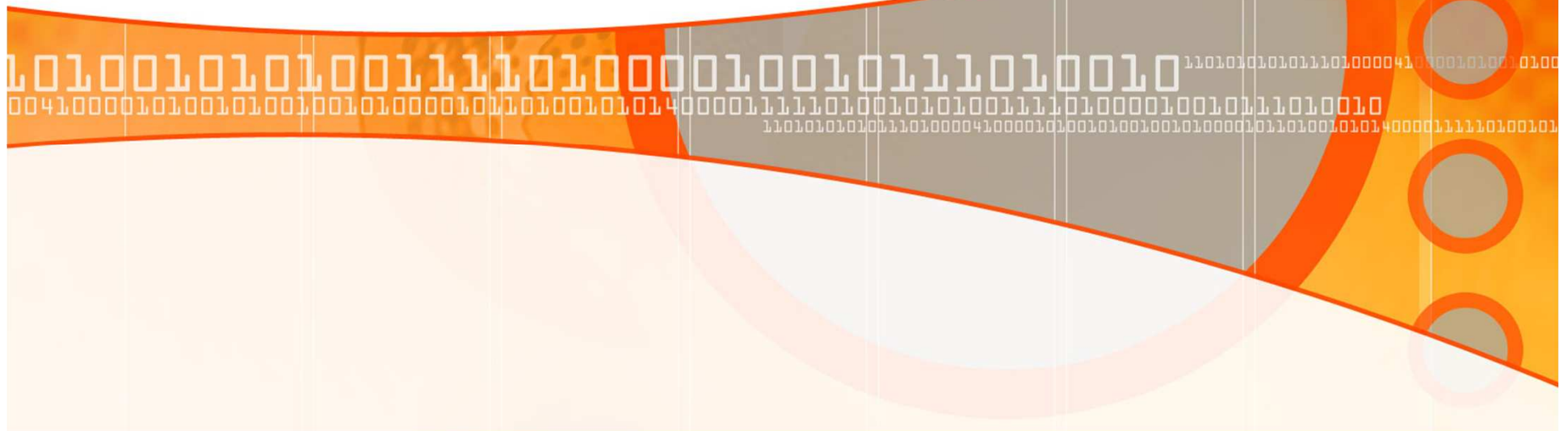


# Language Technology I

Information Retrieval

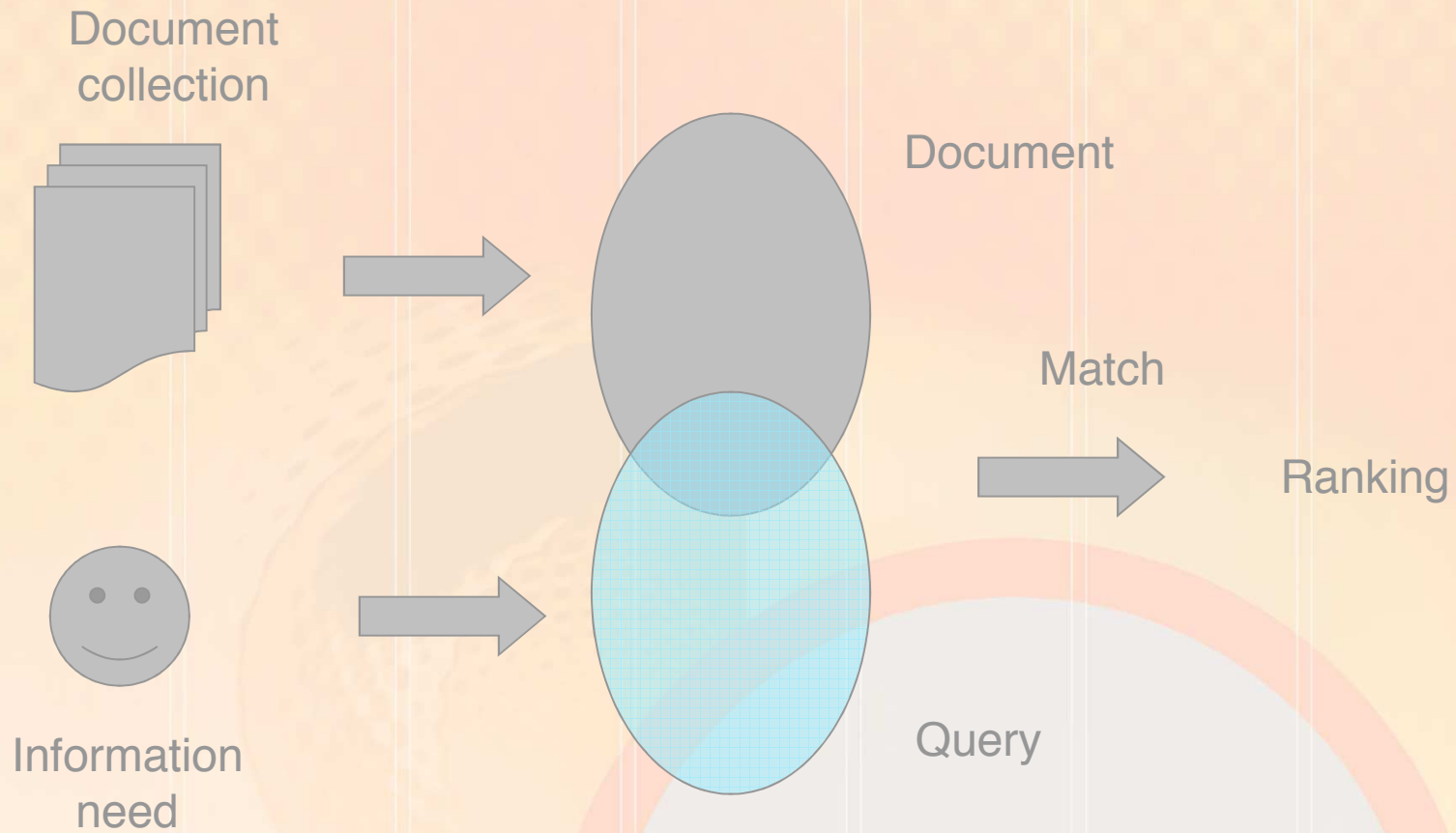


# Information Retrieval

- Traditional information retrieval is basically text search
  - *A collection of text documents*
  - *Documents are generally high-quality and designed to convey information*
  - *Documents are assumed to have no structure beyond words*
- Searches are generally based on meaningful phrases
- The goal is to find the document(s) that best match the search phrase, according to a search model



