CS3245

# Information Retrieval

10

Lecture 10:
Query Refinement
and XML IR

Live Q&A
https://pollev.com/jin

# Last Time

Search engine evaluation

- Benchmark
  - Measures: Precision / Recall / F-measure, Precision-recall graph and single number summaries
  - Documents, queries and relevance judgments
    - Kappa Measure
- A/B Testing
  - Overall evaluation criterion (OEC)

# Today

How to refine the query?

- Relevance Feedback
- Query Expansion

*cat → cat kitten feline -dog*

How to handled structured documents / queries?

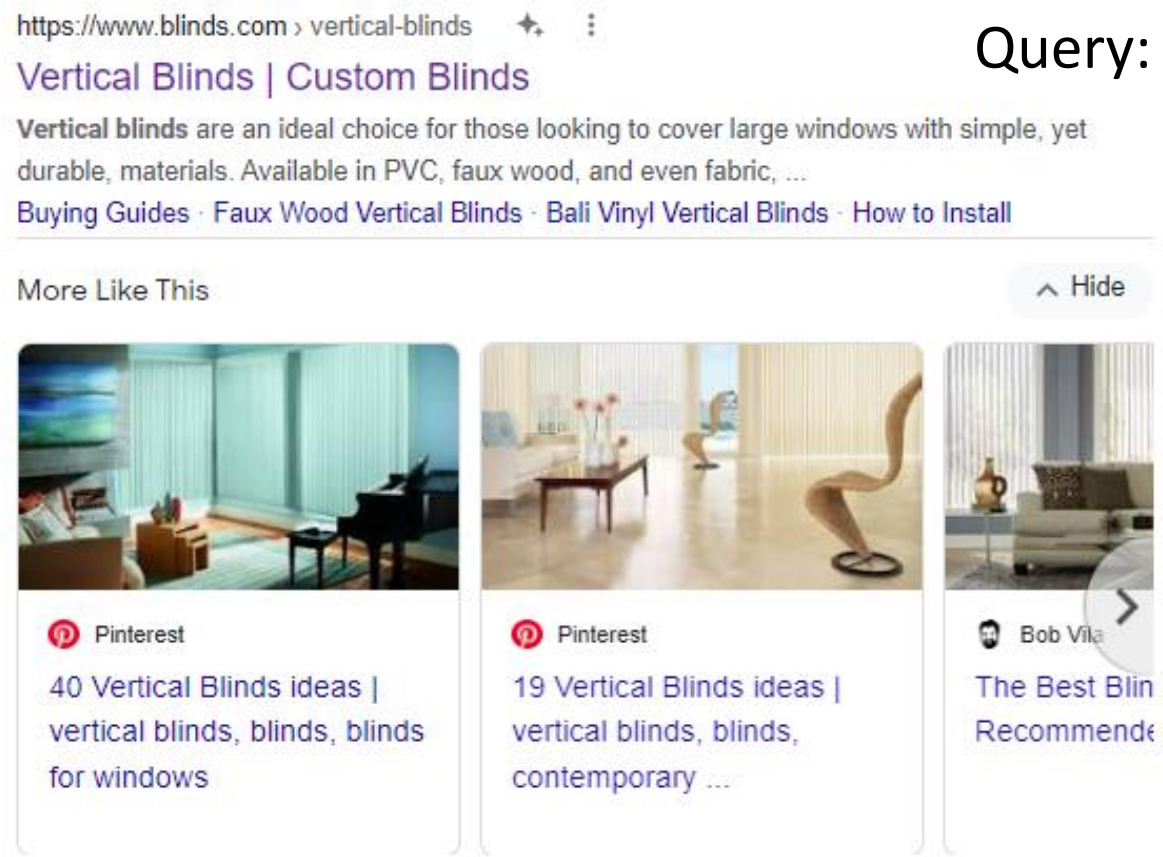
- XML Retrieval

```
<play>
    <author>Shakespeare</author>
    <act number="1">
        <scene number="vii">
            <verse>…</verse>
            <title>Macbeth's Castle</title>
        </scene>
    </act>
    <title>Macbeth</title>
</play>
```

