = 10)
- . . . or measures that reward you more for getting rank 1 right than for getting rank 10 right.
  - NDCG (Normalized Cumulative Discounted Gain)
  - MRR (Mean Reciprocal Rank)
- Search engines also use non-relevance-based measures.
  - Clickthrough on first result
    - Not very reliable if you look at a single clickthrough … but pretty reliable in the aggregate.
  - Studies of user behavior in the lab
  - A/B testing

A/B testing

Purpose: Test a single innovation

Prerequisite: You have a large search engine up and running.

- Have most users use old system
- Divert a small proportion of traffic (e.g., 1%) to the new system that includes the innovation
- Evaluate with an "automatic" overall evaluation criterion (OEC) like clickthrough on first result
- Now we can directly see if the innovation does improve user happiness.
- Probably the evaluation methodology that large search engines trust most
- In principle less powerful than doing a multivariate regression analysis, but easier to understand

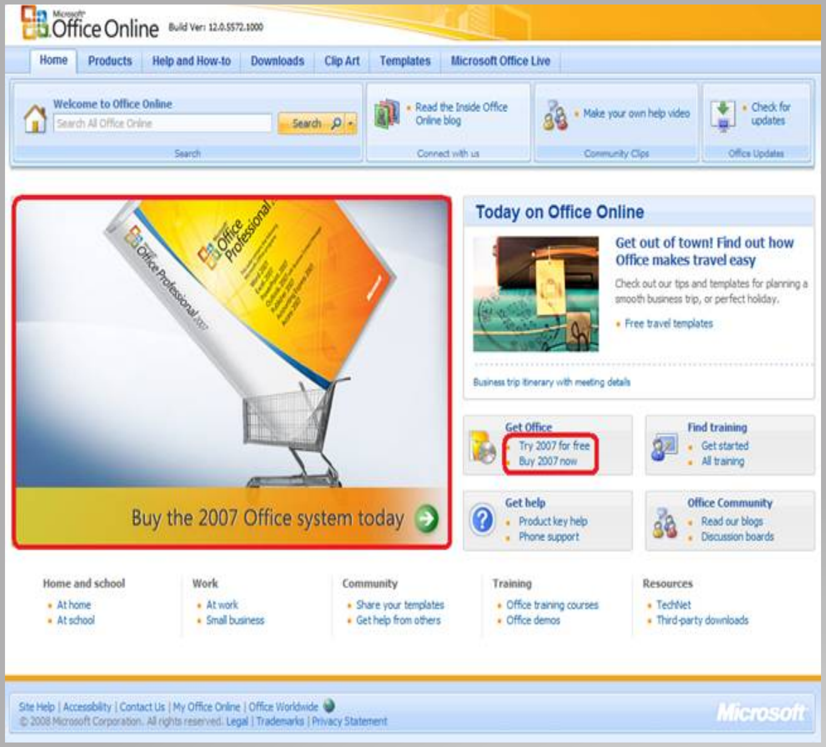
Slide courtesy Microsoft Inc.

Office Online

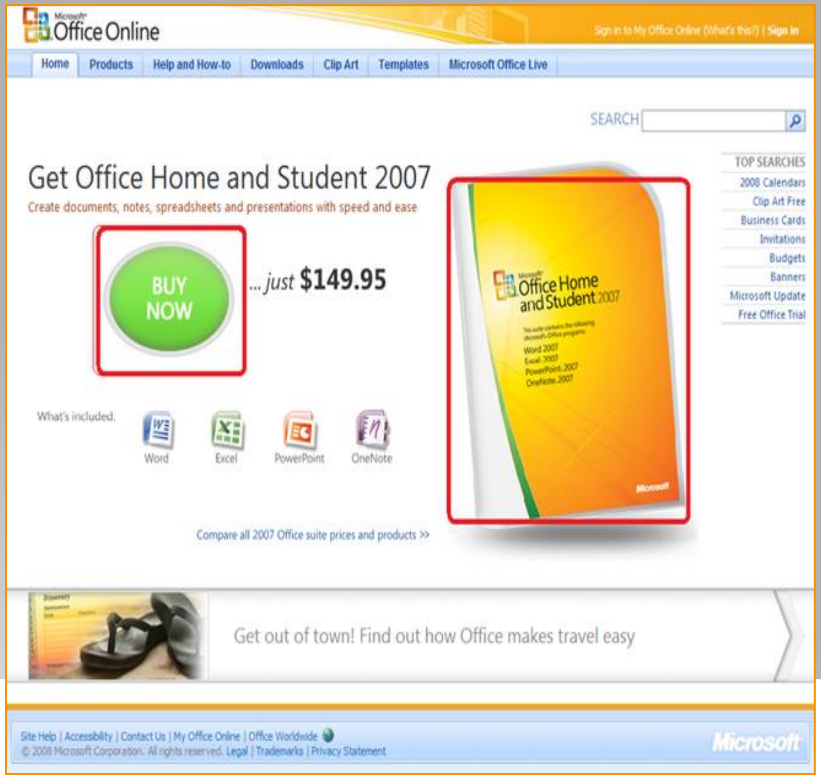
Test new design for Office Online homepage

OEC: Clicks on revenue generating links (red below)

A



B



Is A better, B better, or are they about the same?

Slide courtesy Microsoft Inc.

Office Online

- B was 64% worse
- The Office Online team wrote

  A/B testing is a fundamental and critical Web services… consistent use of A/B testing could save the company millions of dollars

Slide courtesy Microsoft Inc.

The HiPPO

The less data, the stronger the opinions

- Our opinions are often wrong – get the data
- HiPPO stands for the Highest Paid Person's Opinion
- Hippos kill more humans than any other (non-human) mammal (really)
- Don't let HiPPOs in your org kill innovative ideas. ExPeriment!
- We give out these toy HiPPOs at Microsoft

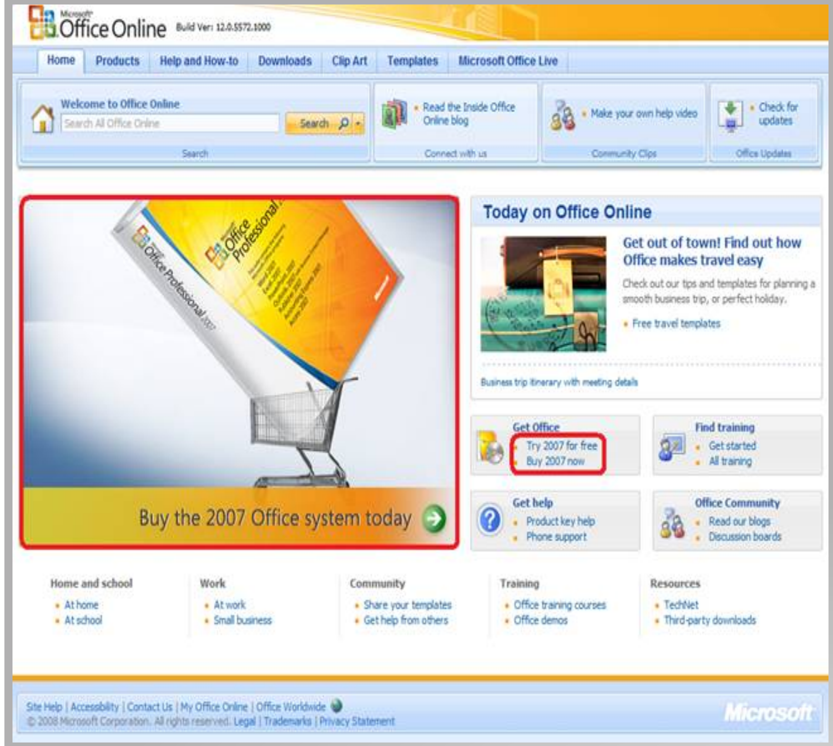
Slide courtesy Microsoft Inc.

