Evaluation of IR systems

Introduction

- How to measure user happiness?

- Depends on many factors:
  - Relevance of results
  - User interface design layout
  - Speed of response
  - Target application
    - Web engine: user finds what they want and return to the engine
      - Can measure rate of return users
    - e-commerce site: user finds what they want and make a purchase
      - Is it the end-user, or the e-commerce site, whose happiness we measure?
      - Measure time to purchase, or fraction of searchers who become buyers?
    - 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 …

Evaluation of IR systems
Introduction

- System quality
  - How fast does the system index?
    - How many documents/hour for a certain distribution of document sizes?
  - How fast does it search?
    - latency as function of index size
  - How large is the document collection?
  - How expressive is its query language? How fast is it on complex queries?
- all but the last criteria are measurable

Evaluation of IR systems

Introduction

- To measure ad hoc information retrieval effectiveness in the standard way, we need:
  - A test document collection
  - A test suite of information needs, expressible as queries
  - A set of relevance judgements
    - which documents are relevant/non-relevant for each query a.k.a. Ground Truth, Gold Standard
- Test collection must be of a reasonable size
  - Need to average performance since results are very variable over different documents and information needs

Introduction

- Relevance is assessed relative to an information need, not to a query.
  - For example, the information need:
    "I'm looking for information on whether drinking red wine is more effective at reducing your risk of heart attack than white wine"

    might be translated into the query:

    white AND red AND wine AND heart AND attack AND effective
- A document is relevant if it addresses the stated information need, *not just because it contains all the word in the query*

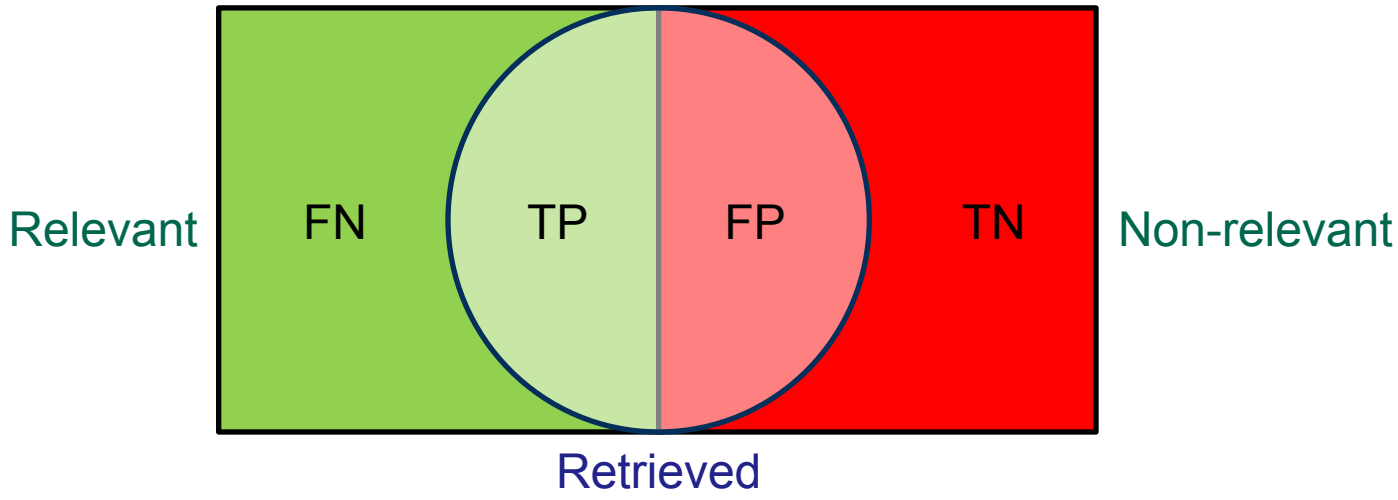
Evaluation of IR systems

Unranked retrieval: TP, FP, FN, TN

	Relevant	Non-relevant
Retrieved	true positive (TP)	false positive (FP)
Not-retrieved	false negative(FN)	true negative(TN)

Retrieved/Not-retrieved: from IR system
Relevant/Non-relevant: from Ground Truth

True: Retrieved/Not-retrieved corresponds to Relevent/Non-relevant
False: Retrieved/Not-retrieved doesn't correspond to Relevent/Non-relevant



Evaluation of IR systems

Unranked retrieval: Precision and Recall

- Precision (P): fraction of retrieved documents that are relevant

$$P = \frac{relevant\ retrieved}{retrieved} = \frac{TP}{TP + FP}$$

  - Measures the “degree of soundness” of the system

- Recall (R): fraction of relevant documents that are retrieved

$$R = \frac{relevant\ retrieved}{relevant} = \frac{TP}{TP + FN}$$

  - Measures the “degree of completeness” of the system

Evaluation of IR systems

Unranked retrieval: Precision and Recall

- An IR system can get high recall (but low precision) by retrieving all documents for all queries
  - Recall is a non-decreasing function of the number of retrieved documents
  - Precision in good IR systems is a decreasing function of the number of retrieved documents
- Precision can be computed at different levels of recall
- Precision-oriented users
  - Web surfers
- Recall-oriented users
  - Professional searchers, paralegals, intelligence analysts

Evaluation of IR systems

Unranked retrieval: F-measure

- F-measure(F): weighted harmonic mean of Precision and Recall

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

  - $\alpha \in [0,1]$
    - $\alpha = 1 \rightarrow F = P$
    - $\alpha = 0 \rightarrow F = R$
    - Usually $\alpha = 0.5 \rightarrow F = \frac{2 \cdot PR}{P+R}$ (balanced F-measure)
  - Trade-off between the “degree of soundness” and the “degree of completeness” of a system

- Weighted harmonic mean: $H = \frac{\sum_{i=1}^{n} \alpha_i}{\sum_{i=1}^{n} \alpha_i \frac{1}{x_i}}$

Evaluation of IR systems

Unranked retrieval: F-measure

- Harmonic mean is a conservative average
  - e.g. 1 document out of 10000 is relevant
  - Retrieving all documents
    - Recall = 100%
    - Precision = 0.01%
    - Arithmetic mean = $\frac{1}{2}(P + R) = 50\%$
    - Harmonic mean (Balanced F-measure) = $\frac{2 \cdot PR}{P+R} = 0.02\%$
- When the value of two number differs, harmonic mean is closer to their minimum than arithmetic or geometric mean

Evaluation of IR systems

Unranked retrieval: Accuracy

- Accuracy(A): fraction of correctly classified documents

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

  - Accuracy is not suitable in the context of IR
    - In many cases data are extremely skewed
      - e.g. 99.99% of documents are Non-relevant
    - In these cases a system tuned to maximize the accuracy will almost always retrieve nothing!
      - Accuracy is 99.99%
      - Recall is $\frac{0}{1}$
      - Precision is $\frac{0}{0}$

Unranked retrieval: Precision and Recall drawbacks

Difficulties in using Precision/Recall

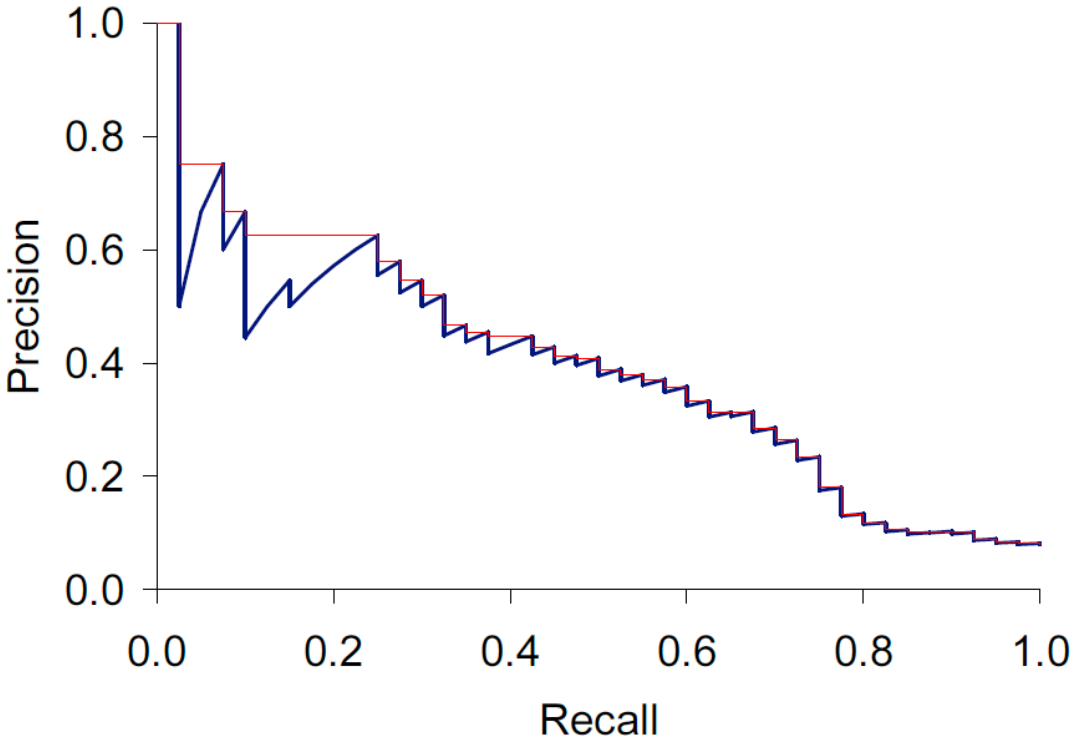
- Average over large corpus/query…
  - Need human relevance assessments
    - People aren't reliable assessors
  - Assessments have to be binary
    - Nuanced assessments?
  - Heavily skewed by corpus/authorship
    - Results may not translate from one domain to another

- *The relevance of one document is treated as independent of the relevance of other documents in the collection*
  - This is also an assumption in most retrieval systems

Evaluation of IR systems

Ranked retrieval: Precision and Recall

- Precision/Recall/F-measure are set-based measures
  - Unordered sets of documents
- In ranked retrieval systems, P and R are values relative to a rank position
  - Evaluation performed by computing Precision as a function of Recall

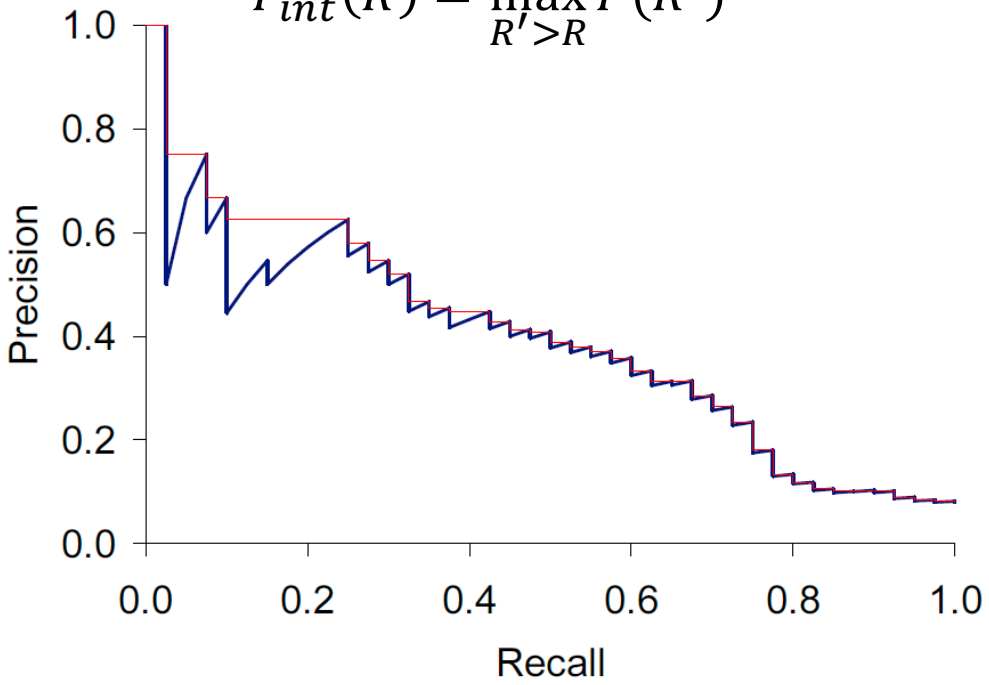


Evaluation of IR systems

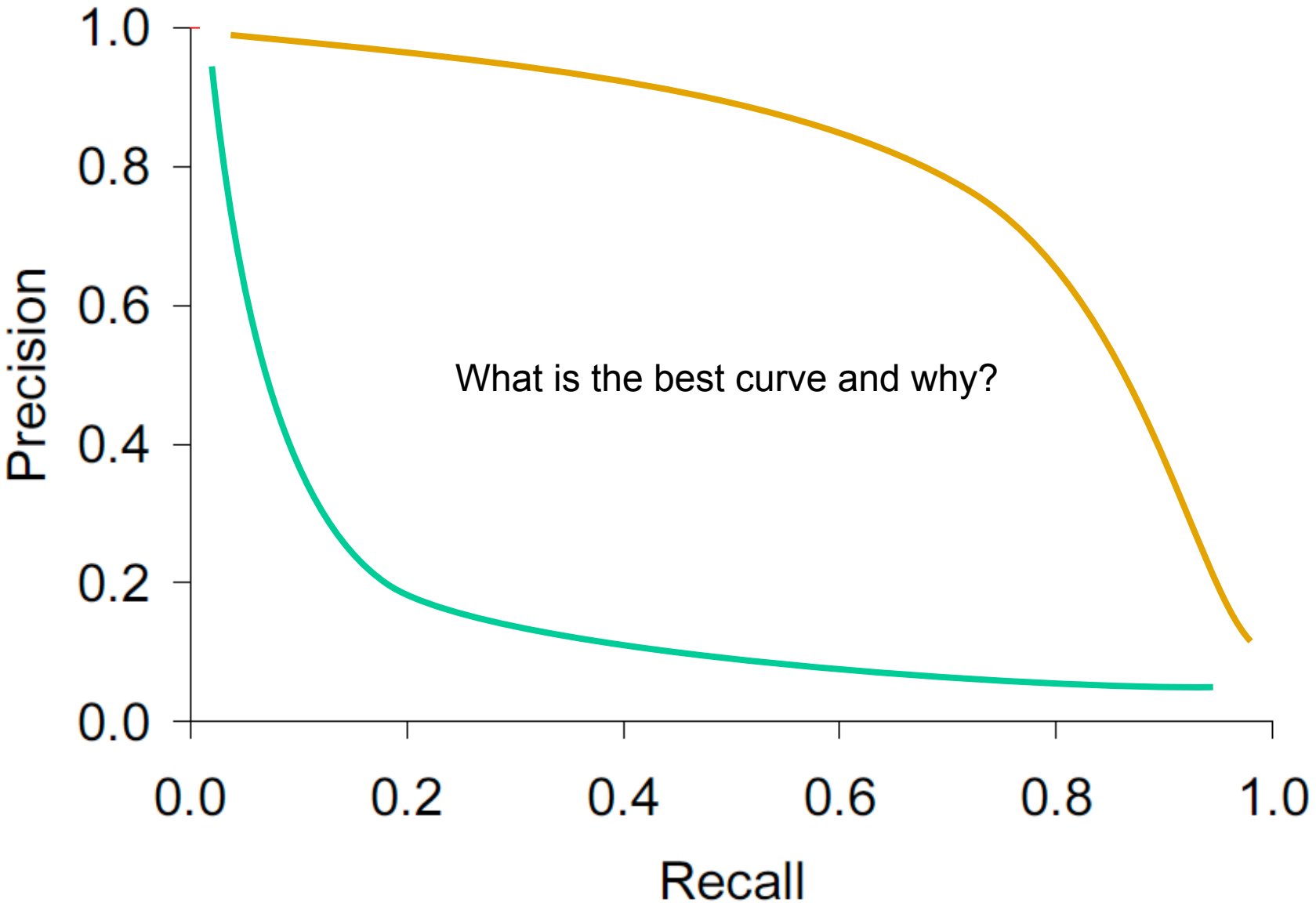
Ranked retrieval: Precision and Recall

- Precision/Recall function
  - If the $(k+1)th$ retrieved document is relevant, then $R(k+1) > R(k)$ and $P(k+1) \geq P(k)$
  - If the $(k+1)th$ retrieved document is non-relevant, then $R(k+1) = R(k)$, but $P(k+1) \leq P(k)$
- To remove the jiggles, use interpolated precision

$$P_{int}(R) = \max_{R' > R} P(R')$$



Ranked retrieval: Precision and Recall



Evaluation of IR systems

Ranked retrieval: Average Precision

- 11-point interpolated average precision
  - measure precision at 11 recall levels $\{0.0, 0.1, 0.2, \ldots, 1.0\}$
  - compute the arithmetic mean of the precision levels

- mean average precision (MAP)
  - Given a set of queries Q, whose cardinality is $|Q|$
  1. Compute the average precision (AP) for each query
     - Average the precision values obtained for the top set of k documents *after each relevant document is retrieved*
     - For a single query, AP is *related* to the area under the un-interpolated Precision/Recall curve
  2. Compute the mean AP over the set of queries

Evaluation of IR systems

Ranked retrieval: Average Precision

- MAP = mean AP over the set of queries

$$MAP(Q) = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\left(\frac{1}{m_i}\sum_{k=1}^{m_i} P(\Re_k)\right)$$

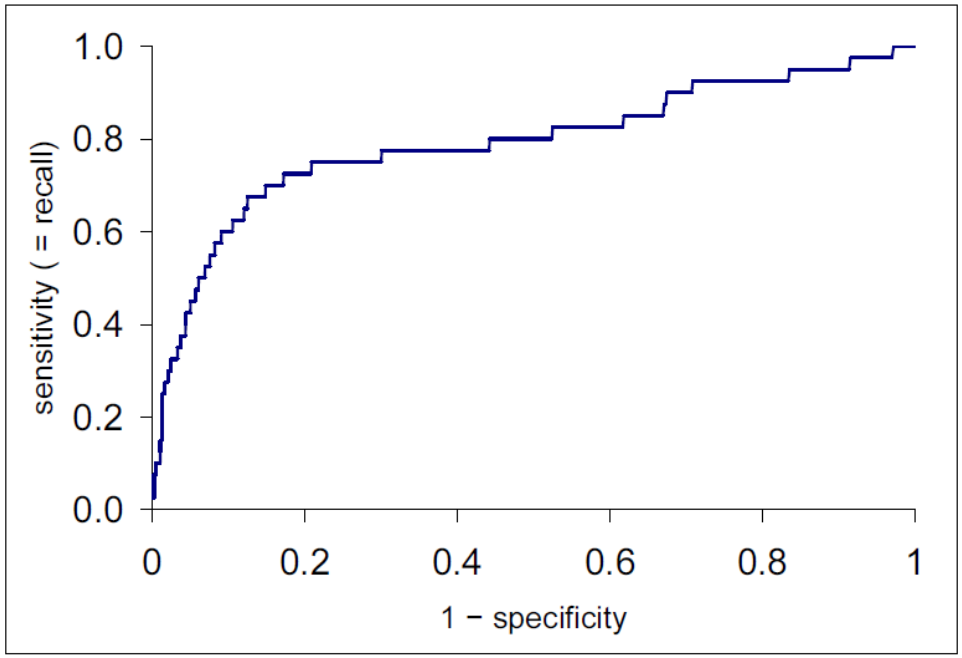
- $\{d_1, \ldots d_{m_i}\}$ documents relevant to query q_i
- $\Re_k$ top-k ranked set of retrieval results

Ranked retrieval: Precision at k, R-precision

- Precision at k
  - Set a fixed value of retrieved results k
  - Compute precision among top-k items
    - pro: does not require any estimate of the size of the set of relevant documents (useful in Web search)
    - con: total number of relevant documents has strong influence on Precision at k
      - e.g. with 8 relevant docs precision at 20 can be at most 0.4
- R-precision
  - Given a relevant set of size Rel
  - Calculate number of relevant documents r in the top-Rel set
    - pros:
      - a perfect system achieves R-precision = 1.0
      - Intuitive meaning: $\frac{r}{Rel}$ = precision at Rel = recall at Rel
    - con: considers only one point on the Precision/Recall curve

Ranked retrieval: Receiver-Operating-Characteristic (ROC)

- True positive rate (*sensitivity*) vs. false positive rate (1 – *specificity*)
- TP rate = *sensitivity* = Recall = $\frac{TP}{TP+FN} = \frac{retrieved\ relevant}{relevant}$
  - fraction of relevant documents that are retrieved
- FP rate = 1 – *specificity* = $\frac{FP}{FP+TN} = \frac{retrieved\ non-relevant}{non-relevant}$
  - fraction of non-relevant documents that are retrieved



Ranked retrieval: example

- An IR system gives the following rankings in response to two queries q_1 and q_2
- The highlighted documents are the ones relevant to the user for a specific query
- Suppose that the whole document collection is shown for each query
  - The total number of relevant and non-relevant documents is known

q_1	q_2
A	C
B	E
F	A
D	D
C	B
E	F

Ranked retrieval: example

- Draw the Precision-Recall curve for each query

q_1

A
B
F
D
C
E

- Query q_1
  - Precision and Recall at 1 $P(1) = \frac{1}{1}$ $\mathrm{R}(1) = \frac{1}{3}$
  - Precision and Recall at 2 $P(2) = \frac{2}{2}$ $\mathrm{R}(2) = \frac{2}{3}$
  - Precision and Recall at 3 $P(3) = \frac{2}{3}$ $\mathrm{R}(3) = \frac{2}{3}$
  - Precision and Recall at 4 $P(4) = \frac{3}{4}$ $\mathrm{R}(4) = \frac{3}{3}$
  - Precision and Recall at 5 $P(5) = \frac{3}{5}$ $\mathrm{R}(5) = \frac{3}{3}$
  - Precision and Recall at 6 $P(6) = \frac{3}{6}$ $\mathrm{R}(6) = \frac{3}{3}$