# Basic Model



# Terminology

- Document
  - *Unit of text indexed in the system*
  - *Result of the retrieval*
- IR systems usually adopt index terms to process queries
- Index term:
  - *a keyword or group of selected words*
  - *any word (more general)*
- An inverted index is built for the chosen index terms
  - *D0 = "it is what it is", D1 = "what is it" and D2 = "it is a banana"*
  - *"a": {D2}*
  - *"banana": {D2}*
  - *"is": {D0, D1, D2}*
  - *"it": {D0, D1, D2}*
  - *"what": {D0, D1}*
- Query
  - *User's information need as a set of terms*

## IR models

- An IR model is characterized by three parameters:
  - *representations for documents and queries*
  - *matching strategies for assessing the relevance of documents to a user query*
  - *methods for ranking query output*
- **Classic models**
  - *Boolean*
  - *Vector space*
  - *Probabilistic*

### Set Theoretic

- *Boolean model*
- *Fuzzy model*
- *Extended boolean model*

### Algebraic

- *Vector space model*
- *Generalized vector model*
- *Latent semantic index*
- *Neural networks model*

### Probabilistic

- *Probabilistic model*
- *Inference network*
- *Belief network*



## Boolean Model

- Based on set theory and Boolean algebra
  - *Documents are sets of terms*
  - *Queries are Boolean expressions on terms*
- **D:** set of words (indexing terms) present in a document
  - *each term is either present (1) or absent (0)*
- **Q:** A boolean expression
  - *terms are index terms*
  - *operators are AND, OR, and NOT*
- **Matching:** Boolean algebra over sets of terms and sets of documents
- No term weighting is allowed

# Boolean Model - Example

---

## Terms:

Index vocabular = {Corsica, Sardinia, Beach, flat, mountains}

## Documents:

document d1 = {Sardinia, Beach, flat}

document d2 = {Corsica, Beach, flat}

document d3 = {Corsica, mountains}

## Queries:

Corsica

flat

flat and Corsica

flat or Corsica

flat and not Corsica

## Yields:

{d2, d3}

{d1, d2}

{d2}

{d1, d2, d3}

{d1}

## Boolean Model example

$((text \vee information) \wedge retrieval \wedge \neg theory)$

- “Information Retrieval”  $X$
- “Information Theory”
- “Modern Information Retrieval: Theory and Practice”
- “Text Compression”

## Boolean Model Disadvantages

- Similarity function is boolean
  - *Exact-match only, no partial matches*
  - *Retrieved documents not ranked*
- All terms are equally important
  - *Boolean operator usage has much more influence than a critical word*
- Query language is expressive but complicated

# Issues for Vector Space Model

- How to determine important words in a document?
  - Word sense?
  - Word n-grams (and phrases, idioms,...) → terms
- How to determine the degree of importance of a term within a document and within the entire collection?
- How to determine the degree of similarity between a document and the query?
- In the case of the web, what is a collection and what are the effects of links, formatting information, etc.?

# The Vector-Space Model

- Assume  $t$  distinct terms remain after preprocessing; call them index terms or the vocabulary.
- These “orthogonal” terms form a vector space.

Dimension =  $t$  = |vocabulary|

- Each term,  $i$ , in a document or query,  $j$ , is given a real-valued weight,  $w_{ij}$ .
- Both documents and queries are expressed as  $t$ -dimensional vectors:

$$d_j = (w_{1j}, w_{2j}, \dots, w_{tj})$$

# Document Collection

- A collection of  $n$  documents can be represented in the vector space model by a term-document matrix.
- An entry in the matrix corresponds to the “weight” of a term in the document; zero means the term has no significance in the document or it simply doesn't exist in the document.

$$\begin{pmatrix} & T_1 & T_2 & \dots & T_t \\ D_1 & w_{11} & w_{21} & \dots & w_{t1} \\ D_2 & w_{12} & w_{22} & \dots & w_{t2} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & w_{1n} & w_{2n} & \dots & w_{tn} \end{pmatrix}$$

# Term Weights: Term Frequency

- More frequent terms in a document are more important, i.e. more indicative of the topic.

$f_{ij}$  = frequency of term  $i$  in document  $j$

- May want to normalize *term frequency* ( $tf$ ) across the entire corpus:

$$tf_{ij} = f_{ij} / \max\{f_{ij}\}$$

# Term Weights: Inverse Document Frequency

- Terms that appear in many *different* documents are *less* indicative of overall topic.

$df_i$  = document frequency of term  $i$

= number of documents containing term  $i$

$idf_i$  = inverse document frequency of term  $i$ ,

=  $\log_2(N/df_i)$

( $N$ : total number of documents)

- An indication of a term's *discrimination* power.
- Log used to dampen the effect relative to  $tf$ .

# TF-IDF Weighting

- A typical combined term importance indicator is *tf-idf weighting*:

$$w_{ij} = tf_{ij} idf_i = tf_{ij} \log_2 (N / df_i)$$

- A term occurring frequently in the document but rarely in the rest of the collection is given high weight.
- Many other ways of determining term weights have been proposed.
- Experimentally, *tf-idf* has been found to work well.

# Computing TF-IDF - An Example

Given a document containing terms with given frequencies:

A(3), B(2), C(1)

Assume collection contains 10,000 documents and document frequencies of these terms are:

A(50), B(1300), C(250)

Then:

A:  $tf = 3/3$ ;  $idf = \log(10000/50) = 5.3$ ;  $tf-idf = 5.3$

B:  $tf = 2/3$ ;  $idf = \log(10000/1300) = 2.0$ ;  $tf-idf = 1.3$

C:  $tf = 1/3$ ;  $idf = \log(10000/250) = 3.7$ ;  $tf-idf = 1.2$

# Similarity Measure

- A **similarity measure** is a function that computes the *degree of similarity* between two vectors.
- Using a similarity measure between the query and each document:
  - It is possible to rank the retrieved documents in the order of presumed relevance.
  - It is possible to enforce a certain threshold so that the size of the retrieved set can be controlled.

# Similarity Measure - Inner Product

- Similarity between vectors for the document  $\mathbf{d}_j$  and query  $\mathbf{q}$  can be computed as the vector inner product:

$$\text{sim}(\mathbf{d}_j, \mathbf{q}) = \sum_{i=1}^t \mathbf{d}_j \cdot \mathbf{q} = w_{ij} \cdot w_{iq}$$

where  $w_{ij}$  is the weight of term  $i$  in document  $j$  and  $w_{iq}$  is the weight of term  $i$  in the query

- For binary vectors, the inner product is the number of matched query terms in the document (size of intersection).
- For weighted term vectors, it is the sum of the products of the weights of the matched terms.