Pitfall 1: Wrong Success Metric

Remember this example?

OEC: Clicks on revenue generating links (red below)

A


Office Online Build Ver: 12.0.5572.1000

Home | Products | Help and How-to | Downloads | Clip Art | Templates | Microsoft Office Live

Welcome to Office Online — Search All Office Online — Search

Read the Inside Office Online blog — Connect with us

Make your own help video — Community Clips

Check for updates — Office Updates

Buy the 2007 Office system today

Today on Office Online

Get out of town! Find out how Office makes travel easy

Check out our tips and templates for planning a smooth business trip, or perfect holiday.

- Free travel templates

Business trip itinerary with meeting details

Get Office
- Try 2007 for free
- Buy 2007 now

Find training
- Get started
- All training

Get help
- Product key help
- Phone support

Office Community
- Read our blogs
- Discussion boards

Home and school
- At home
- At school

Work
- At work
- Small business

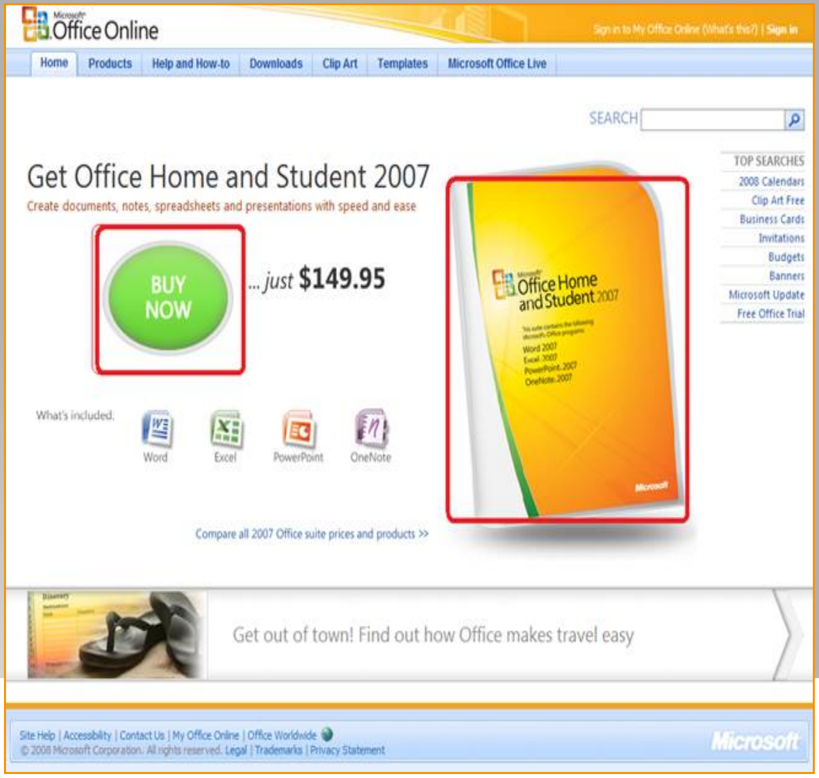
Community
- Share your templates
- Get help from others

Training
- Office training courses
- Office demos

Resources
- TechNet
- Third-party downloads

Site Help | Accessibility | Contact Us | My Office Online | Office Worldwide
© 2008 Microsoft Corporation. All rights reserved. Legal | Trademarks | Privacy Statement

B


Office Online — Sign in to My Office Online (What's this?) | Sign in

Home | Products | Help and How-to | Downloads | Clip Art | Templates | Microsoft Office Live

SEARCH

Get Office Home and Student 2007

Create documents, notes, spreadsheets and presentations with speed and ease

BUY NOW ... just $149.95

What's included: Word, Excel, PowerPoint, OneNote

Compare all 2007 Office suite prices and products >>

TOP SEARCHES
- 2008 Calendars
- Clip Art Free
- Business Cards
- Invitations
- Budgets
- Banners
- Microsoft Update
- Free Office Trial

Get out of town! Find out how Office makes travel easy

Site Help | Accessibility | Contact Us | My Office Online | Office Worldwide
© 2008 Microsoft Corporation. All rights reserved. Legal | Trademarks | Privacy Statement

Slide courtesy Microsoft Inc.

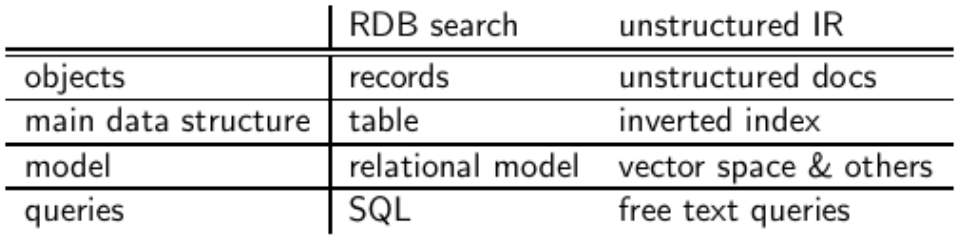
Pitfall: Wrong Overall Evaluation Criterion (OEC)

- B had a drop in the OEC of 64%
- Were sales correspondingly less also?
- No. The experiment is valid if the conversion from a click to purchase is similar
- The price was shown only in B, sending more qualified purchasers to the pipeline
- Lesson: measure what you really need to measure, even if it’s difficult!

XML RETRIEVAL

IR and relational databases

- IR systems often contrasted with relational databases (RDB).
- Traditionally, IR systems retrieve information from *unstructured text* ("raw" text without markup).
- RDB systems are used for querying *relational data*: sets of records that have values for predefined attributes such as employee number, title and salary.



	RDB search	unstructured IR
objects	records	unstructured docs
main data structure	table	inverted index
model	relational model	vector space & others
queries	SQL	free text queries

- Some structured data sources containing text are best modeled as structured documents rather than relational data (Structured retrieval).

Structured retrieval

Premise: queries are structured or unstructured; documents are structured.

