# Information Retrieval Basics

**Kira Radinsky**

Many of the following slides are courtesy of Ronny Lempel (Yahoo!),
Aya Soffer and David Carmel (IBM)

# Indexing/Retrieval – Basics

- 2 Main Stages
  - Indexing process - involves pre-processing and storing of information into a repository – an *index*
  - Retrieval/runtime process - involves issuing a query, accessing the index to find documents *relevant* to the query
- Basic Concepts:
  - Document – any piece of information (book, article, database record, Web page, image, video, song)
    - usually textual data
  - Query – some text representing the user's information need
  - Relevance – a binary relation (a predicate) between documents and queries R(d,q)
    - Obviously a simplification of subjective quality with many shades of gray

# Defining Document Relevance to a Query

- Although Relevance is the basic concept of IR, it lacks a precise definition

- Difficulties :
  - User/intent dependent
  - Time/place dependent
  - In practice, depends on what other documents are deemed relevant
    - And even on what other documents were retrieved for the same query

- Simplified assumption:
  - D - the set of all documents in the collection (corpus),
  - Q - the set of all possible queries,
  - R: D X Q $\rightarrow$ {0,1} is well defined
  - d is "relevant" to q *iff* R(d,q) = 1
- Some models try to measure the probability of relevance: Pr(R|d,q)

# Index building: Text Profiling

- Documents/Queries are *profiled* to generate a canonical representation
- The profile is usually based on the set of indexing units (terms) in the text
- Indexing units (terms) are generally representative words in the text
  - How to select representative units?
  - For the moment, let’s take all the words in the given document/query

In the beginning God created the heaven and the earth . And the earth was without form and void.

..,and,,beginning,,created,,earth,,form,,god,,heaven,,in,,the,,void,,was,,without
.., 3 ,, 1 ,,1 ,, 2 ,, 1 ,, 1 ,, 1 ,, 1,, 4 ,, 1 ,, 1 ,, 1

# More Formally

Given a collection of documents (a corpus)

- All the terms in the collection can be labeled $t_1, t_2, \ldots, t_N$
- The profile of document $d_j$ is an N-dimensional vector,

$$d_j \rightarrow (w_{1j}, w_{2j}, \ldots, w_{Nj})$$

  - where
    - $w_{ij} = 0$ if $t_i$ does not appear in $d_j$
    - $w_{ij} > 0$ otherwise
- The N-dimensional vector space is conceptual – implementations will not actually manipulate such large vectors

# Index Representation as a (Sparse) Matrix

| A(t,d) | $d_1$ | $d_2$ | … | $d_M$ |
|---|---|---|---|---|
| $t_1$ | $w_{11}$ | $w_{12}$ | | $w_{1M}$ |
| $t_2$ | | | | |
| | | | | |
| $t_N$ | $w_{N1}$ | | | $w_{NM}$ |

Most entries are zero – certainly for large corpora

# Using the Index for Relevance Retrieval

- Assumption: a document not containing any query term is not relevant
- Given a simple query of one term $q=\{t_i\}$
- Use the index for retrieval:
  - 1. Retrieve all documents $d_j$ with $w_{ij} > 0$
  - 2. Sort them in decreasing order (in some models)
  - 3. Return the (ranked) list of "relevant" documents to the user
- In general: given a user's query $q=\{t_1 \ldots t_k\}$:
  - Disjunctive queries: return a (ranked) list of documents containing at least one of the query terms
  - Conjunctive queries: return a (ranked) list of documents containing all of the query terms

# The Boolean Model

- Simple model based on Set Theory

- Queries are specified as Boolean expressions
  - indexing units are words
  - Boolean Operator: OR, AND, NOT
  - Example:
    q =“java” AND “compilers” AND (“unix” OR “linux”)

- Relevance: A document is relevant to the query if it satisfies the query Boolean expression

# Boolean Model- Example

| A(t,d) | $d_1$ | $d_2$ | $d_3$ | $d_4$ | $d_5$ |
|---|---|---|---|---|---|
| a | 1 | 1 | 1 | 0 | 1 |
| b | 0 | 1 | 0 | 1 | 1 |
| c | 0 | 0 | 1 | 0 | 1 |

q = a AND (b OR (NOT c))

# Search Using an Inverted Index

- a → d1,d2,d3,d5
- b → d2,d4,d5
- c → d3,d5

- a → 1,1,1,0,1
- b → 0,1,0,1,1
- NOT c → 1,1,0,1,0

AND

1,1,0,0,1

OR

1,1,0,1,1

q = a AND (b OR (NOT c))