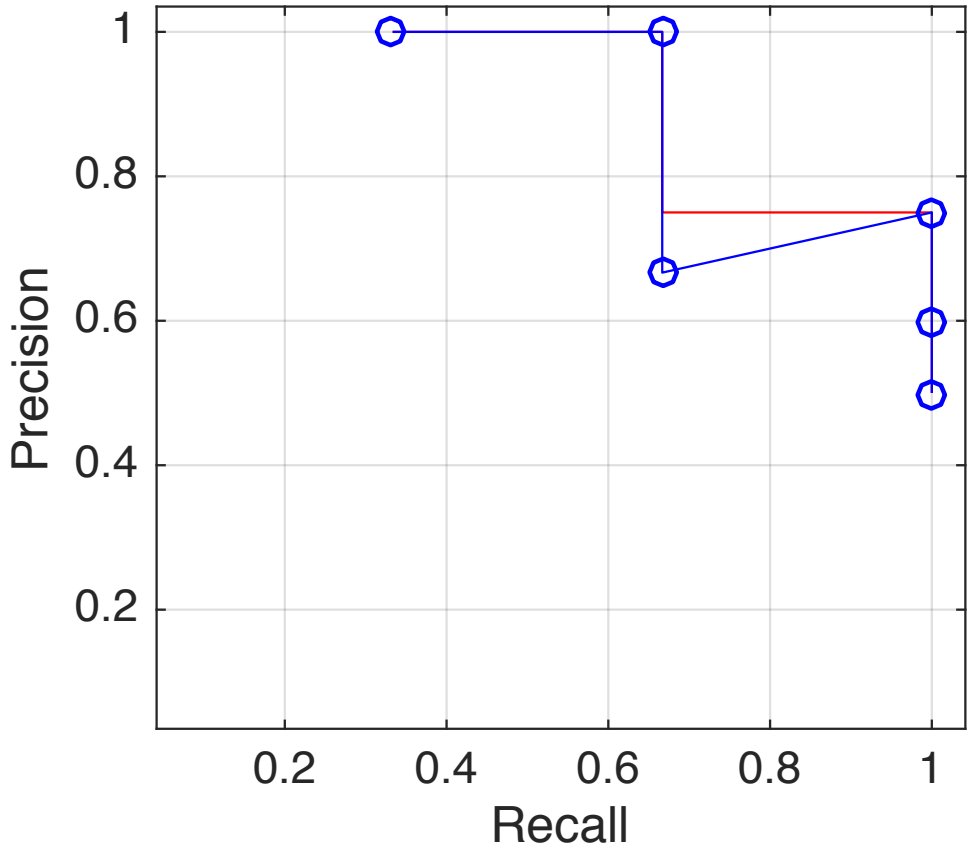
Ranked retrieval: example

- Draw the Precision-Recall curve for each query (continue…)

q_1

A
B
F
D
C
E

- Query q_1



Ranked retrieval: example

- Draw the Precision-Recall curve for each query (continue…)

q_2

C
E
A
D
B
F

- Query q_2
  - Precision and Recall at 1 $P(1) = \frac{0}{1}$ $\mathrm{R}(1) = \frac{0}{2}$
  - Precision and Recall at 2 $P(2) = \frac{1}{2}$ $\mathrm{R}(2) = \frac{1}{2}$
  - Precision and Recall at 3 $P(3) = \frac{1}{3}$ $\mathrm{R}(3) = \frac{1}{2}$
  - Precision and Recall at 4 $P(4) = \frac{1}{4}$ $\mathrm{R}(4) = \frac{1}{2}$
  - Precision and Recall at 5 $P(5) = \frac{2}{5}$ $\mathrm{R}(5) = \frac{2}{2}$
  - Precision and Recall at 6 $P(6) = \frac{2}{6}$ $\mathrm{R}(6) = \frac{2}{2}$

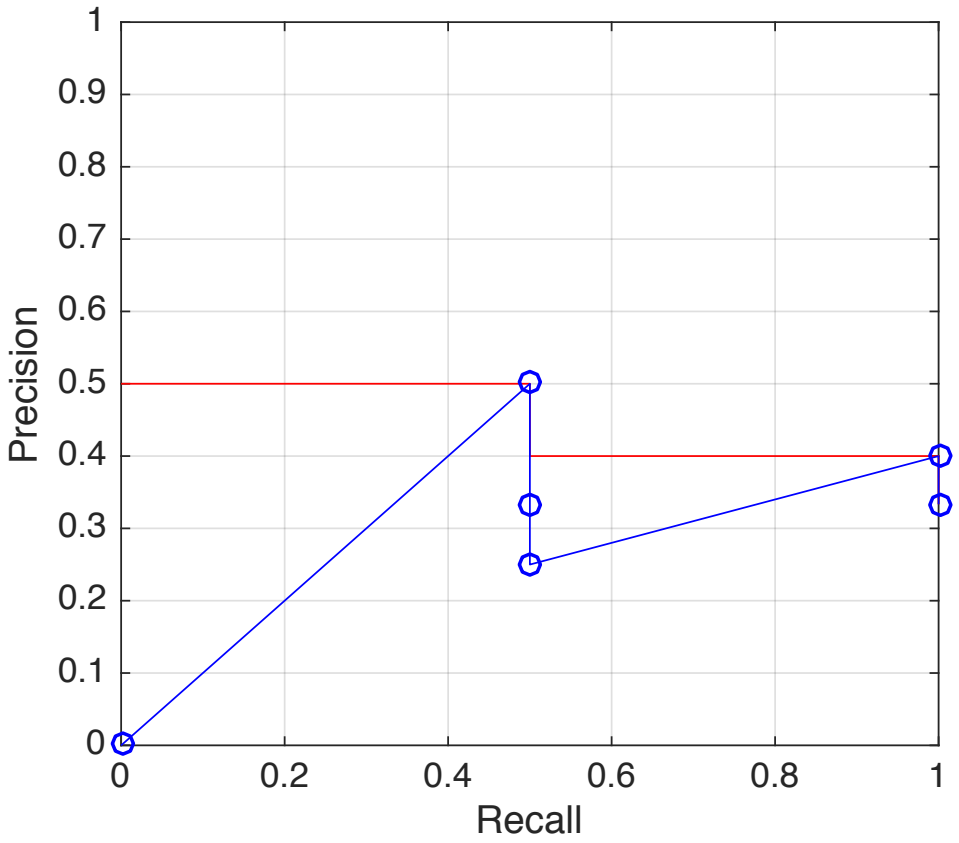
Ranked retrieval: example

- Draw the Precision-Recall curve for each query (continue…)

q_2

C
E
A
D
B
F

- Query q_2



Evaluation of IR systems

Ranked retrieval: example

- Determine the R-precision for each query
  - Query q_1
    - $Rel = 3 \rightarrow$ R-precision = $P(3) = \frac{2}{3}$
  - Query q_2
    - $Rel = 2 \rightarrow$ R-precision = $P(2) = \frac{1}{2}$

- Calculate the Mean Average Precision
  - $AP_1 = \frac{1}{3}\big(P(1) + P(2) + P(4)\big) = \frac{11}{12}$
  - $AP_2 = \frac{1}{2}\big(P(2) + P(5)\big) = \frac{9}{20}$
  - $MAP = \frac{1}{2}(AP_1 + AP_2) = \frac{41}{60}$

Ranked retrieval: example

- Draw the Receiver-Operating-Characteristic for each query

q_1

A
B
F
D
C
E

- Query q_1
  - $TP_{rate}(1) = R(1) = \frac{1}{3}$ $\quad FP_{rate}(1) = \frac{0}{3}$
  - $TP_{rate}(2) = R(2) = \frac{2}{3}$ $\quad FP_{rate}(2) = \frac{0}{3}$
  - $TP_{rate}(3) = R(3) = \frac{2}{3}$ $\quad FP_{rate}(3) = \frac{1}{3}$
  - $TP_{rate}(4) = R(4) = \frac{3}{3}$ $\quad FP_{rate}(4) = \frac{1}{3}$
  - $TP_{rate}(5) = R(5) = \frac{3}{3}$ $\quad FP_{rate}(5) = \frac{2}{3}$
  - $TP_{rate}(6) = R(6) = \frac{3}{3}$ $\quad FP_{rate}(6) = \frac{3}{3}$

Evaluation of IR systems

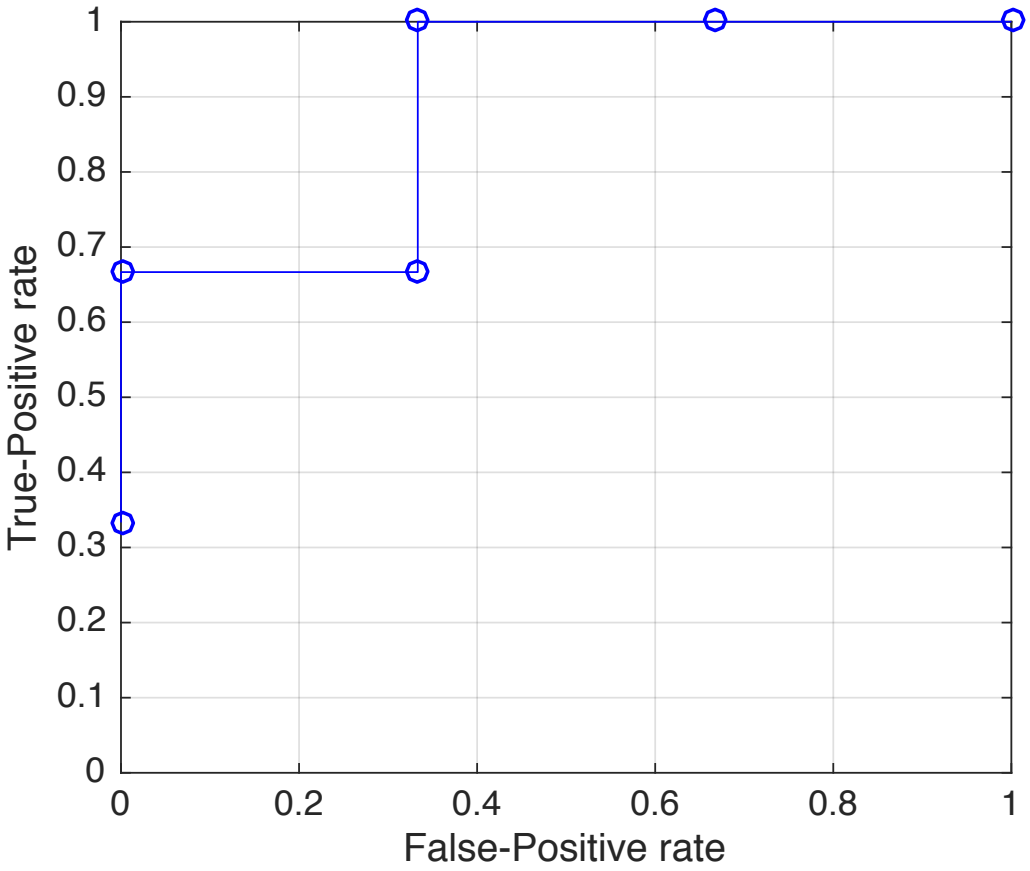
Ranked retrieval: example

- Draw the Receiver-Operating-Characteristic for each query

q_1

A
B
F
D
C
E

- Query q_1



Ranked retrieval: example

- Draw the Receiver-Operating-Characteristic for each query

q_2

C
E
A
D
B
F

- Query q_2
  - $TP_{rate}(1) = R(1) = \frac{0}{2}$ $\quad FP_{rate}(1) = \frac{1}{4}$
  - $TP_{rate}(2) = R(2) = \frac{1}{2}$ $\quad FP_{rate}(2) = \frac{1}{4}$
  - $TP_{rate}(3) = R(3) = \frac{1}{2}$ $\quad FP_{rate}(3) = \frac{2}{4}$
  - $TP_{rate}(4) = R(4) = \frac{1}{2}$ $\quad FP_{rate}(4) = \frac{3}{4}$
  - $TP_{rate}(5) = R(5) = \frac{2}{2}$ $\quad FP_{rate}(5) = \frac{3}{4}$
  - $TP_{rate}(6) = R(6) = \frac{2}{2}$ $\quad FP_{rate}(6) = \frac{4}{4}$

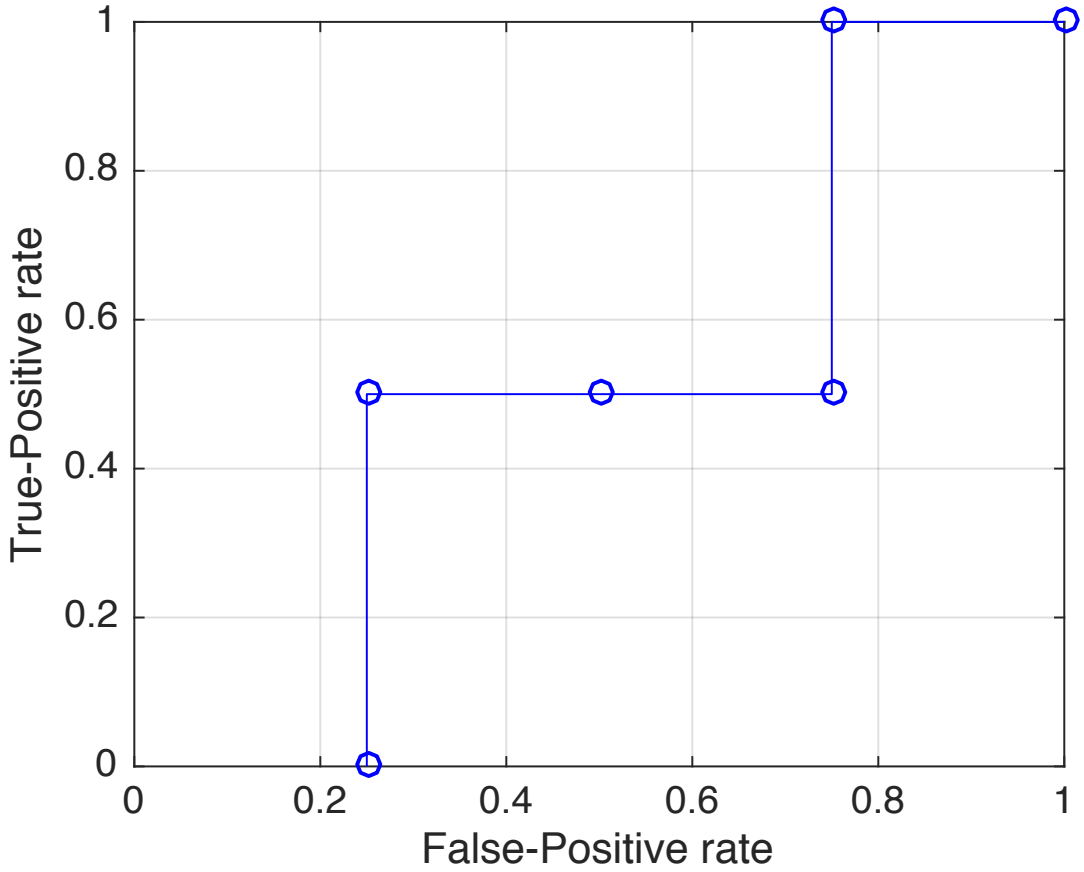
Evaluation of IR systems
Ranked retrieval: example

- Draw the Receiver-Operating-Characteristic for each query

q_2

C
E
A
D
B
F

- Query q_2



References

- [Baeza-Yates and Ribeiro-Nieto, 1999] R. Baeza-Yates and B. Ribeiro-Nieto, “Modern Information Retrieval”, 1999 (http://www.mir2ed.org/)
- [Manning et al., 2008] C.D. Manning, P. Raghavan and H. Schütze, “Introduction to Information Retrieval”, Cambridge University Press, 2008 (http://nlp.stanford.edu/IR-book/)

Monsoon 2022

Relevance Feedback

Relevance Feedback, Pseudo Relevance Feedback and Query Expansion Techniques

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

1st November 2022 (rajendra.2power3.com)

> Topics to be covered

- Recap:
  - Phrase Queries / Proximity Search
  - Spell Correction / Noisy Channel Modelling
  - Index Construction
  - Ranking – Scoring
  - Vector Space Models / Probabilistic IR
  - Information Retrieval Evaluation

- Relevance Feedback
- Pseudo Relevance Feedback
- Query Expansion

  - More topics to come up … Stay tuned …!!

Overview

Recap: Information Retrieval

- **Information Retrieval (IR)** is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

- These days we frequently think first of **web search**, but there are many other cases:
  - E-mail search
  - Searching your laptop
  - Corporate knowledge bases
  - Legal information retrieval
  - and so on . . .

Bag of words model

- We do not consider the order of words in a document.
- John is quicker than Mary and Mary is quicker than John are represented in the same way.
- This is called a bag of words model.
- In a sense, this is a step back: The positional index was able to distinguish these two documents.
- We will look at "recovering" positional information later in this course.
- For now: bag of words model

Measure of Closeness of Vectors

- Measure the closeness between two vectors
- Two texts are semantically related if they share some vocabulary
  - **More Vocabulary they share, the stronger is the relationship**
- This implies that the measure of closeness increases with the number of words matches between two texts
- If matching terms are important then vectors should be considered closer to each other

Modern Vector Space Models

- The length of the sub-vector in dimension - i is used to represent the importance or the weigh of word – i in a text
- Words that are absent in a text get a weight – 0 (zero)
- Apply Vector Inner Product measure between two vectors:
- This vector inner product increases:
  - **# words match between two texts**
  - **Importance of the matching terms**

How can we improve recall in search?

- Main Topic: two ways of improving recall: relevance feedback and query expansion

- As an example consider query q: [aircraft] . . .
- . . . and document d containing "plane", but not containing "aircraft"

- A simple IR system will not return d for q.
- Even if d is the most relevant document for q!

- We want to change this:
  - **Return relevant documents even if there is no term match with the (original) query**

Recall

- Loose definition of recall in this lecture: "increasing the number of relevant documents returned to user"
- This may actually decrease recall on some measures, e.g., when expanding "jaguar" with "panthera"

. . .which eliminates some relevant documents, but increases relevant documents returned on top pages

Options for improving recall

- **Local:** Do a "local", on-demand analysis for a user query
  - Main local method: **relevance feedback**
- **Global:** Do a global analysis once (e.g., of collection) to produce **thesaurus**
  - Use thesaurus for **query expansion**

Google examples for query expansion

- One that works well

  ~flights -flight

- One that doesn't work so well

  ~hospitals -hospital

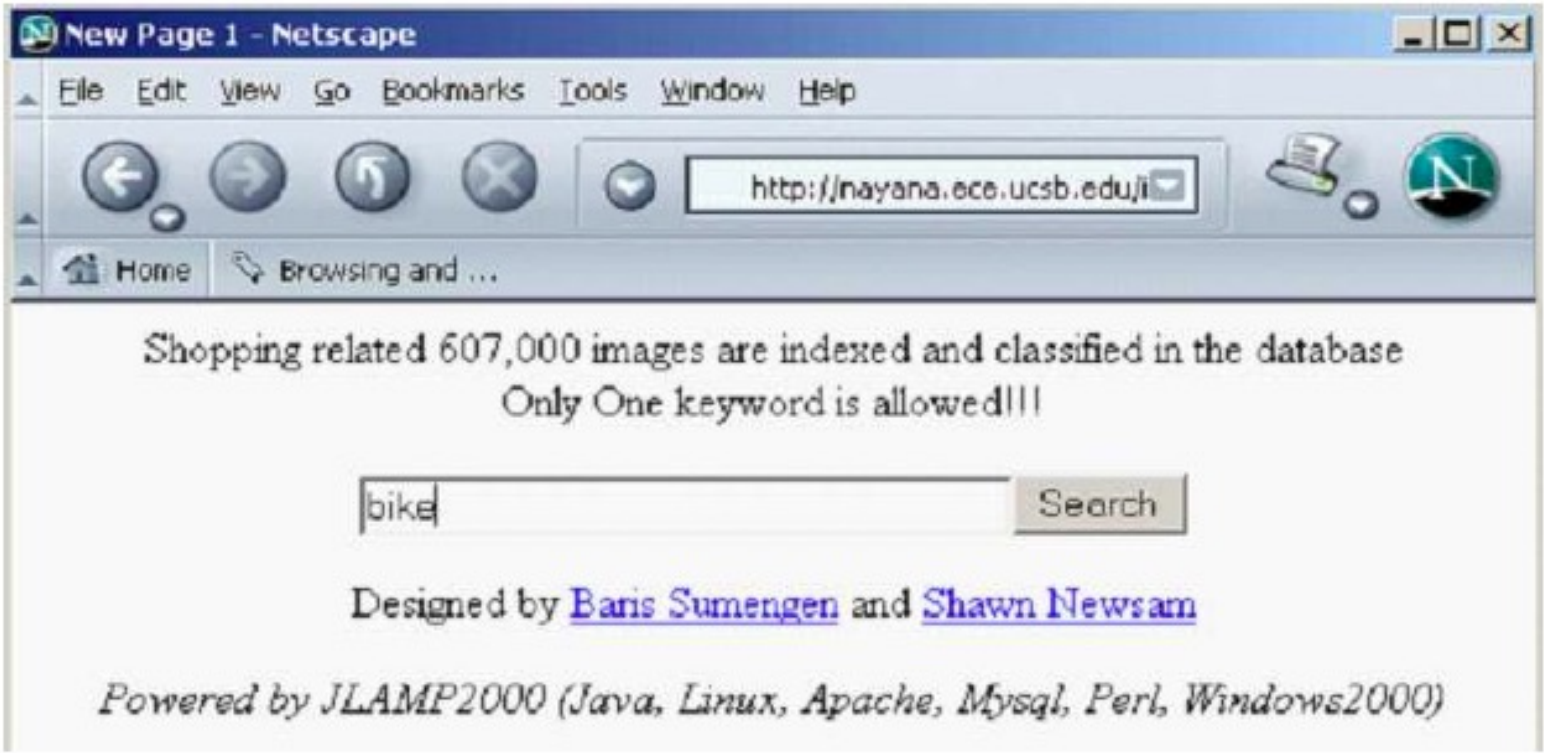
Relevance feedback: Basic idea

- The user issues a (short, simple) query.
- The search engine returns a set of documents.
- User marks some docs as relevant, some as nonrelevant.
- Search engine computes a new representation of the information need. Hope: better than the initial query.
- Search engine runs new query and returns new results.
- New results have (hopefully) better recall.