Applications of structured retrieval

Digital libraries, patent databases, blogs, tagged text with entities like persons and locations (named entity tagging)

Example

- Digital libraries: *give me a full-length article on fast fourier transforms*
- Patents: *give me patens 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*

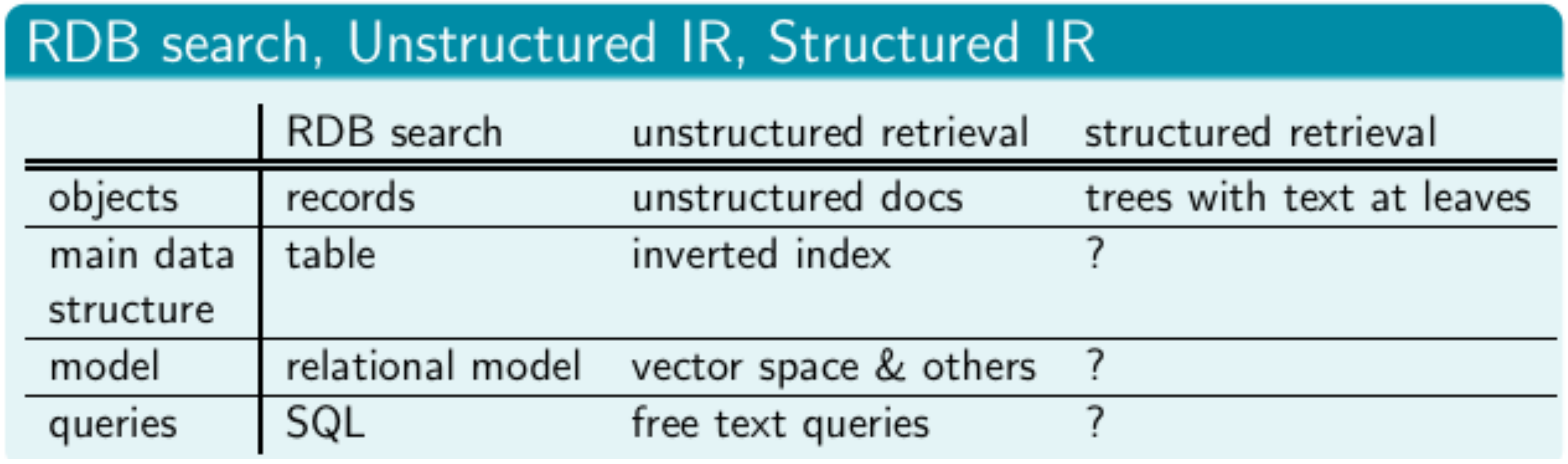
Why RDB is not suitable in this case

Three main problems

1. An unranked system (like a DB) can return a large set leading to information overload
2. Users often don't precisely state structural constraints – may not know possible structure elements are supported
   - *tours* AND (COUNTRY: *Vatican* OR LANDMARK: *Coliseum*)?
   - *tours* AND (STATE: *Vatican* OR BUILDING: *Coliseum*)?
3. Users may be unfamiliar with structured search and the necessary advanced search interfaces or syntax

Solution: adapt ranked retrieval to structured documents

Structured Retrieval



RDB search, Unstructured IR, Structured IR

	RDB search	unstructured retrieval	structured retrieval
objects	records	unstructured docs	trees with text at leaves
main data structure	table	inverted index	?
model	relational model	vector space & others	?
queries	SQL	free text queries	?

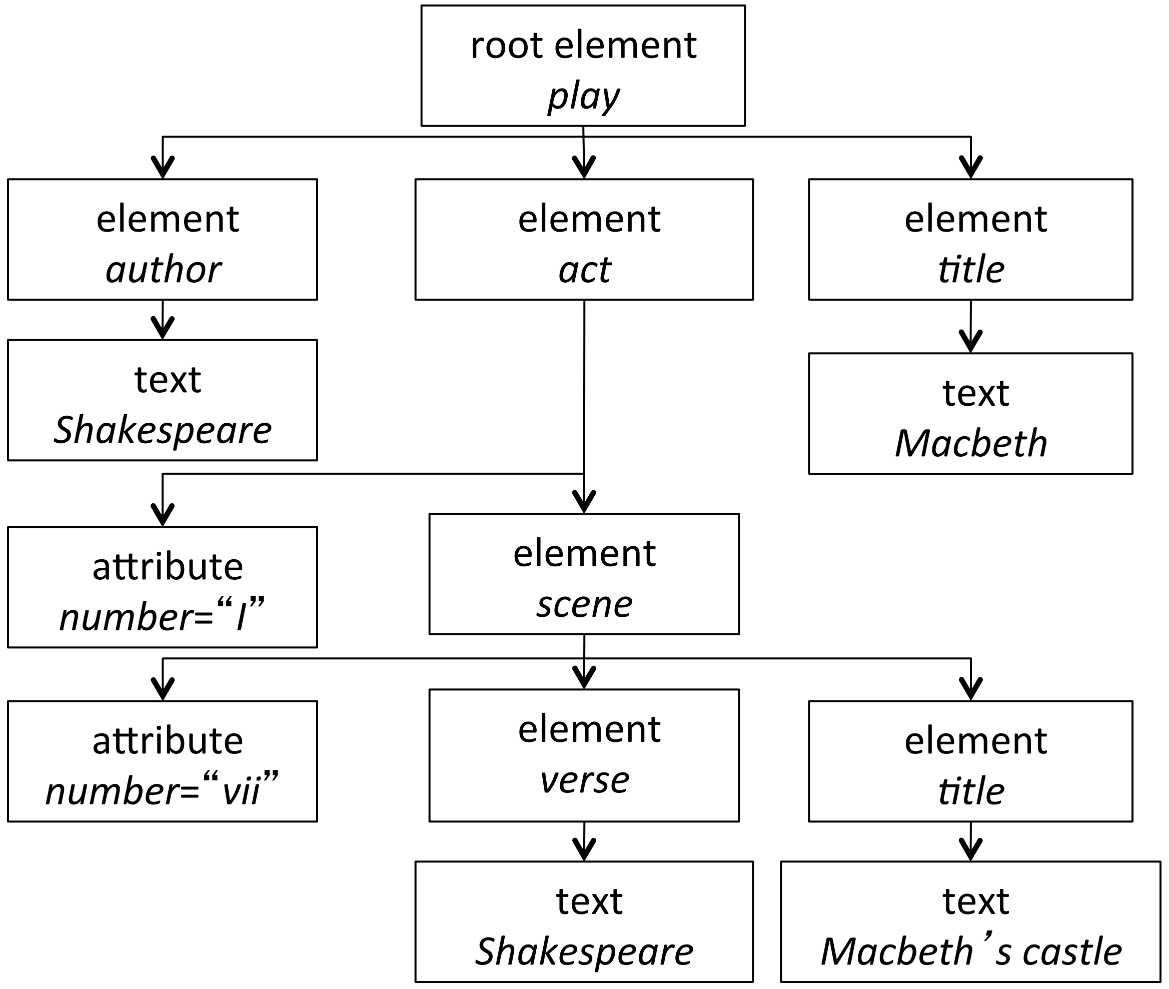
- Standard for encoding structured documents: Extensible Markup Language (XML)
  - structured IR ➔ XML IR
  - also applicable to other types of markup (HTML, SGML, …)

XML Documents

- 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)

