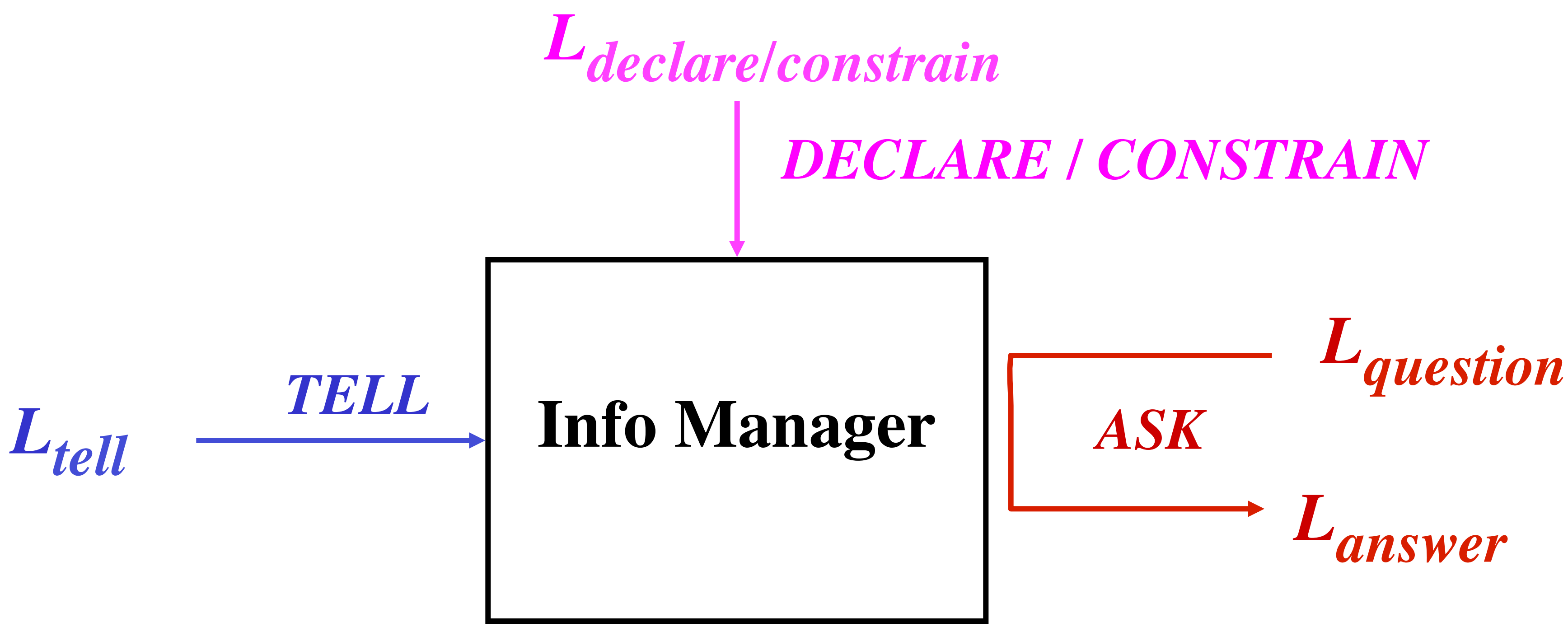
Information Retrieval

(Slides occasionally based on those of Prof. Rao Kambhampati)

Data retrieval with files of text (multimedia)
Functional View



- L_{tell} : collection of "documents" (<u>unstructured data):</u>
  email, news article, paragraph, journal article, book
- $L_{question}$: user's "information needs"
- L_{answer} : collection of "relevant" documents
- *query answering spec.*: definition of "relevant", ...