# RELEVANCE FEEDBACK

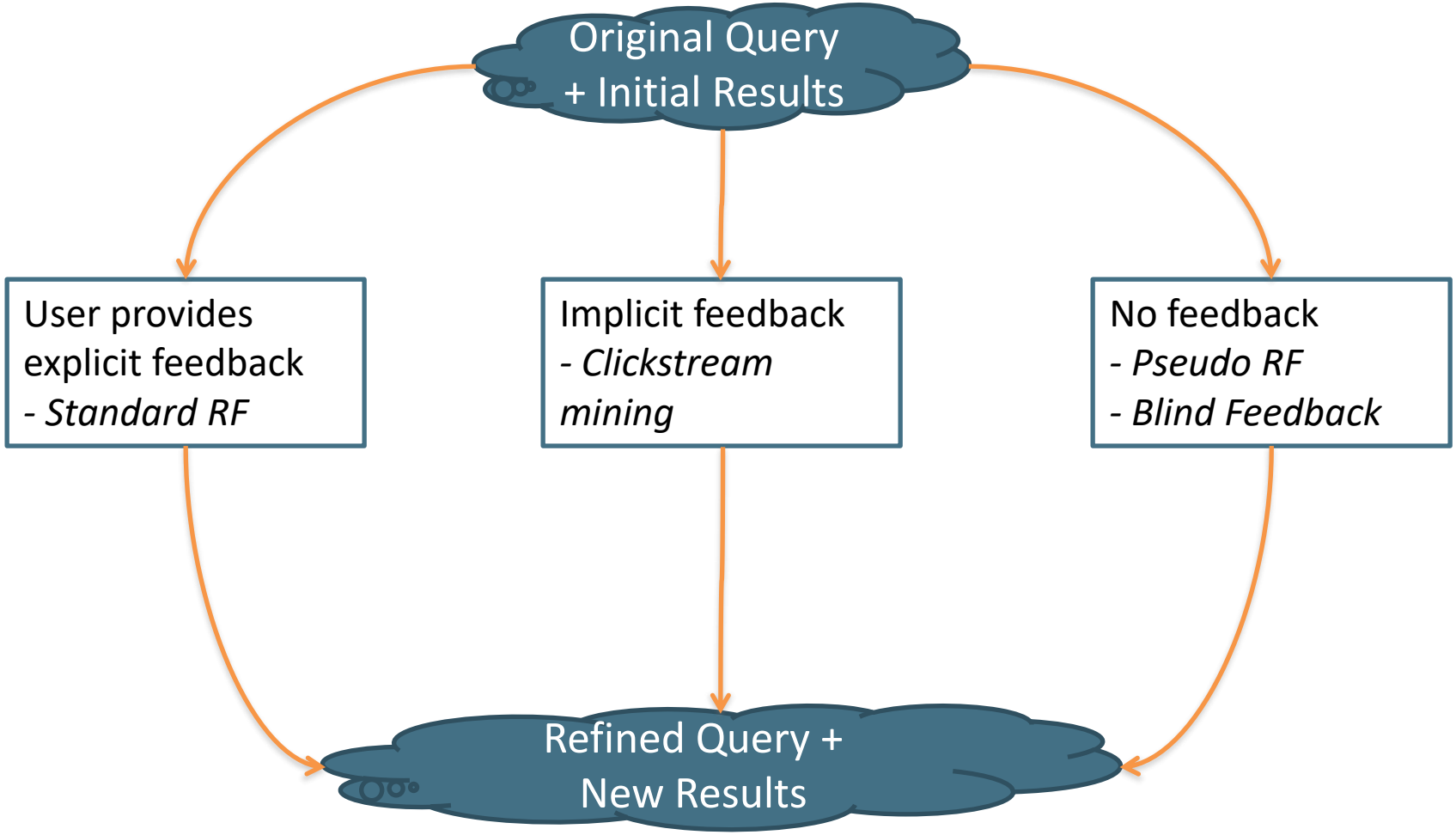
# Relevance Feedback

Query: vertical blinds



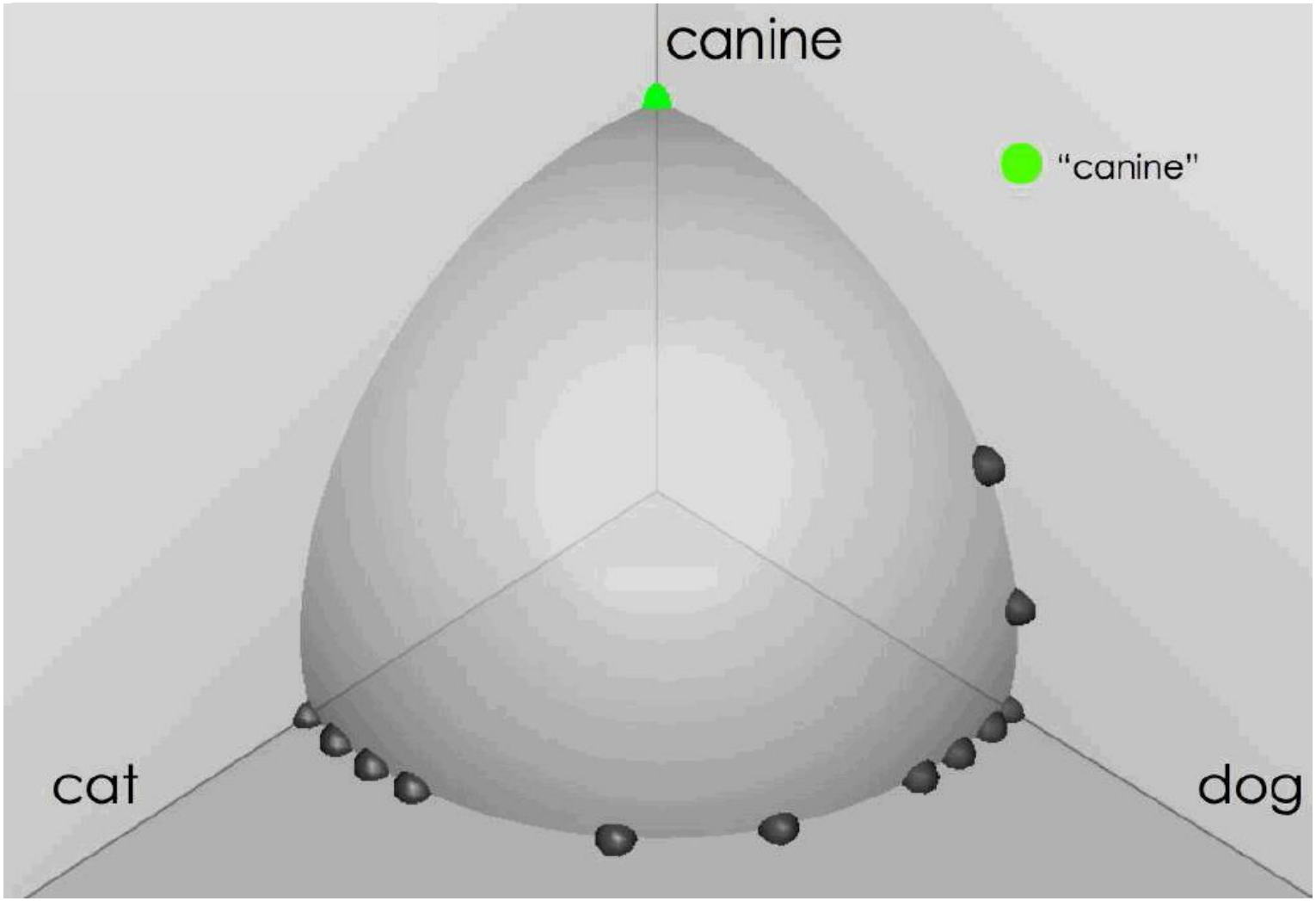

https://www.seroundtable.com/google-more-like-this-star-search-feature-34176.html