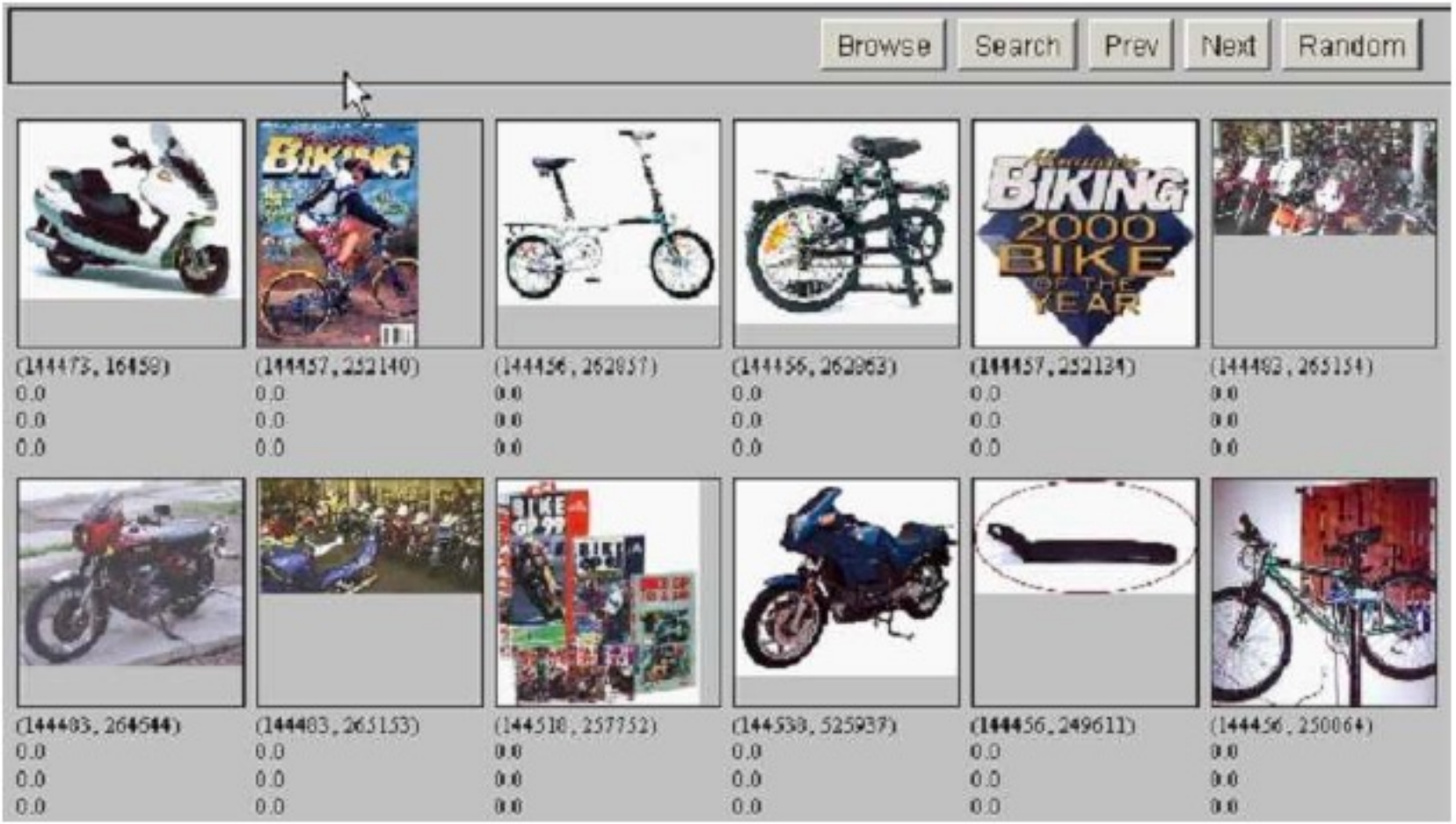
Relevance feedback

- We can iterate this: several rounds of relevance feedback.
- We will use the term **ad hoc retrieval** to refer to regular retrieval without relevance feedback.
- We will now look at three different examples of relevance feedback that highlight different aspects of the process.

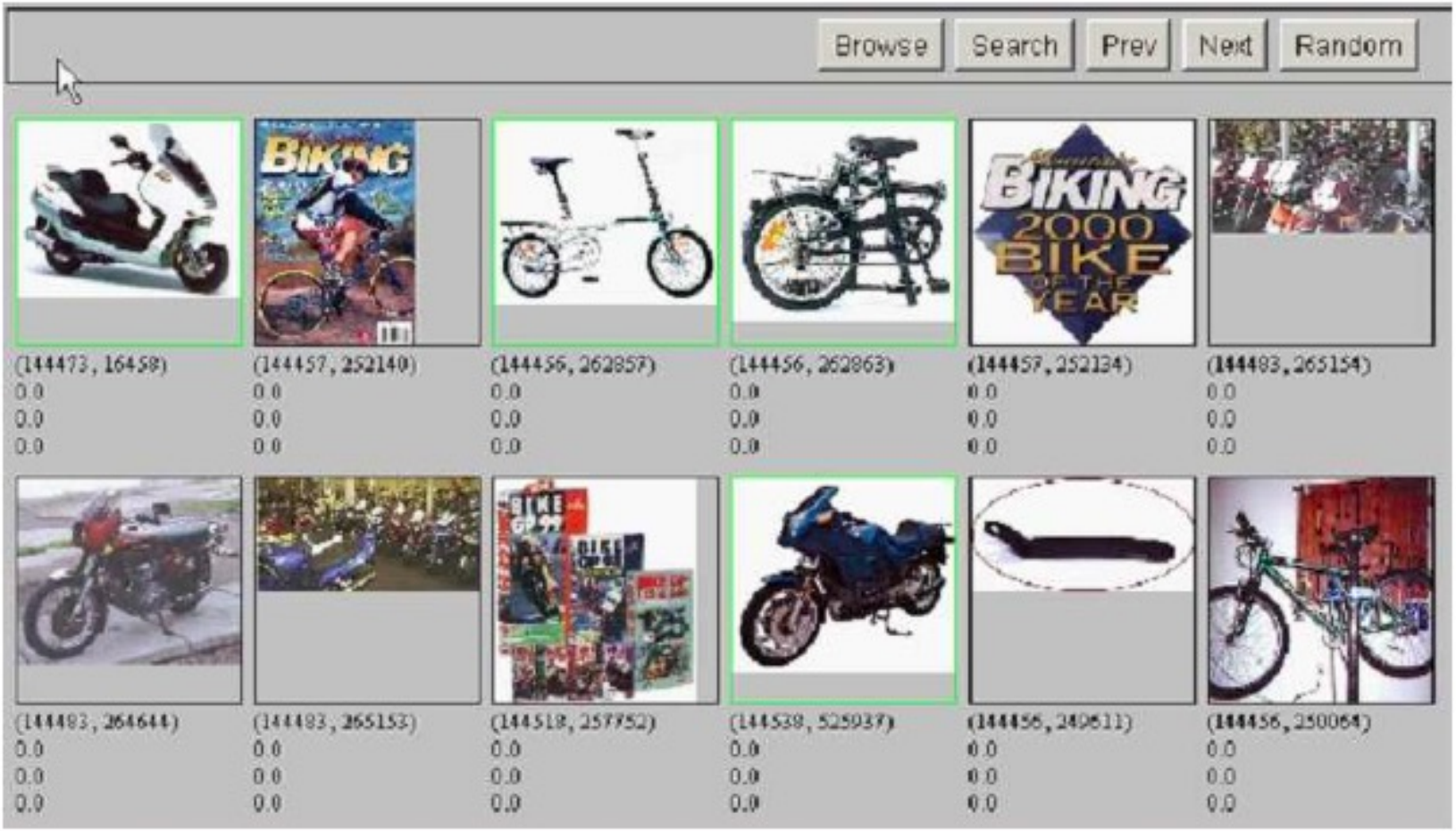
Relevance feedback: Example 1



Results for initial query



User feedback: Select what is relevant



Results after relevance feedback

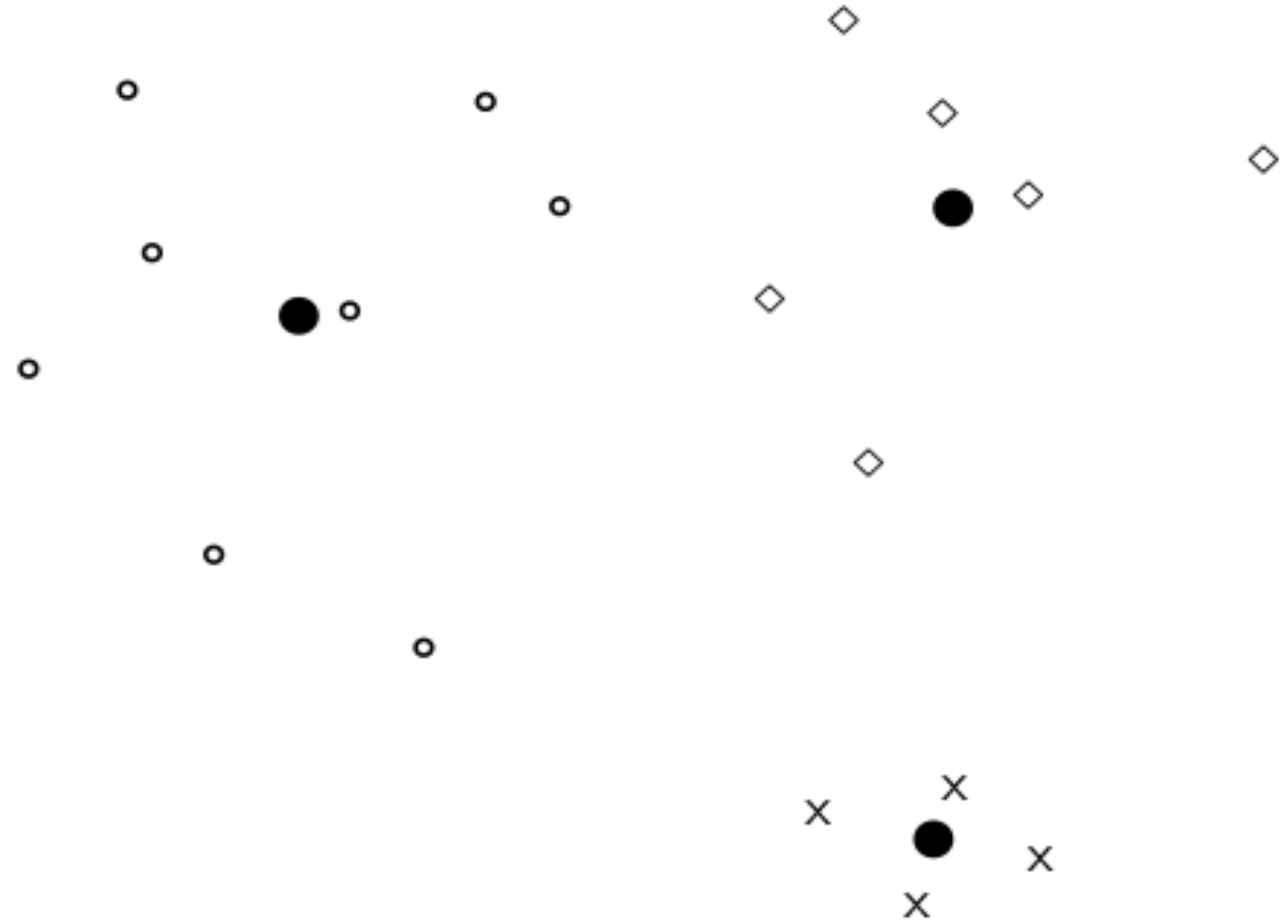
Key concept for Relevance Feedback: Centroid

- The centroid is the center of mass of a set of points.
- Recall that we represent documents as points in a high-dimensional space.
- Thus: we can compute centroids of documents.
- **Definition:**

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

- where D is a set of documents and $\vec{v}(d) = \vec{d}$ is the vector we use to represent document d.

Centroid: Example



Rocchio' algorithm

- The Rocchio' algorithm implements relevance feedback in the vector space model.
- Rocchio' chooses the query $\vec{q}_{opt}$ that maximizes

$$\vec{q}_{opt} = \arg\max_{\vec{q}}[\text{sim}(\vec{q}, \mu(D_r)) - \text{sim}(\vec{q}, \mu(D_{nr}))]$$

- Dr : set of relevant docs; Dnr : set of nonrelevant docs
- Intent: ~qopt is the vector that separates relevant and nonrelevant docs maximally.
- Making some additional assumptions, we can rewrite $\vec{q}_{opt}$ as:

$$\vec{q}_{opt} = \mu(D_r) + [\mu(D_r) - \mu(D_{nr})]$$

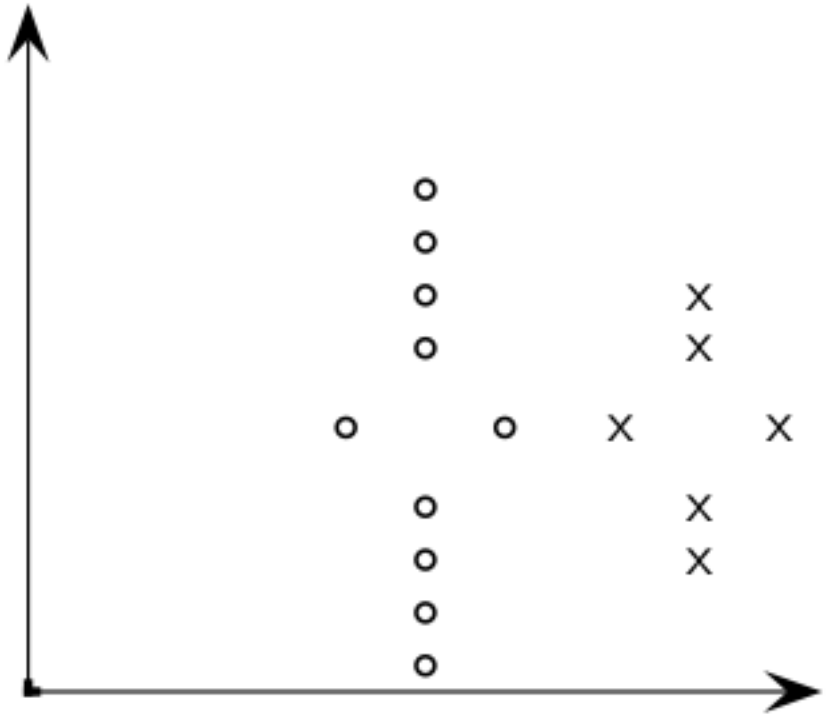
Rocchio' algorithm

- The optimal query vector is:

$$\begin{aligned}\vec{q}_{opt} &= \mu(D_r) + [\mu(D_r) - \mu(D_{nr})] \\ &= \frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r} \vec{d}_j + [\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r} \vec{d}_j - \frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}} \vec{d}_j]\end{aligned}$$

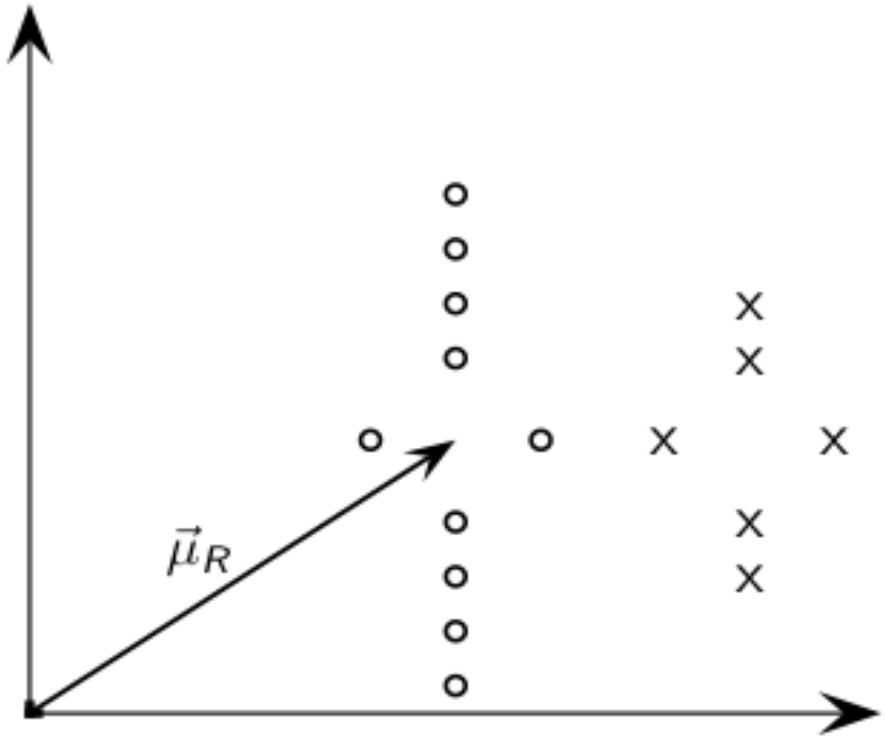
- We move the centroid of the relevant documents by the difference between the two centroids.

Exercise: Compute Rocchio' vector



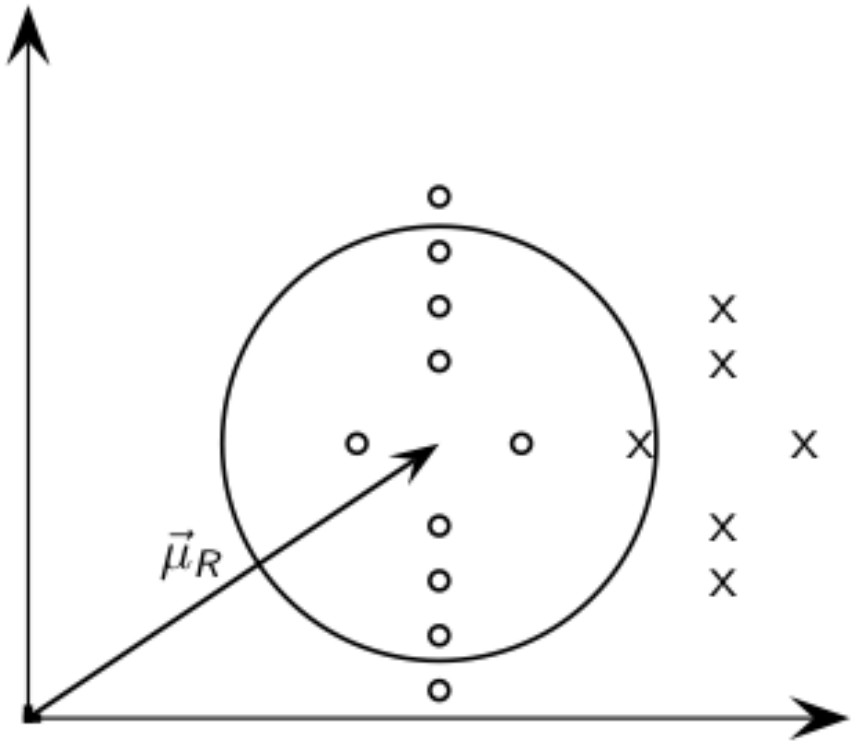
circles: relevant documents, Xs: nonrelevant documents

Rocchio' illustrated



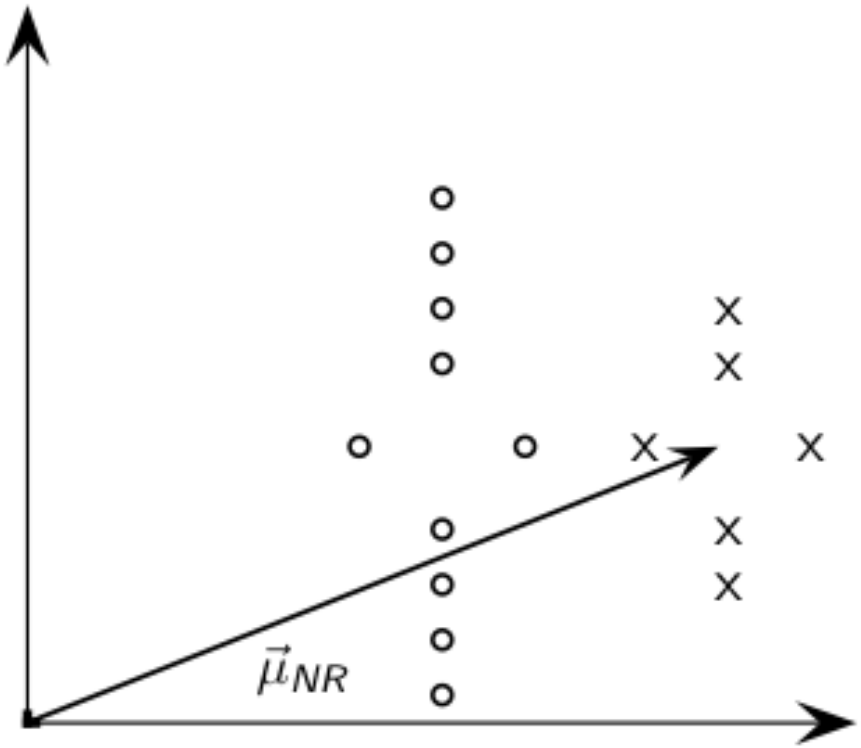
$\vec{\mu}_R$: centroid of relevant documents

Rocchio' illustrated



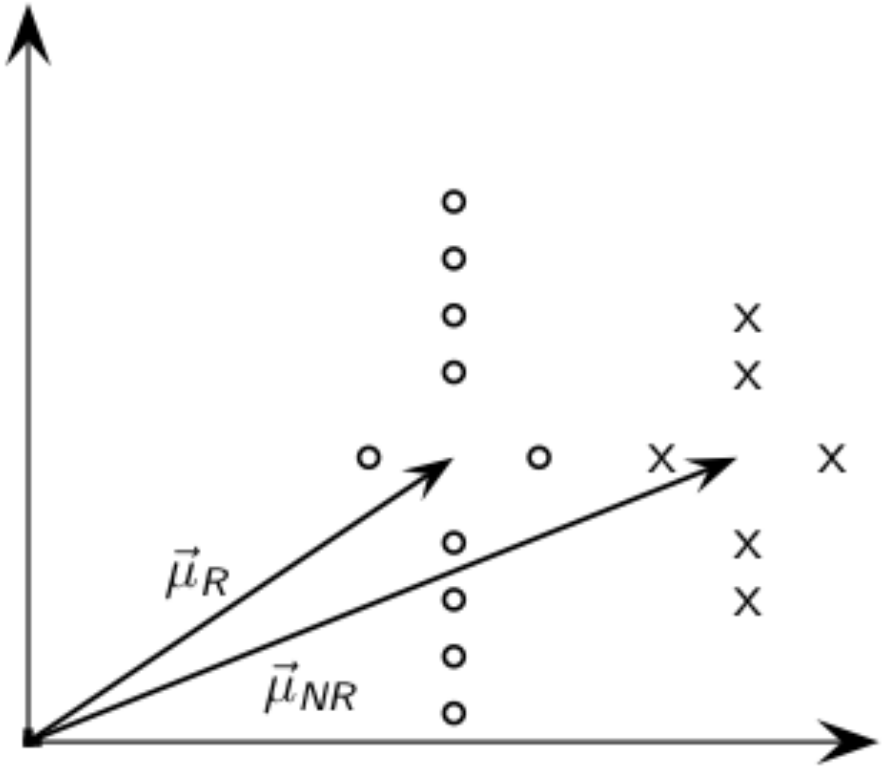
$\vec{\mu}_R$ does not separate relevant / nonrelevant.