I. Boolean retrieval

- $L_{question}$ **: Boolean expression of *words***
  - e.g., "tiramisu **and** liqueur **and not** cake"
- L_{answer} **: collection of "relevant" documents**
- ***Specification of relevant:***
  - reduce formula to Disjunctive Normal Form
    $(w_{11} \wedge w_{12} \wedge \ldots) \vee (w_{21} \wedge w_{22} \wedge \ldots) \vee (w_{n1} \wedge w_{n2} \wedge \ldots)$
  - treat docs as sets of words; return all docs with every word in *some* conjunct $(w_{k1} \wedge w_{k2} \wedge \ldots)$

Boolean retrieval

***Implementation*:**

- could be done using relational databases with clobs & "*like*"
- index (hash, B+ tree) from words to documents ("simple inverted file")

***Examples:* Lexus/Nexus, medical reports, AltaVista**

***Problems*:**

1. users have "information", not "data" needs
   - **word variants** (liquor, liqueur, liqueurs) **often not relevant**
   - **polysemy; ambiguity; word location might be relevant** (AltaVista "near")
   - **too brittle** (single missing word makes document ineligible; word might not have been so important)
2. naive users seem to have problems expressing their needs in this semi-formal notation
3. the number of documents returned is too large for users to examine individually

Information Retrieval: “normalization”

To address problems 1:

a) Lexical analysis: normalize “words”

- **eliminate hyphens** (*but* MS-DOS ?)
- **punctuation marks** (*but* John’s vs Johns, ‘03)
- **normalize case of letters** (*but* us vs US)
- *Another problem:* users can’t tell what system has done

(check out google, altavista, other web search engines and see what they do)

b) Stemming

Identify morphological variants, creating groups

- *system / systems*
- *forget / forgetting / forgetful*
- *analyse / analysed / analysing / analysis / analytical*

Possible uses:

- replace word by group representative (in document)
- replace word by all variants in its group (in query)

Well known algorithm by Porter, makes 5 passes; based on condition-action rules (available in public Bow collection)

IT IS HEURISTIC!!! **(because it does not use a dictionary, to make it fast)**

Too aggressive

- organization / organ
- policy / police
- army / arm
- executive / execute

Too timid

- european / europe
- cylindrical / cylinder

c) Enriching/normalizing

- **Forming compound nouns:** *'computer science'*
- **Thesaurus:**
  create non-morphologically related group of words ("*tree*"):
  - synonyms *(arbor),*
  - hypernyms -more general/broader than *(plant),*
  - hyponyms - more specific/narrower than *(sapling)*

  e.g.,
    - ***Roget's Thesaurus* - more useful for literature**
    - ***WordNet* [Miller] - becoming widely used as a simple ontology**
- **Domain-specific thesaurus:**
  - more powerful: terminology of comp science
  - automatically generated thesaurus from the given document corpus: based on correlated occurrence of terms in the same "context" (what is context: document, paragraph, sentence structure?); works well statistically, when there are MANY documents

II. Boolean Retrieval with Controlled Vocab.

- **Alternative approach to "information needs" problem:**
  describe ahead of time **what text is about**

e.g., rather than view text as a collection of its words, assign to each document a *small* collection of words from a controlled vocabulary (e.g., the NASA thesaurus for the aerospace discipline, the MESH thesaurus for medicine, CACM/dmoz/yahoo subject hierarchy,) **representing its content**

L_{tell} **= (doc, {keyword1,keyword2,...}) pairs**

- **Who annotates the documents?**
  - author, librarian, machine ((semi) automatic classifier, possibly based on machine learning)

Approximate Query Answering

- **Note that all the previous steps are *heuristic*: they may improve answers, but occasionally they can cause problems (e.g., introduce additional ambiguity)**
- **So we are giving up on the idea of "perfect data answer", as in databases, in order to get better "information answer"**
- **Additional possibilities:**
  - provide ranked list of answers: present first those most sure to be of interest to the user; this addresses problem #3 (*"too many answers"*)
  - co-operative answering (e.g., iterative refinement, automatic weakening when answer set is empty)