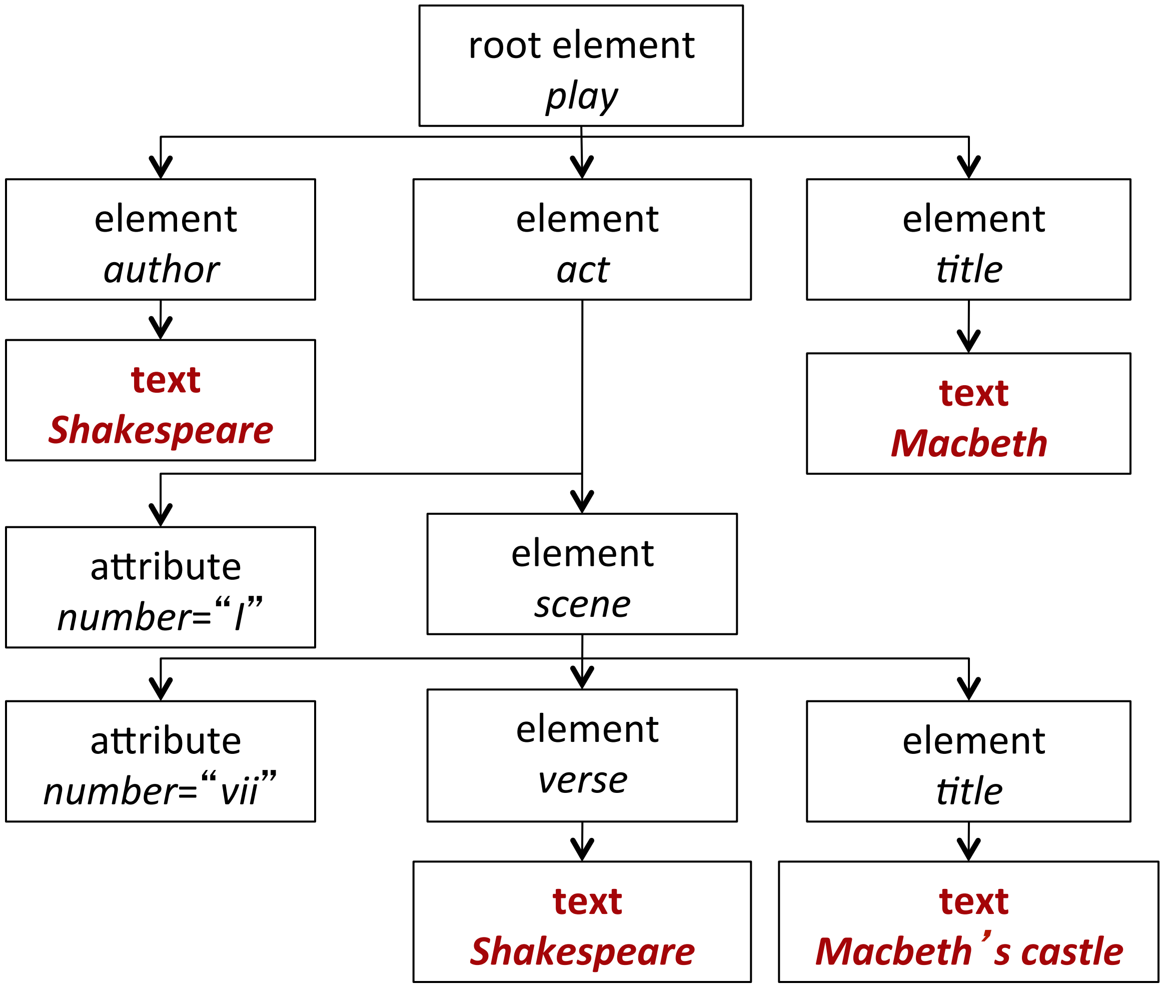
```
<play>
<author>Shakespeare</author>
<title>Macbeth</title>
<act number="I">
<scene  number=""vii">
<title>Macbeth's castle</title>
<verse>Will I with wine
…</verse>
</scene>
</act>
</play>
```

XML Document

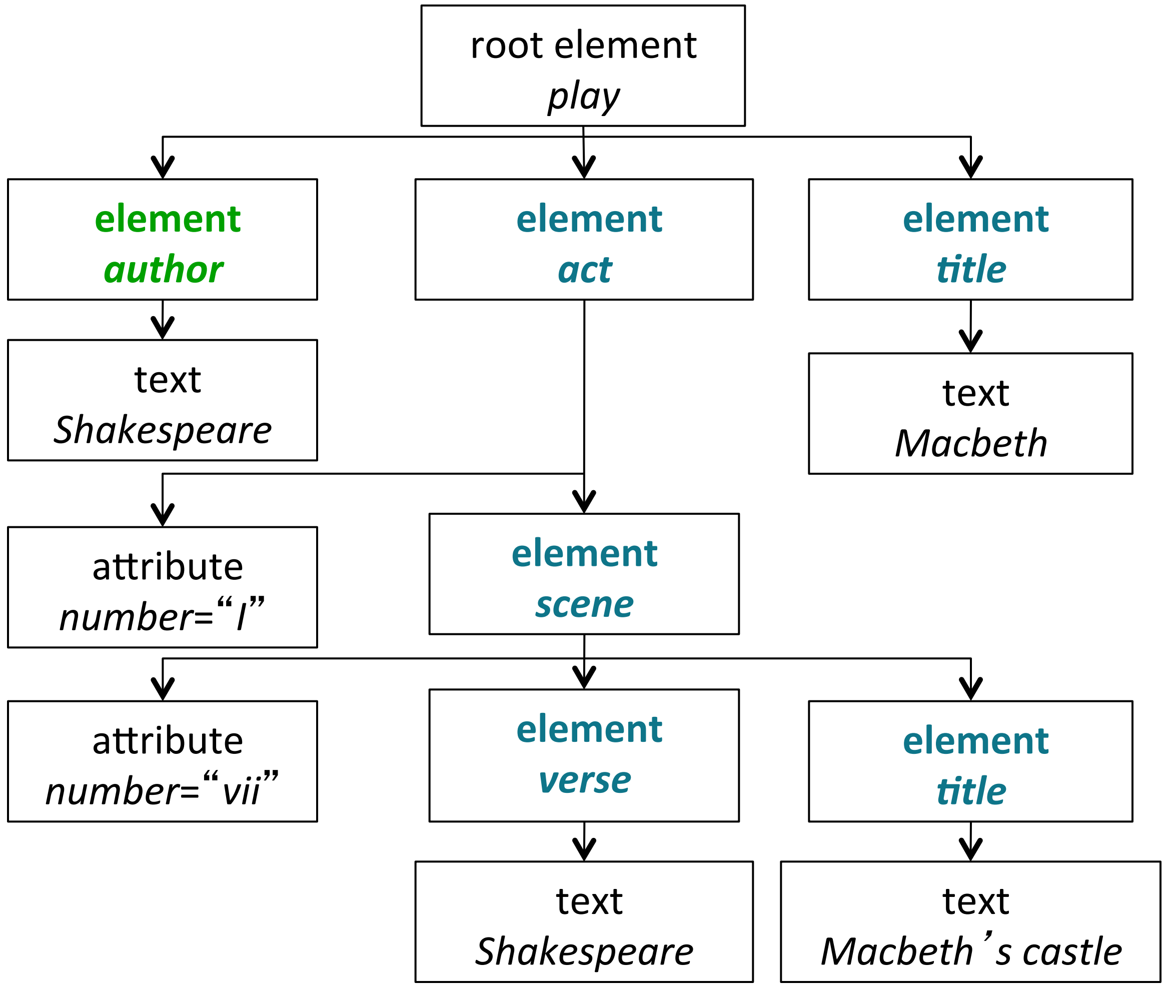


XML Document

The **leaf nodes** consist of text



XML Document



The internal nodes encode **document structure** or **metadata** functions

CHALLENGES IN XML RETRIEVAL

First challenge: Document parts to retrieve

- Structured or XML retrieval: users want parts of documents (i.e., XML elements), not the entire thing.