Rocchio' illustrated

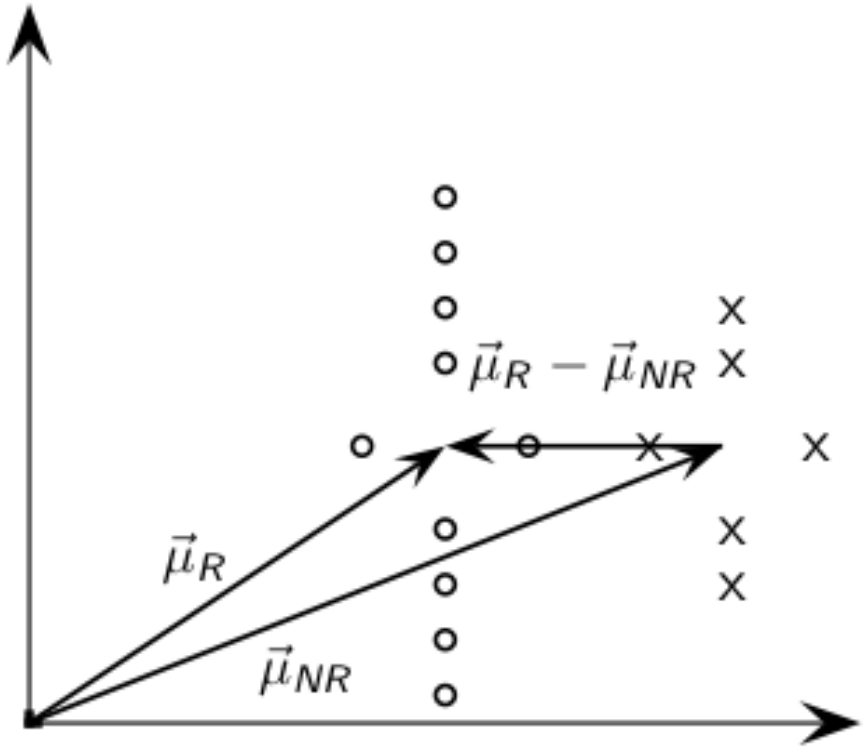


$\vec{\mu}_{NR}$: centroid of nonrelevant documents.

Rocchio' illustrated

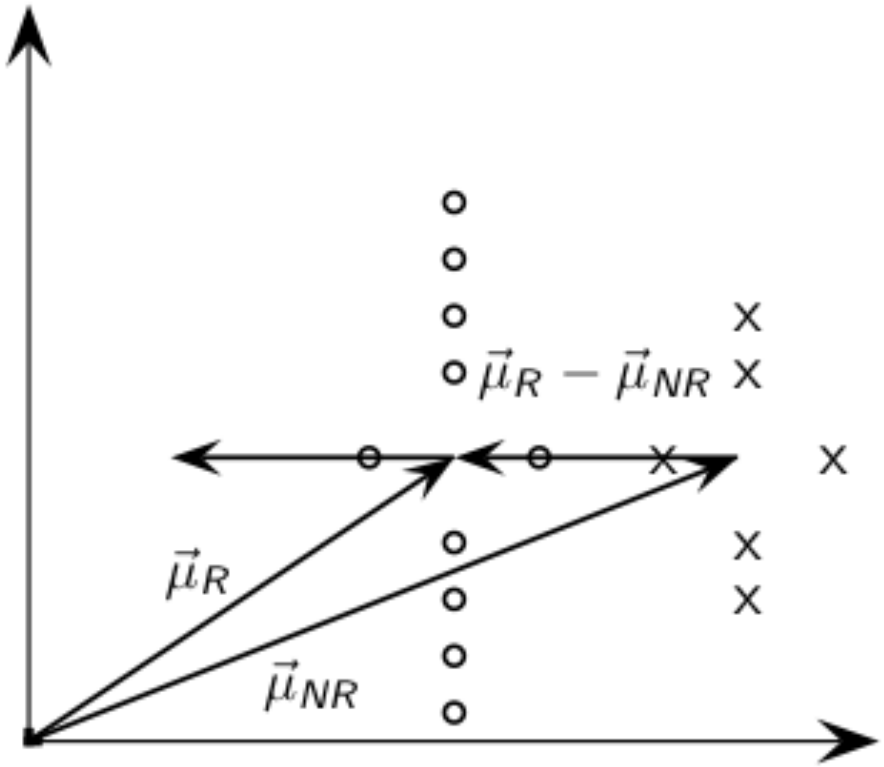


Rocchio' illustrated



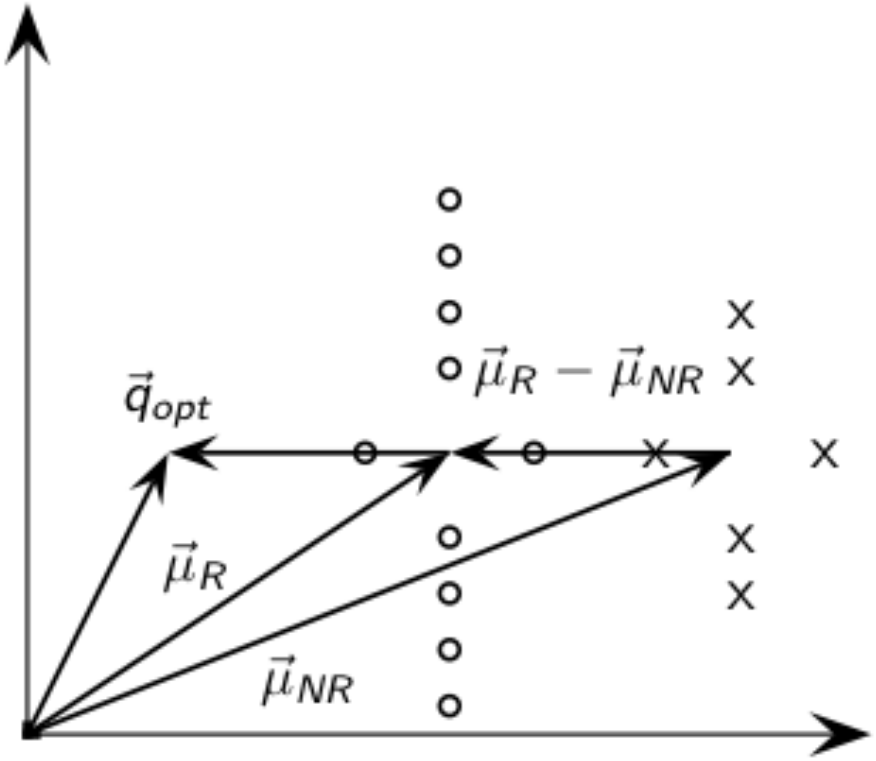
$\vec{\mu}_R$ - $\vec{\mu}_{NR}$: difference vector

Rocchio' illustrated



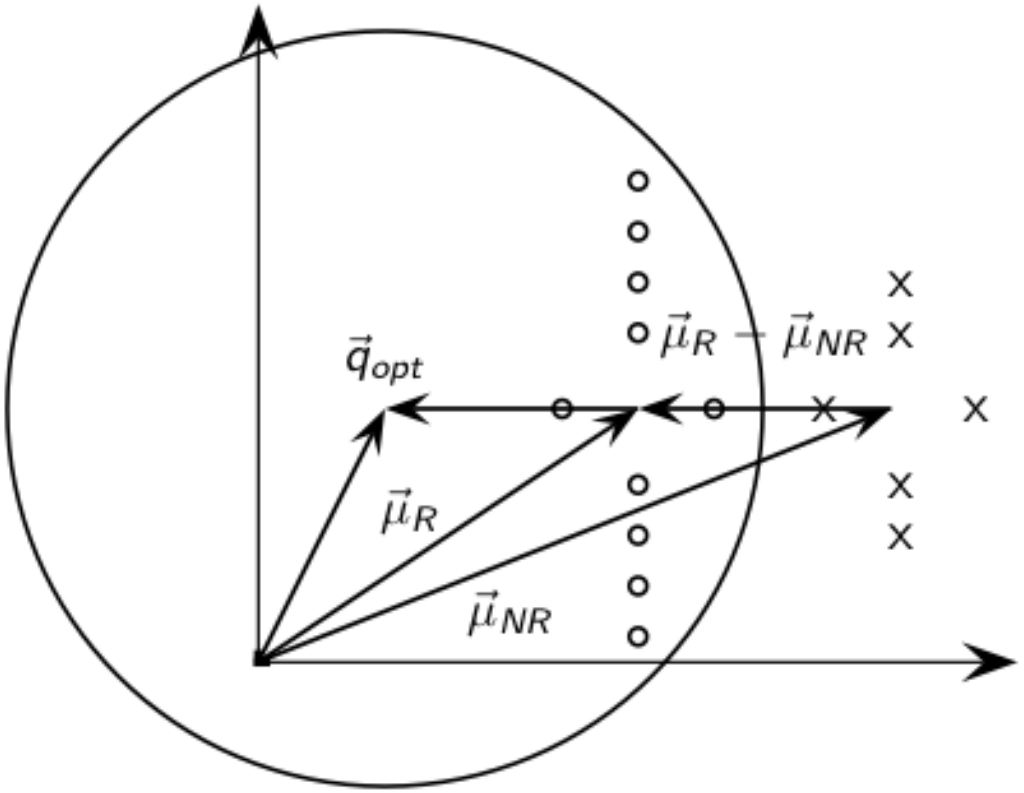
Add difference vector to $\vec{\mu}_R$ ...

Rocchio' illustrated



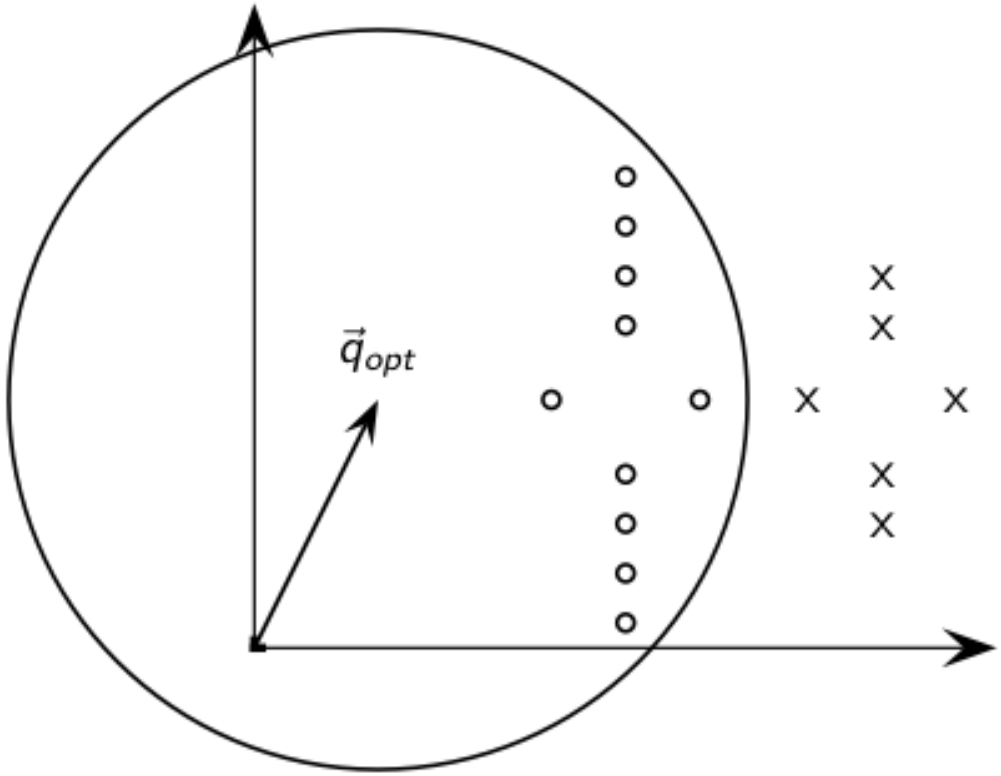
... to get $\vec{q}_{opt}$

Rocchio' illustrated



$\vec{q}_{opt}$: separates relevant / nonrelevant perfectly.

Rocchio' illustrated



$\vec{q}_{opt}$: separates relevant / nonrelevant perfectly

Terminology

- We use the name Rocchio' for the theoretically better motivated original version of Rocchio.
- The implementation that is actually used in most cases is the SMART implementation – we use the name Rocchio (without prime) for that.

Rocchio 1971 algorithm (SMART)

Used in practice:

$$\begin{aligned}\vec{q}_m &= \alpha\vec{q}_0 + \beta\mu(D_r) - \gamma\mu(D_{nr}) \\ &= \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j \in D_{nr}}\vec{d}_j\end{aligned}$$

- q_m: modified query vector; q_0: original query vector; D_r and D_{nr}: sets of known relevant and nonrelevant documents respectively; α, β, and γ: weights
- New query moves towards relevant documents and away from nonrelevant documents.
- Tradeoff α vs. β/γ: If we have a lot of judged documents, we want a higher β/γ.
- Set negative term weights to 0.
- "Negative weight" for a term doesn't make sense in the vector space model.

Positive vs. negative relevance feedback

- Positive feedback is more valuable than negative feedback.
- For example, set $\beta = 0.75$, $\gamma = 0.25$ to give higher weight to positive feedback.
- Many systems only allow positive feedback.

Relevance feedback: Assumptions

- When can relevance feedback enhance recall?
- **Assumption A1:** The user knows the terms in the collection well enough for an initial query.
- **Assumption A2:** Relevant documents contain similar terms (so I can "hop" from one relevant document to a different one when giving relevance feedback).

Violation of A1

- Assumption A1: The user knows the terms in the collection well enough for an initial query.
- Violation: Mismatch of searcher's vocabulary and collection vocabulary
- Example: cosmonaut / astronaut

Violation of A2

- Assumption A2: Relevant documents are similar.
- Example for violation: [contradictory government policies]

- Several unrelated "prototypes"
  - Subsidies for tobacco farmers vs. anti-smoking campaigns
  - Aid for developing countries vs. high tariffs on imports from developing countries

- Relevance feedback on tobacco docs will not help with finding docs on developing countries.

Relevance feedback: Evaluation

- Pick one of the evaluation measures from last lecture, e.g., precision in top 10: P@10
- Compute P@10 for original query q_0
- Compute P@10 for modified relevance feedback query q_1
- In most cases: q_1 is spectacularly better than q_0!
- Is this a fair evaluation?

Relevance feedback: Evaluation

- Fair evaluation must be on "residual" collection: docs not yet judged by user.
- Studies have shown that relevance feedback is successful when evaluated this way.
- Empirically, one round of relevance feedback is often very useful. Two rounds are marginally useful.

Evaluation: Caveat

- True evaluation of usefulness must compare to other methods taking the **same amount of time**.
- Alternative to relevance feedback: User revises and resubmits query.
- Users may prefer revision/resubmission to having to judge relevance of documents.
- There is no clear evidence that relevance feedback is the "best use" of the user's time.

- **Exercise:**
  - Do search engines use relevance feedback?
  - Why?

Relevance feedback: Problems

- Relevance feedback is expensive.
  - Relevance feedback creates long modified queries.
  - Long queries are expensive to process.
- Users are reluctant to provide explicit feedback.
- It's often hard to understand why a particular document was retrieved after applying relevance feedback.
- The search engine Excite had full relevance feedback at one point, but abandoned it later.

Pseudo-relevance feedback

- Pseudo-relevance feedback automates the "manual" part of true relevance feedback.
- Pseudo-relevance algorithm:
  - Retrieve a ranked list of hits for the user's query
  - **Assume that the top k documents are relevant.**
  - Do relevance feedback (e.g., Rocchio)
- Works very well on average
- But can go horribly wrong for some queries.
- Several iterations can cause query drift.

Pseudo-relevance feedback at TREC4

- Cornell SMART System
- Results show number of relevant documents out of top 100 for 50 queries (so total number of documents is 5000):

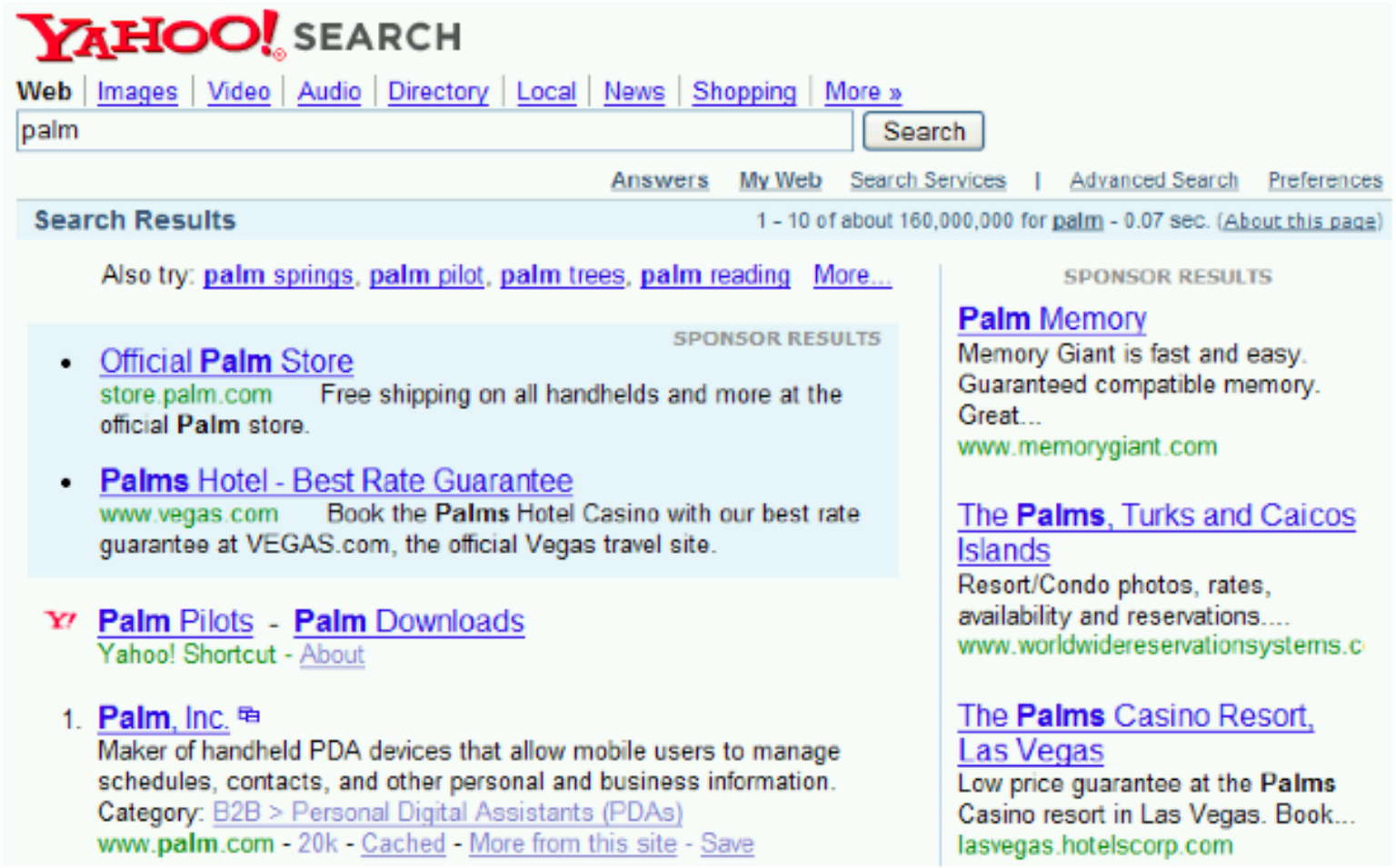
method	number of relevant documents
lnc.ltc	3210
lnc.ltc-PsRF	3634
Lnu.ltu	3709
Lnu.ltu-PsRF	4350

- Results contrast two length normalization schemes (L vs. l) and pseudo-relevance feedback (PsRF).
- The pseudo-relevance feedback method used added only 20 terms to the query. (Rocchio will add many more.)
- This demonstrates that pseudo-relevance feedback is effective on average.

Query expansion

- Query expansion is another method for **increasing recall**.
- We use "global query expansion" to refer to "global methods for query reformulation".
- In global query expansion, the query is modified based on some global resource, i.e. a resource that is not query-dependent.
- Main information we use: (near-)synonymy
- A publication or database that collects (near-)synonyms is called a **thesaurus**.
- We will look at two types of thesauri: manually created and automatically created.