Alternate Model of IR

- $\mathcal{L}_{question}$ **: another (unstructured) document, even if it is short (e.g., English sentence):** describes user interests *//addresses problem #2: difficulty of expressing queries*
- $\mathcal{L}_{answer}$ **:** ***ranked/ordered list*** of relevant documents { *doc*$_i$ }
- ***Specification of "ranking"*** **(and hence "relevance")** ***:***
  - based on a similarity function ***sim(doc$_i$, query)***

II. Vector Space model of similarity

- **Document = set of words/index terms.**

  represent collection as term/document boolean matrix W[j, k]

a: System and human system engineering testing of EPS

b: A survey of user opinion of computer system response time

c: The EPS user interface management system

d: Human machine interface for ABC computer applications

e: Relation of user perceived response time to error measurement

f: The generation of random, binary, ordered trees

g: The intersection graph of paths in trees

h: Graph minors IV: Widths of trees and well-quasi-ordering

	a	b	c	d	e	f	g	h
Interface	0	0	1	0	0	0	0	0
User	0	1	1	0	1	0	0	0
System	1	1	1	0	0	0	0	0
Human	1	0	0	1	0	0	0	0
Computer	0	1	0	1	0	0	0	0
Response	0	1	0	0	1	0	0	0
Time	0	1	0	0	1	0	0	0
EPS	1	0	1	0	0	0	0	0
Survey	0	1	0	0	0	0	0	0
Trees	0	0	0	0	0	1	1	1
Graph	0	0	0	0	0	0	1	1
Minors	0	0	0	0	0	0	0	1

q: User interface management systems

Alternatively, consider each document k as a binary vector w_k [j]

II. Vector Space model of Similarity

e.g., Collection of 6 documents, with term occurrences:

- **Doc A** **care, cat, persian**
- **Doc B** **care, care, care, cat, cat, cat, persian, persian, persian**
- **Doc C** **cat, cat, cat, cat, cat, cat, cat, cat, cat**
- **Doc D** **care, cat, dog, dog, dog, dog, dog, dog, persian**
- **Doc E** **care, cat, dog**
- **Doc F** **care**

General idea: each document will be represented by a vector of *weights* -- one corresponding to each term.

(i)e.g., Binary Vector of term occurrences in documents

Put terms in some <u>order</u> (alphabetical often):

"care", "cat", "dog" "persian".

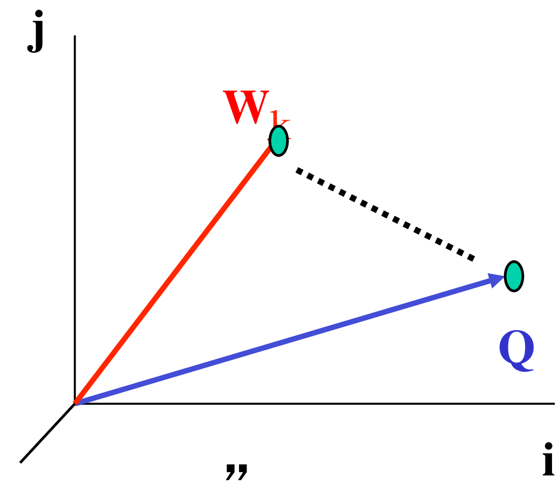
Use 0 or 1 as weights.

- DocVec_ A = <1, 1, 0, 1>
- DocVec_ B = <1, 1, 0, 1>
- DocVec_ C = <0, 1, 0, 0>
- DocVec_ D = <1, 1, 1, 1>
- DocVec_ E = <1, 1, 1, 0>
- DocVec_ F = <1, 0, 0, 0>

Vector space model

What are reasonable models of "similarity" in this case?

Think of each document $Doc_{\mathbf{k}}$ as vector $W_{\mathbf{k}}$ in n-dimensional space of index terms. (Query Q will also be thought of as a vector.)