# Properties of Inner Product

- Favors long documents with a large number of unique terms.
- Measures how many terms matched but not how many terms are *not* matched.

# Inner Product -- Examples

Binary:

retrieval  
database  
architecture  
computer  
text  
management  
information

- D = 1, 1, 1, 0, 1, 1, 0

- Q = 1, 0, 1, 0, 0, 1, 1

Size of vector = size of vocabulary = 7

0 means corresponding term not found in document or query

$$\text{sim}(D, Q) = 3$$

Weighted:

$$D_1 = 2T_1 + 3T_2 + 5T_3 \quad D_2 = 3T_1 + 7T_2 + 1T_3$$

$$Q = 0T_1 + 0T_2 + 2T_3$$

$$\text{sim}(D_1, Q) = 2*0 + 3*0 + 5*2 = 10$$

$$\text{sim}(D_2, Q) = 3*0 + 7*0 + 1*2 = 2$$

# Comments on Vector Space Models

- Simple, mathematically based approach.
- Considers both local (*tf*) and global (*idf*) word occurrence frequencies.
- Provides partial matching and ranked results.
- Tends to work quite well in practice despite obvious weaknesses.
- Allows efficient implementation for large document collections.

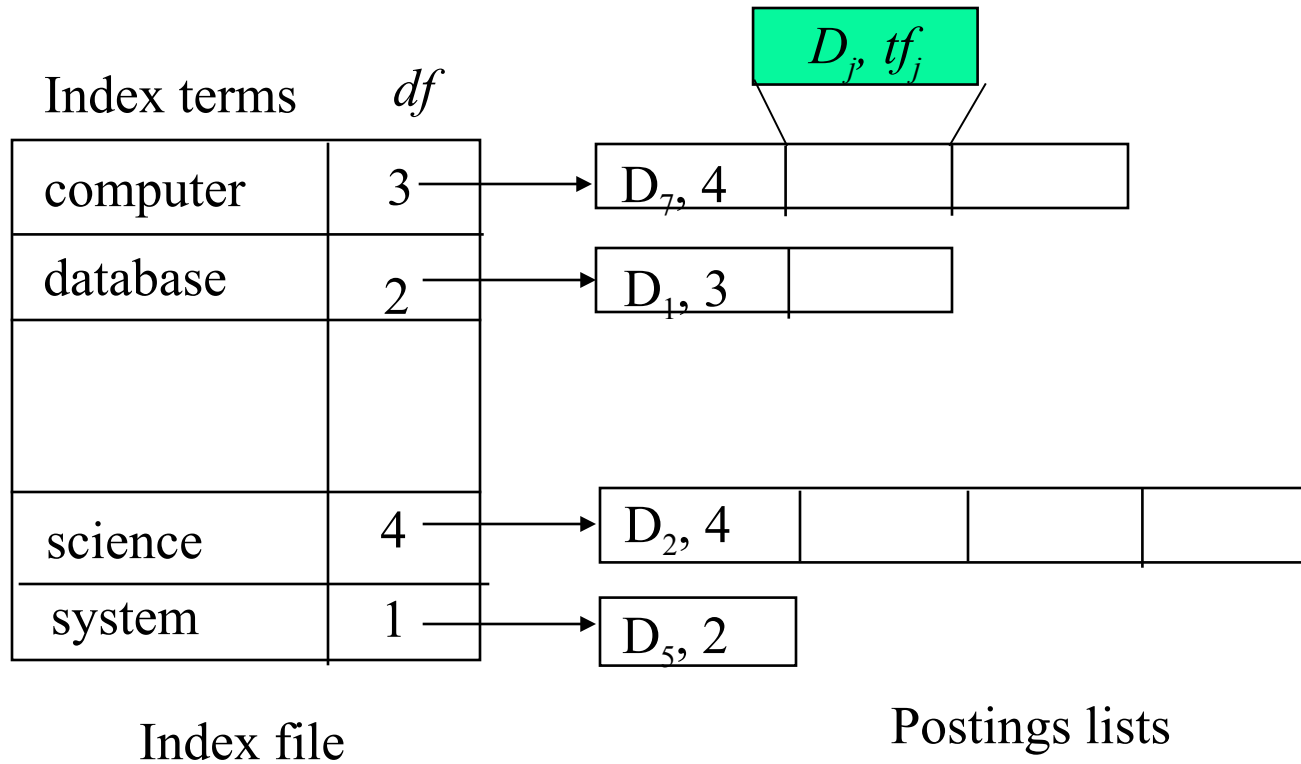
# Sparse Vectors

- Vocabulary and therefore dimensionality of vectors can be very large,  $\sim 10^4$ .
- However, most documents and queries do not contain most words, so vectors are sparse (i.e. most entries are 0).
- Need efficient methods for storing and computing with sparse vectors.

# Implementation Based on Inverted Files

- In practice, document vectors are not stored directly; an inverted organization provides much better efficiency.
- The keyword-to-document index can be implemented as a hash table, a sorted array, or a tree-based data structure (trie, B-tree).
- Critical issue is logarithmic or constant-time access to token information.

# Inverted Index

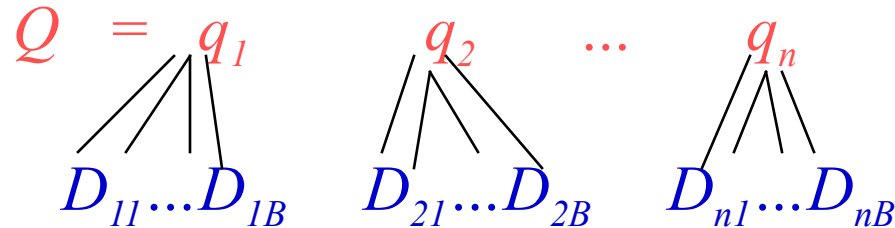


# Retrieval with an Inverted Index

- Tokens that are not in both the query and the document do not effect cosine similarity.
  - Product of token weights is zero and does not contribute to the dot product.
- Usually the query is fairly short, and therefore its vector is *extremely* sparse.
- Use inverted index to find the limited set of documents that contain at least one of the query words.

# Inverted Query Retrieval Efficiency

- Assume that, on average, a query word appears in  $B$  documents:



- Then retrieval time is  $O(|Q| B)$ , which is typically, **much** better than naïve retrieval that examines all  $N$  documents,  $O(|V| N)$ , because  $|Q| \ll |V|$  and  $B \ll N$ .