Example

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

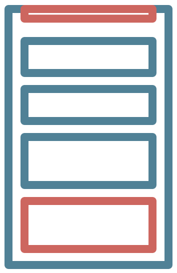
  - In this case, the user is probably looking for the scene.
  - However, an otherwise unspecified search for *Macbeth* should return the play of this name, not a subunit.
- Solution: structured document retrieval principle

Structured document retrieval principle

Structured document retrieval principle

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.
- 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*.
  - In this case, the title (higher node) is preferred.
  - Difficult to decide which level of the tree best satisfies the query.

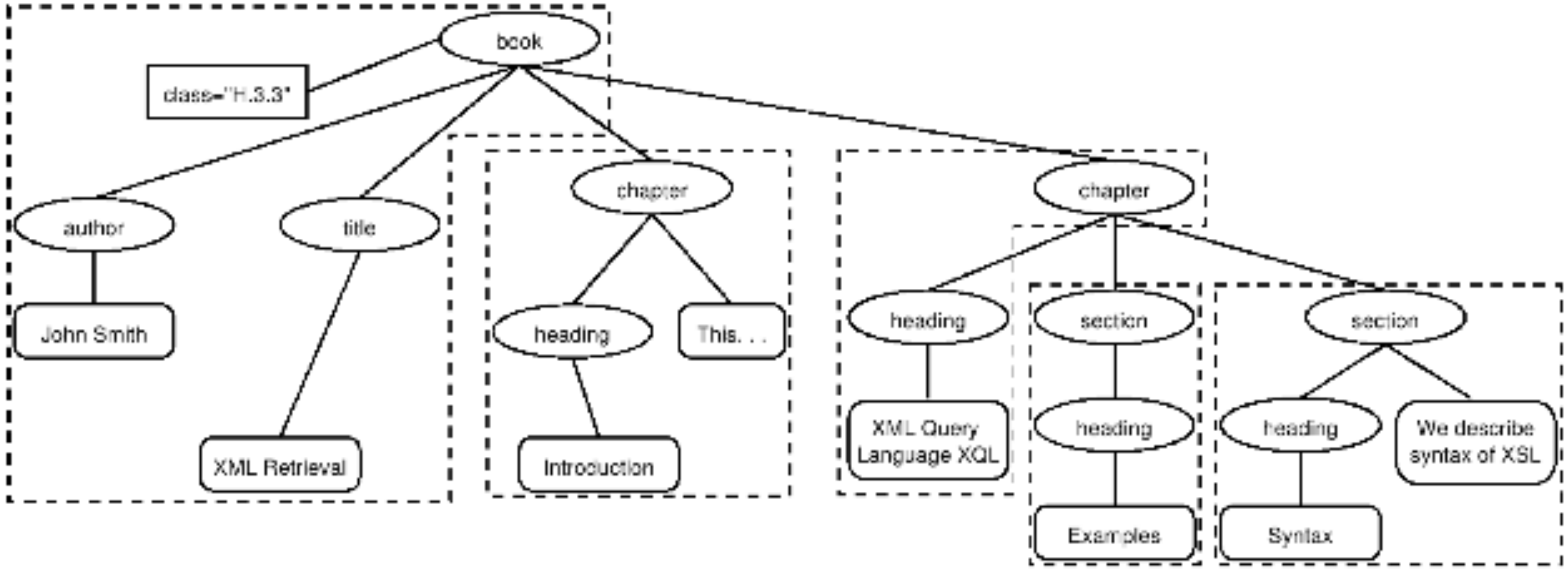


Second challenge: Document part to index

- 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, etc.
    - In structured retrieval, there are four main different approaches to defining the indexing unit:
        1. Non-overlapping pseudo-documents
        2. Top down
        3. Bottom up
        4. All units

1) Non-overlapping pseudodocuments

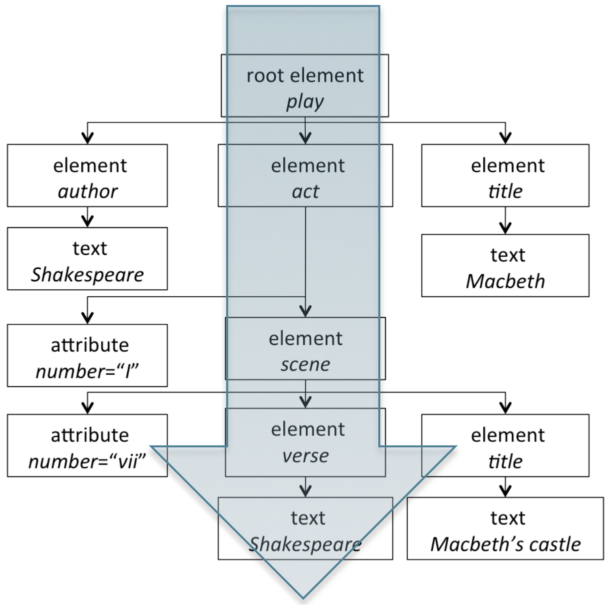
Group nodes into non-overlapping subtrees



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

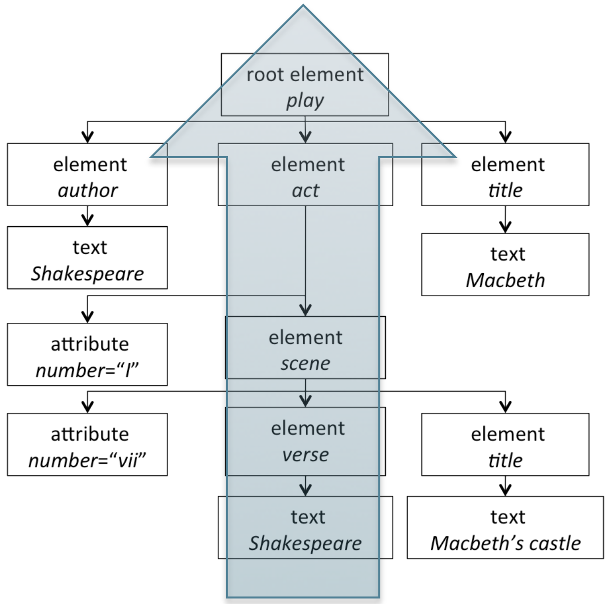
2) Top down

- A 2-stage process:
  1. Start with one of the largest elements as the indexing unit, e.g. the **<book>** element in a collection of books
  2. Then postprocess search results to find for each book the subelement that is the best hit.
- This two-stage process often fails to return the best subelement
  - the relevance of a whole book is often not a good predictor of the relevance of subelements within it.



3) Bottom Up

- We can search all leaves, select the most relevant ones and then extend them to larger units in postprocessing (bottom up).
- 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.



4) Index all elements

- The least restrictive approach, but also problematic:
- Many XML elements are not meaningful search results, e.g., an ISBN number, bolded text
- Indexing all elements means that search results will be highly redundant.

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.

- **Nested elements** are elements that are contained within each other. Returning redundant nested elements is not very user-friendly.

Third challenge: Nested elements

Due to the redundancy of nested elements, it is common to restrict the set of elements eligible for retrieval.

Restriction strategies include:

- discard all small elements
- discard all elements that **users** do not look at (from examining retrieval system logs)
- discard all elements that **assessors** generally do not judge to be relevant (when relevance assessments are available)
- only keep elements that a **system designer or librarian** has deemed to be useful

In most of these approaches, result sets will still contain nested elements.