Intuitively, want similarity measures that

1. ***allow partial match***
2. ***favor documents with more words in common***
3. ***have bounded value (e.g, between 0 and 1)***
4. ***for ease of similarity comparison***

Mathematicians have studied lots of "distance measures"

- "Euclidean distance": $\sqrt{\sum_{\mathbf{j}}(w_k[j] - q[j])^2}$
- "Dot product"

  $W_{\mathbf{k}} \bullet Q = \sum_{\mathbf{j}}(w_k[j] \times q[j])$

  for Boolean vectors, = count of all shared index terms

  (*good for 1. and 2.*)

Vector space model

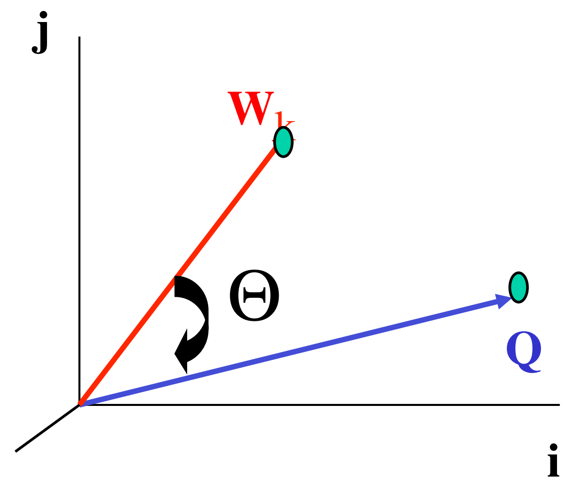
Desiderata 4: *longer documents are likely to contain more words in common with the query (though such docs are not likely to be more relevant)* **-- so should "normalize" for this** use length |W|

$$\frac{W \bullet Q}{|W||Q|}$$

- Another possible measure of similarity between vectors is the angle Θ between the vectors

Interestingly:

$$\frac{W \bullet Q}{|W||Q|} = \textbf{cosine}(\Theta) \quad !!!$$



As angle decreases from 90 to 0, cosine increases from 0 (less sim.) to 1 (more sim.), so there is no need to find angle itself – can compare cosines.

Vector Space model of similarity - 2

- **Desiderata 5:** *prefer documents in which shared terms occur more often! (⇒ treat document as* **bag** *of words, and vectors as count of terms* ***term frequency*** **TF**)
  - **Doc A** **care, cat, persian**
  - **Doc B** **care, care, care, cat, cat, cat, persian, persian, persian**
  - **Doc C** **cat, cat, cat, cat, cat, cat, cat, cat, cat**
  - **Doc D** **care, cat, dog, dog, dog, dog, dog, dog, persian**
  - **Doc E** **care, cat, dog**
  - **Doc F** **care**

Terms, in order: "care", "cat", "dog" "persian".

(Within document frequency) TF weight vectors:

- TF_ A = <1, 1, 0, 1>
- TF_ B = <3, 3, 0, 3>
- TF_ C = <0, 9, 0, 0>
- TF_ D = <1, 1, 6, 1>
- TF_ E = <1, 1, 1, 0>
- TF_ F = <1, 0, 0, 0>

Vector Space model of similarity - 3

- **Desiderata 6. for similarity measure:**
  - more frequent words (e.g., *the*, *computer*) are likely to be shared, yet not significant. So shared **infrequent** words are more significant.

To address this, add a document (in)frequency factor into the weighing: **INVERSE DOCUMENT FREQUENCY idf**

idf[j] = measures how **infrequently** term t_j appears in the *entire document set*

example (temporary)

Terms, in order: *"care", "cat", "dog" "persian".*

- Within document frequency, TF vectors:
  - TF_ A = <1, 1, 0, 1>
  - TF_ B = <3, 3, 0, 3>
  - TF_ C = <0, 9, 0, 0>
  - TF_ D = <1, 1, 6, 1>
  - TF_ E = <1, 1, 1, 0>
  - TF_ F = <1, 0, 0, 0>

- Number of document occurrences per term:

