# Boolean and Vector Space Retrieval Models

1

## Retrieval Models

- A retrieval model specifies the details of:
  - Document representation
  - Query representation
  - Retrieval function
- Determines a notion of relevance.
- Notion of relevance can be binary or continuous (i.e. *ranked retrieval*).

2

## Classes of Retrieval Models

- Boolean models (set theoretic)
  - Extended Boolean
- Vector space models (statistical/algebraic)
  - Generalized VS
  - Latent Semantic Indexing
- Probabilistic models

3

## Other Model Dimensions

- Logical View of Documents
  - Index terms
  - Full text
  - Full text + Structure (e.g. hypertext)
- User Task
  - Retrieval
  - Browsing

4

## Retrieval Tasks

- Ad hoc retrieval: Fixed document corpus, varied queries.
- Filtering: Fixed query, continuous document stream.
  - User Profile: A model of relative static preferences.
  - Binary decision of relevant/not-relevant.
- Routing: Same as filtering but continuously supply ranked lists rather than binary filtering.

5

## Common Preprocessing Steps

- Strip unwanted characters/markup (e.g. HTML tags, punctuation, numbers, etc.).
- Break into tokens (keywords) on whitespace.
- Stem tokens to "root" words
  - computational → comput
- Remove common stopwords (e.g. a, the, it, etc.).
- Detect common phrases (possibly using a domain specific dictionary).
- Build inverted index (keyword → list of docs containing it).

6

## Boolean Model

- A document is represented as a set of keywords.
- Queries are Boolean expressions of keywords, connected by AND, OR, and NOT, including the use of brackets to indicate scope.
  - [[Rio & Brazil] | [Hilo & Hawaii]] & hotel & !Hilton]
- Output: Document is relevant or not. No partial matches or ranking.

7

## Boolean Retrieval Model

- Popular retrieval model because:
  - Easy to understand for simple queries.
  - Clean formalism.
- Boolean models can be extended to include ranking.
- Reasonably efficient implementations possible for normal queries.

8

## Boolean Models – Problems

- Very rigid: AND means all; OR means any.
- Difficult to express complex user requests.
- Difficult to control the number of documents retrieved.
  - *All* matched documents will be returned.
- Difficult to rank output.
  - *All* matched documents logically satisfy the query.
- Difficult to perform relevance feedback.
  - If a document is identified by the user as relevant or irrelevant, how should the query be modified?

9

## Statistical Models

- A document is typically represented by a *bag of words* (unordered words with frequencies).
- Bag = set that allows multiple occurrences of the same element.
- User specifies a set of desired terms with optional weights:
  - Weighted query terms:
    Q = < database 0.5; text 0.8; information 0.2 >
  - Unweighted query terms:
    Q = < database; text; information >
  - No Boolean conditions specified in the query.

10

## Statistical Retrieval

- Retrieval based on *similarity* between query and documents.
- Output documents are ranked according to similarity to query.
- Similarity based on occurrence *frequencies* of keywords in query and document.
- Automatic relevance feedback can be supported:
  - Relevant documents “added” to query.
  - Irrelevant documents “subtracted” from query.

11

## 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.?

12

# 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:

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



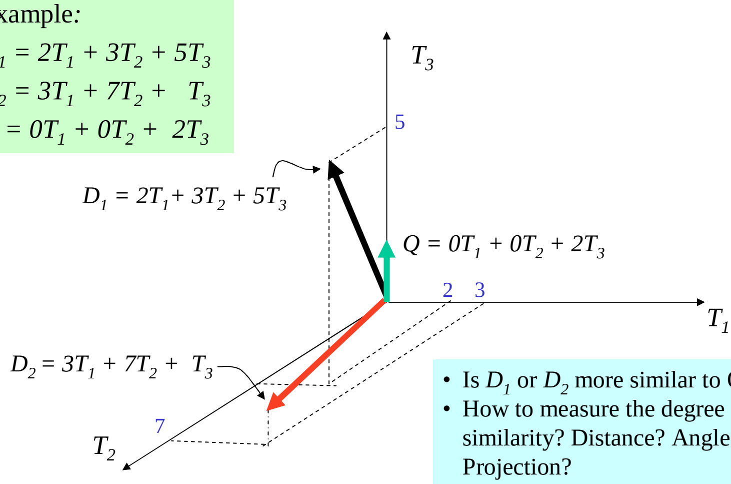
# Graphic Representation

Example:

$D_1 = 2T_1 + 3T_2 + 5T_3$

$D_2 = 3T_1 + 7T_2 + T_3$

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




- Is $D_1$ or $D_2$ more similar to Q?
- How to measure the degree of similarity? Distance? Angle? Projection?