Third challenge: Nested elements

Further techniques:

- 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.

Highlighting

- 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.

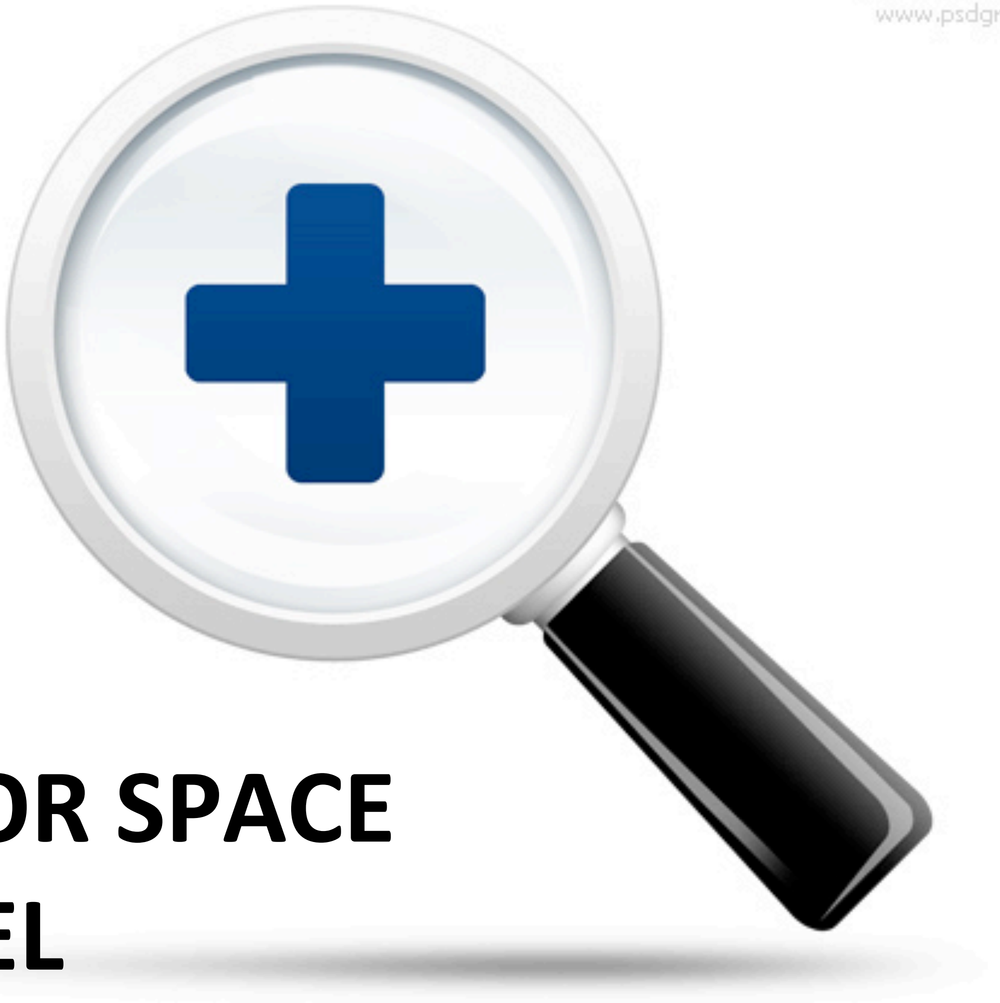
Nested elements and term statistics

- Further challenge related to nesting: 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.

- 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.