# Relevance Feedback



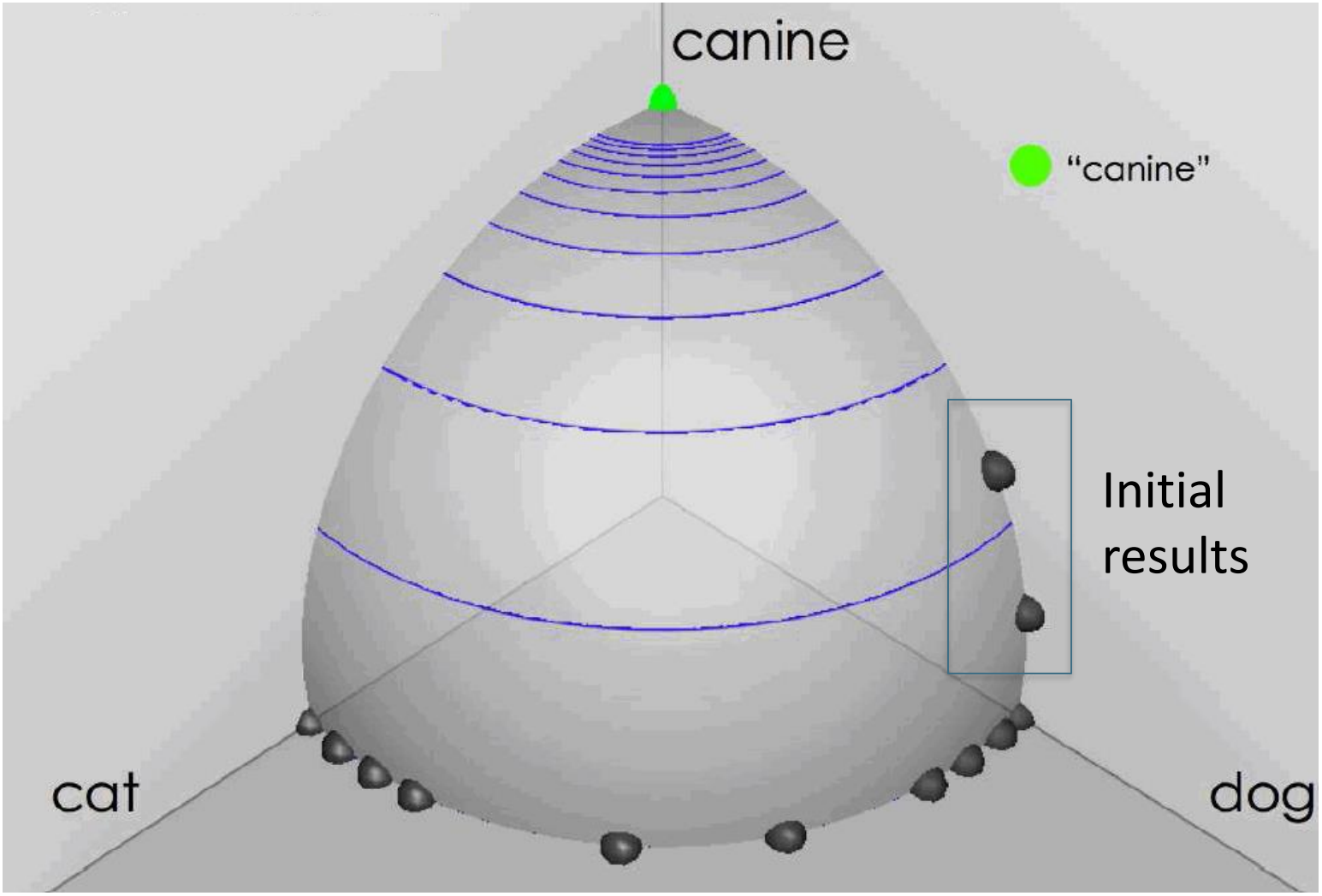
# Initial results for query *canine*

source: Fernando Diaz



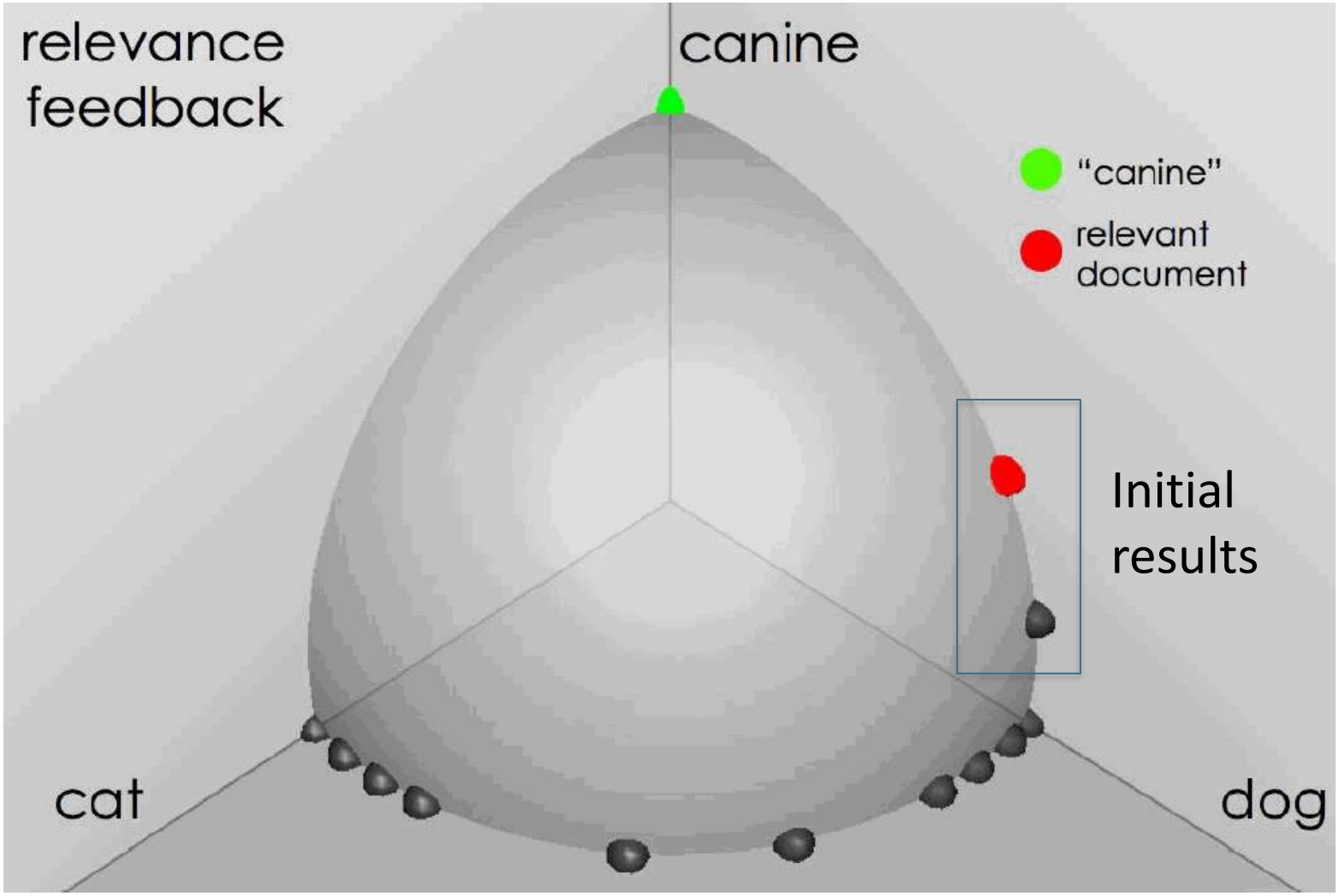
# Initial results for query *canine*

source: Fernando Diaz



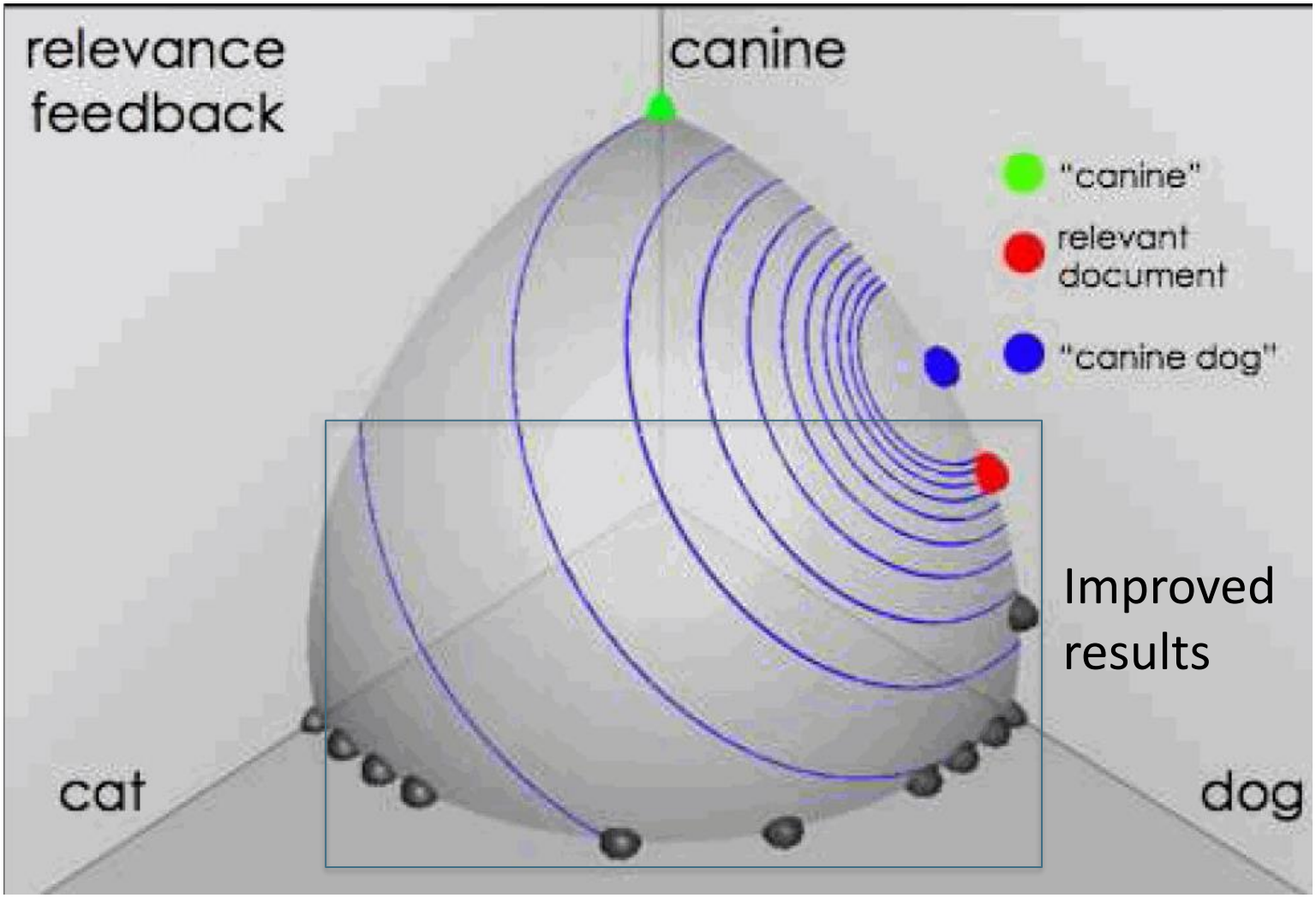
# User feedback: Select what is relevant

source: Fernando Diaz



# Results after relevance feedback

source: Fernando Diaz



# Initial query/results

Initial query: *New space satellite applications*

User marks relevant items

4.2 new
15.4 satellite
12.6 space
8.5 application

} Original terms with initial weights

\+ 1. 0.539, 08/13/91, NASA Hasn't Scrapped Imaging Spectrometer
\+ 2. 0.533, 07/09/91, NASA Scratches Environment Gear From Satellite Plan
– 3. 0.528, 04/04/90, Science Panel Backs NASA Satellite Plan, But Urges Launches of Smaller Probes
– 4. 0.526, 09/09/91, A NASA Satellite Project Accomplishes Incredible Feat: Staying Within Budget
– 5. 0.525, 07/24/90, Scientist Who Exposed Global Warming Proposes Satellites for Climate
– 6. 0.524, 08/22/90, Report Provides Support for the Critics Of Using Big Satellites to Study Climate
– 7. 0.516, 04/13/87, Arianespace Receives Satellite Launch Pact From Telesat Canada
\+ 8. 0.509, 12/02/87, Telecommunications Tale of Two Companies