## Vector Space Model, Summarized

- The best term-weighting schemes tf-idf weights:

$$w_{ij} = f(i,j) * \log(N/n_i)$$

- For the query term weights, a suggestion is

$$w_{iq} = (0.5 + [0.5 * \text{freq}(i,q) / \max(\text{freq}(l,q))]) * \log(N / n_i)$$

- This model is very good in practice:
  - tf-idf works well with general collections
  - Simple and fast to compute
  - Vector model is usually as good as the known ranking alternatives

## Pros & Cons of Vector Model

- Advantages:
  - term-weighting improves quality of the answer set
  - partial matching allows retrieval of docs that approximate the query conditions
  - cosine ranking formula sorts documents according to degree of similarity to the query
- Disadvantages:
  - assumes independence of index terms; not clear if this is a good or bad assumption

## Comparison of Classic Models

- **Boolean model** does not provide for partial matches and is considered to be the weakest classic model
- Some experiments indicate that the **vector model** outperforms the third alternative, the **probabilistic model**, in general
  - Recent IR research has focused on improving probabilistic models – but these haven't made their way to Web search
- Generally we use a variation of the vector model in most text search systems

## Why evaluate IR systems?

- *There are many retrieval models/ algorithms/ systems, which one is the best?*
- *What is the best component for:*
  - *Ranking function (dot-product, cosine, ...)*
  - *Term selection (stopword removal, stemming...)*
  - *Term weighting (TF, TF-IDF,...)*
- *How far down the ranked list will a user need to look to find some/all relevant documents?*

## Difficulties in Evaluating IR Systems

- *Effectiveness is related to the **relevancy** of retrieved items.*
- *Relevancy is not typically binary but continuous.*
- *Even if relevancy is binary, it can be a difficult judgment to make.*
- *Relevancy, from a human standpoint, is:*
  - *Subjective: Depends upon a specific user's judgment.*
  - *Situational: Relates to user's current needs.*
  - *Cognitive: Depends on human perception and behavior.*
  - *Dynamic: Changes over time.*

## **Human Labeled Corpora (Gold Standard)**

- *Start with a corpus of documents.*
- *Collect a set of queries for this corpus.*
- *Have one or more human experts exhaustively label the relevant documents for each query.*
- *Typically assumes binary relevance judgments.*
- *Requires considerable human effort for large document/query corpora.*

# What to Evaluate?

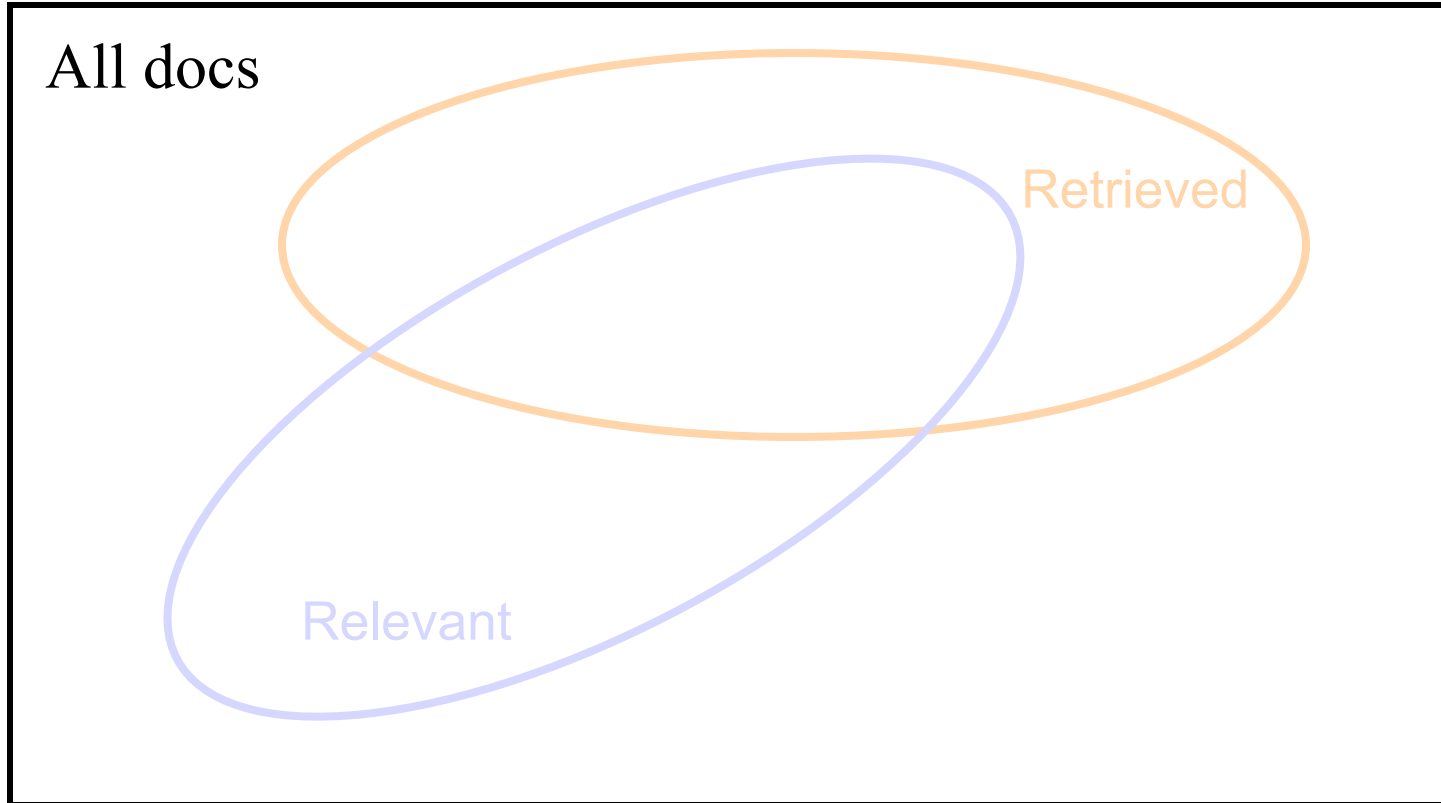
What can be measured that reflects users' ability to use system? (Cleverdon 66)

- Coverage of Information
- Form of Presentation
- Effort required/Ease of Use
- Time and Space Efficiency

effectiveness

- Recall
  - proportion of relevant material actually retrieved
- Precision
  - proportion of retrieved material actually relevant