## 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.

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

15

## 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:

  $tf_{ij} = f_{ij} \; / \, max_i \, \{f_{ij}\}$

16

## 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*.

17

## 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.

18

## 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

19

## Query Vector

- Query vector is typically treated as a document and also tf-idf weighted.
- Alternative is for the user to supply weights for the given query terms.

20

## 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.

21

## Similarity Measure - Inner Product

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

$$\mathrm{sim}(\boldsymbol{d}_j,\boldsymbol{q}) = \boldsymbol{d}_{\mathrm{j}} \bullet \boldsymbol{q} = \sum_{i=1}^{t} w_{ij} \cdot w_{iq}$$

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.

22

# Properties of Inner Product

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

23

# 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

sim(D, Q) = 3

Weighted:

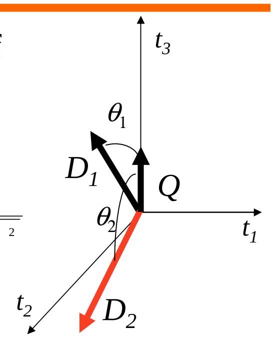
$D_1 = 2T_1 + 3T_2 + 5T_3$ $\quad D_2 = 3T_1 + 7T_2 + 1T_3$
$Q = 0T_1 + 0T_2 + 2T_3$

sim($D_1$ , $Q$) = 2*0 + 3*0 + 5*2 = 10
sim($D_2$ , $Q$) = 3*0 + 7*0 + 1*2 = 2

24

## Cosine Similarity Measure

- Cosine similarity measures the cosine of the angle between two vectors.
- Inner product normalized by the vector lengths.




$$\text{CosSim}(\boldsymbol{d}_j, \boldsymbol{q}) = \frac{\vec{d}_j \cdot \vec{q}}{\left|\vec{d}_j\right| \cdot \left|\vec{q}\right|} = \frac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^{2} \cdot \sum_{i=1}^{t} w_{iq}^{2}}}$$

$D_1 = 2T_1 + 3T_2 + 5T_3$ CosSim($D_1$ , $Q$) = 10 / $\sqrt{(4+9+25)(0+0+4)}$ = 0.81
$D_2 = 3T_1 + 7T_2 + 1T_3$ CosSim($D_2$ , $Q$) = 2 / $\sqrt{(9+49+1)(0+0+4)}$ = 0.13
$Q = 0T_1 + 0T_2 + 2T_3$

$D_1$ is 6 times better than $D_2$ using cosine similarity but only 5 times better using inner product.

25

## Naïve Implementation

Convert all documents in collection D to tf-idf weighted vectors, $\boldsymbol{d_j}$, for keyword vocabulary V.

Convert query to a tf-idf-weighted vector $\boldsymbol{q}$.

For each $\boldsymbol{d_j}$ in D do

  Compute score $s_j$ = cosSim($\boldsymbol{d_j}$, $\boldsymbol{q}$)

Sort documents by decreasing score.

Present top ranked documents to the user.

Time complexity: O(|V|·|D|) Bad for large V & D !

|V| = 10,000; |D| = 100,000; |V|·|D| = 1,000,000,000

26

## 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.



## Problems with Vector Space Model

- Missing semantic information (e.g. word sense).
- Missing syntactic information (e.g. phrase structure, word order, proximity information).
- Assumption of term independence (e.g. ignores synonomy).
- Lacks the control of a Boolean model (e.g., *requiring* a term to appear in a document).
  - Given a two-term query "A B", may prefer a document containing A frequently but not B, over a document that contains both A and B, but both less frequently.

# Exercise

The corpus C consists in the following three documents:

- d1: “new york times”
- d2: “new york post”
- d3: “los angeles times”

1. Assuming that the term frequencies are normalized by the maximum frequency in a given document, calculate the tf-idf scores for all the terms in C. Assume the words in the vectors are ordered alphabetically.

2. Given the following query: “new new times”, calculate the tf-idf vector for the query, and compute the score of each document in C relative to this query, using the cosine similarity measure. Assume that term frequencies are normalized by the maximum frequency in a given query.