Assume others as nonrelevant

# Refined query after relevance feedback

| | | |
|---|---|---|
| 2.074 new | 15.10 space | Original terms with adjusted weights |
| 30.81 satellite | 5.660 application | |
| 5.991 nasa | 5.196 eos | New terms with weights |
| 4.196 launch | 3.972 aster | |
| 3.516 instrument | 3.446 arianespace | |
| 3.004 bundespost | 2.806 ss | |
| 2.790 rocket | 2.053 scientist | |
| 2.003 broadcast | 1.172 earth | |
| 0.836 oil | 0.646 measure | |

# Results for the expanded query

2 1. 0.513, 07/09/91, NASA Scratches Environment Gear From Satellite Plan

1 2. 0.500, 08/13/91, NASA Hasn't Scrapped Imaging Spectrometer

3. 0.493, 08/07/89, When the Pentagon Launches a Secret Satellite, Space Sleuths Do Some Spy Work of Their Own

4. 0.493, 07/31/89, NASA Uses 'Warm' Superconductors For Fast Circuit

8 5. 0.492, 12/02/87, Telecommunications Tale of Two Companies

6. 0.491, 07/09/91, Soviets May Adapt Parts of SS-20 Missile For Commercial Use

7. 0.490, 07/12/88, Gaping Gap: Pentagon Lags in Race To Match the Soviets In Rocket Launchers

8. 0.490, 06/14/90, Rescue of Satellite By Space Agency To Cost $90 Million

Original Positions of Marked Relevant Documents

# How to refine a query?

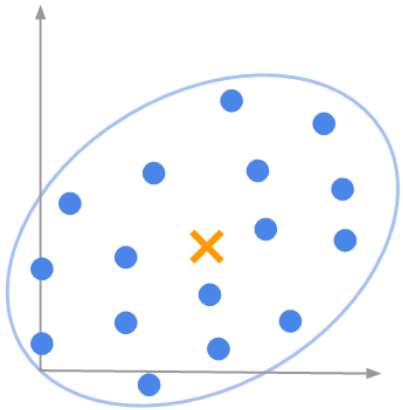
- We have …
  - $q_0$ = *the initial query*
    - For retrieving some initial docs
  - $D_r$ = a (small) set of known relevant doc vectors
  - $D_{nr}$ = a (small) set of known irrelevant doc vectors
    - From the relevant feedback on the initial docs
- We want to find …
  - $q_m$ = the modified query

NUS National University of Singapore

# Centroid

- The center of mass of a set of documents.

$$\vec{\mu}(D) = \frac{1}{|D|}\sum_{d \in D} \vec{d}$$



  - |D| = the number of documents in the set.

- Example:
  - D = {$d_1$, $d_2$, $d_3$} with $d_1$ = (1, 2), $d_2$ = (3, 5), $d_3$ = (2, 2)
  - Centroid of D: ((1+3+2)/3, (2+5+2)/3) = (2, 3)

# Rocchio (1971)

*Popularized in the SMART system (Salton)*

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

*Centroid of $D_r$* *Centroid of $D_{nr}$*

- $\{\alpha, \beta, \gamma\}$ = weights (hand-chosen or set empirically)
  - Tradeoff α vs. β/γ: What if we have only a few judged documents?
  - Tradeoff β vs. γ: Which is more valuable?
- Term weights in the query vector can go negative
  - Set the weights to 0 or exclude documents which contain such terms

# Evaluation of relevance feedback

Use $q_m$ and compute precision recall graph

1. Assess on all documents in the collection
   - Spectacular improvements, but … it's cheating!
2. Use documents in residual collection (set of documents minus those assessed relevant)
   - Lower results but more realistic
   - Compare the relative performance instead

- Best: use two collections each with their own relevance assessments
  - $q_0$ and user feedback from first collection
  - $q_m$ run on second collection and measured

# When does RF work?

Empirically, a round of RF is often very useful. Two rounds is sometimes marginally useful.

The two assumptions should hold:

1. User's initial query at least partially works.

2. (Non)-relevant documents are similar.

# Pseudo relevance feedback (PRF)

- Blind feedback automates the "manual" part of true RF, by assuming the top *k* is actually relevant.

- Algorithm:
  - Retrieve a ranked list of hits for the user's query
  - Assume that the top *k* documents are relevant.
  - Do relevance feedback

- Works very well on average
  - But can go horribly wrong for some queries
  - Several iterations can cause query drift

# QUERY EXPANSION

# Query Expansion

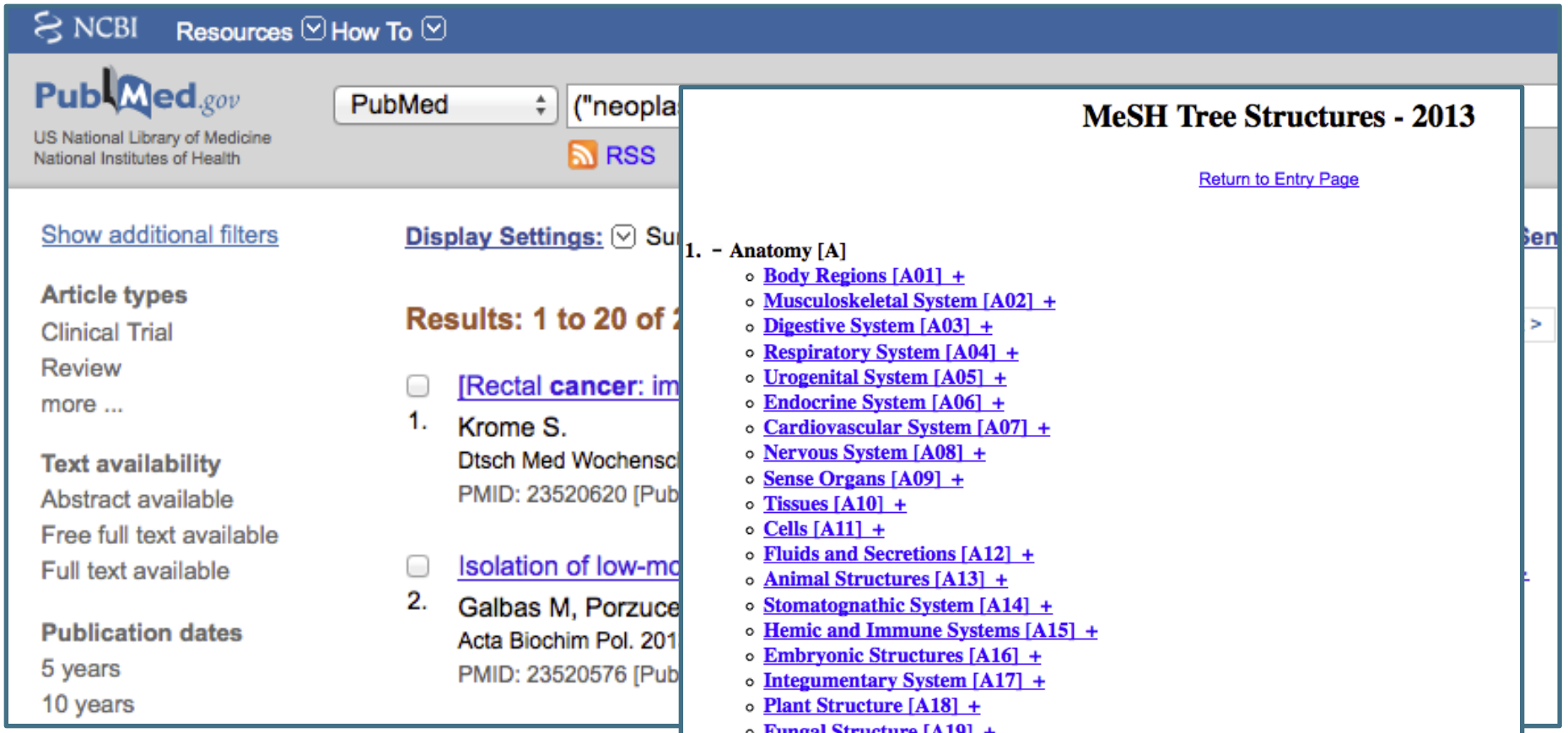
- For each query term, expand it with the related words of *t* from a thesaurus
  - The thesaurus can be **manually compiled** or **automatically generated**.