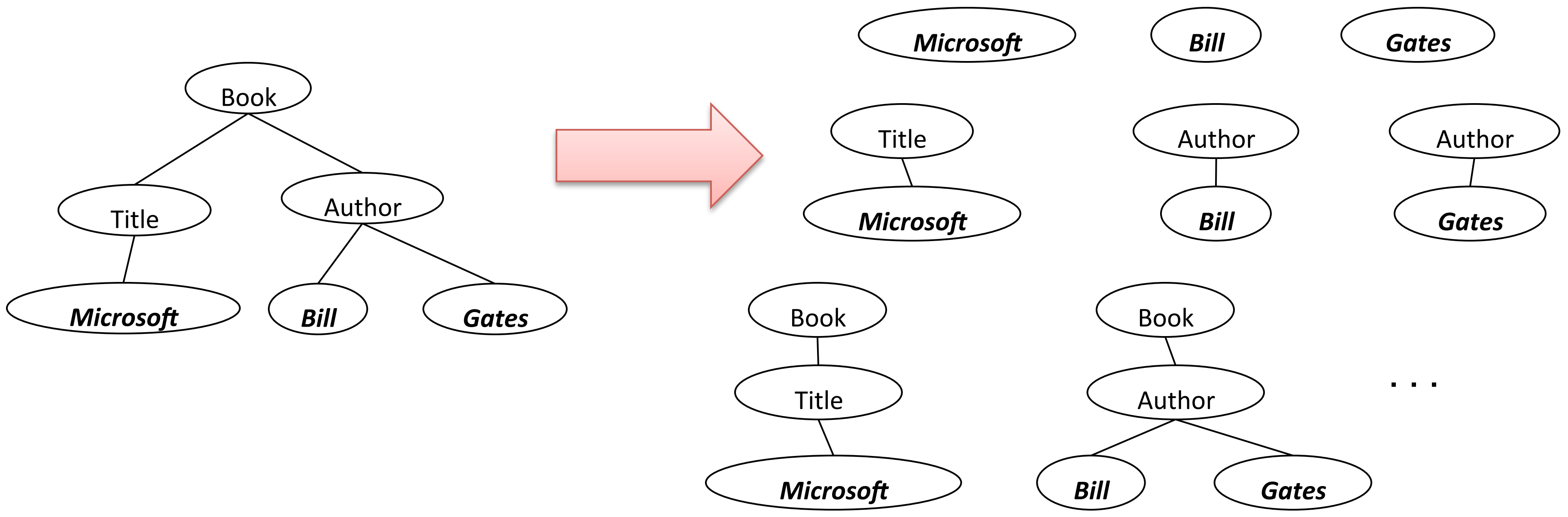


VECTOR SPACE MODEL FOR XML IR

Main idea: lexicalized subtrees

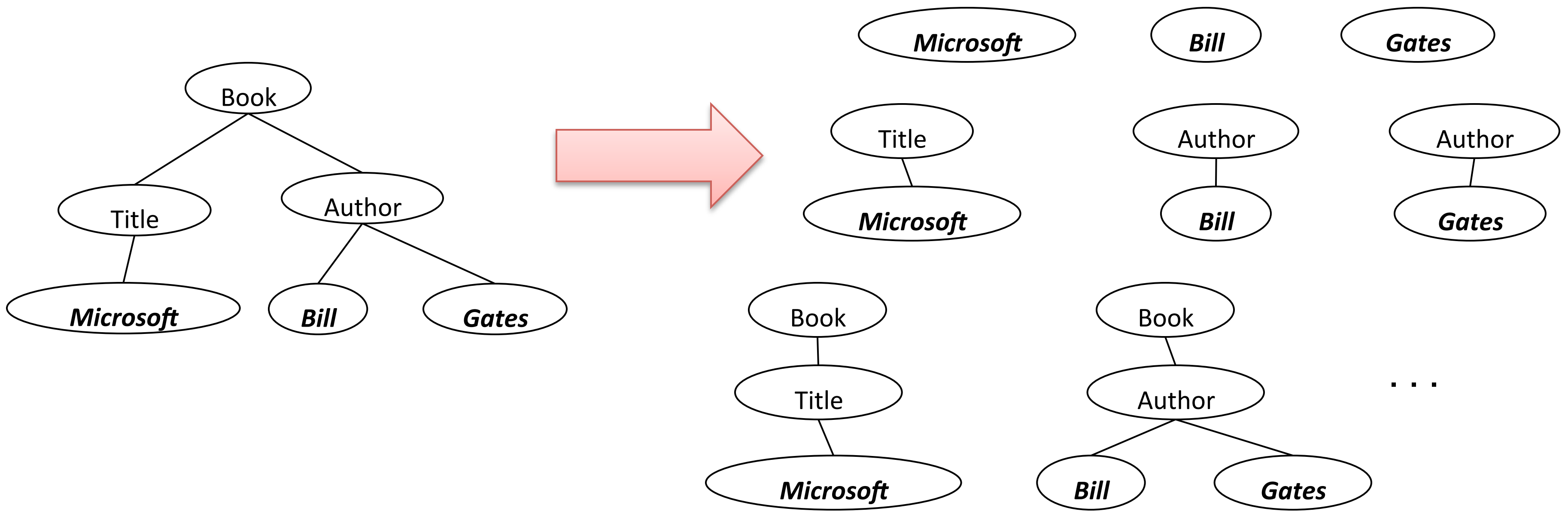
- Aim: to have each dimension of the vector space encode a word together with its position within the XML tree.
- How: Map XML documents to lexicalized subtrees.

“With words”



Creating 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.



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.

Structural term

- There is a tradeoff between the dimensionality of the space and the accuracy of query results.

Feast or Famine

  - If we restrict dimensions to vocabulary terms, then the VSM retrieval system 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 in the collection, the dimensionality becomes too large.
- Compromise: index all paths that end in a single vocabulary term (i.e., all XML-context term pairs). We call such an XML-context term pair a structural term and denote it by **<c, t>**: a pair of XML-context **c** and vocabulary term **t**.

Context resemblance

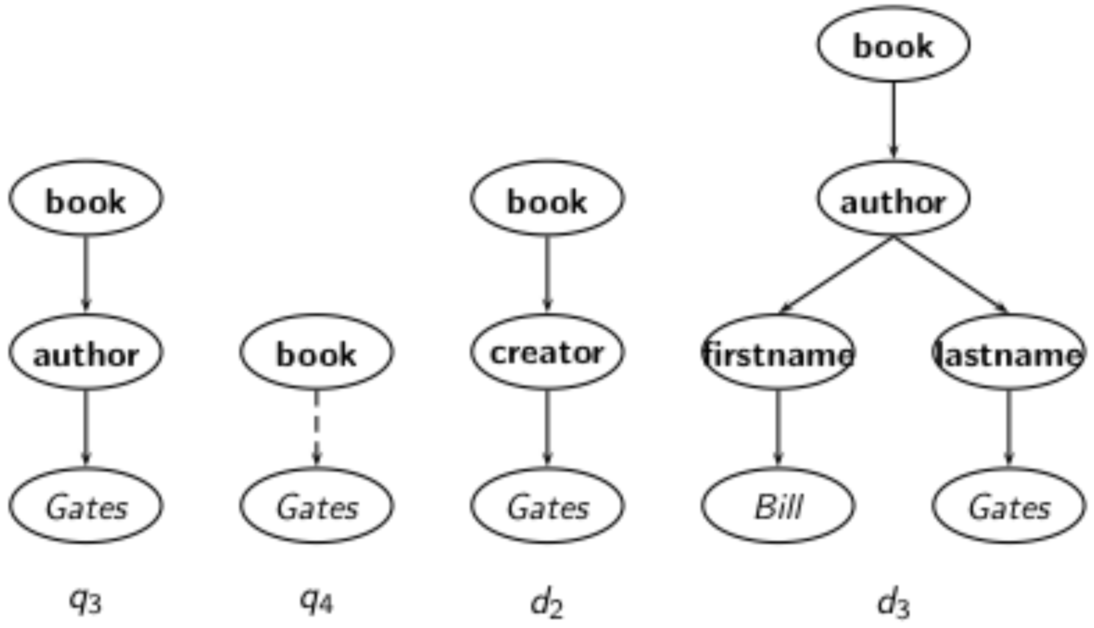
- A simple measure of the similarity of a path c_q in a query and a path c_q in a document is the following ***context resemblance*** function CR:

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

  $|c_q|$ and $|c_d|$ are the number of nodes in the query path and document path, respectively

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

Context resemblance example

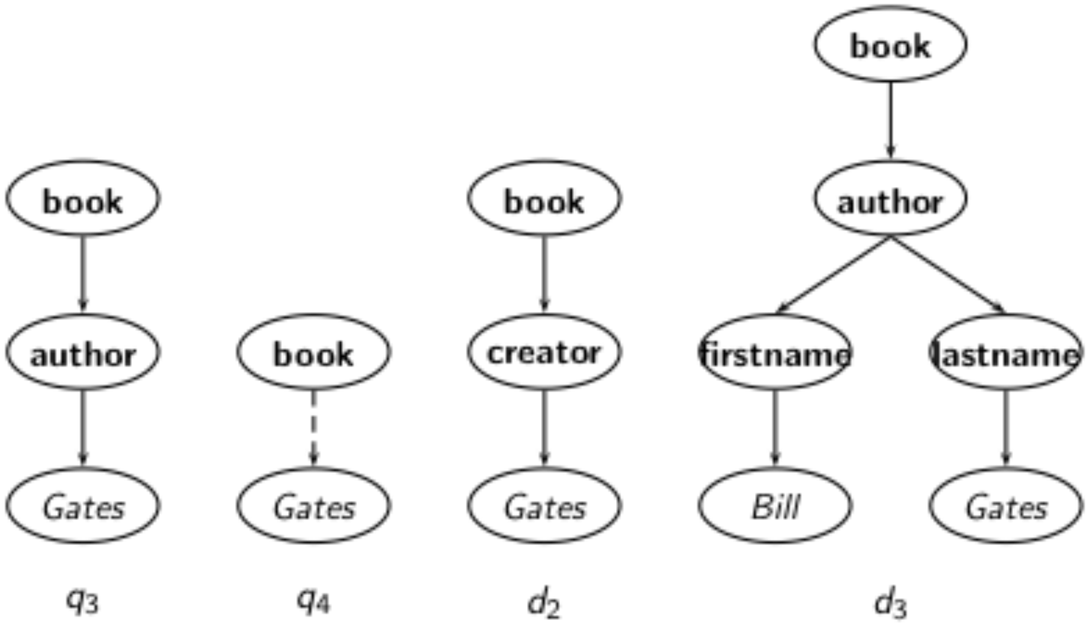


$$\mathrm{CR}(c_q, c_d) = \begin{cases} \frac{1+|c_q|}{1+|c_d|} & \text{if } c_q \text{ matches } c_d \\ 0 & \text{if } c_q \text{ does not match } c_d \end{cases}$$

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

Blanks on slides, you may want to fill in

Context resemblance example



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

- CR($c_{q?}$, $c_{d?}$) = CR(c_q, c_d) = 3/5 = 0.6.