Query expansion: Example



Types of user feedback

- User gives feedback on **documents.**
  - More common in relevance feedback
- User gives feedback on **words** or **phrases**.
  - More common in query expansion

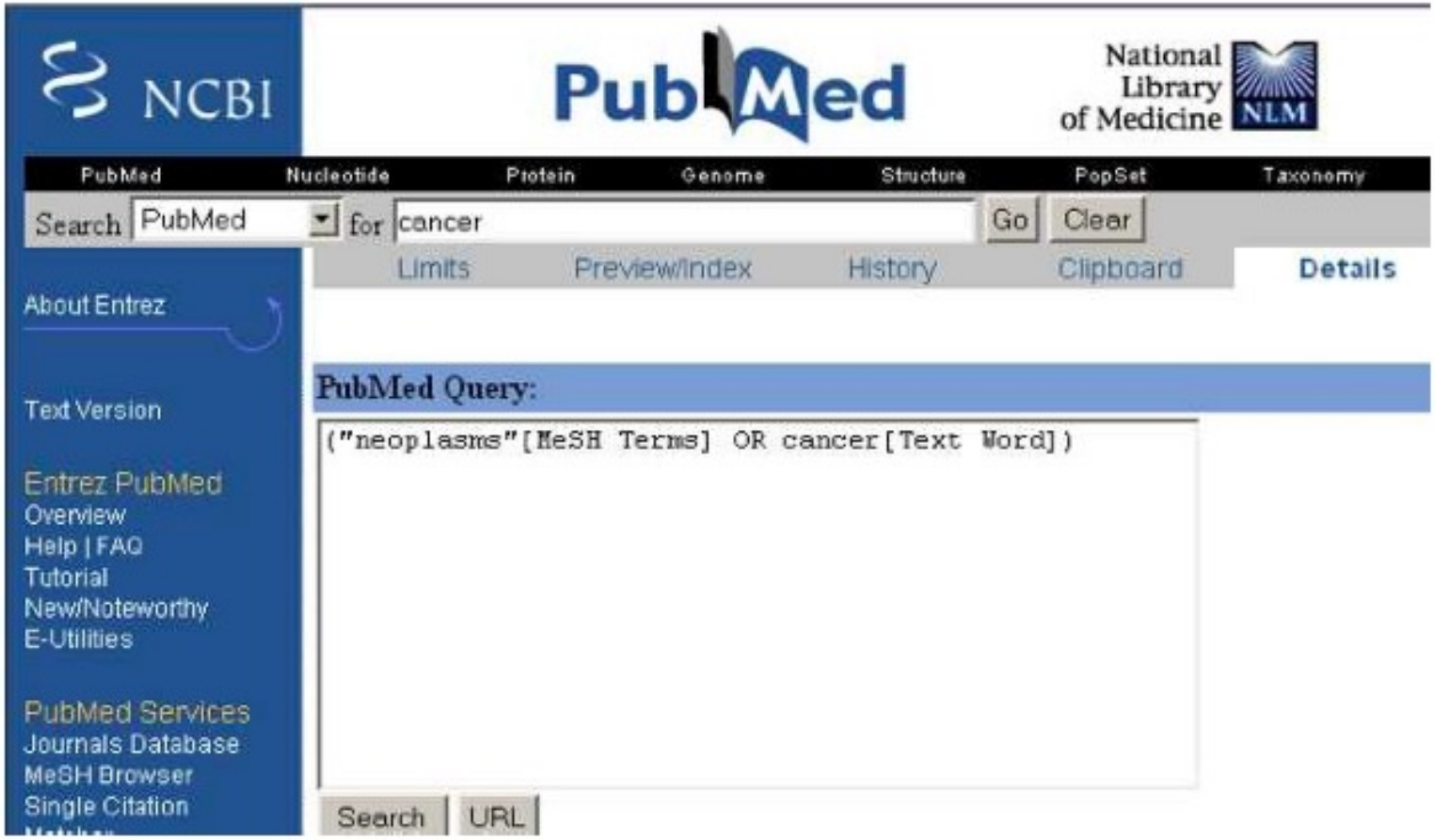
Types of query expansion

- Manual thesaurus (maintained by editors, e.g., PubMed)
- Automatically derived thesaurus (e.g., based on co-occurrence statistics)
- Query-equivalence based on query log mining (common on the web as in the "palm" example)

Thesaurus-based query expansion

- For each term t in the query, expand the query with words the thesaurus lists as semantically related with t.
- Example from earlier: HOSPITAL → MEDICAL
- Generally increases recall
- May significantly decrease precision, particularly with ambiguous terms
  - INTEREST RATE → INTEREST RATE FASCINATE
- Widely used in specialized search engines for science and engineering
- It's very expensive to create a manual thesaurus and to maintain it over time.
- A manual thesaurus has an effect roughly equivalent to annotation with a **controlled vocabulary**.

Example for manual thesaurus: PubMed



Automatic thesaurus generation

- Attempt to generate a thesaurus automatically by analyzing the distribution of words in documents
- Fundamental notion: similarity between two words
- **Definition 1:** Two words are **similar if they co-occur with similar words**.
  - "car" ≈ "motorcycle" because both occur with "road", "gas" and "license", so they must be similar.
- **Definition 2:** Two words are **similar if they occur in a given grammatical relation with the same words**.
  - You can harvest, peel, eat, prepare, etc. apples and pears, so apples and pears must be similar.
- Co-occurrence is more robust, grammatical relations are more accurate.

Co-occurence-based thesaurus: Examples

Word	Nearest neighbors
absolutely	absurd whatsoever totally exactly nothing
bottomed	dip copper drops topped slide trimmed
captivating	shimmer stunningly superbly plucky witty
doghouse	dog porch crawling beside downstairs
makeup	repellent lotion glossy sunscreen skin gel
mediating	reconciliation negotiate case conciliation
keeping	hoping bring wiping could some would
lithographs	drawings Picasso Dali sculptures Gauguin
pathogens	toxins bacteria organisms bacterial parasite
senses	grasp psyche truly clumsy naive innate

Query expansion at search engines

- Main source of query expansion at search engines: query logs

- **Example 1:** After issuing the query [herbs], users frequently search for [herbal remedies].

  → "herbal remedies" is potential expansion of "herb".

- **Example 2:** Users searching for [flower pix] frequently click on the URL photobucket.com/flower. Users searching for [flower clipart] frequently click on the same URL.

  → "flower clipart" and "flower pix" are potential expansions of each other.

Exercise – Try Yourself

- Consider a collection of n documents
- Let n be sufficiently large (at least 100 docs)
  - You can take our dataset
  - Sports News Dataset (ths-181-dataset.zip)

- **Find Two Annotators:**
  - The most frequency words and
  - The least frequent words
  - Form k (=10) queries each with exactly 3-words taken from above lists (at least one from each)
  - Compute Similarity between each query and and documents

Summary

In this class, we focused on:

(a) Recap: Positional Indexes

i. Wild card Queries
ii. Spelling Correction
iii. Noisy Channel modelling for Spell Correction

(b) Various Indexing Approaches

i. Block Sort based Indexing Approach
ii. Single Pass In Memory Indexing Approach
iii. Distributed Indexing using Map Reduce
iv. Examples

Summary

In this class, we focused on:

(a) Recap: Positional Indexes

i. Wild card Queries

ii. Spelling Correction

iii. Noisy Channel modelling for Spell Correction

(b) Various Term Weighting

i. Jaccard

ii. Term Frequency

iii. Term Weighting

iv. Vector Space Models

Many more to come up …

Acknowledgements

Thanks to ALL RESEARCHERS:

- Modern Information Retrieval Baeza-Yates and Ribeiro-Neto, Addison Wesley, 1999.
- **Introduction to Information Retrieval Manning, Raghavan and Schutze, Cambridge University Press, 2008.**
- Search Engines Information Retrieval in Practice W. Bruce Croft, D. Metzler, T. Strohman, Pearson, 2009.
- Information Retrieval Implementing and Evaluating Search Engines Stefan Büttcher, Charles L. A. Clarke and Gordon V. Cormack, MIT Press, 2010.
- Many Authors who contributed to SIGIR / WWW / KDD / ECIR / CIKM / WSDM and other top tier conferences
- **Prof. Mandar Mitra, Indian Statistical Institute, Kolkatata (https://www.isical.ac.in/~mandar/)**

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may ask for one-to-one meeting

Best Approach

- You may leave me an email any time
  (email is the best way to reach me faster)

Questions
It's Your Time

Contact Information

Dr. Rajendra Prasath
IIIT Sri City, Chittoor

Monsoon 2022

Web Crawling

- Including Distributed Crawling and (Near) Duplicate Elimination

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

4th November 2022 (rajendra.2power3.com)

> Topics to be covered

- Recap:
  - Phrase Queries / Proximity Search
  - Spell Correction / Noisy Channel Modelling
  - Index Construction / Ranking – Scoring
  - Vector Space Models / Probabilistic IR
  - Information Retrieval Evaluation
  - Relevance Feedback / PRF / Query Expansion

- Web Crawling
  - Basics of Web Crawling
  - Basic Architecture
  - URL Frontier / Fetcher and Parser
  - (Near) Duplicate Eliminations

    - More topics to come up … Stay tuned …!!

Recap: Information Retrieval

- **Information Retrieval (IR)** is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

- These days we frequently think first of **web search**, but there are many other cases:
  - E-mail search
  - Searching your laptop
  - Corporate knowledge bases
  - Legal information retrieval
  - and so on . . .

Recap: Vector Space Models

- The length of the sub-vector in dimension - i is used to represent the importance or the weigh of word – i in a text
- Words that are absent in a text get a weight – 0 (zero)
- Apply Vector Inner Product measure between two vectors:
- This vector inner product increases:
  - **# words match between two texts**
  - **Importance of the matching terms**

Recap: How can we improve recall in search?

- Main Topic: two ways of improving recall: relevance feedback and query expansion
- As an example consider query q: [aircraft] . . .
- . . . and document d containing "plane", but not containing "aircraft"
- A simple IR system will not return d for q.
- Even if d is the most relevant document for q!
- We want to change this:
  - **Return relevant documents even if there is no term match with the (original) query**

Recap: Relevance Feedback (RF)

- The user issues a (short, simple) query.
- The search engine returns a set of documents.
- User marks some docs as relevant, some as nonrelevant.
- Search engine computes a new representation of the information need. Hope: better than the initial query.
- Search engine runs new query and returns new results.
- New results have (hopefully) better recall.

Recap: Pseudo Relevance Feedback (PRF)

- Pseudo-relevance feedback automates the "manual" part of true relevance feedback.

- Pseudo-relevance algorithm:
  - Retrieve a ranked list of hits for the user's query
  - **Assume that the top k documents are relevant.**
  - Do relevance feedback (e.g., Rocchio)

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

Recap: Query expansion

- Query expansion is another method for **increasing recall**.
- We use "global query expansion" to refer to "global methods for query reformulation".
- In global query expansion, the query is modified based on some global resource, i.e. a resource that is not query-dependent.

- Main information we use: (near-)synonymy
- A publication or database that collects (near-)synonyms is called a **thesaurus**.
- We will look at two types of thesauri: manually created and automatically created.

Recap: XML Retrieval

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

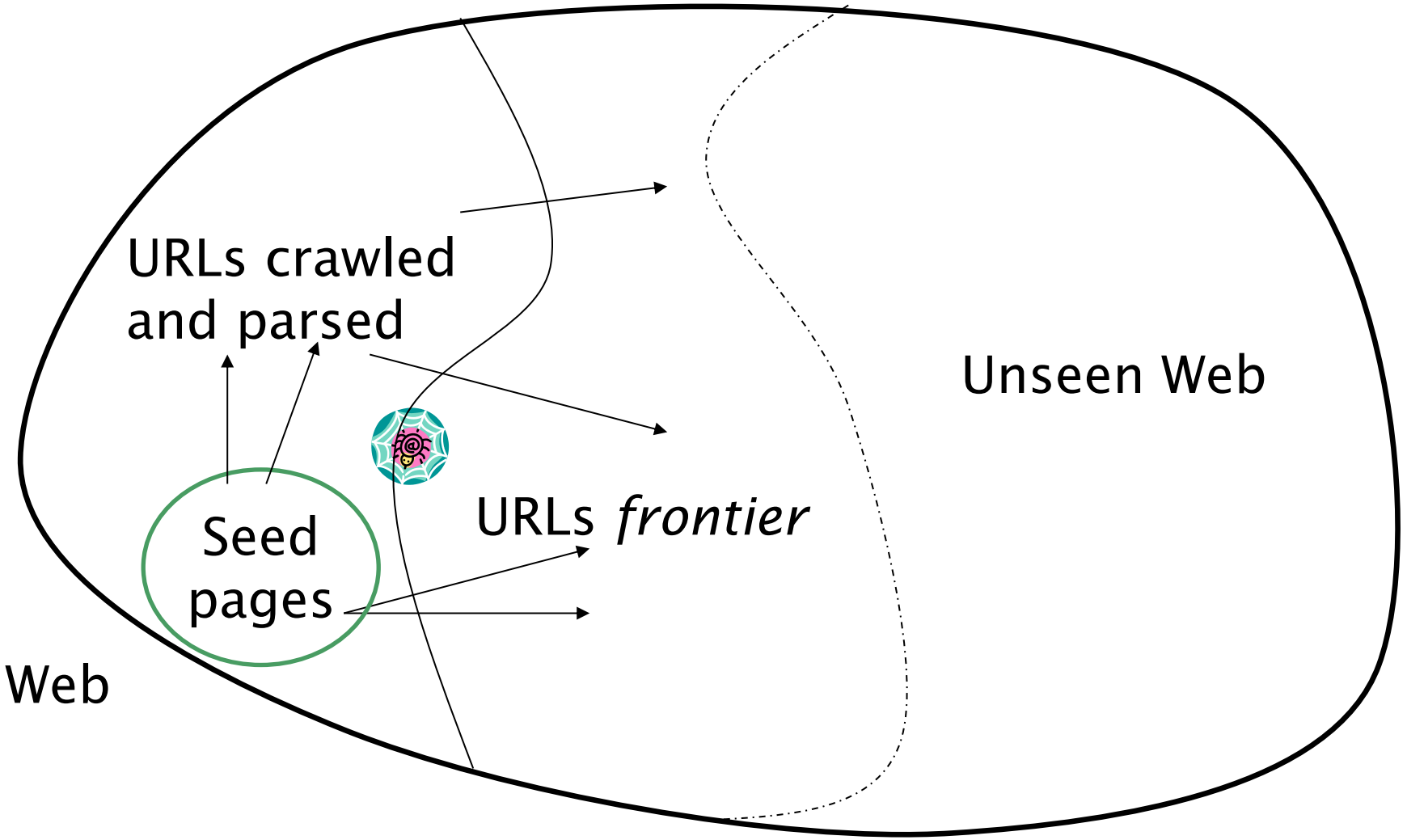
Web Crawling

- Basics of Web Crawling
- Challenges in Crawling
- Basic Crawling Architecture
- URL Frontiers
- Duplicate URL Elimination
- Back Queues

Basic crawler operation

- Begin with known "seed" URLs
- Fetch and parse them
  - **Extract URLs they point to**
  - **Place the extracted URLs on a queue**
- Fetch each URL on the queue and repeat

Crawling picture



Simple picture – complications

- Web crawling isn't feasible with one machine
  - All of the above steps distributed
- Malicious pages
  - Spam pages
  - Spider traps – incl dynamically generated
- Even non-malicious pages pose challenges
  - Latency/bandwidth to remote servers vary
  - Webmasters' stipulations
    - How "deep" should you crawl a site's URL hierarchy?
  - Site mirrors and duplicate pages
- Politeness – don't hit a server too often

What any crawler must do?

- **Be Robust:** Be immune to spider traps and other malicious behavior from web servers
- **Be Polite:** Respect implicit and explicit politeness considerations

Explicit and implicit politeness

- Explicit politeness: specifications from webmasters on what portions of site can be crawled
  - robots.txt

- Implicit politeness: even with no specification, avoid hitting any site too often

Robots.txt

- Protocol for giving spiders ("robots") limited access to a website, originally from 1994
  - www.robotstxt.org/robotstxt.html

- Website announces its request on what can(not) be crawled
  - For a server, create a file `/robots.txt`
  - This file specifies access restrictions

Robots.txt example

- No robot should visit any URL starting with "/yoursite/temp/", except the robot called “searchengine":

```
User-agent: *
Disallow: /yoursite/temp/

User-agent: searchengine
Disallow:
```

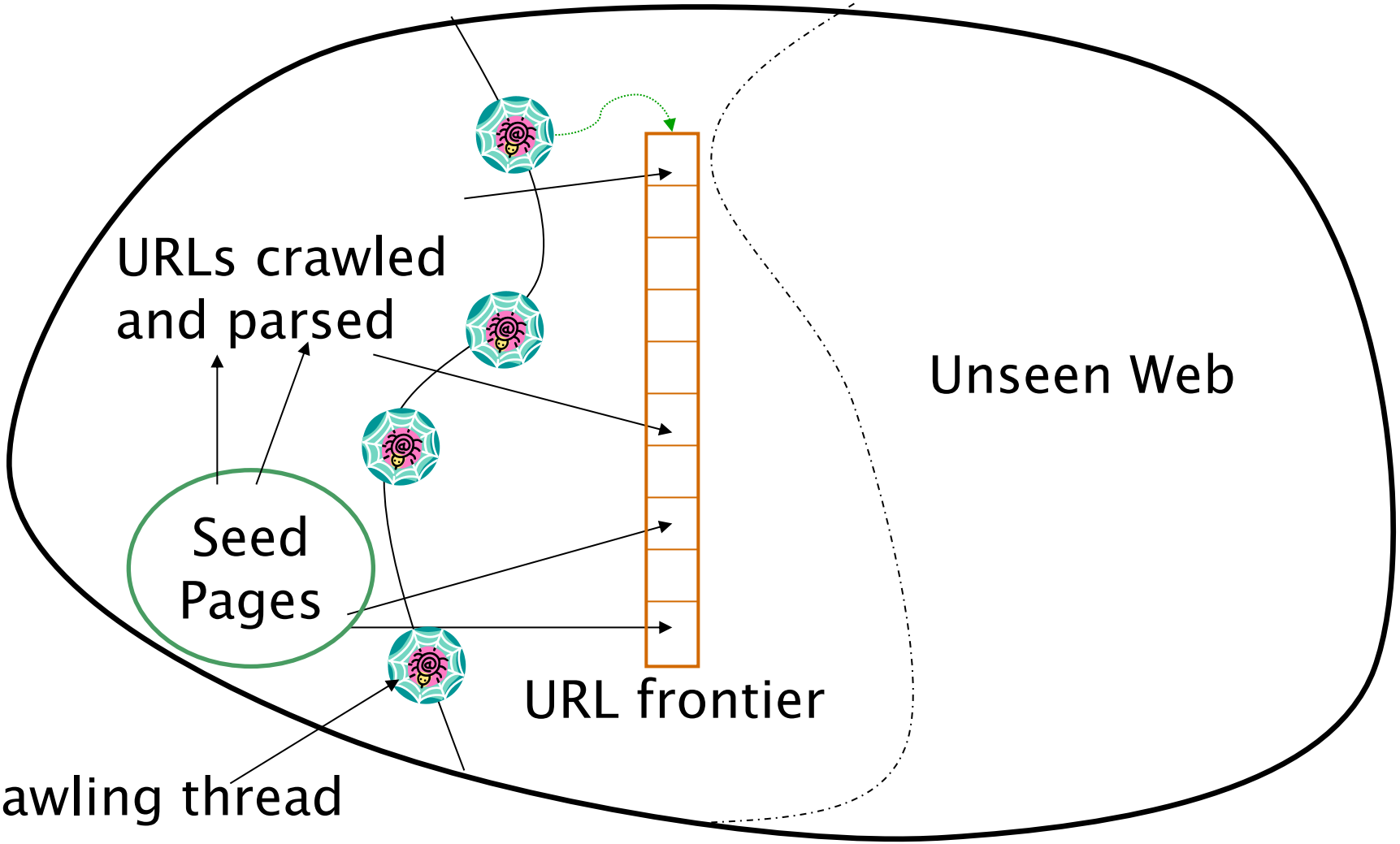
What any crawler should do?

- Be capable of distributed operation: designed to run on multiple distributed machines
- Be scalable: designed to increase the crawl rate by adding more machines
- Performance/efficiency: permit full use of available processing and network resources

What any crawler should do?

