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.

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.

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:

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

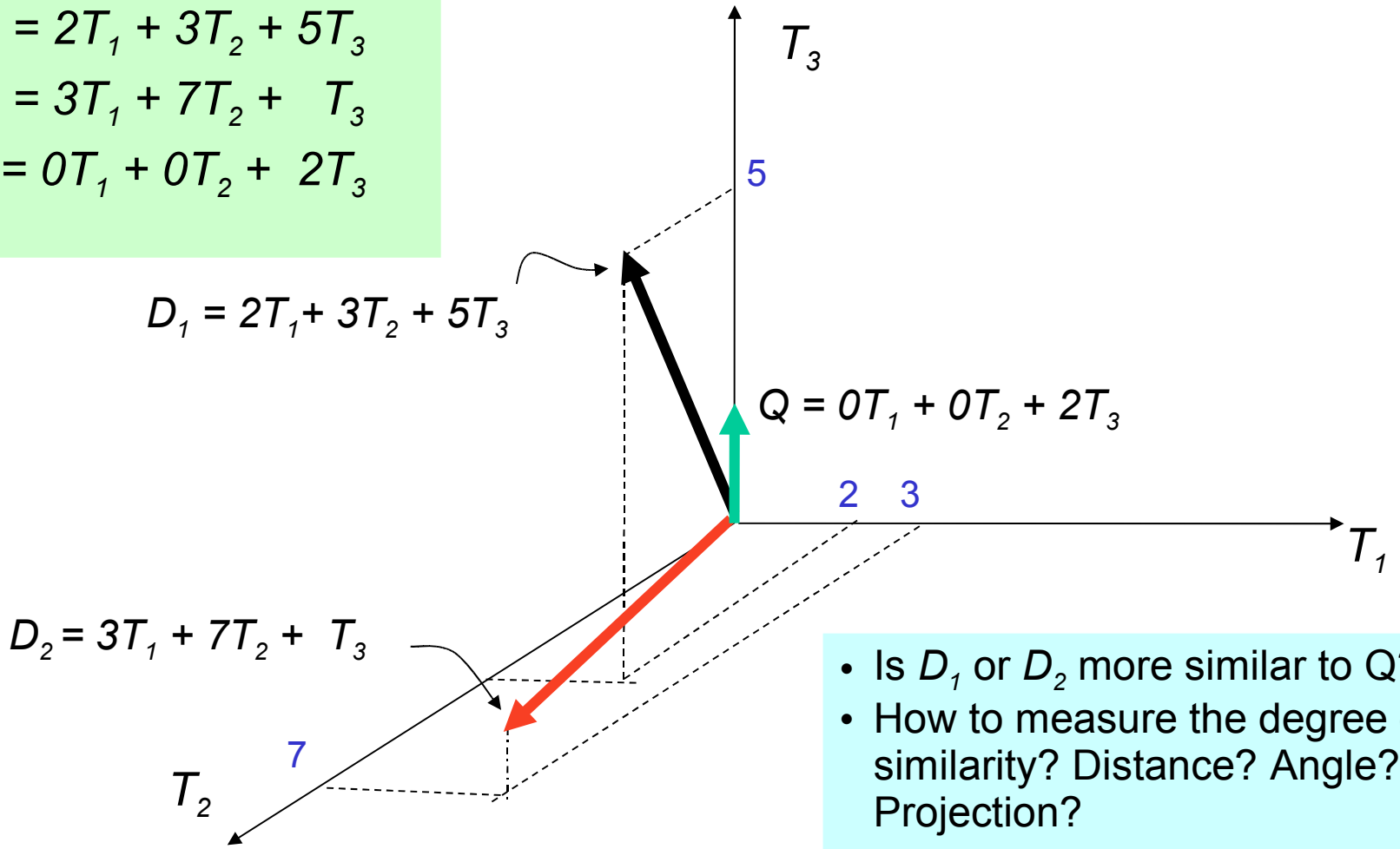
Graphic Representation

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.

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

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*.
- Note similarity to Luhn's Idea

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

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.

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 $\boldsymbol{d}_j$ and query $\boldsymbol{q}$ can be computed as the vector inner product:

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

  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.

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.