- Examples
  - feline → feline cat
  - interest rate → interest rate fascinate evaluate

S: (adj) **feline** (of or relating to cats) *"feline fur"*

- Generally increases recall, but may decrease precision when terms are ambiguous.

# Manually compiled thesauri: MeSH



# Manually compiled thesaurii: WordNet

WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: washing machine Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

**Noun**

- S: (n) washer, automatic washer, **washing machine** (a home appliance for washing clothes and linens automatically)
  - *direct hypernym* / ***inherited hypernym*** / *sister term*
    - S: (n) white goods (large electrical home appliances (refrigerators or washing machines etc.) that are typically finished in white enamel)
      - S: (n) home appliance, household appliance (an appliance that does a particular job in the home)
        - S: (n) appliance (durable goods for home or office use)
          - S: (n) durables, durable goods, consumer durables (consumer goods that are not destroyed by use)
            - S: (n) consumer goods (goods (as food or clothing) intended for direct use or

```
from nltk.corpus import wordnet as wn

wn.synsets("motorcar")
wn.synsets("car.n.01").lemma_names
```

# Automatic Thesaurus Generation

You shall know a word by the company it keeps
– John R. Firth

- You can "harvest", "peel", "eat" and "prepare" apples and pears, so apples and pears must be similar
- Generate a thesaurus by analyzing the documents
- Assumption: distributional similarity
  - i.e., Two words are similar if they **co-occur** / **share same grammatical relations** with similar words.

Co-occurrences are more robust; grammatical relations are more accurate. Why?

# Co-occurrence Thesaurus

In NLTK! ☺
Have a look!

A concordance permits us to see words in context. For example, we saw that then inserting the relevant word in parentheses:

```
>>> text1.similar("monstrous")
Building word-context index...
subtly impalpable pitiable curious imperial perilous trustv
abundant untoward singular lamentable few maddens horrible
mystifying christian exasperate puzzled
>>> text2.similar("monstrous")
Building word-context index...
very exceedingly so heartily a great good amazingly as swe
remarkably extremely vast
>>>
```

Observe that we get different results for different texts. Austen uses this word

The term `common_contexts` allows us to examine just the contexts that are sh

```
>>> text2.common_contexts(["monstrous", "very"])
be_glad am_glad a_pretty is_pretty a_lucky
>>>
```

NUS National University of Singapore

# XML RETRIEVAL

# Unstructured vs. Structured

Macbeth

Shakespeare

Act 1, Scene vii

Macbeth's Castle

…

```
<play>
    <author>Shakespeare</author>
    <act number="1">
        <scene number="vii">
            <verse>…</verse>
            <title>Macbeth's Castle</title>
        </scene>
    </act>
    <title>Macbeth</title>
</play>
```

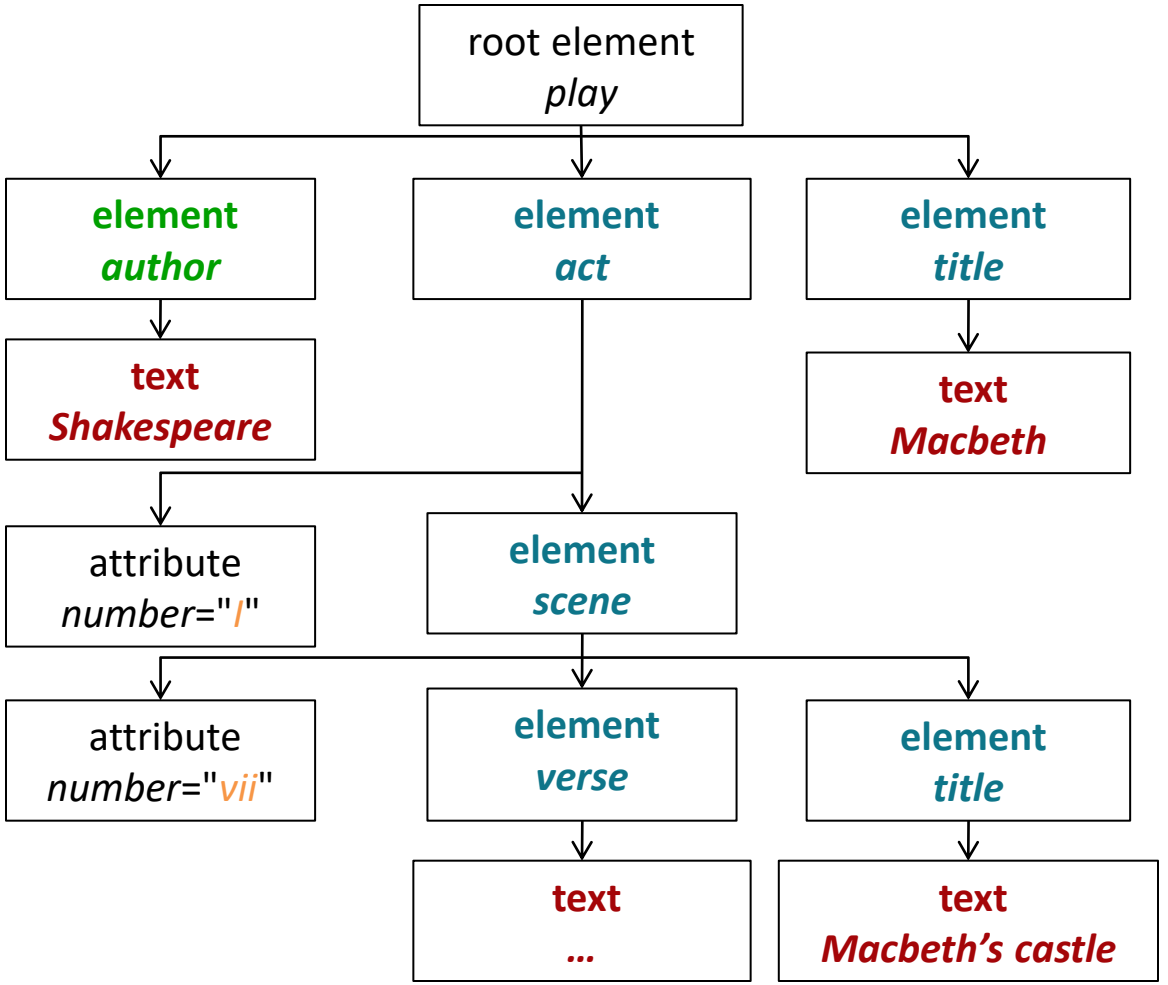
# XML Document

Internal nodes encode **document structure** or **metadata**

An element can have one or more **attributes** and sub elements

**Leaf nodes** consist of text

Possible **queries** which match with **(part of)** this document:
Macbeth
scene/title#castle



# Structured Retrieval

## 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 patents whose claims mention RSA public key encryption and that cite US Patent 4,405,829*
- Entity-tagged text: *give me articles about sightseeing tours of the Vatican and the Coliseum*

# Common Problems

- What is the unit of retrieval?
  - E.g., the whole document or a **component** of it.
- Do the users know about the structure of the documents well?
- How to rank the items in the result list?
- How to evaluate the retrieval performance?