# Relevant vs. Retrieved

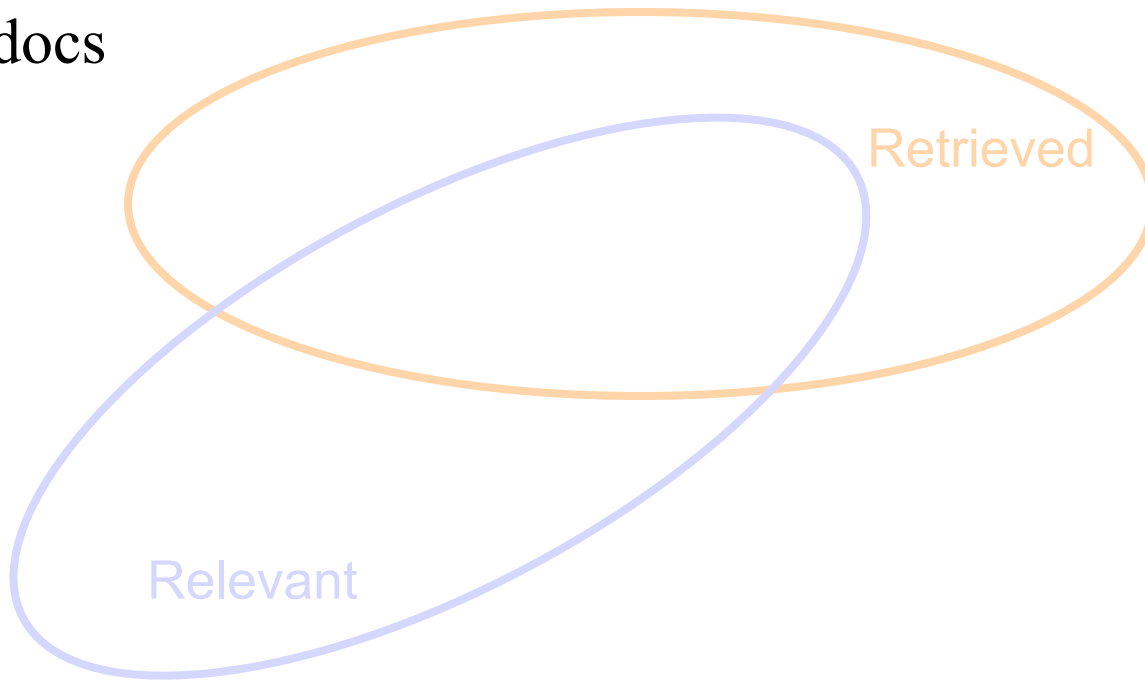


# Precision vs. Recall

$$\text{Precision} = \frac{|\text{RelRetrieved}|}{|\text{Retrieved}|}$$

$$\text{Recall} = \frac{|\text{RelRetrieved}|}{|\text{Rel in Collection}|}$$

All docs

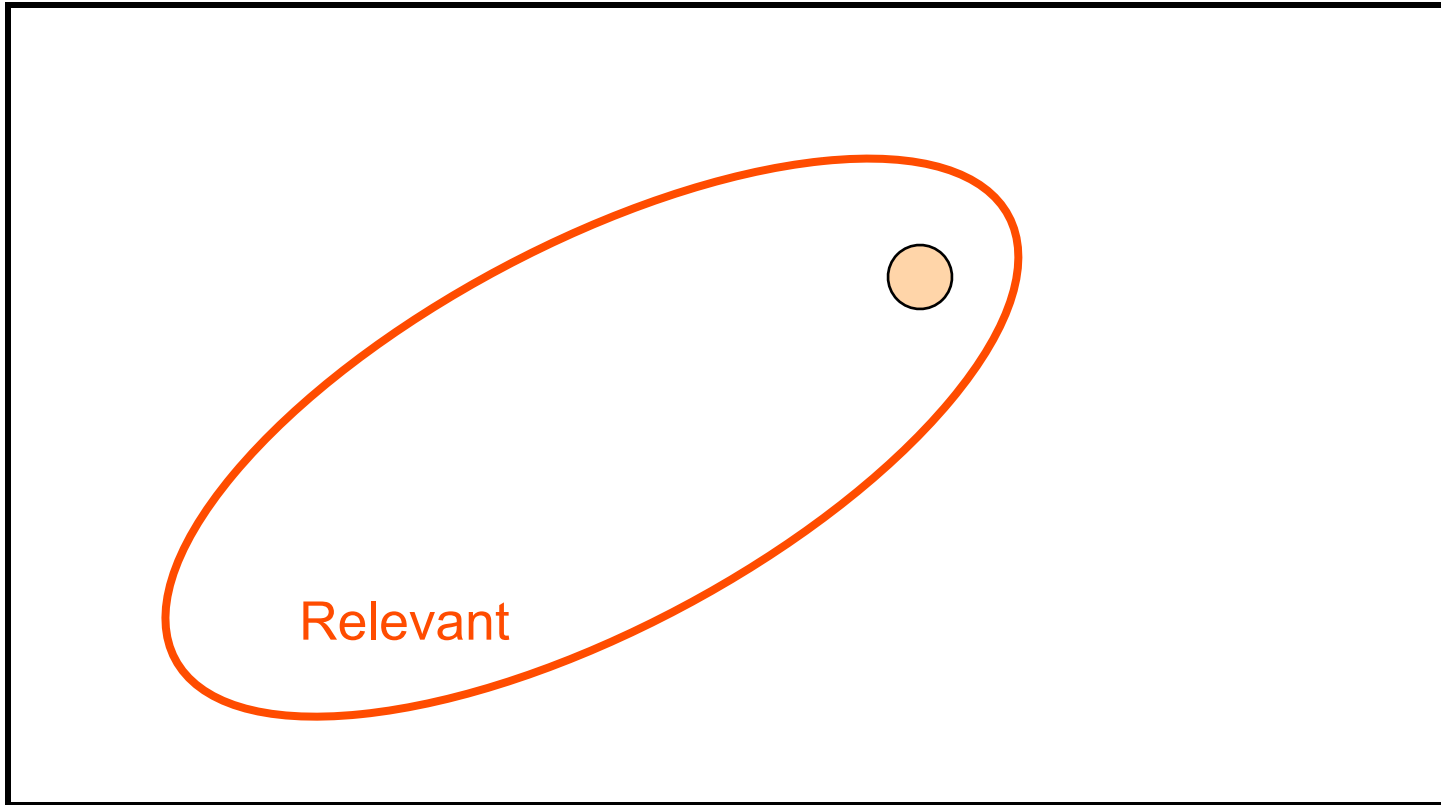


# Why Precision and Recall?

Get as much good stuff while at the same time getting as little junk as possible.