  $n_{care}=5$, $n_{cat}=5$, $n_{dog}=2$, $n_{persian}=3$

- So, one might try

  IDF['care']=1/5, IDF['cat']=1/5, IDF['dog']=1/2

Vector Space model of similarity - 3

- **Combine TF and IDF**

TF-IDF (TermFrequency -InverseDocumentFrequency) **model**

- *general form of weight for j'th index term* t_j *in doc*$_{\mathbf{k}}$ **,** *which used to be*

$$w_k[j] = tf_k[j] = \text{frequencyOfTerm}[j,k]$$

becomes

$$w_k[j] = tf_k[j] \times idf[j]$$

TF-IDF

The following formula is one of many; developed *empirically*

- **Let**
  - N be the total number of docs in the collection
  - n_j be the number of docs which contain index term t_j
  - $freq(j,k)$ number of times term t_j occurs in document d_k
- $tf_k[j] = freq(j,k)$ (or some scaled version like freq(j,k)/max freq(i,k))
- **The *idf* factor for term t_j is computed as**

$$idf[j] = \log (N/n_j)$$

  the *log* is used to reduce the weight of *idf*. It can also be interpreted as the *amount of information* associated with the term t_j.

- **(For the *query document q*, one might use a different variant)**

Now use cosine distance between vectors $w_k[\,]$, $q[\,]$ ***to rank answers***

Example TF-IDF Computation

Collection of 6 documents, with term occurrences:

- **Doc A** **care, cat, persian**
- **Doc B** **care, care, care, cat, cat, cat, persian, persian, persian**
- **Doc C** **cat, cat, cat, cat, cat, cat, cat, cat, cat**
- **Doc D** **care, cat, dog, dog, dog, dog, dog, dog, persian**
- **Doc E** **care, cat, dog**
- **Doc F** **care**

total number of docs N=6

Example

Terms, in order: *"care", "cat", "dog" "persian".*

- TF_ A = <1, 1, 0, 1>
- TF_ B = <3, 3, 0, 3>
- TF_ C = <0, 9, 0, 0>
- TF_ D = <1, 1, 6, 1>
- TF_ E = <1, 1, 1, 0>
- TF_ F = <1, 0, 0, 0>

- Number of document occurrences per term:

  $n_{care}=5, \quad n_{cat}=5, \quad n_{dog}=2, \quad n_{persian}=3$

- Number of documents $N = 6$

$$\boxed{\mathrm{IDF(term)} = \log_2 (N/n_{\mathbf{term}})}$$

$idf_{care}=log(6/5)=0.26, \quad ..., \quad idf_{persian}=log(6/3)=1.00$

IDF vector $< \log_2(6/5), \log_2(6/5), \log_2(6/2), \log_2(6/3) >$

$= < 0.26, 0.26, 1.58, 1.00 >$

Example

Terms, in order: *"care", "cat", "dog" "persian".*

- TF_ A = <1, 1, 0, 1>
- TF_ B = <3, 3, 0, 3>
- TF_ C = <0, 9, 0, 0>
- TF_ D = <1, 1, 6, 1>
- TF_ E = <1, 1, 1, 0>
- TF_ F = <1, 0, 0, 0>

IDF **= < 0.26, 0.26, 1.58, 1.00 >**

WT_A = <1 × 0.26, 1 × 0.26, 0 × 1.58, 1 × 1.00>
= <0.26, 0.26, 0.00, 1.00>

WT_B = <3 × 0.26, 3 × 0.26, 0 × 1.58, 3 × 1.00>
= <0.79, 0.79, 0.00, 3.00>

$$\mathbf{WT_k = TF_k \times IDF}$$

WT_C = <0 × 0.26, 9 × 0.26, 0 × 1.58, 0 × 1.00>
= <0.00, 2.37, 0.00, 0.00>

WT_D = <1 × 0.26, 1 × 0.26, 6 × 1.58, 1 × 1.00>
= <0.26, 0.26, 9.51, 1.00>

WT_E = <1 × 0.26, 1 × 0.26, 1 × 1.58, 0 × 1.00>
= <0.26, 0.26, 1.58, 0.00>

WT_F = <1 × 0.26, 0 × 0.26, 0 × 1.58, 0 × 1.00> = <0.26, 0.00, 0.00, 0.00>

Example

Terms, in order: "care", "cat", "dog" "persian".
Doc A WT_A **= <0.26, 0.26, 0.00, 1.00>**
...