Results: d1, d2, d5

# Boolean Model – Pros & Cons

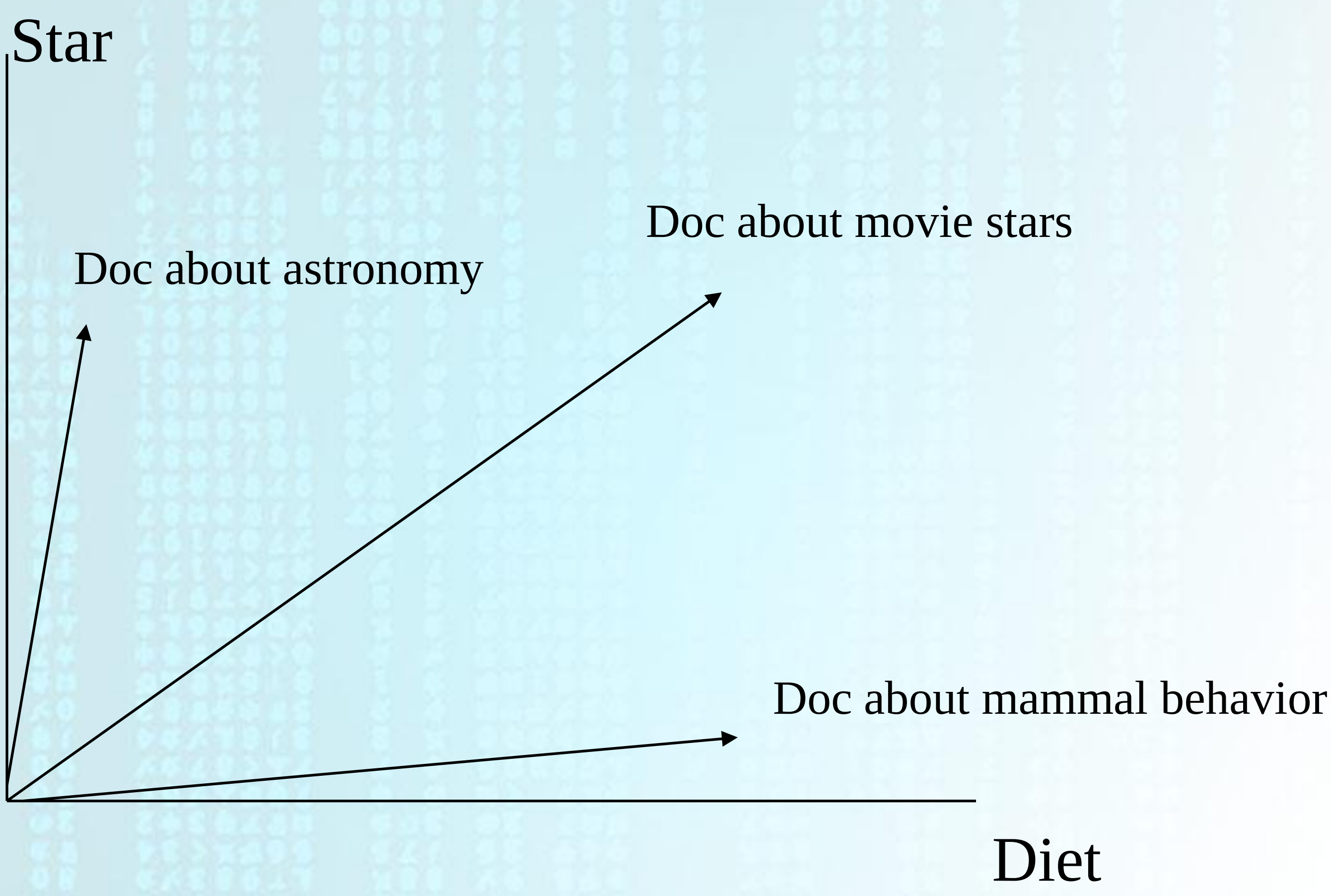
- Pros:
  - Fast (bitmap vector operations)
  - Binary decision (doc is "relevant" or not)
  - Some extensions are easy (e.g. synonym support)

- Cons:
  - Binary decision - what about ranking?
  - Who speaks Boolean?

# Vector Space Model

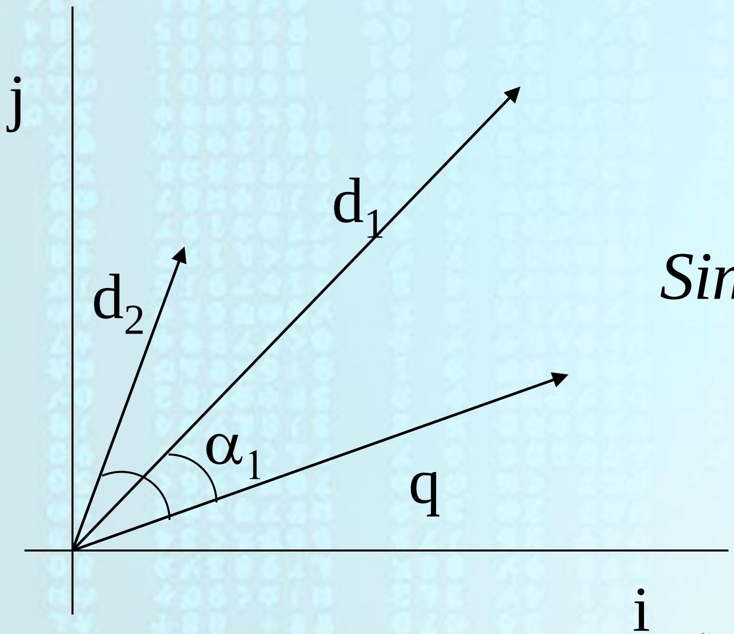
- Documents are represented as vectors in a (huge) N-dimensional space
  - N is the number of terms in the corpus, i.e. size of the lexicon/dictionary
- Query is a document like any other document
- Relevance – measured by similarity:
  - A document is relevant to the query if its representative vector is similar to the query’s representative vector