Document similarity measure

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

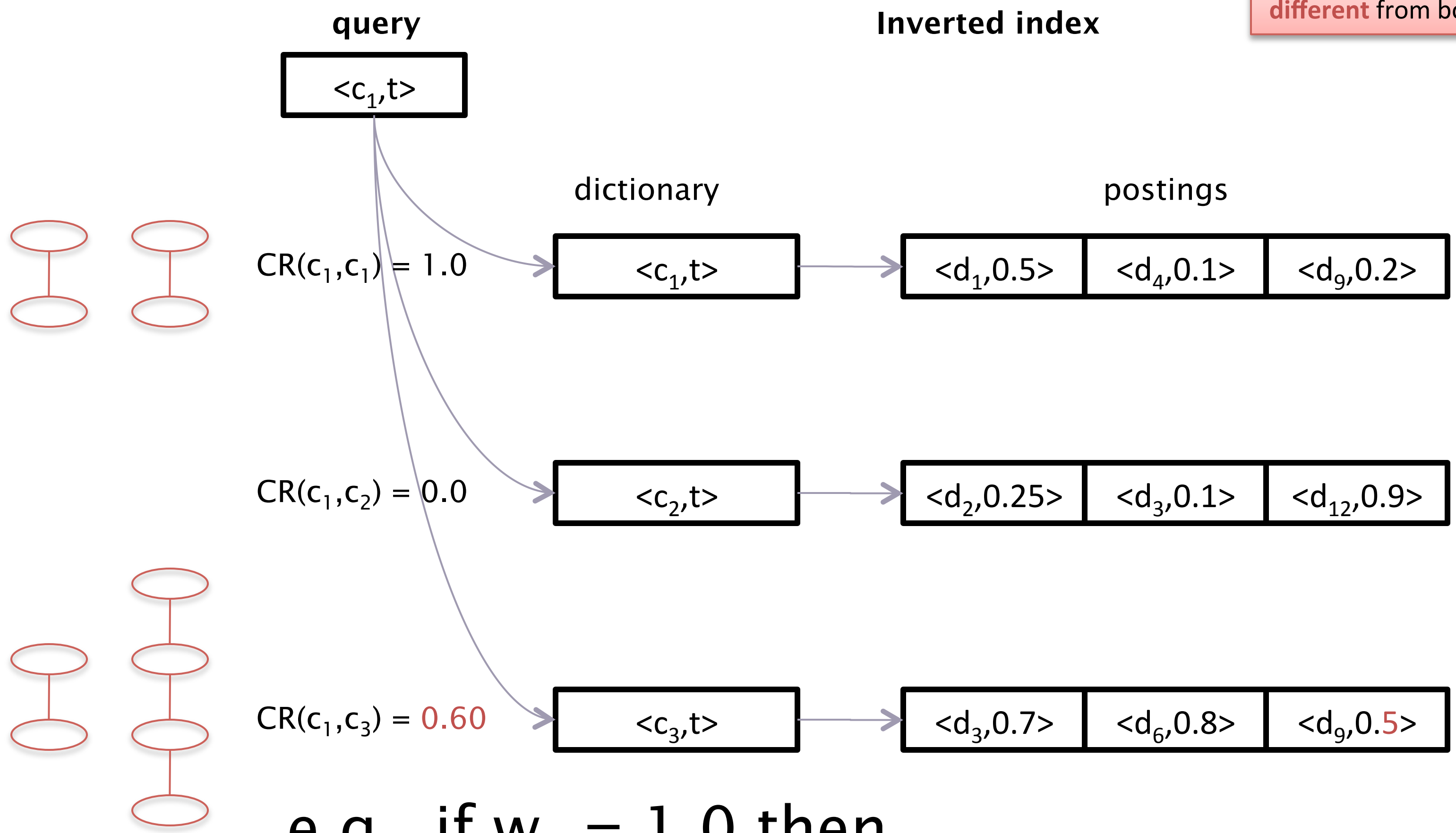
$$\sum_{c_k \in B} \sum_{c_l \in B} \mathrm{CR}(c_k, c_l) \sum_{t \in V} \mathrm{weight}(q, t, c_k) \frac{\mathrm{weight}(d, t, c_l)}{\sqrt{\sum_{c \in B, t \in V} \mathrm{weight}^2(d, t, c)}}$$

- *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. idf_t x $\mathrm{wf}_{t,d}$, 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.

Blanks on slides, you may want to fill in

SIMNOMERGE example

example slightly different from book



e.g., if $w_q = 1.0$ then
$sim(q,d_9) = 1.0\ ((1.0 \times 0.2) + (0.6 \times 0.5)) = .5$

“No Merge” because each context is separately calculated

SIMNOMERGE algorithm

SCOREDOCUMENTSWITHSIMNOMERGE(*q*, *B*, *V*, *N*,

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

XML IR EVALUATION

Initiative for the Evaluation of XML retrieval (INEX)

INEX: standard benchmark evaluation (yearly) that has produced test collections (documents, sets of queries, and relevance judgments).
Based on IEEE journal collection (since 2006 INEX uses the much larger English Wikipedia test collection).
The relevance of documents is judged by human assessors.

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

INEX Topics

- Two types:

1. content-only or **CO topics**: regular keyword queries as in unstructured information retrieval
2. 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

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.

- We distinguish four cases:

1. Exact coverage (E): The information sought is the main topic of the component and the component is a meaningful unit of information.
2. 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.
3. Too large (L): The information sought is present in the component, but is not the main topic.
4. No coverage (N): The information sought is not a topic of the component.

INEX relevance assessments

- The **topical relevance** dimension also has four levels: highly relevant (3), fairly relevant (2), marginally relevant (1) and nonrelevant (0).

Combining the relevance dimensions

Components are judged on both dimensions and the judgments are then combined into a digit-letter code, e.g. **2S** is a fairly relevant component that is too small. In theory, there are 16 combinations of coverage and relevance, but many cannot occur. For example, a nonrelevant component cannot have exact coverage, so the combination **3N** is not possible.

INEX relevance assessments

- The relevance-coverage combinations are quantized as follows:

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

- This evaluation scheme takes account of the fact that binary relevance judgments are not appropriate for XML retrieval. The quantization function **Q** instead allows us to grade each component as partially relevant. The number of relevant components in a retrieved set *A* of components can then be computed as:

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

INEX evaluation measures

- 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.

Drawback

Overlap is not accounted for. Accentuated by the problem of multiple nested elements occurring in a search result.

- Recent INEX focus: develop algorithms and evaluation measures that return non-redundant results lists and evaluate them properly.

Summary

Evaluation

Different schemes for lab versus in-the-wild testing

- Benchmark testing
- A/B testing

Resources:

- IIR 8, MIR Chapter 3, MG 4.5
- Carbonell and Goldstein (1998) The use of MMR, diversity-based reranking for reordering documents and producing summaries. SIGIR 21.

XML

- Structured or XML IR: effort to port unstructured IR know-how to structured (DB-like) data
- Specialized applications such as patents and digital libraries

ResourcesL

- IIR Ch 10
- http://inex.is.informatik.uni-duisburg.de/