Query Q: "Do cats care for other cats?"

- **TF_ Q = <1, 2, 0, 0>**

We do not weight the query by idf (empirically seems better), so this is also **WT_Q = <1, 2, 0, 0>**

To compare query Q and document A, compute cosine

$$COS_{Q,A} = \frac{\sum_{j=1}^{j=M_T} \left(W_Q[j] \times W_A[j]\right)}{\sqrt{\sum_{j=1}^{j=M_T} \left(W_Q[j]^2\right) \times \sum_{j=1}^{j=M_T} \left(W_A[j]^2\right)}}$$

W_Q • W_A = 1×0.26 + 2×0.26 + 0×0 + 0×1 = .78

|W_Q| = sqrt(1+4+0+0) = 2.23

|W_A| = sqrt(.26×.26+.26×.26+0+1) = 1.07

sim(Q,A)
= .78/(2.23×1.07)
= .33

IV. InfoRetr with Relevance Feedback

- $\mathcal{L}_{question}$: query document (as before)
- $\mathcal{L}_{answer}$: *ranked/ordered list* of documents (but hopefully more useful/relevant to user)
- $\mathcal{L}_{tell2}$: **preliminary list of docs** (*presumed* relevant by the system) **annotated by human with +, – to indicate *actual* relevance (i.e. $\mathcal{L}_{tell2}$ = {(D_1,+),(D_2,+),(D_3,–),(D_4,+),...}**

Idea: improve notion of "relevance" being used for that query

Relevance feedback for vector model

- **Can be shown that *if* you knew complete set of relevant documents, the optimal query for it would be**

$$Q_{opt} = \frac{1}{|Cr|} \sum_{dj \in Cr} dj - \frac{1}{N - |Cr|} \sum_{dj \notin Cr} dj$$

- **Rocchio method** $$Q_1 = \alpha Q_0 + \frac{\beta}{|Dr|} \sum_{dj \in Dr} dj - \frac{\gamma}{|Dn|} \sum_{dj \in Dn} dj$$

  Q_0 is initial query. Q_1 is "improved query"
  $D_{\mathbf{r}}$ = set of docs retrieved marked relevant by user
  $D_{\mathbf{n}}$ = set of irrelevant docs retrieved
  α =1; β =.75, γ =.25 typically.

- **So, terms in original query are "reweighted", <u>and</u> query is "expanded" with terms appearing in relevant documents, and somewhat "trimmed" of terms in irrelevant documents**
- **Simple, gives reasonable results empirically, but unprincipled**

Measuring the *Performance* of Retrieval

Precision -

- what percentage of the retrieved documents are relevant to the query
  - low precision --> many irrelevant documents for the user to look at and discard --> bad

Recall -

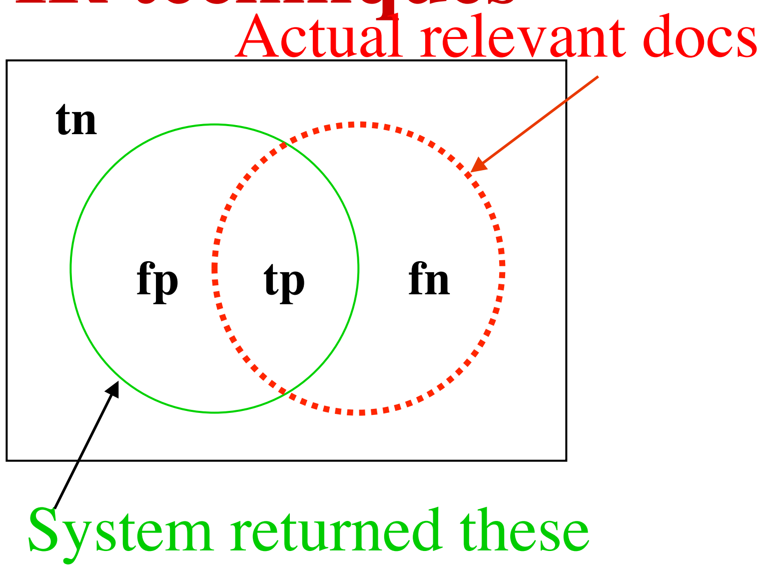
- what percentage of the documents relevant to the query (from the point of view of the user) were retrieved
  - low recall --> many documents missed --> very bad