- Fetch pages of "higher quality" first
- Continuous operation: Continue fetching fresh copies of a previously fetched page
- Extensible: Adapt to new data formats, protocols

Updated crawling picture



URL frontier

- Can include multiple pages from the same host
- Must avoid trying to fetch them all at the same time
- Must try to keep all crawling threads busy

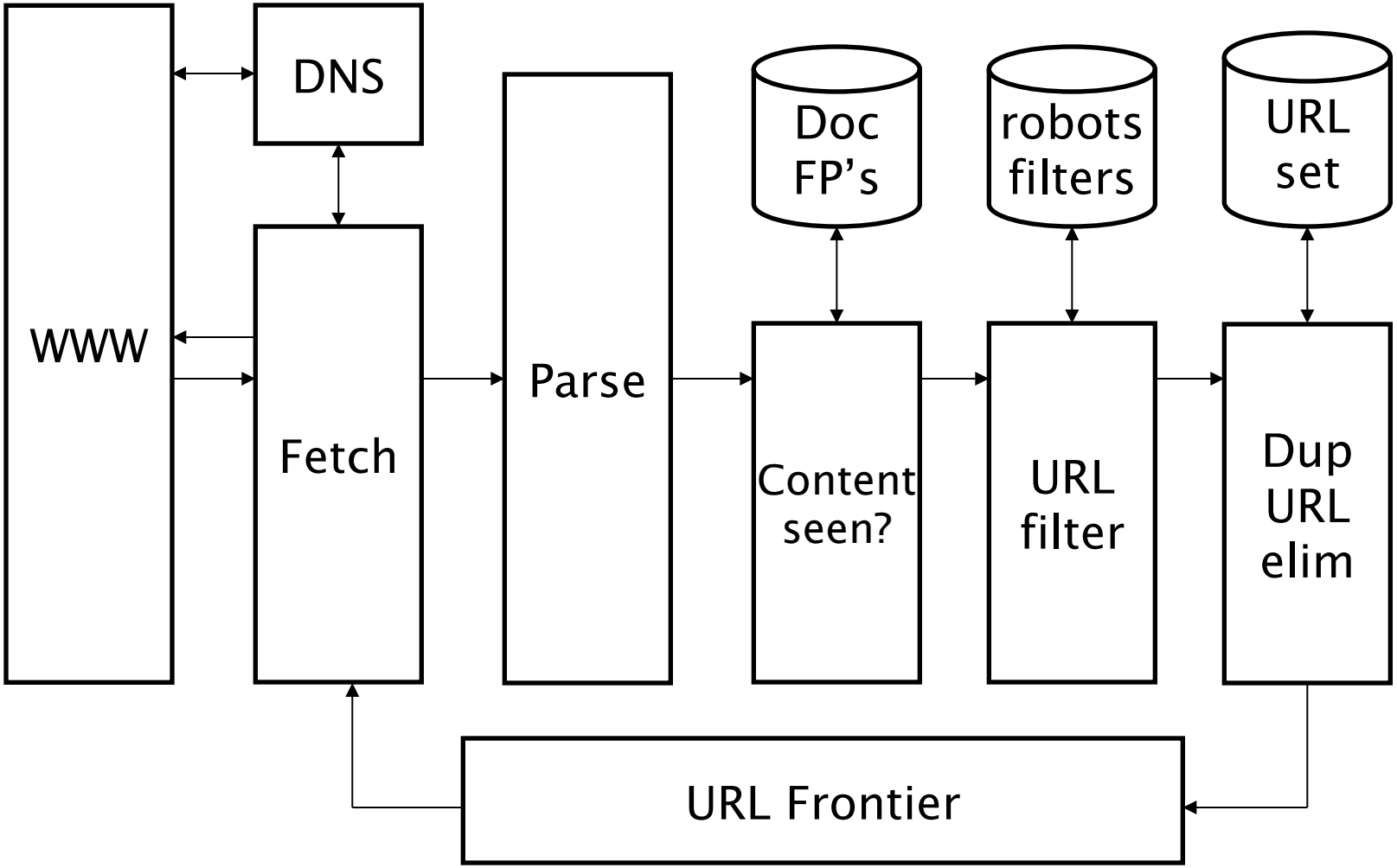
Processing steps in crawling

- Pick a URL from the frontier
- Fetch the document at the URL
- Parse the URL
  - Extract links from it to other docs (URLs)
- Check if URL has content already seen
  - If not, add to indexes
- For each extracted URL
  - Ensure it passes certain URL filter tests
  - Check if it is already in the frontier (duplicate URL elimination)

Which one?

E.g., only crawl .edu, obey robots.txt, etc.

Basic crawl architecture



DNS (Domain Name Server)

- A lookup service on the internet
  - Given a URL, retrieve its IP address
  - Service provided by a distributed set of servers – thus, lookup latencies can be high (even seconds)
- Common OS implementations of DNS lookup are *blocking*: only one outstanding request at a time
- Solutions
  - DNS caching
  - Batch DNS resolver – collects requests and sends them out together

Parsing: URL normalization

- When a fetched document is parsed, some of the extracted links are *relative* URLs
- E.g., http://en.wikipedia.org/wiki/Main_Page has a relative link to /wiki/Wikipedia:General_disclaimer which is the same as the absolute URL http://en.wikipedia.org/wiki/Wikipedia:General_disclaimer
- During parsing, must normalize (expand) such relative URLs

Content seen?

- Duplication is widespread on the web
- If the page just fetched is already in the index, do not further process it
- This is verified using document fingerprints or shingles

Filters and robots.txt

- Filters – regular expressions for URLs to be crawled/not
- Once a robots.txt file is fetched from a site, need not fetch it repeatedly
  - Doing so burns bandwidth, hits web server
- Cache robots.txt files

Duplicate URL elimination

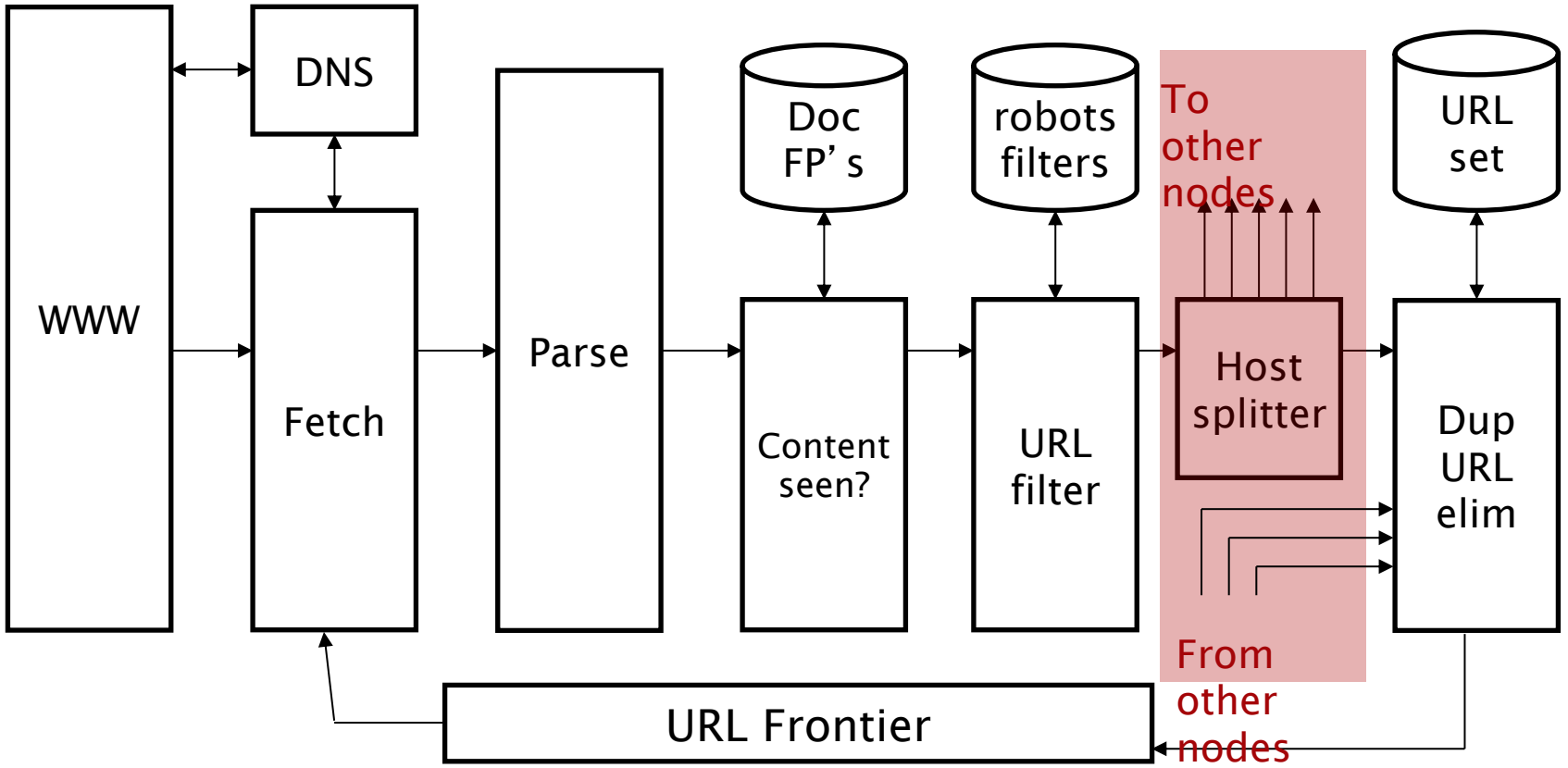
- For a non-continuous (one-shot) crawl, test to see if an extracted+filtered URL has already been passed to the frontier

- For a continuous crawl – see details of frontier implementation

Distributing the crawler

- Run multiple crawl threads, under different processes – potentially at different nodes
  - Geographically distributed nodes
- Partition hosts being crawled into nodes
  - Hash used for partition
- How do these nodes communicate and share URLs?

Communication between nodes



- Output of the URL filter at each node is sent to the Dup URL Eliminator of the appropriate node

URL frontier: two main considerations

- Politeness: do not hit a web server too frequently
- Freshness: crawl some pages more often than others
  - E.g., pages (such as News sites) whose content changes often

These goals may conflict with each other.

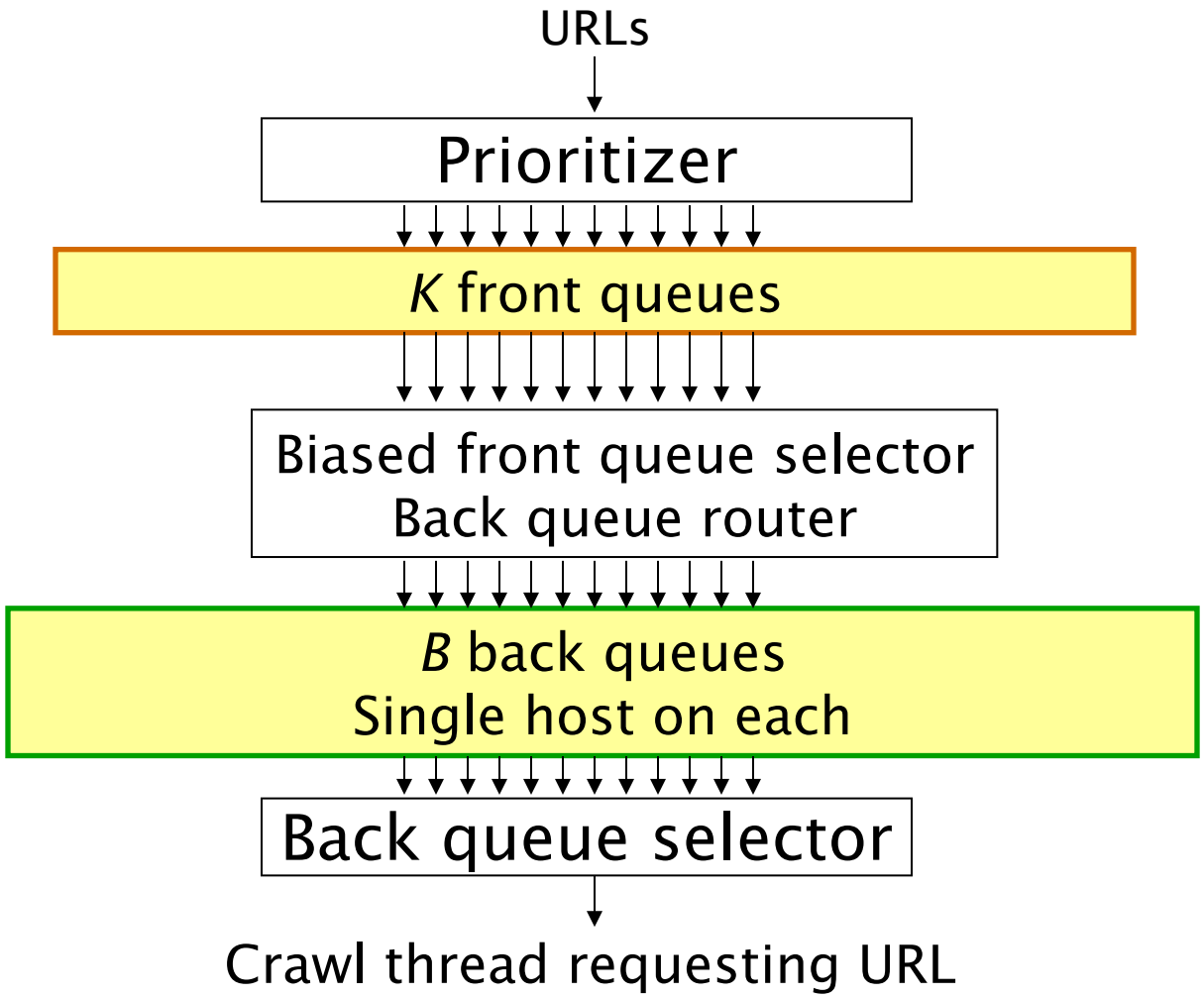
(E.g., simple priority queue fails – many links out of a page go to its own site, creating a burst of accesses to that site.)

Politeness – challenges

- Even if we restrict only one thread to fetch from a host, can hit it repeatedly

- Common heuristic: insert time gap between successive requests to a host that is >> time for most recent fetch from that host

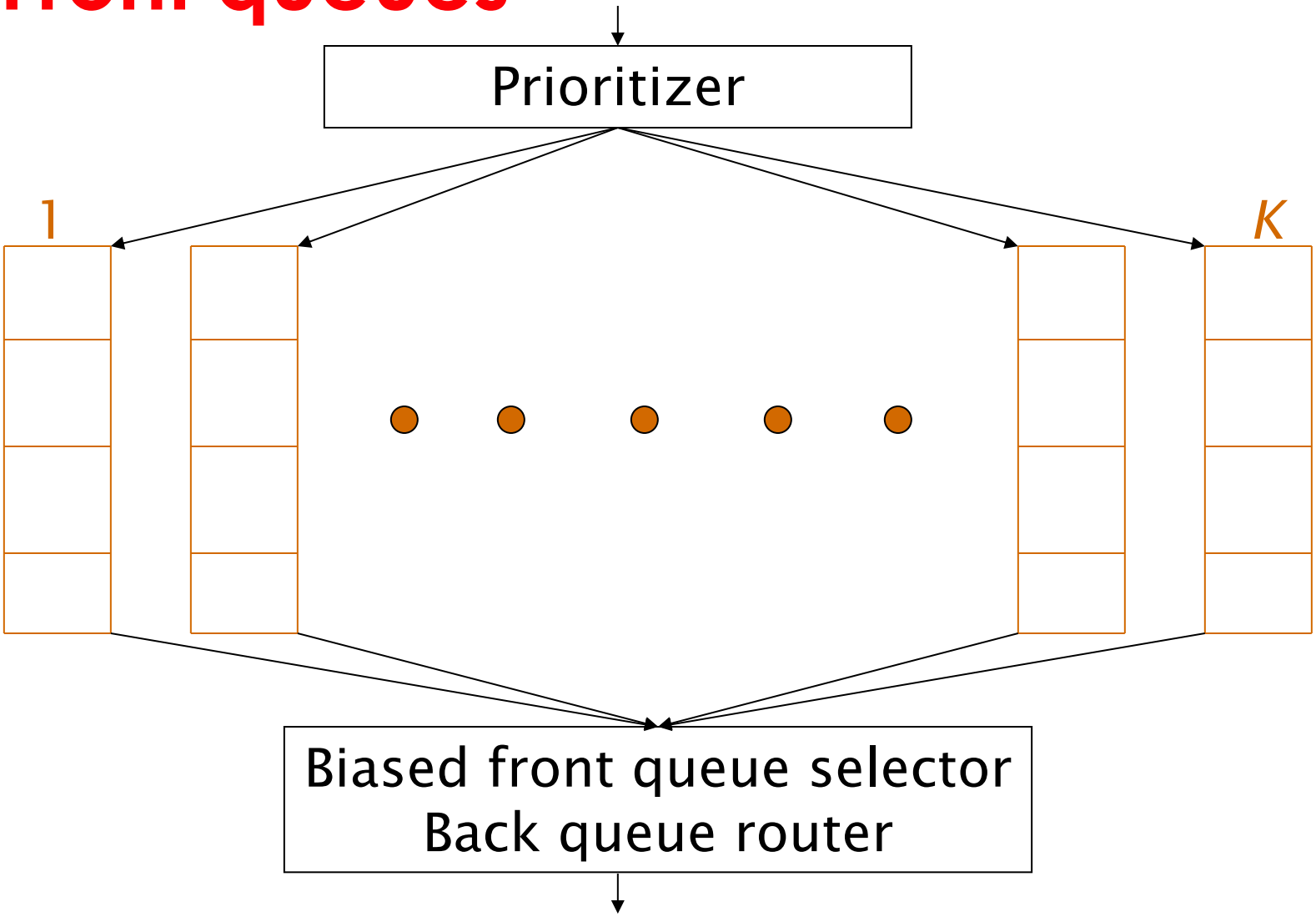
URL frontier: Scheme



URL frontier

- URLs flow in from the top into the frontier
- Front queues manage prioritization
- Back queues enforce politeness
- Each queue is FIFO

Front queues



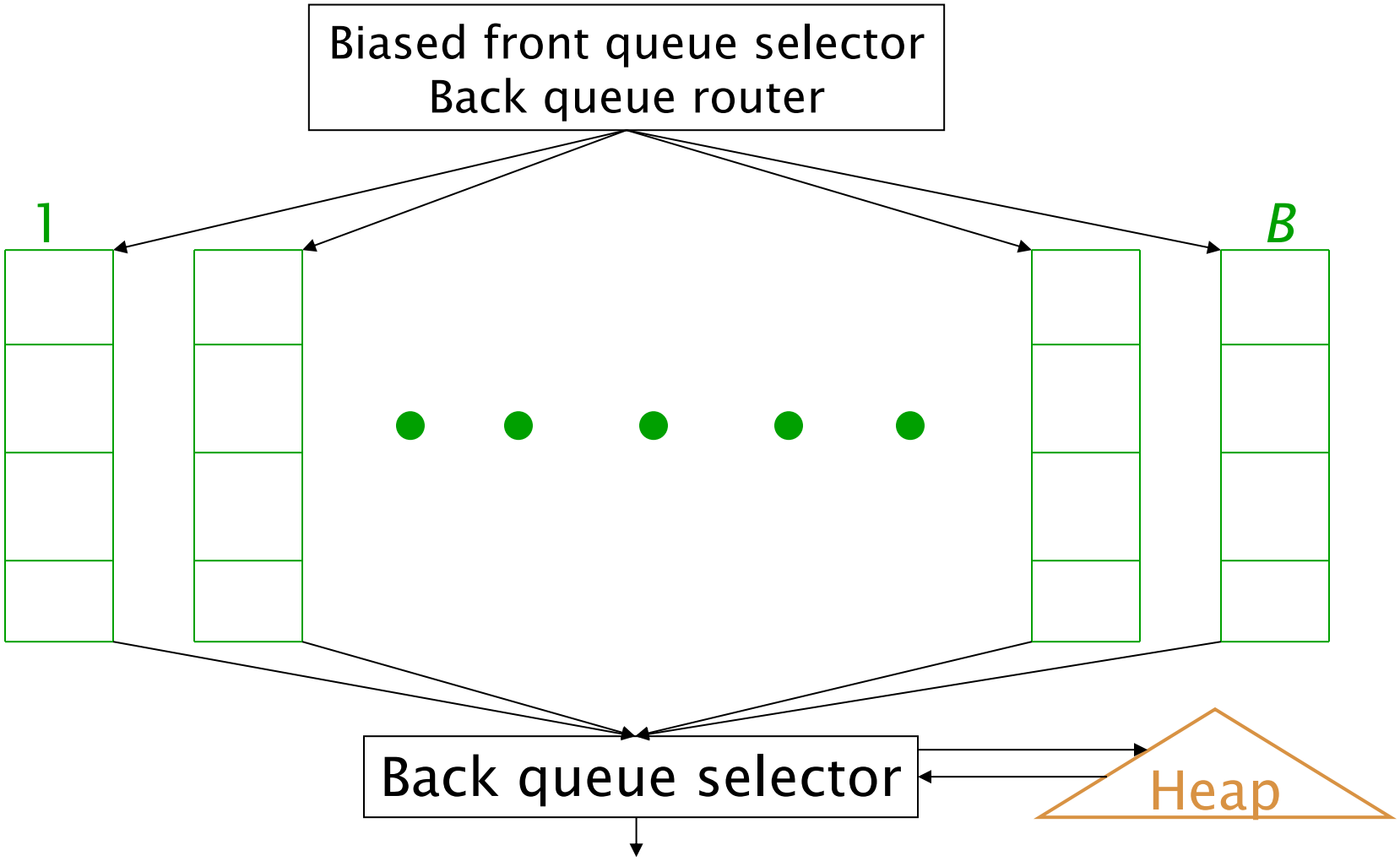
Front queues

- Prioritizer assigns to URL an integer priority between 1 and *K*
  - Appends URL to corresponding queue
- Heuristics for assigning priority
  - Refresh rate sampled from previous crawls
  - Application-specific (e.g., "crawl news sites more often")

Biased front queue selector

- When a back queue requests a URL (in a sequence to be described): picks a front queue from which to pull a URL
- This choice can be round robin biased to queues of higher priority, or some more sophisticated variant
  - Can be randomized

Back queues


Biased front queue selector
Back queue router
1
B
Back queue selector
Heap

Back queue invariants

- Each back queue is kept non-empty while the crawl is in progress

- Each back queue only contains URLs from a single host
  - Maintain a table from hosts to back queues

Host name	Back queue
…	3
	1
	B

Back queue heap

- One entry for each back queue
- The entry is the earliest time t_e at which the host corresponding to the back queue can be hit again
- This earliest time is determined from
  - Last access to that host
  - Any time buffer heuristic we choose

Back queue processing

- A crawler thread seeking a URL to crawl:
- Extracts the root of the heap
- Fetches URL at head of corresponding back queue *q* (look up from table)
- Checks if queue *q* is now empty – if so, pulls a URL *v* from front queues
  - If there's already a back queue for *v*'s host, append *v* to it and pull another URL from front queues, repeat
  - Else add *v* to *q*
- When *q* is non-empty, create heap entry for it

Number of back queues B

- Keep all threads busy while respecting politeness
- Recommendation: three times as many back queues as crawler threads

Summary

In this class, we focused on:

(a) Recap:

- ✓ Relevance Feedback
- ✓ Pseudo Relevance Feedback
- ✓ Query Expansion

(b) Web Crawling

- ➢ World Wide Web
- ➢ Basic Architecture
- ➢ URL Frontier
- ➢ Duplicate URL Elimination
- ➢ URL Queues

Many more to come up … ! Stay Tuned !!