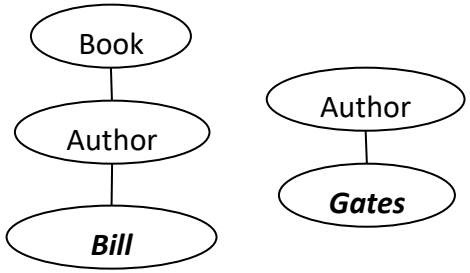
# VECTOR SPACE MODEL FOR XML IR

# Key idea: Structural terms

- An unstructured document / query
  - Consists of one or more terms
  - Is a vector in a high-dimensional space where each dimension corresponds to a term

***Bill Gates***

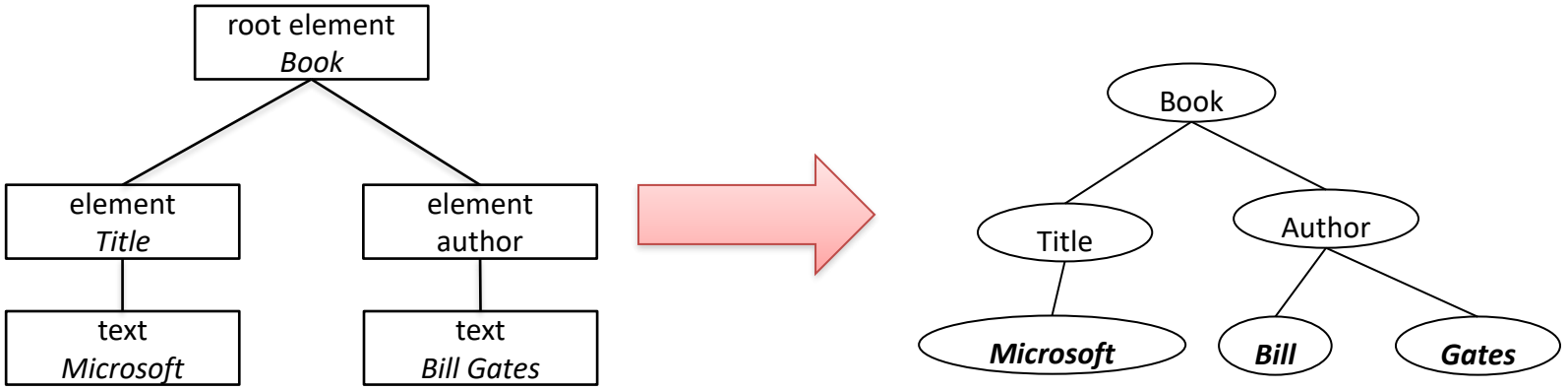
- A structured document / query
  - Consists of one or more structural terms
  - Is a vector in a high-dimensional space where each dimension corresponds to a structural term




A structural term <c, t> is a pair of XML-context c and vocabulary term t.

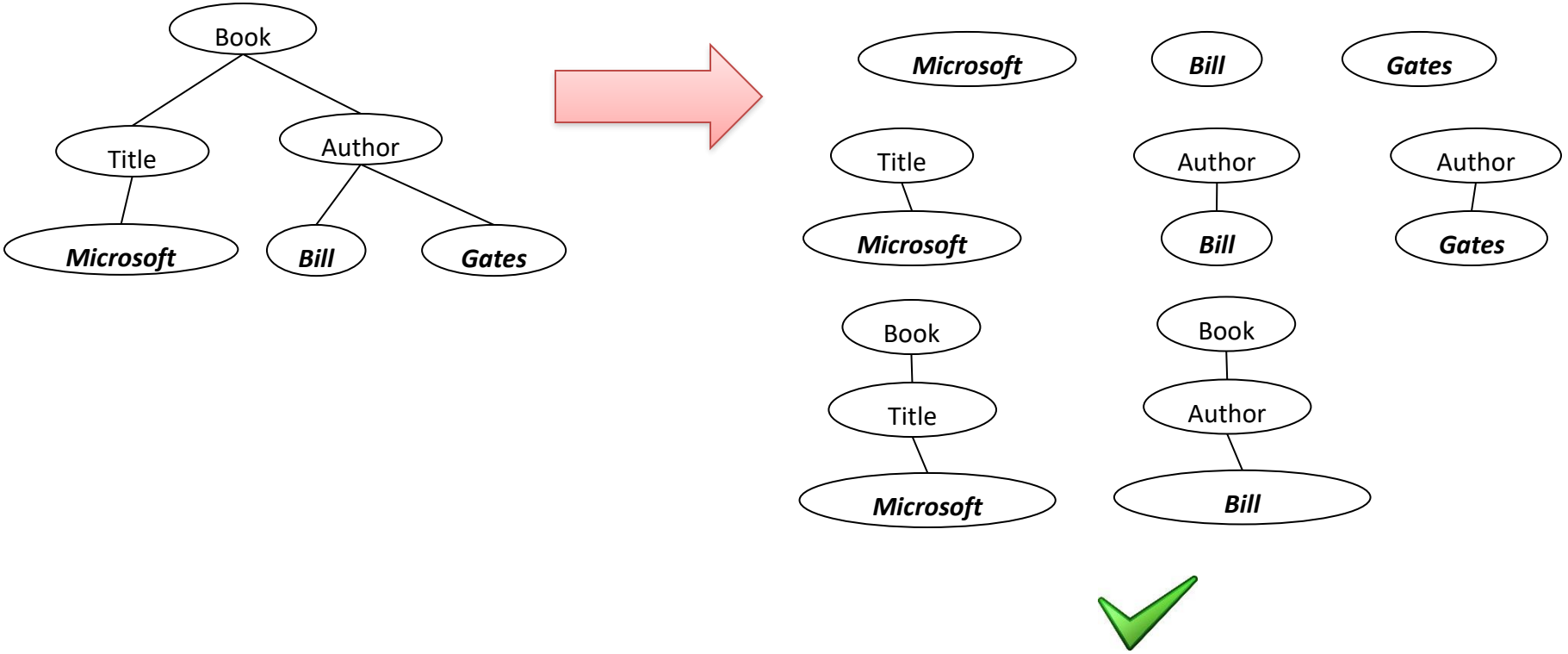
# Structural terms extraction

- Step 1: Take each text node (leaf) and break it into multiple nodes, one for each word. E.g. split **Bill Gates** into **Bill** and **Gates**



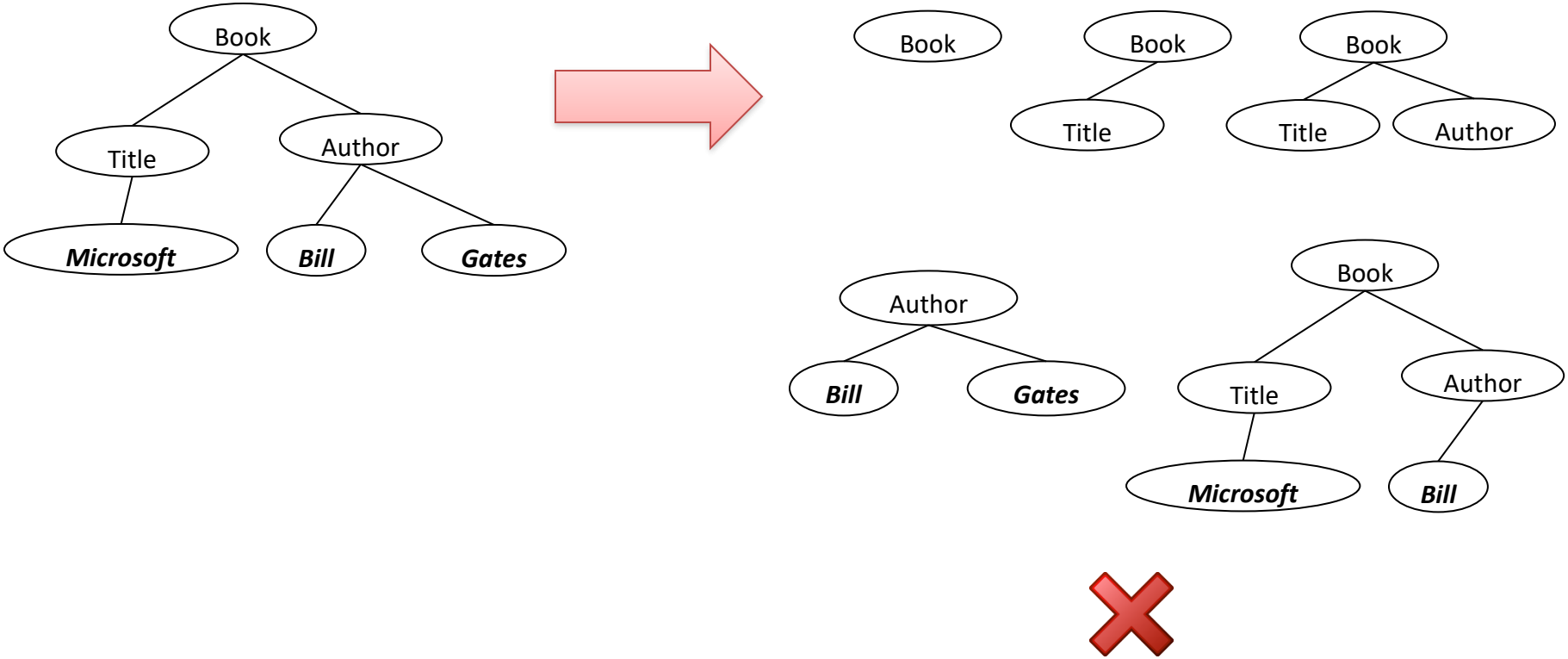
# Structural terms extraction

- Step 2: Extract all paths that end in a single vocabulary term as **structural terms**