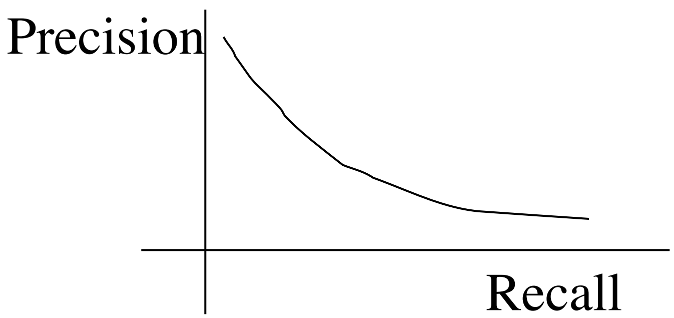
Recall vs. precision

One could increase recall by retrieving many documents (down to a low level of relevance ranking), but then many irrelevant documents would be fetched, reducing precision.

Measuring *Performance* of IR techniques

- **Precision** $\frac{tp}{tp + fp}$
  - Proportion of selected items that are correct
- **Recall** $\frac{tp}{tp + fn}$
  - Proportion of target items that were selected
- **Precision-Recall curve**
  - But a system could returns just 1 doc, sure to be right!? Or return all docs to be fully precise!?
  - Prevision vs Recall curve





Precision/Recall Curves - one approach

11-point recall-precision curve

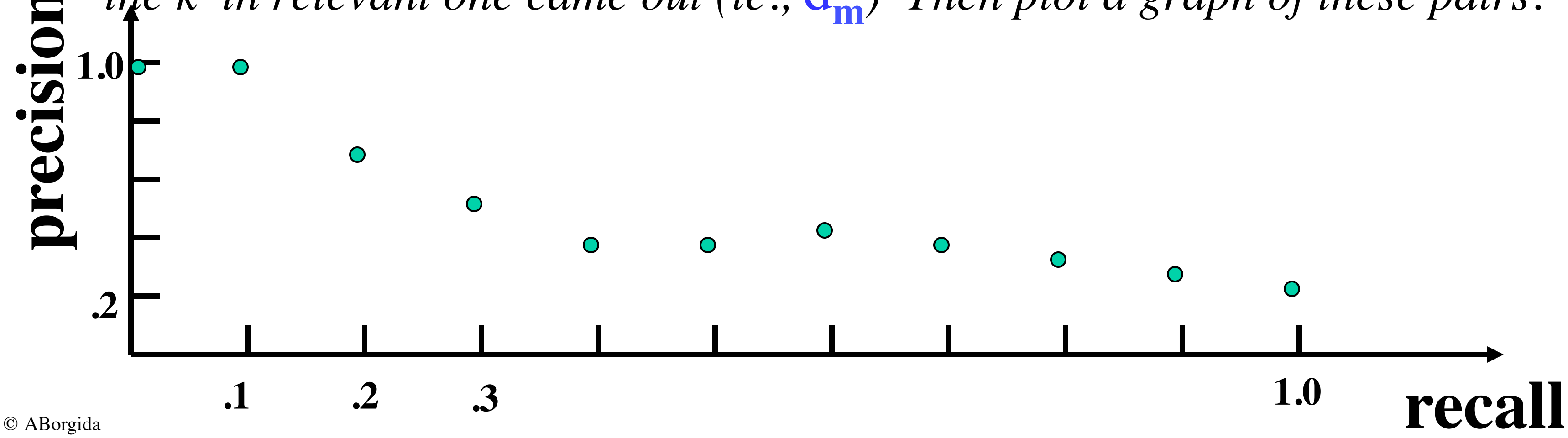
Example: Suppose for a given query, 10 documents are relevant (in blue below). Suppose when all documents are ranked in descending similarities, we have