Acknowledgements

Thanks to ALL RESEARCHERS:

- Modern Information Retrieval Baeza-Yates and Ribeiro-Neto, Addison Wesley, 1999.
- **Introduction to Information Retrieval Manning, Raghavan and Schutze, Cambridge University Press, 2008.**
- Search Engines Information Retrieval in Practice W. Bruce Croft, D. Metzler, T. Strohman, Pearson, 2009.
- Information Retrieval Implementing and Evaluating Search Engines Stefan Büttcher, Charles L. A. Clarke and Gordon V. Cormack, MIT Press, 2010.
- Many Authors who contributed to SIGIR / WWW / KDD / ECIR / CIKM / WSDM and other top tier conferences
- **Prof. Mandar Mitra, Indian Statistical Institute, Kolkatata (https://www.isical.ac.in/~mandar/)**

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may ask for one-to-one meeting

Best Approach

- You may leave me an email any time
  (email is the best way to reach me faster)

Questions
It's Your Time

Contact Information

Dr. Rajendra Prasath
IIIT Sri City, Chittoor

Monsoon 2022

Distributional Semantics

- Including distributional hypothesis and word space models

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

15th November 2022 (rajendra.2power3.com)

Recap: Information Retrieval

- **Information Retrieval (IR)** is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

- These days we frequently think first of web search, but there are many other cases:
  - E-mail search
  - Searching your laptop
  - Corporate knowledge bases
  - Legal information retrieval
  - and so on . . .

Recap: Vector Space Models

- The length of the sub-vector in dimension - i is used to represent the importance or the weigh of word – i in a text
- Words that are absent in a text get a weight – 0 (zero)
- Apply Vector Inner Product measure between two vectors:
- This vector inner product increases:
  - **# words match between two texts**
  - **Importance of the matching terms**

Recap: How can we improve recall in search?

- Main Topic: two ways of improving recall: relevance feedback and query expansion

- As an example consider query q: [aircraft] . . .
- . . . and document d containing "plane", but not containing "aircraft"

- A simple IR system will not return d for q.
- Even if d is the most relevant document for q!

- We want to change this:
  - **Return relevant documents even if there is no term match with the (original) query**

Recap: Relevance Feedback (RF)

- The user issues a (short, simple) query.
- The search engine returns a set of documents.
- User marks some docs as relevant, some as nonrelevant.
- Search engine computes a new representation of the information need. Hope: better than the initial query.
- Search engine runs new query and returns new results.
- New results have (hopefully) better recall.

Recap: Pseudo Relevance Feedback (PRF)

- Pseudo-relevance feedback automates the "manual" part of true relevance feedback.

- Pseudo-relevance algorithm:
  - Retrieve a ranked list of hits for the user's query
  - **Assume that the top k documents are relevant.**
  - Do relevance feedback (e.g., Rocchio)

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

Recap: Query expansion

- Query expansion is another method for **increasing recall**.
- We use "global query expansion" to refer to "global methods for query reformulation".
- In global query expansion, the query is modified based on some global resource, i.e. a resource that is not query-dependent.

- Main information we use: (near-)synonymy
- A publication or database that collects (near-)synonyms is called a **thesaurus**.
- We will look at two types of thesauri: manually created and automatically created.

✧ Topics Covered So Far

- ✧ Bi-Word Index
- ✧ Wild Card Queries
- ✧ Permuterm Index
- ✧ K-gram Index (k = 2 → Bigram Index)
- ✧ Spell Correction
- ✧ Term Weighting
- ✧ Vector Space Models
- ✧ Evaluation Metrics
- ✧ Relevance Feedback

✧ Now: Distributed Semantics

Recap: Overview

- Why Ranked Retrieval?
- Term Frequency
- Term Weighting
- TF-IDF Weighting
- The Vector Space Model
- Relevance Feedback
- Pseudo Relevance Feedback

Recap: Vector Space Models

- The length of the sub-vector in dimension - i is used to represent the importance or the weigh of word – i in a text
- Words that are absent in a text get a weight – 0 (zero)
- Apply Vector Inner Product measure between two vectors:
- This vector inner product increases:
- # words match between two texts
- Importance of the matching terms

Word Space Models

- How do we identify semantically related information that improves documents retrieval better
- User query terms are expanded based on terms with similar word senses that are discovered by the "**associatedness**" of the document context with that of the given query
- Can we capture the term contexts using higher order term associations and assists the effective retrieval of news documents

Motivations

- **Associatedness** - guided by word space models (Kanerva et al 2000)
- The word-space model computes the meaning of terms by implicitly utilizing the contexts of words collected over large text data
- The distributional patterns represent semantic similarity between words in terms of their spatial proximity in the context space
  - Words → context vectors whose relative directions are assumed to indicate semantic similarity
- **Distributional hypothesis:**
  - words with similar meanings are assumed to have similar contexts
  - word space methodology makes semantics computable
  - Underlying models do not require linguistic or semantic expertise

Interesting Contributions

- Similarity assessment is conjectured to involve higher-order relationships, particularly in the models of analogical reasoning (Gentner and Forbus, 1991)
- Discovered higher-order distributional relations for textual CBR (Chakraborti et al., 2007; Deerwester et al., 1990)
- RI is an alternative to **Latent Semantic Indexing (LSI)** that reduces dimensionality (Kanerva et al., 2000; Sahlgren, 2005)
- Semantic behavior, word order information can be learned in an unsupervised way, using **Holographic Reduced Representations(HRR)** (Plate, 1995; Jones and Mewhort, 2007)
- We follow the evaluation measures proposed for the standard IR systems (Singhal et al., 1996; Raghunathan et al., 2008)

Interesting Contributions (contd)

- Bruninghaus et al. [5] NLP & IE methods to automatically extract relevant factual information without converting into structured documents.
- Harman [12] - Abductive inference reasons upon incomplete or inconsistent information
- Baddeley [3] - find context relationship between documents
- Kanerva [18, 17] - scalable distributional model – meaningful implicit relationships between terms in queries and documents
- Harris [13] - semantically similar terms occur in similar contexts → semantically similar docs (no topic info)
- Johnson - Lindenstrauss Lemma [16] - a way of encoding textual information in the form of random projections.

LSI vs Random Indexing

- Latent Semantic Indexing (LSI) :
  - Sparse term - document matrix and Singular Value
  - Decomposition (SVD)
  - Not incremental
  - Dimensionality Reduction
- **Random Indexing (RI):** uses distributional statistics for identifying the semantic similarity
  - Random index and context vectors - one for each term /
  - sentence / para - of fixed size
  - Incremental
  - Identifies implicit semantic relations

Distributional Hypothesis

Johnson-Lindenstrauss Lemma [1984]:

- A set of points in a high dimensional vector space can be mapped down into a reduced dimensional space such that the distance between any two points changes but not significantly

- This inherently leads to:
  - Dimensionality Reduction
  - Random Projections

Word Relations

- Associative relations - immediate relations to adjacent words:

eat ⟶ food

- Synonymy relations - second order relations to words that share contexts:

eat

↓

drink

- word space methodology makes semantics computable and constitutes a purely descriptive approach to semantic modeling

Random Indexing

- **Index Vectors:**
  - each context (e.g. each document or each word) is assigned a unique and randomly generated representation
  - index vectors are sparse, high-dimensional, and ternary
  - their dimensionality (d) is in the order of thousands, and that they consist of a small number of randomly distributed +1s and -1s, with the rest of the elements of the vectors set to 0.
- **Context Vectors:**
  - context vectors are produced by scanning through the text
  - each time a word occurs in a context (within a sliding context window), that context's d-dimensional index vector is added to the context vector for the word in question.
  - Words are thus represented by d-dimensional context vectors that are effectively the sum of the words' contexts

Random Indexing (contd)

Context Vectors:

- For each occurrence of a given feature in all cases, we focus on a fixed window of size (2 × k) + 1 centered at the given feature [suggested window size is 5 (= term + / - k terms)]
- Then feature context vector for featurei is computed using the following equation:

$$C_{feature_i} = C_{feature_i} + \sum_{j=-k;j\neq 0}^{+k} I_{feature_{(i+j)}} \times \frac{1}{d^{|j|}}$$

- where 1/d|j| is the weight proportion w.r.to size j of window (d=2)

- Superposition is used while updating the context vector
- Adding two vectors x and y yields a vector z where z = x + y & the cosine similarities between x & z, and y & z will be high

RI – An Example

New Case: "The fisherman caught a big salmon today"
window size k = 2 , and we are training the feature **big**

The windowed sentence for the feature big looks like this:
The, [fisherman, caught, big, salmon, today].
The feature-context-vector C_{big} for big becomes now:

$$C_{big} = C_{big} + (0.25 \times I_{fisherman}) + (0.5 \times I_{caught}) + (0.5 \times I_{salmon}) + (0.25 \times I_{today})$$

Meaning of a case is captured in the collective representation of the constituent features

Doc Context Vectors:

- Doc context vector = a weighted superposition of context vectors of features that occur in the case, as follows:

$$C_{case} = \sum_{i=1} f_i \times C_{feature_i}$$

where f_i is the number of occurrences of $feature_i$ in case.

Example: Consider 2 Sentences

Two sample sentences:

the weather is **fine** in Hong Kong

the weather is **nice** in Hong Kong

Generate random keys for each word within some context

Window: weather { 0 -1 +1 0 }

is	{ 0	0	+1	-1 }
in	{ +1	0	0	-1 }
hong	{ +1	-1	0	0 }

Collect sums for words of interest:

	d1	d2	d3	d4	
weather { 0	-1	+1	0 }		
is	{ 0	0	+1	-1 }	
in	{ +1	0	0	1 }	
hong	{ +1	1	0	0 }	
					+
fine	{ +2	2	+2	2 }	
nice	{ +2	2	+2	2 }	

Advantages of RI

- **RI is an incremental method**
  - Similarity computation even with a few examples
- Dimensionality d of the vectors is a parameter
- Random Indexing uses "implicit" dimension reduction
  - [Constant (much lower) dimensionality]
- Random Indexing can be used with any type of context
  - Other word space models typically use either documents or words as contexts

Summary

In this class, we focused on:

(a) Words / Terms / Lexical Units
(b) Preparing Term – Document matrix
(c) Boolean Retrieval
(d) Inverted Index Construction

(e) Distributional Semantics
   i. Hypothesis
   ii. Word-Space Models
   iii. Random Indexing
   iv. Illustrations

Acknowledgements

Thanks to ALL RESEARCHERS:

1. Introduction to Information Retrieval Manning, Raghavan and Schutze, Cambridge University Press, 2008.
2. Search Engines Information Retrieval in Practice W. Bruce Croft, D. Metzler, T. Strohman, Pearson, 2009.
3. Information Retrieval Implementing and Evaluating Search Engines Stefan Büttcher, Charles L. A. Clarke and Gordon V. Cormack, MIT Press, 2010.
4. Modern Information Retrieval Baeza-Yates and Ribeiro-Neto, Addison Wesley, 1999.
5. Many Authors who contributed to SIGIR / WWW / KDD / ECIR / CIKM / WSDM and other top tier conferences
6. Prof. Mandar Mitra, Indian Statistical Institute, Kolkatata (https://www.isical.ac.in/~mandar/)

Questions
It's Your Time

Contact Information:

Dr. Rajendra Prasath
IIIT Sri City, Chittoor

Monsoon 2022

XML Retrieval

- Including SimNoMerge Algorithm

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

16th November 2022 (rajendra.2power3.com)

> Topics to be covered

- Recap:
  - Phrase Queries / Proximity Search
  - Spell Correction / Noisy Channel Modelling
  - Index Construction / Ranking – Scoring
  - Vector Space Models / Probabilistic IR
  - Information Retrieval Evaluation
  - Relevance Feedback / PRF / Query Expansion
  - Web Crawling / Distributional Semantics

- XML Retrieval
  - 4 Different Approaches
  - Basics of XML Retrieval
  - Context Resemblance

- More topics to come up … Stay tuned …!!

Overview

Recap: Information Retrieval

- **Information Retrieval (IR)** is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

- These days we frequently think first of **web search**, but there are many other cases:
  - E-mail search
  - Searching your laptop
  - Corporate knowledge bases
  - Legal information retrieval
  - and so on . . .

Recap: Vector Space Models

- The length of the sub-vector in dimension - i is used to represent the importance or the weigh of word – i in a text
- Words that are absent in a text get a weight – 0 (zero)
- Apply Vector Inner Product measure between two vectors:
- This vector inner product increases:
  - **# words match between two texts**
  - **Importance of the matching terms**

Recap: How can we improve recall in search?

- Main Topic: two ways of improving recall: relevance feedback and query expansion
- As an example consider query q: [aircraft] . . .
- . . . and document d containing "plane", but not containing "aircraft"
- A simple IR system will not return d for q.
- Even if d is the most relevant document for q!
- We want to change this:
  - **Return relevant documents even if there is no term match with the (original) query**

Recap: Relevance Feedback (RF)

- The user issues a (short, simple) query.
- The search engine returns a set of documents.
- User marks some docs as relevant, some as nonrelevant.
- Search engine computes a new representation of the information need. Hope: better than the initial query.
- Search engine runs new query and returns new results.
- New results have (hopefully) better recall.

Recap: Pseudo Relevance Feedback (PRF)

- Pseudo-relevance feedback automates the "manual" part of true relevance feedback.

- Pseudo-relevance algorithm:
  - Retrieve a ranked list of hits for the user's query
  - **Assume that the top k documents are relevant.**
  - Do relevance feedback (e.g., Rocchio)

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

Recap: Query expansion

- Query expansion is another method for **increasing recall**.
- We use "global query expansion" to refer to "global methods for query reformulation".
- In global query expansion, the query is modified based on some global resource, i.e. a resource that is not query-dependent.

- Main information we use: (near-)synonymy
- A publication or database that collects (near-)synonyms is called a **thesaurus**.
- We will look at two types of thesauri: manually created and automatically created.

XML Retrieval

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

IR and relational databases

- IR systems are 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

- Basic setting: 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

- An unranked system (DB) would return a potentially large number of articles that mention the Vatican, the Coliseum and sightseeing tours without ranking them by relevance to query.
- Difficult for users to precisely state structural constraints – may not know which structured elements are supported by the system.
  - tours AND (COUNTRY: Vatican OR
  - LANDMARK: Coliseum)?
  - tours AND (STATE: Vatican OR BUILDING: Coliseum)?
- Users may be completely unfamiliar with structured search and advanced search interfaces or unwilling to use them.
- Solution: adapt ranked retrieval to structured documents to address these problems.

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)

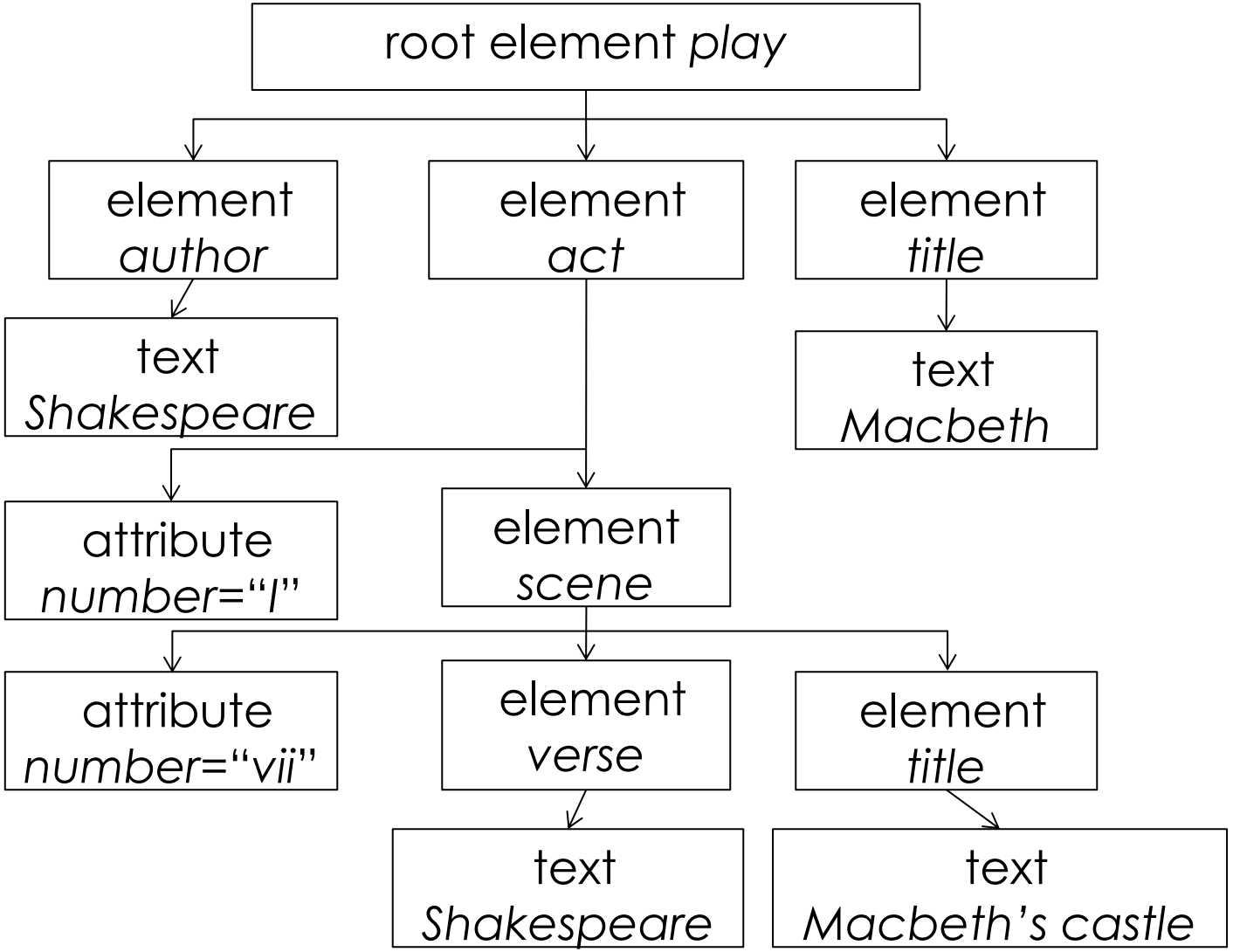
- ✧ structured IR → XML IR
- ✧ Applicable to other types of markup (HTML, SGML, ...)

XML document

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

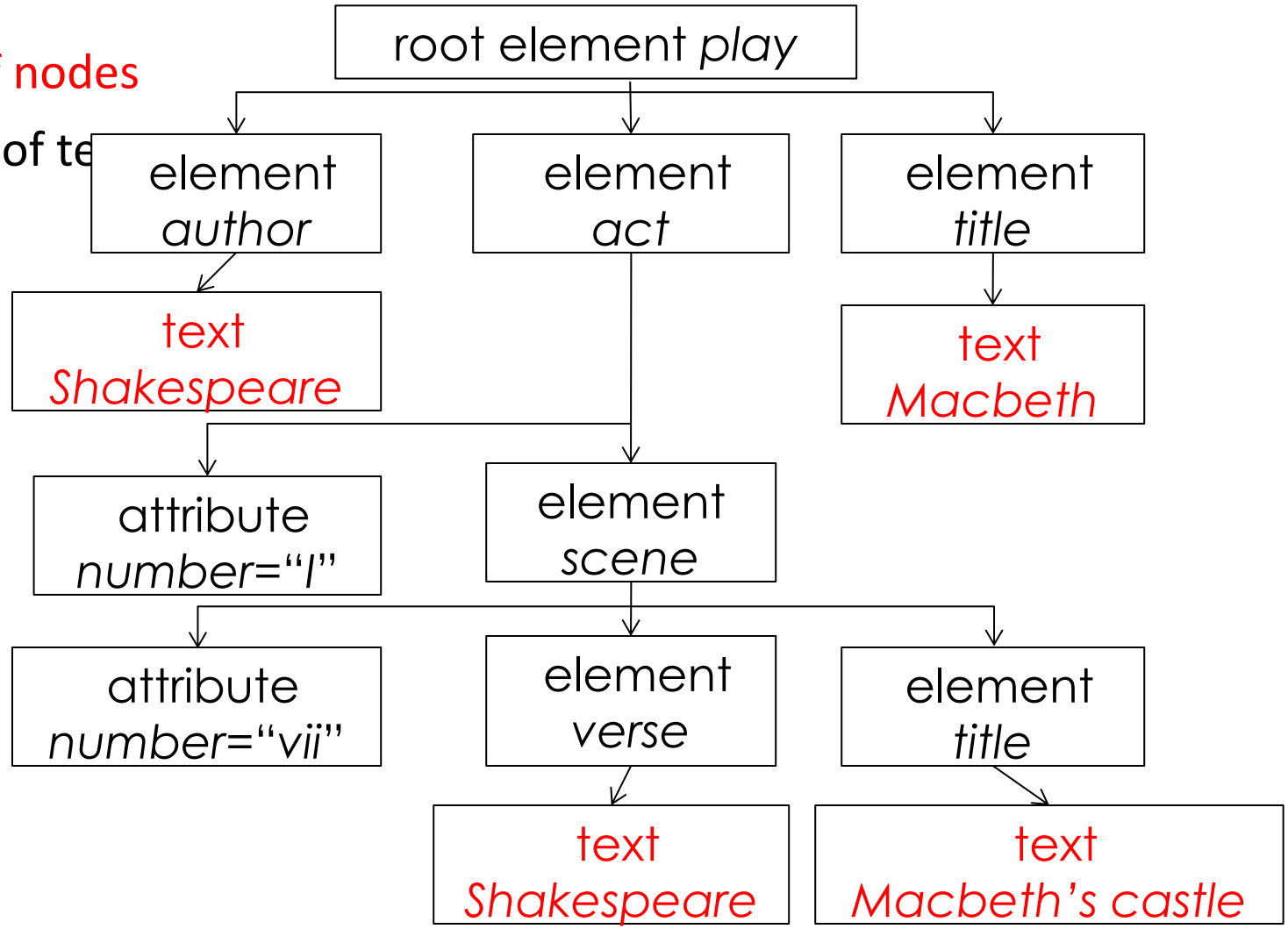
```
<play>
<author>Shakespeare</author>
<title>Macbeth</title>
<act number="I">
<scene  number=""vii">
<title>Macbeth's castle</title>
<verse>Will I with wine
…</verse>
</scene>
</act>
</play>
```