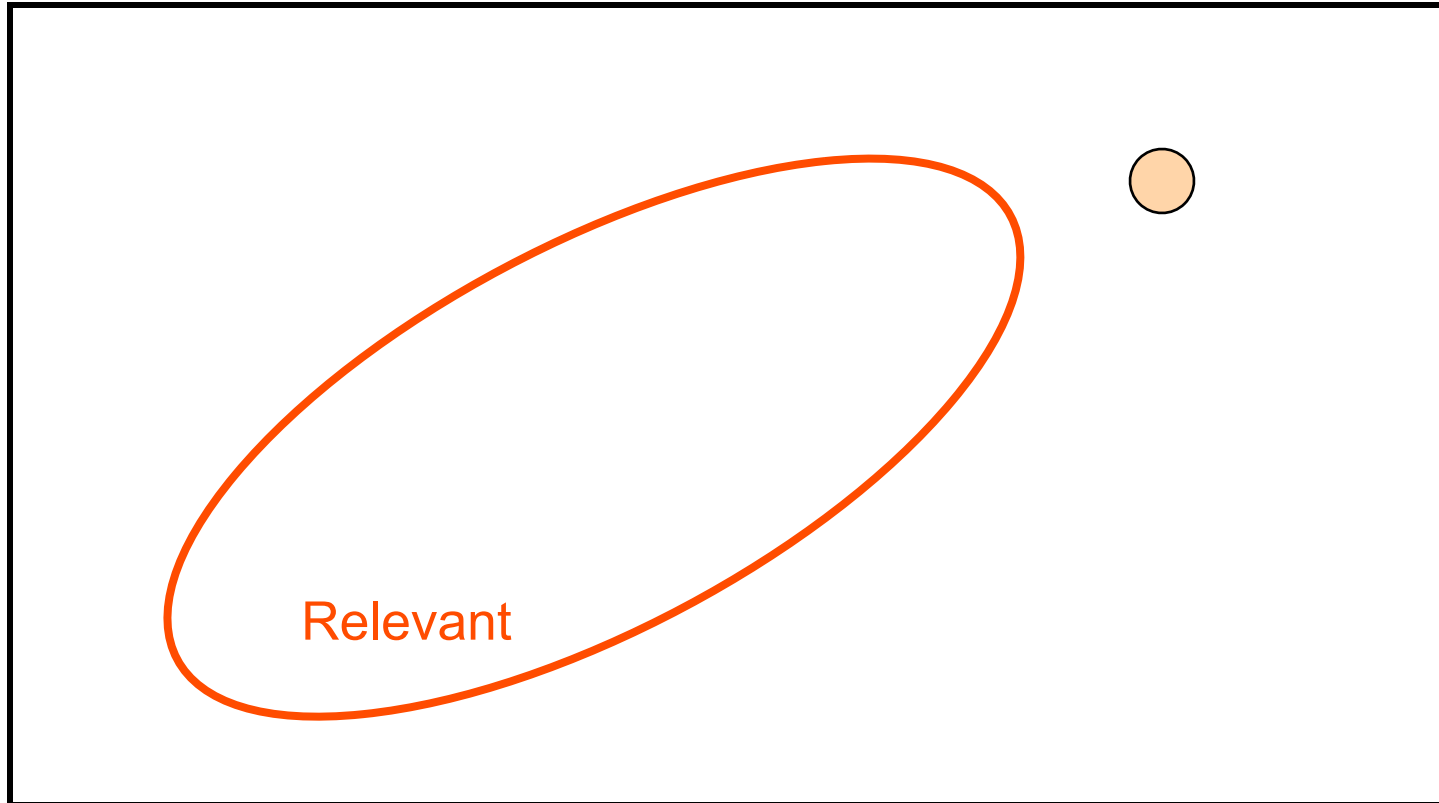
# Retrieved vs. Relevant Documents

Very high precision, very low recall



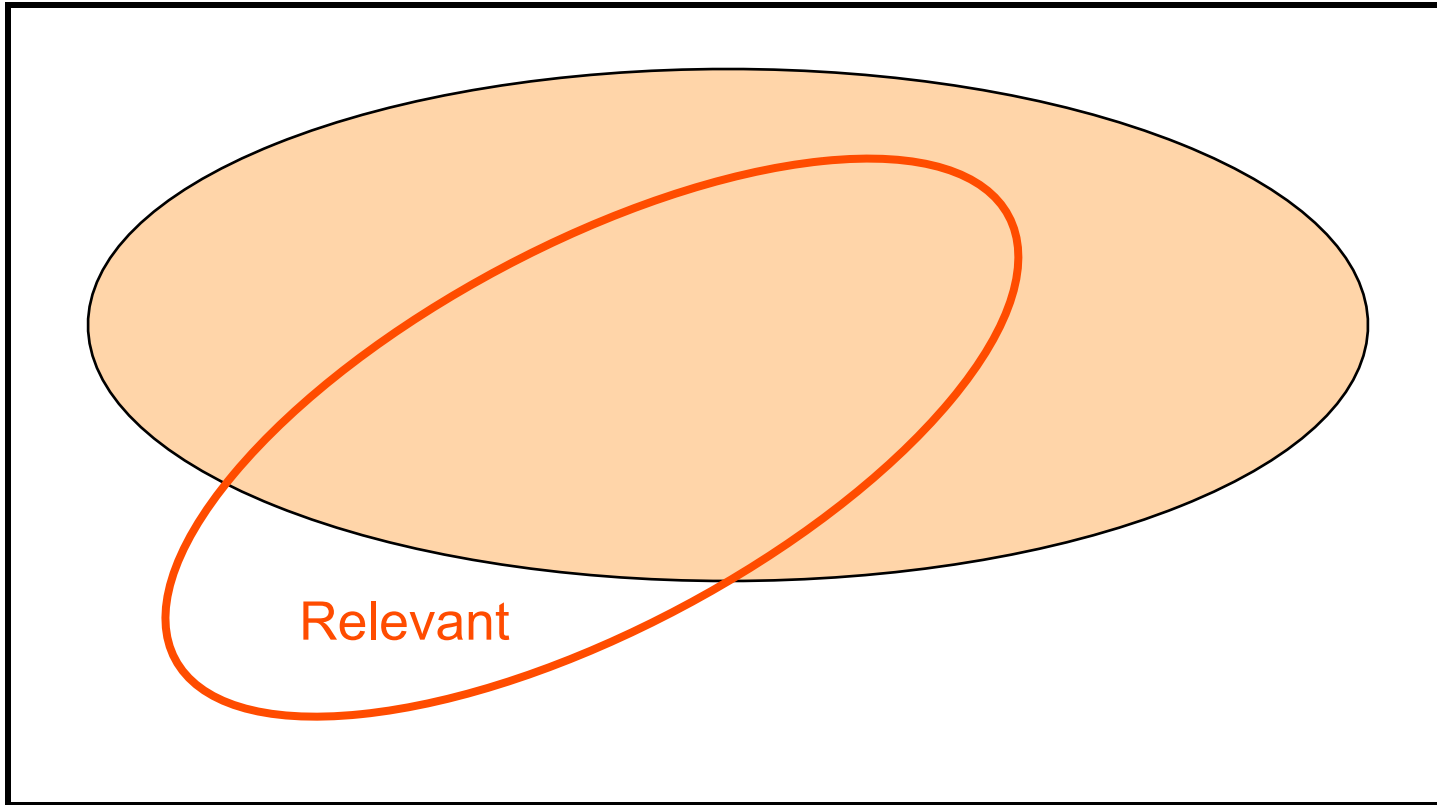
# Retrieved vs. Relevant Documents

Very low precision, very low recall (0 in fact)