d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9 d_{10} d_{11} d_{12} d_{13} d_{14} d_{15} d_{16} d_{17} d_{18} d_{19} d_{20} d_{21} d_{22} d_{23} d_{24} d_{25} d_{26} d_{27} d_{28} d_{29} d_{30} d_{31} …

After each relevant *blue* document, compute precision & recall *up to that point:*

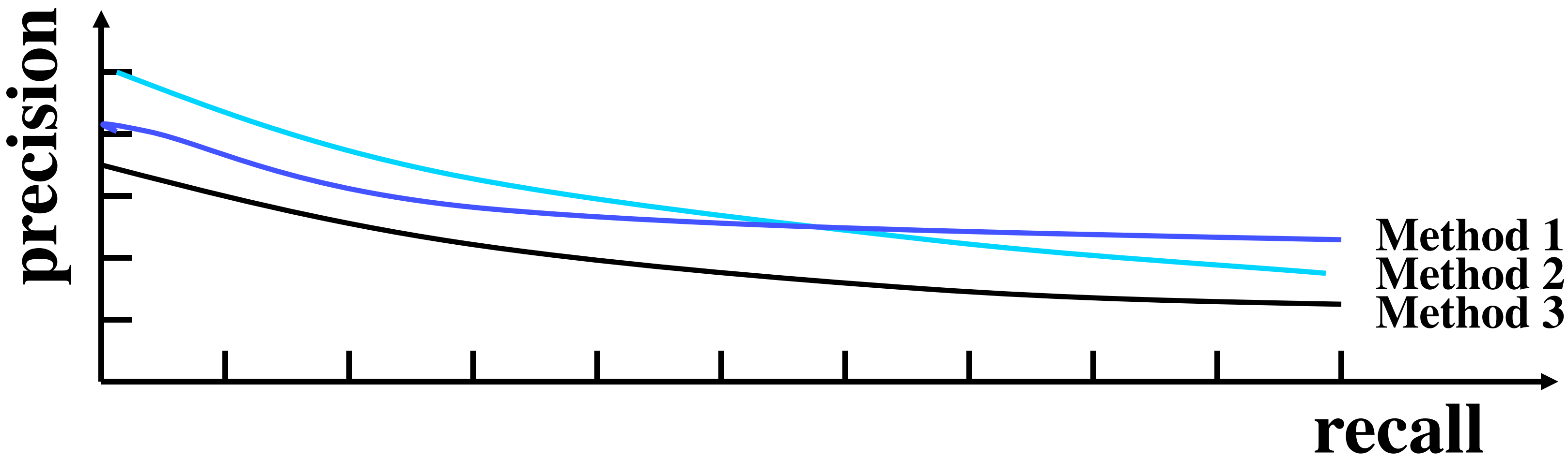
(1/1,1/10), (2/3,2/10), (3/6,3/10), (4/10,4/10),(5/12,5/10),...,(10/29,10/10)

Note pattern: (k/m,k/10) where m is how many docs where retrieved by the time the k'th relevant one came out (ie., d_m) Then plot a graph of these pairs.



Precision Recall Curves…

When evaluating the retrieval effectiveness of different text retrieval systems or methods, a large number of queries are used and their average 11-point recall-precision curve is plotted.



- Methods 1 and 2 are better than method 3.
- Method 1 is better than method 2 when high recall is needed.