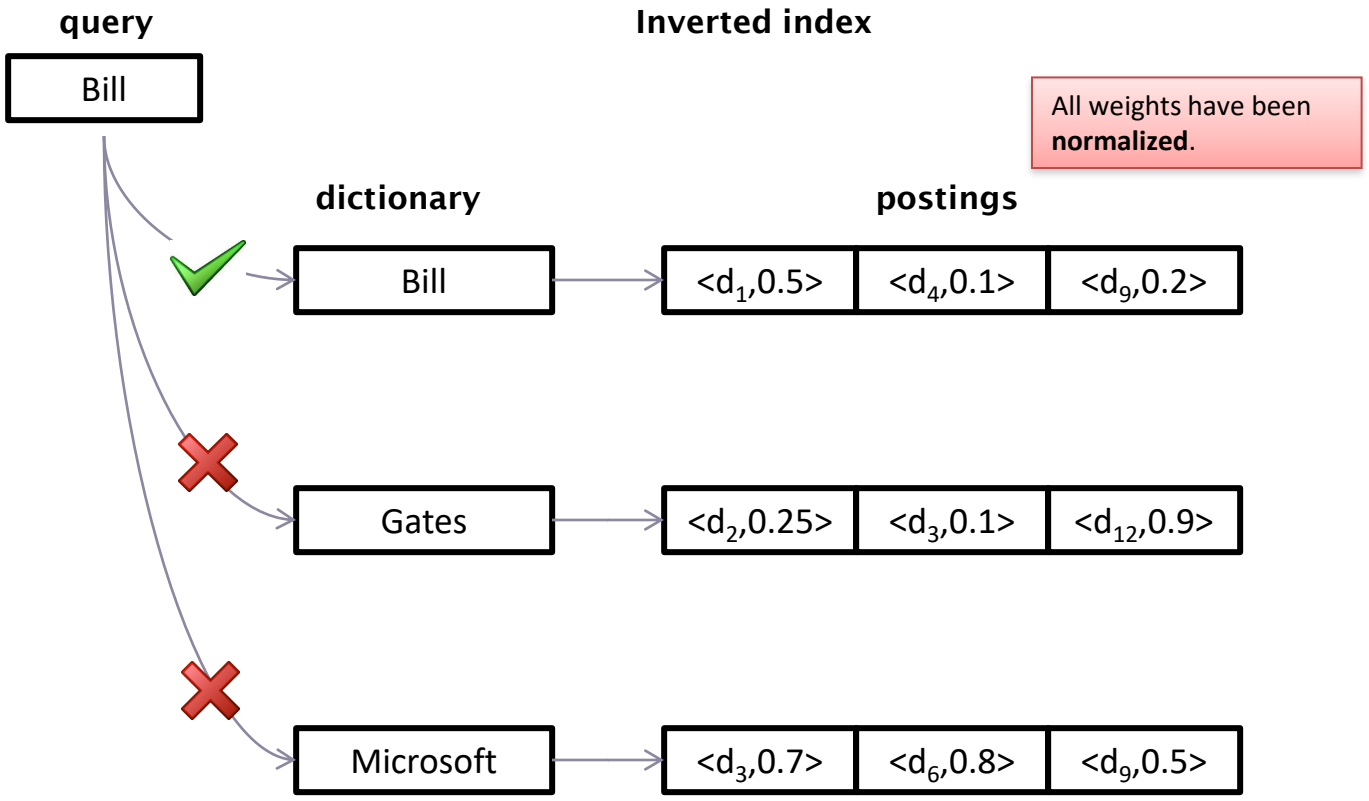


# Structural terms extraction

- Step 2: Extract all paths that end in a single vocabulary term as **structural terms**



# Recap: Cosine Similarity

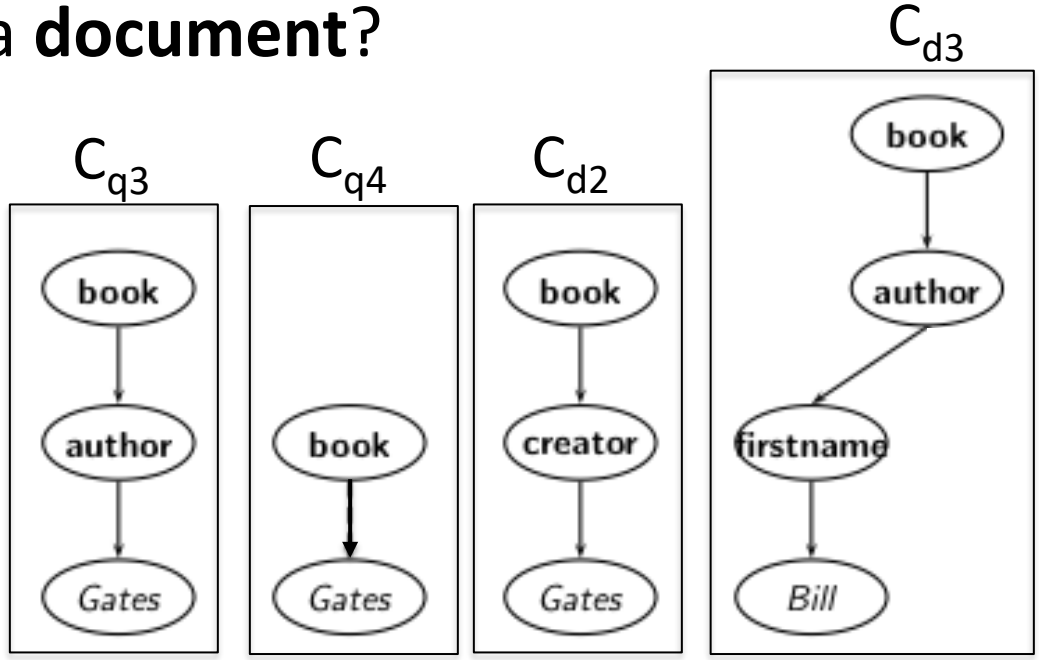



if $w_q = 1.0$, then score($d_9$) += (1.0 x 0.2) = 0.2

Query Term Weight * Document Term Weight

# Matching between structural terms

- Can $C_{q3}$ and $C_{q4}$ from a **query** match with $C_{d2}$ and $C_{d3}$ from a **document**?



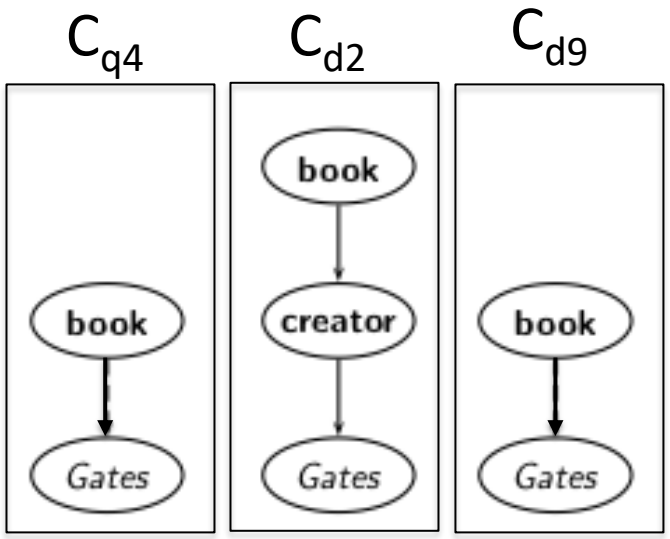

- $c_q$ matches $c_d$ **iff** we can transform $c_q$ into $c_d$ by inserting additional nodes.