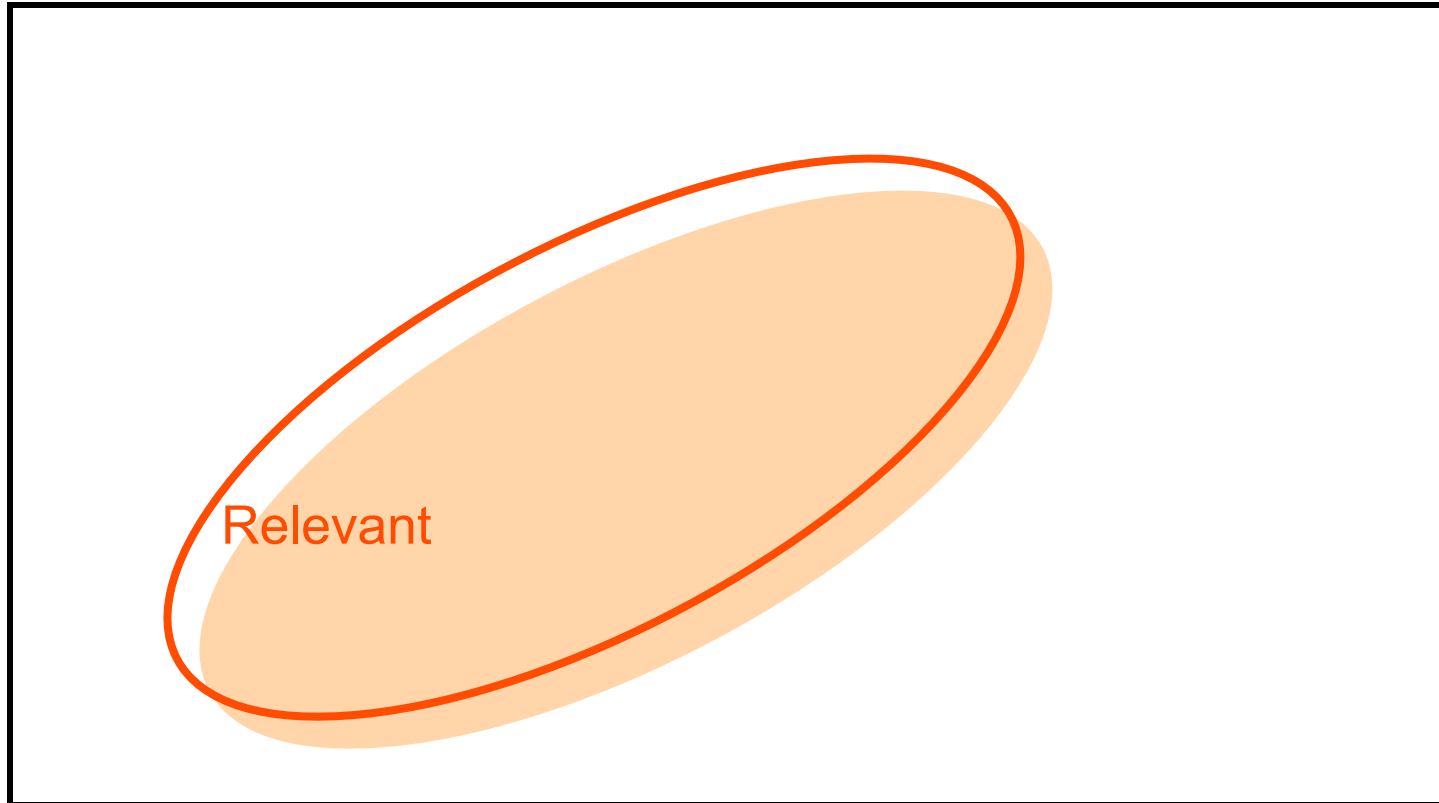
# Retrieved vs. Relevant Documents

High recall, but low precision



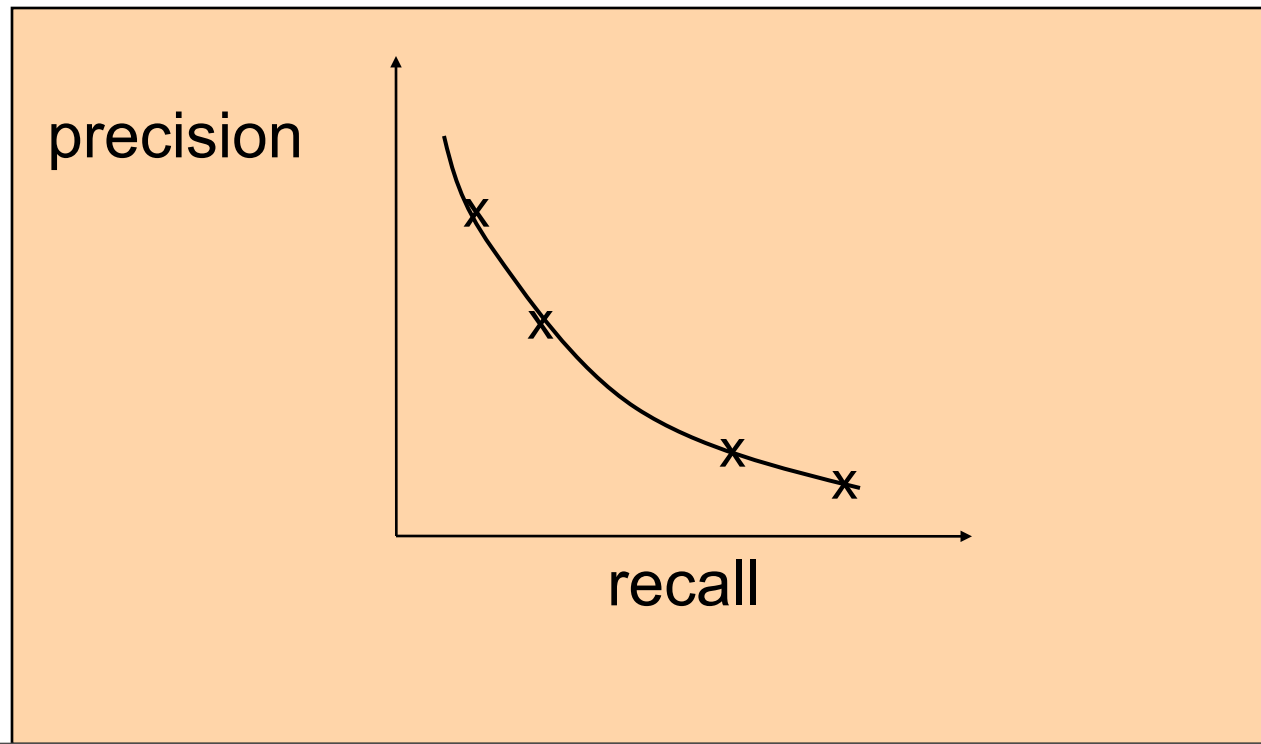
# Retrieved vs. Relevant Documents

High precision, high recall (at last!)



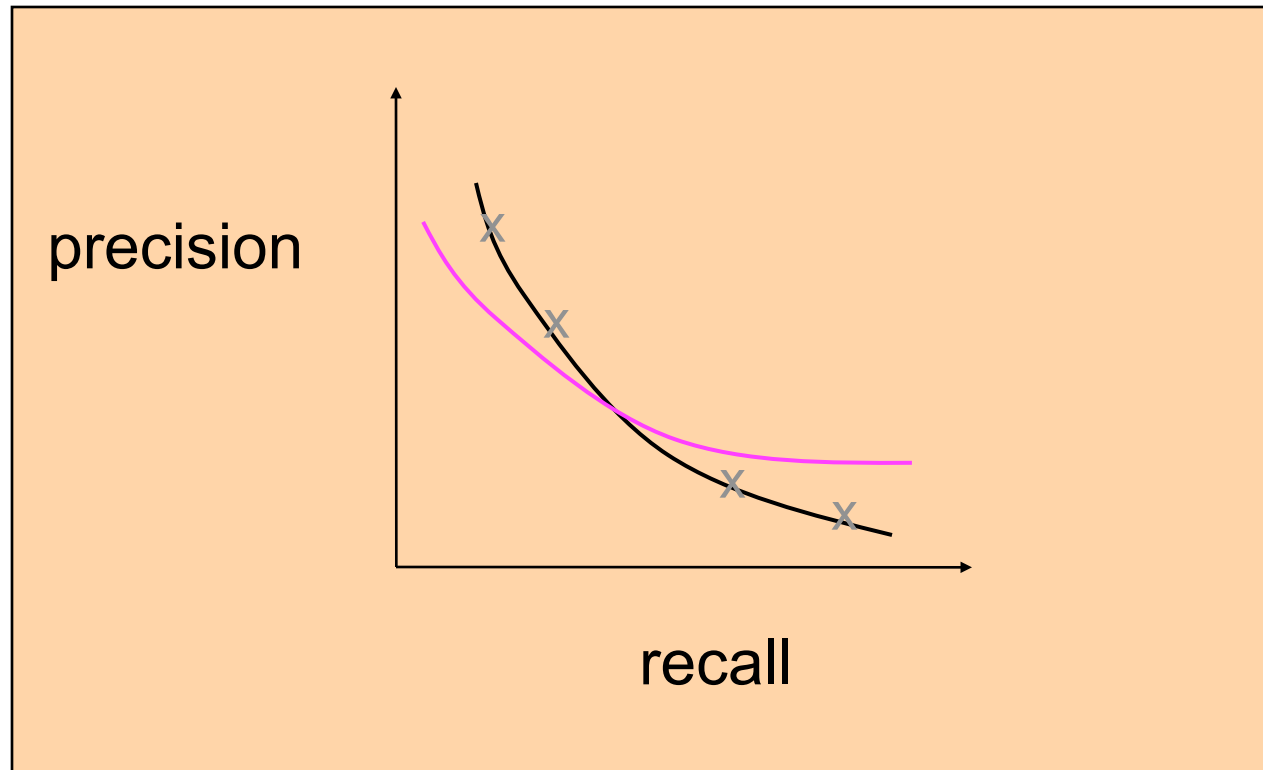
# Precision/Recall Curves

- There is a tradeoff between Precision and Recall
- So measure Precision at different levels of Recall
- Note: this is an AVERAGE over MANY queries



# Precision/Recall Curves

- Difficult to determine which of these two hypothetical results is better:





# Document Cutoff Levels

- Another way to evaluate:
  - Fix the number of documents retrieved at several levels:
    - top 5
    - top 10
    - top 20
    - top 50
    - top 100
    - top 500
  - Measure precision at each of these levels
  - Take (weighted) average over results
- This is a way to focus on how well the system ranks the first k documents.

# Problems with Precision/Recall

- Can't know true recall value
  - except in small collections
- Precision/Recall are related
  - A combined measure sometimes more appropriate
- Assumes batch mode
  - Interactive IR is important and has different criteria for successful searches
- Assumes a strict rank ordering matters.

# Relation to Contingency Table

	Doc is Relevant	Doc is NOT relevant
Doc is retrieved	a	b
Doc is NOT retrieved	c	d

- Accuracy:  $(a+d) / (a+b+c+d)$
- Precision:  $a/(a+b)$
- Recall: ?
- Why don't we use Accuracy for IR?
  - (Assuming a large collection)
  - Most docs aren't relevant
  - Most docs aren't retrieved
  - Inflates the accuracy value

# F-Measure

- One measure of performance that takes into account both recall and precision.
- Harmonic mean of recall and precision:

$$F = \frac{2PR}{P + R} = \frac{2}{\frac{1}{R} + \frac{1}{P}}$$

- Compared to arithmetic mean, both need to be high for harmonic mean to be high.

# E Measure (parameterized F Measure)

- A variant of F measure that allows weighting emphasis on precision over recall:

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

- Value of  $\beta$  controls trade-off:
  - $\beta = 1$ : Equally weight precision and recall ( $E=F$ ).
  - $\beta > 1$ : Weight recall more.
  - $\beta < 1$ : Weight precision more.