# Documents as Vectors



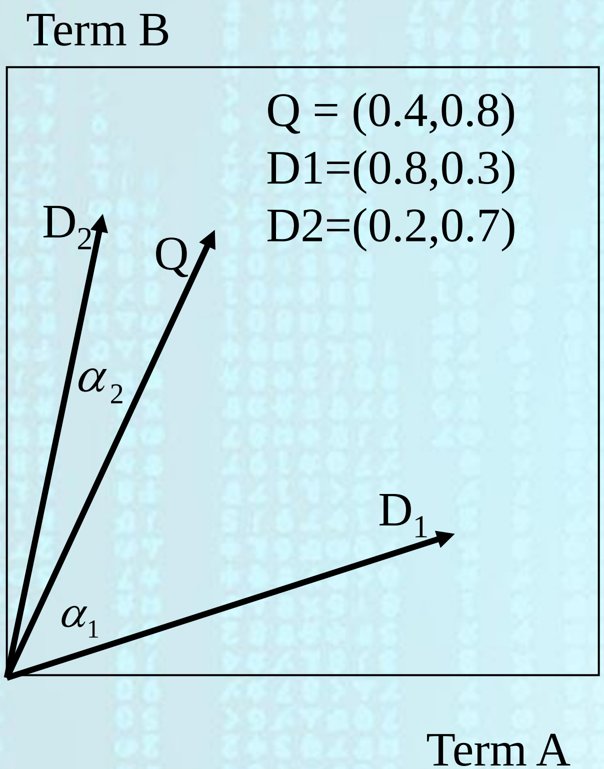
# Vector-space Model

- “Relevance” is measured by similarity - the cosine of the angle between doc-vectors and the query vector
- Need to represent the query as a vector in the same vector-space as the documents




$$Sim(q,d) = \frac{q \bullet d}{|q| \bullet |d|} = \frac{\sum_i w_{iq} w_{id}}{\sqrt{\sum_i w_{iq}^2 \sum_i w_{id}^2}}$$

# Example




$$Sim(q,d) = \frac{q \bullet d}{|q| \bullet |d|} = \frac{\sum_i w_{iq} w_{id}}{\sqrt{\sum_i w_{iq}^2 \sum_i w_{id}^2}}$$

$$sim(q,d_2) = \frac{(0.4 \cdot 0.2) + (0.8 \cdot 0.7)}{\sqrt{[(0.4)^2 + (0.8)^2] \cdot [(0.2)^2 + (0.7)^2]}}$$

$$= \frac{0.64}{\sqrt{0.42}} = 0.98$$

$$sim(q,d_1) = \frac{.56}{\sqrt{0.58}} = 0.74$$

# How to Determine the w(t,d) Weights?

- Binary weights:
  - $w_{i,j}$= 1 iff document $d_j$ contains term $t_i$, otherwise 0.
  - (e.g. the Boolean model)
- Term frequency (tf):
  - $w_{i,j}$= (number of occurrences of $t_i$ in $d_j$)
- What about term importance?
  - E.g. q="galaxy in space".
  - Should an occurrence of the query term "in" in a document contribute the same as an occurrence of the query term "galaxy"?

# Determining the w(t,d) Weights (cont)

tf x idf weighting scheme (Salton 73)

- tf – a monotonic function of the term frequency in the document,
  - e.g. $tf(t,d) = \log(freq(t,d) + 1)$
- idf – the inverse document frequency of a term – a decreasing function of the term total freq $N_t$
  - e.g: $idf(t) = \log(N / N_t)$ (for terms appearing at least once, N- #documents, $N_t$ -#documents with t)
  - Intuition: query terms that are rare in the corpus better distinguish the relevant documents from the irrelevant ones
  - $W_{i,j} = tf(t_i,d_j) * idf(t_i)$

# Vector Space Pros & Cons

- Pros
  - Terms weighting scheme improves retrieval effectiveness
  - Allows for approximate query matching
  - Cosine similarity is a good ranking measure
  - Simple and elegant, with a solid mathematical foundation

- Cons
  - Terms are not really orthogonal dimensions due to strong term relationships and dependencies
  - Ranking does not guarantee multiple term containment
    - Default semantics of search engines is “AND” for multi-term queries (conjunction queries)
  - Term weighting schemes sometimes difficult to maintain in incremental settings, e.g. idf values and document norms frequently change

# Practical Considerations

- Document length approximations
- Incorporating proximity considerations of query terms occurrences in result documents into the formulae
- Stop-word elimination
  - Stop-word examples: and, the, or, of, in, a, an, to, …
- Linguistic processing of terms (stemming, lemmatization, synonym expansion, compounds) and their effects on recall/precision