# Similarity between structural terms

- Context Resemblance:
  - A simple measure of the similarity of a structural term $c_q$ in a query and a structural term $c_d$ in a document

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

  - $|c_q|$ and $|c_d|$ are the number of nodes in the terms, respectively.
  - Examples
    - CR($c_{q4}$, $c_{d2}$) = (1 + 2) / (1 + 3) = 0.75
    - CR($c_{q4}$, $c_{d9}$) = 3 / 3 = 1



# SimNoMerge

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

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

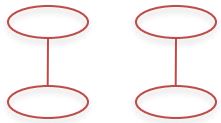
*Context resemblance*

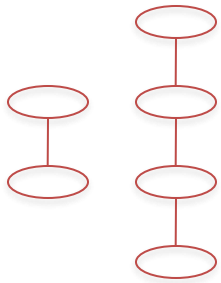
*Query structural term weight*

*Normalized document structural term weight*

- $V$ 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. $\mathrm{idf}_t$ x $\mathrm{wf}_{t,d}$, where $\mathrm{idf}_t$ depends on which elements we use to compute $\mathrm{df}_t$.)
- SimNoMerge ($q$, $d$) is not a true cosine measure since its value can be larger than 1.0.

# SimNoMerge example





query

<$c_1$,t>

e.g., author#Bill

Inverted index

This example is slightly different from book

All weights have been **normalized**.

dictionary

postings

CR($c_1$,$c_1$) = 1.0

<$c_1$,t>

<$d_1$,0.5> <$d_4$,0.1> <$d_9$,0.2>

e.g., author#Bill

OK to ignore Query vs <c2, t> since CR = 0.0

CR($c_1$,$c_2$) = 0.0

<$c_2$,t>

<$d_2$,0.25> <$d_3$,0.1> <$d_{12}$,0.9>

e.g., title#"Bill"

$d_9$ contains two structural terms <$c_1$, t> and <$c_3$, t>.

CR($c_1$,$c_3$) = 0.60

<$c_3$,t>

<$d_3$,0.7> <$d_6$,0.8> <$d_9$,0.5>

e.g., book/author/firstname#Bill

Context Resemblance * Query Term Weight * Document Term Weight

if $w_q$ = 1.0, then score($d_9$) +=

(1.0 × 1.0 x 0.2) + (0.6 × 1.0 x 0.5) = 0.5

Query vs <$c_1$,t>

Query vs <$c_3$,t>

"No Merge" because each context is separately calculated

# SimNoMerge algorithm

NUS National University of Singapore

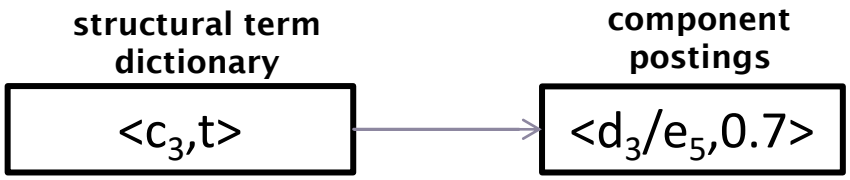
ScoreDocumentsWithSimNoMerge (q, *B*, *V*, *N*, *normalizer*)

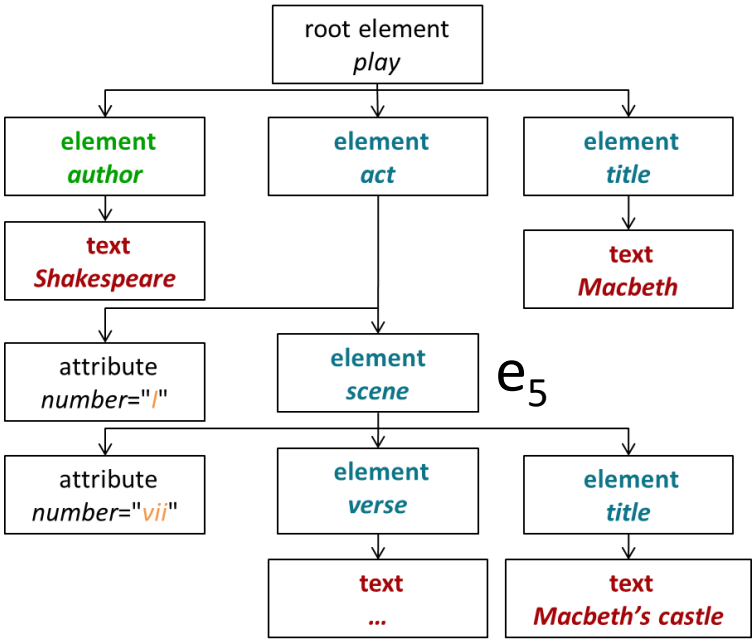
```
1  for n ← 1 to N
2  do score[n] ← 0
3  for each ⟨c_q, t⟩ ∈ q
4  do w_q ← WEIGHT(q, t, c_q)
5      for each c ∈ B
6      do if CR(c_q, c) > 0
7            then postings ← GETPOSTINGS(⟨c, t⟩)
8                 for each posting ∈ postings
9                 do x ← CR(c_q, c) * w_q * weight(posting)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

# From document to component

- The same idea applies to indexing and retrieving components (i.e., elements) in XML documents.

- E.g.,
  - Element $e_5$ in $d_3$ can be indexed and retrieved by itself.



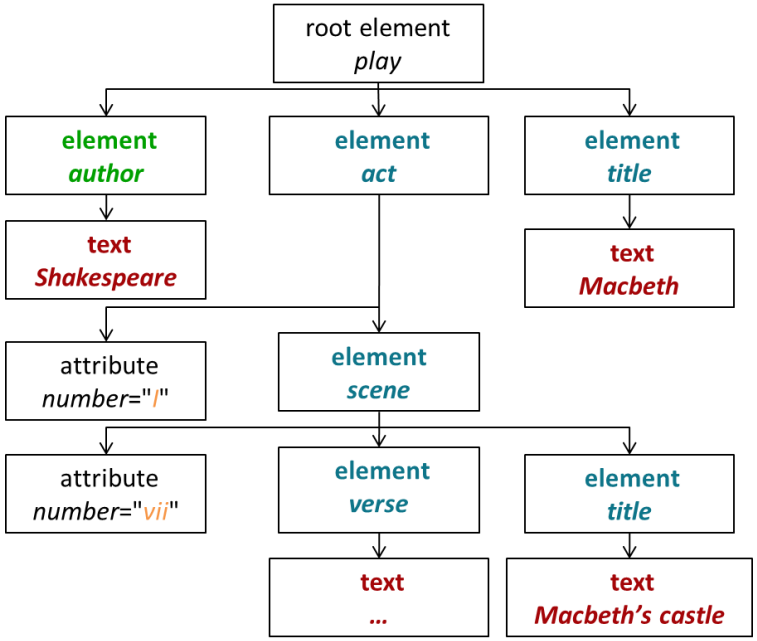
# XML IR EVALUATION

# XML IR Evaluation

- Component-based
- Two aspects: **Component Coverage + Topical Relevance**.

**Component coverage**

Evaluates whether the element retrieved is "structurally" correct, i.e., neither too low nor too high in the tree.



# Component Coverage

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

# Topical Relevance

- Four levels:
  - Highly relevant (3)
  - Fairly relevant (2)
  - Marginally relevant (1)
  - Nonrelevant (0)

# Combining the relevance dimensions

- A digit-letter code
  - E.g., **2S** is a fairly relevant component that is too small.
- 16 combinations in theory but many cannot occur.
  - E.g., a nonrelevant component cannot have exact coverage, so the combination **0E** is not possible.

# INEX relevance assessments

- The relevance-coverage combinations are quantized as

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

- The number of relevant components in a retrieved set *A* of components can then be computed as:

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

- Example: If the 5 components retrieved are assessed as {3E, 3E, 0N, 1E, 1S}, the precision is (1 + 1 + 0 + 0.5 + 0.25) / 5 = 0.55

# Summary

1. Query Refinement
   - Relevance Feedback – "Documents"
   - Query Expansion – "Terms"

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

- Resources
  - IIR Ch 9/10
  - MG Ch. 4.7 and MIR Ch. 5.2 – 5.4
  - http://inex.is.informatik.uni-duisburg.de/