XML document

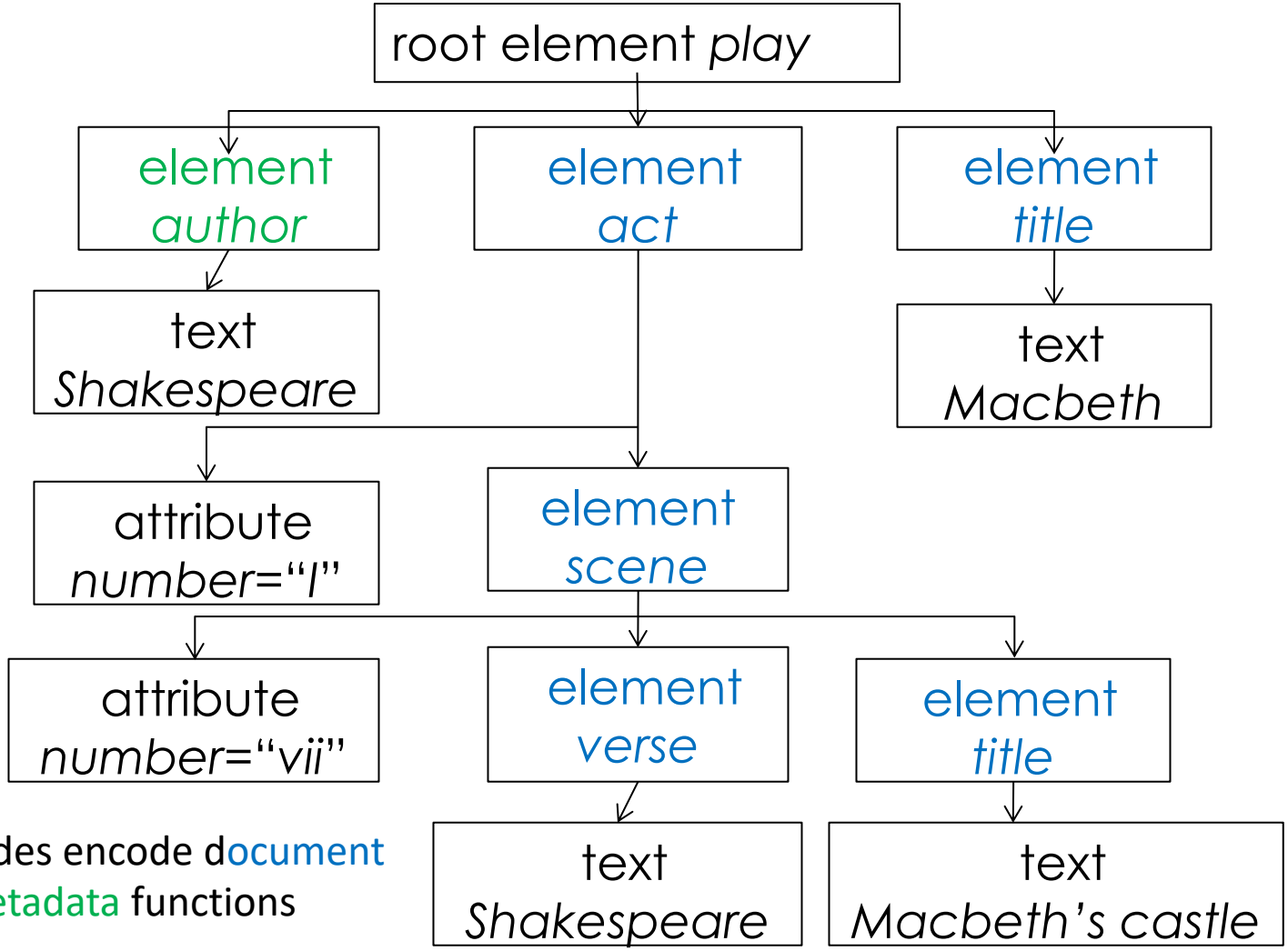


XML document

The leaf nodes consist of te



XML document



The internal nodes encode document structure or metadata functions

XML basics

- **XML Documents Object Model (XML DOM)**: standard for accessing and processing XML documents
  - The DOM represents elements, attributes and text within elements as nodes in a tree.
  - With a DOM API, we can process an XML documents by starting at the root element and then descending down the tree from parents to children.
- **XPath**: standard for enumerating path in an XML document collection.
  - We will also refer to paths as XML contexts or simply contexts
- **Schema**: puts constraints on the structure of allowable XML documents. E.g. a schema for Shakespeare's plays: scenes can occur as children of acts.
  - Two standards for schemas for XML documents are: XML DTD (document type definition) and XML Schema.

First challenge: document parts to retrieve

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

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

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

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

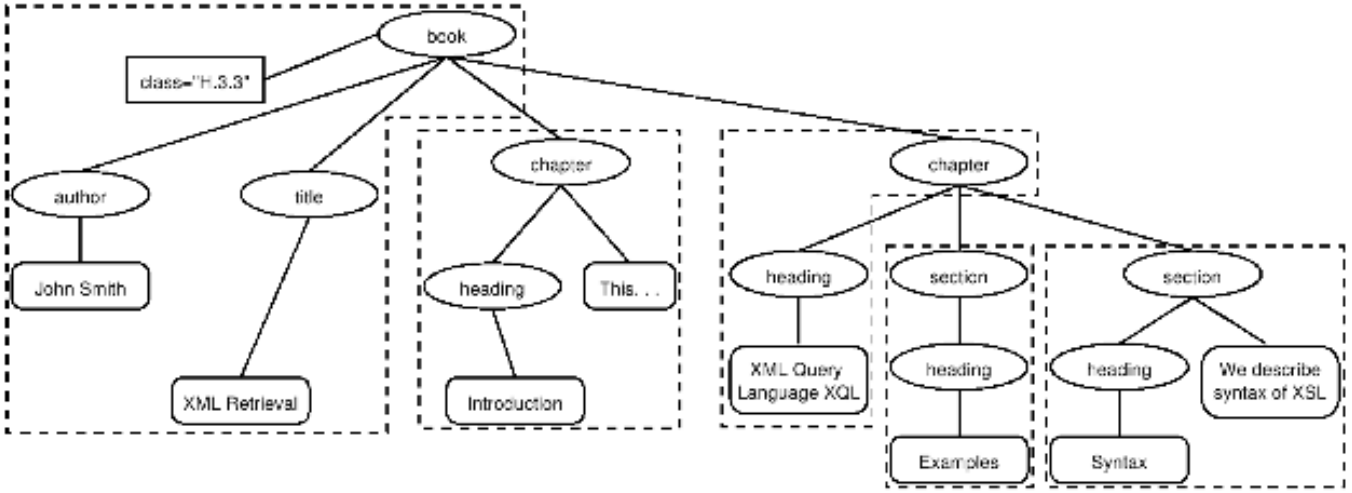
- Motivates a retrieval strategy that returns the smallest unit that contains the information sought, but does not go below this level.
- Hard to implement this principle algorithmically. E.g. query: title:Macbeth can match both the title of the tragedy, Macbeth, and the title of Act I, Scene vii, Macbeth's castle.
  - But in this case, the title of the tragedy (higher node) is preferred.
  - Difficult to decide which level of the tree satisfies the query.

Second challenge: document parts to index

- Central notion for indexing and ranking in IR: documents unit or indexing unit.
- In unstructured retrieval, usually straightforward: files on your desktop, email massages, web pages on the web etc.
- In structured retrieval, there are four main different approaches to defining the indexing unit
  - non-overlapping pseudodocuments
  - top down
  - bottom up
  - all

XML indexing unit: approach 1

- Group nodes into non-overlapping pseudo documents



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

XML indexing unit: approach 2

- Top down (2-stage process):
  - Start with one of the latest elements as the indexing unit, e.g. the book element in a collection of books
  - Then, postprocess search results to find for each book the subelement that is the best hit.

- This two-stage retrieval process often fails to return the best subelement because the relevance of a whole book is often not a good predictor of the relevance of small subelements within it.

XML indexing unit: approach 3

Bottom up:

- Instead of retrieving large units and identifying subelements (top
- down), we can search all leaves, select the most relevant ones and
- then extend them to larger units in postprocessing.
- Similar problem as top down: the relevance of a leaf element is
- often not a good predictor of the relevance of elements it is
- contained in.

XML indexing unit: approach 4

Index all elements: the least restrictive approach. Also problematic:

- ✧ Many XML elements are not meaningful search results, e.g., an ISBN number.
- ✧ 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.

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

Third challenge: nested elements

- Because of the redundancy caused by the nested elements it is common to restrict the set of elements eligible for retrieval.
- Restriction strategies include:
  - discard all small elements
  - discard all element types that users do not look at (working XML retrieval system logs)
  - discard all element types that assessors generally do not judge to be relevant (if relevance assessments are available)
  - only keep element types that a system designer or librarian has deemed to be useful search results
- In most of these approaches, result sets will still contain nested elements.

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

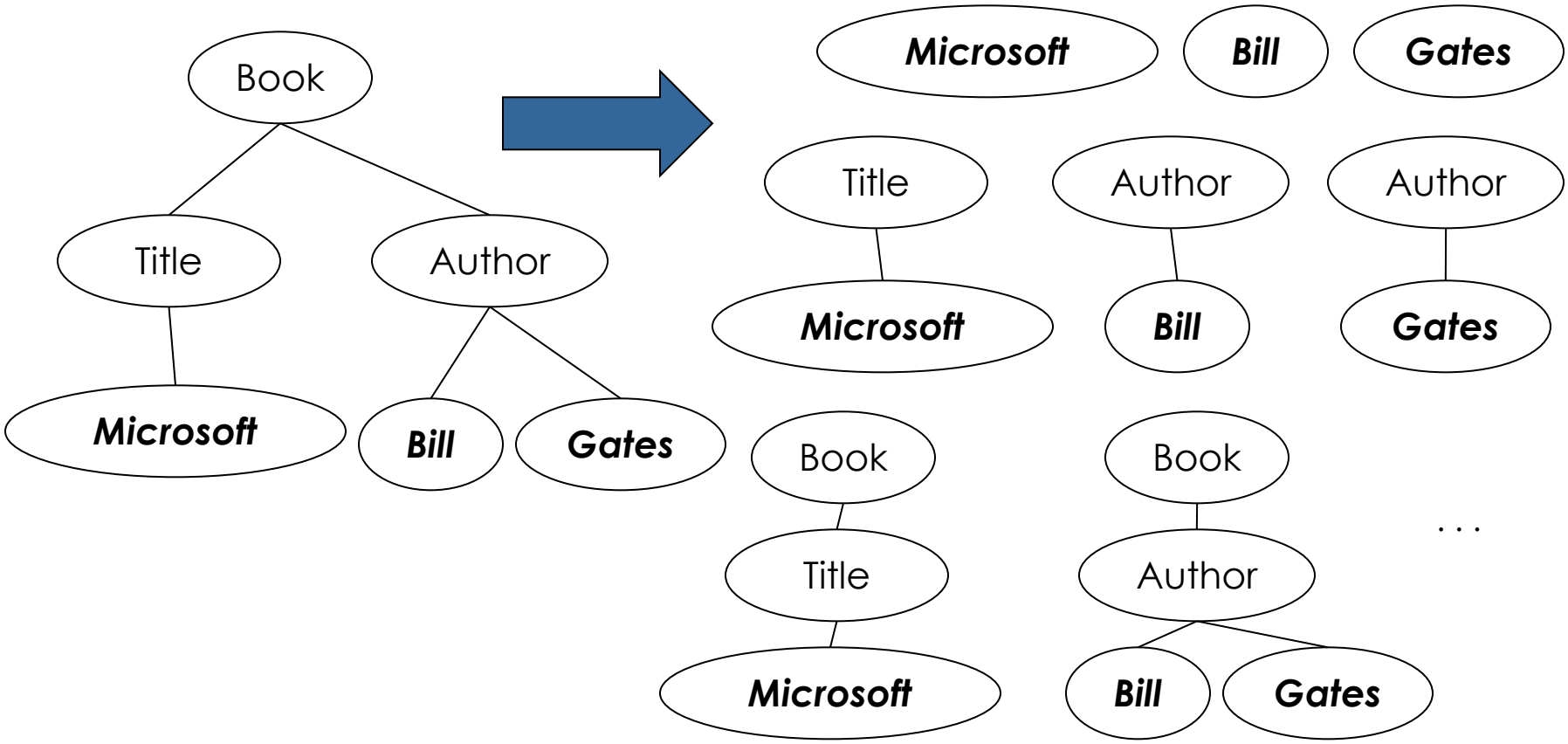
Example

- 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.

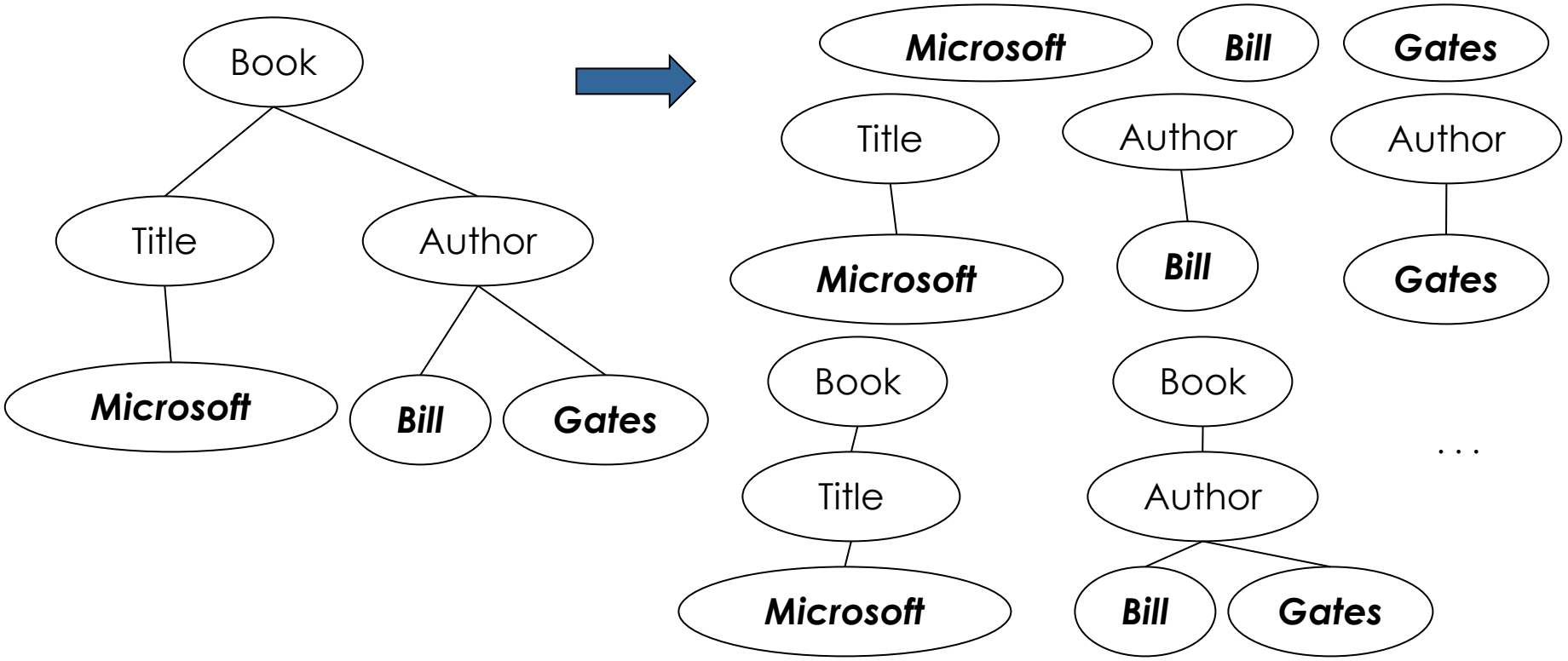
Main idea: lexicalized subtrees

- 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.



Main idea: 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.

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

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

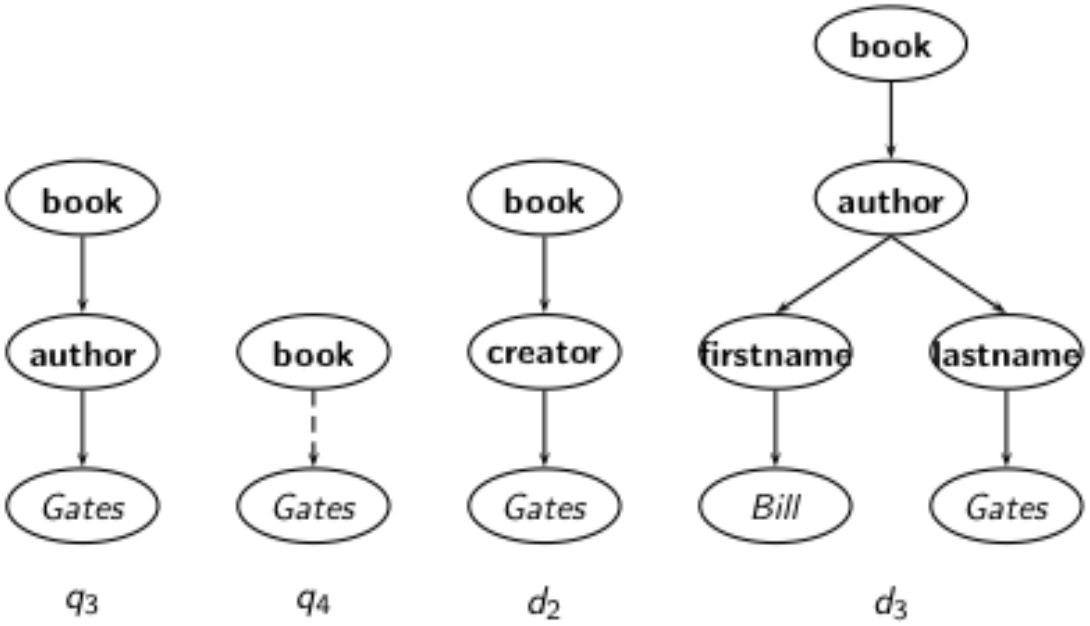
Context resemblance

- 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 C_R:

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

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

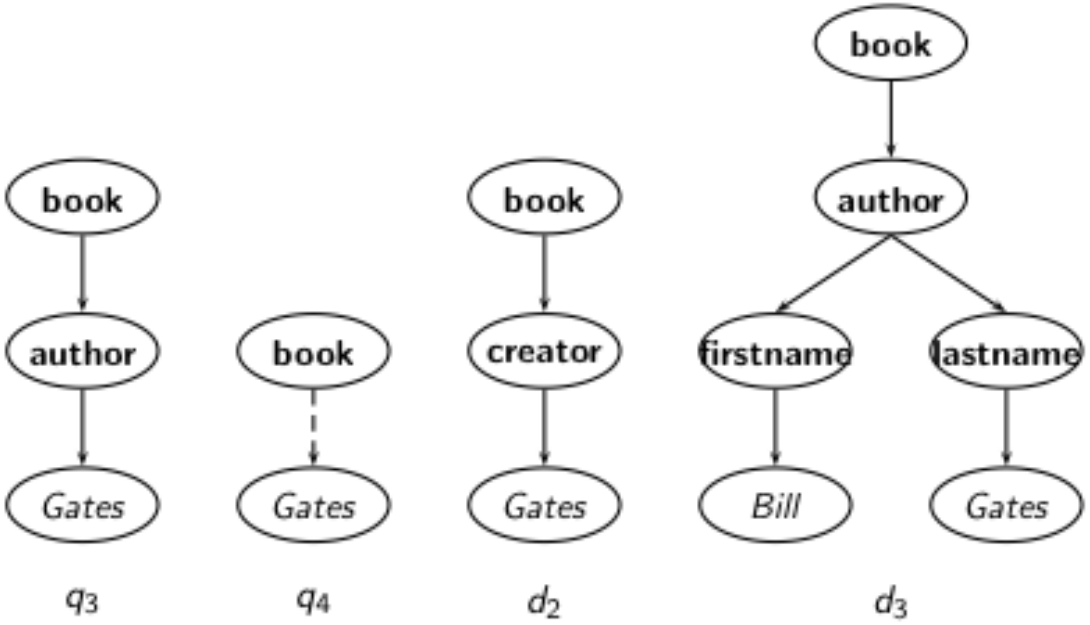
Context resemblance example



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

$C_R(c_q, c_d)$ = 3/4 = 0.75. The value of $C_R(c_q, c_d)$ is 1.0 if q and d are identical.

Context resemblance example



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

$C_R(c_q, c_d)$ = ? $C_R(c_q, c_d)$ = 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) =

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

- V is the vocabulary of non-structural terms
- B is the set of all XML contexts
- weight (q, t, c), weight(d, t, c) are the weights of term t in XML context c in query q and document d, resp. (standard weighting e.g. idf_t x $wf_{t,d}$, 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.

SimNoMerge algorithm

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
```

Applications

- Structured or XML IR: effort to port unstructured (standard) IR know-how onto a scenario that uses structured (DB-like) data
- Specialized applications (e.g. patents, digital libraries)
- A decade old, unsolved problem

Summary

In this class, we focused on:

(a) Recap:

- ✓ Relevance Feedback / PRF
- ✓ Query Expansion / Web Crawling

(b) XML Retrieval

- ➢ 4 Different Approaches
- ➢ Lexicalized Subtrees
- ➢ Context Resemblance
- ➢ Document Similarity
- ➢ XML Retrieval
- ➢ SimNoMerge Algorithm
- ➢ Examples and Illustrations

Many more to come up … ! Stay Tuned !!

Acknowledgements

Thanks to ALL RESEARCHERS:

- Modern Information Retrieval Baeza-Yates and Ribeiro-Neto, Addison Wesley, 1999.
- **Introduction to Information Retrieval Manning, Raghavan and Schutze, Cambridge University Press, 2008.**
- Search Engines Information Retrieval in Practice W. Bruce Croft, D. Metzler, T. Strohman, Pearson, 2009.
- Information Retrieval Implementing and Evaluating Search Engines Stefan Büttcher, Charles L. A. Clarke and Gordon V. Cormack, MIT Press, 2010.
- Many Authors who contributed to SIGIR / WWW / KDD / ECIR / CIKM / WSDM and other top tier conferences
- **Prof. Mandar Mitra, Indian Statistical Institute, Kolkatata (https://www.isical.ac.in/~mandar/)**

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may ask for one-to-one meeting

Best Approach

- You may leave me an email any time
  (email is the best way to reach me faster)

Questions
It's Your Time

Contact Information

Dr. Rajendra Prasath
IIIT Sri City, Chittoor