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:

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

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

sim(D_1 , Q) = 2*0 + 3*0 + 5*2 = 10

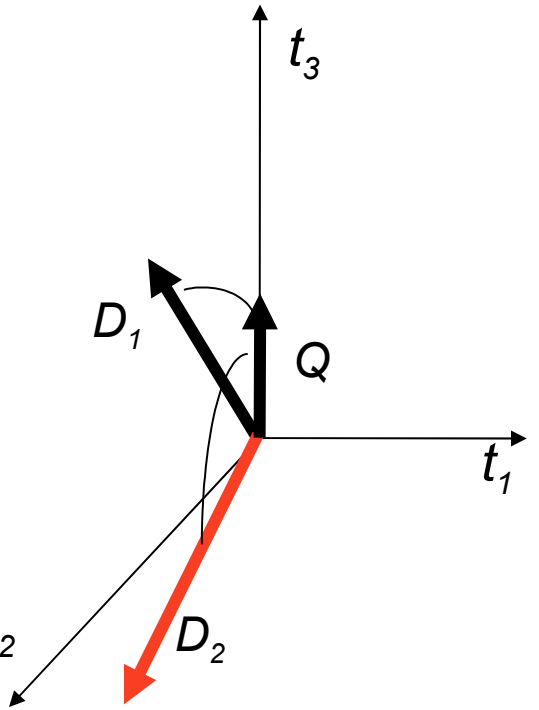
sim(D_2 , Q) = 3*0 + 7*0 + 1*2 = 2

Cosine Similarity Measure

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

CosSim($\boldsymbol{d}_j$, $\boldsymbol{q}$) =

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



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

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 $s_j = \text{cosSim}(\boldsymbol{d}_j, \boldsymbol{q})$

Sort documents by decreasing score.

Present top ranked documents to the user.

Time complexity: $O(|V| \cdot |D|)$ Bad for large V & D !

$|V| = 10{,}000$; $|D| = 100{,}000$; $|V| \cdot |D| = 1{,}000{,}000{,}000$

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.

Basic Probability

- Conditional – or Posterior - Probability
  - P(A|B) == the probability that A will occur given that B has occurred
  - P(A|B) = P(A and B both occur) / P(B)
    - For example: in a sentence from the “Wizard of Oz” what is the probability of the next word being “witch” given that the previous word was “wicked”
      - P(witch | wicked)
- Prior Probability
  - The probability of something given no information
  - P(“the”) = 0.07 in English text

Bayes Rule

- Bayes Theorem
  - $P(A|B) = (P(B|A)/P(B))\ P(A)$
    - $P(B|A)/P(B)$ is called the Likelihood of A given B
- Example
  - 2 bowls of cookies
    - Bowl A: 30 Chocolate chip, 10 plain
    - Bowl B: 20 Chocolate chip, 20 plain
  - Suppose randomly pick a bowl and then randomly select a cookie from that bowl. When you do so, you get a chocolate chip
  - What is the probability that you picked from bowl A?
    - $P(A \mid cc)$
  - Know: $P(cc|A) = 0.75$, $P(cc|B) = 0.5$, $P(cc)=0.675$, $P(A) = 0.5$
  - $P(A|cc) = P(cc|A)*P(A)/P(cc) = 0.75*0.5/0.675 = 0.6$

Bayes and IR

- Want to compute
  - P(is relevant to Q | document D)
    = (P(D|rel) * P(rel)) / P(D)
- So, by Bayes would need:
  - P(D | relevant to Q), P(relevant to Q), P(D)
  - P(is relevant to Q) = probability of picking a relevant document from among all documents. (This is the same for all documents)
  - P(D) = P(t1) * ... for each word in D
    - *This we know, but it drops out as a constant term*

- P(D | relevant to Q)
  - Assumes independence
    - So P(D|rel) = P(ti1|rel) * P(t2|rel)*...
  - For each word ti in D
    - pi = P(relevant | ti), qi = P(not relevant | ti)
    - Then let ci = log ((pi(i-qi)) / (qi(1-pi))
      - Roughly, ci is how predictive of relevancy a given word is.
        - The ***relevance weight***
      - Simple model says pi=1 iff word ti in in Q,
        - problem – this leads to division by 0 and is just wrong
          - When would this be correct?
      - Problem – words are all equally weighted.
        - Is this really a problem?
  - Summing the ci gives an indicator of relevance of the document. Roughly, if >0 then relevant
  - Problem, where do you get all of the pi and qi
  - The sum gives a total ordering of documents.

TF-IDF vs Probabilistic Models

- IDF term looks a lot like divisor in Bayes rule
- TF term looks a lot like – what?
  - It is related to P(ti | D) but this is not used.
- What in Probabilistic model corrects for the benefit of big documents?
  - Unlike cosine distance which is strictly non-negative the ci terms may be less than 0. How?
  - So, VS only accumulates evidence in favor according to the intersection of document and query. Probabilistic accumulates evidence for and against according to all words in document.

Self Organization of the Web ...

- Community: “a collection of web pages such that each member page has more hyperlinks ... within the community than outside”
- Hypothesis: “the web self organizes such that link structure allows efficient identification of communities”
- Postulate: a link contains a lot of unambiguous semantic content
  - home pages .. “things I am interested in”
- Community Uses: automatic web portals, Focused search engines, content filtering, sociology

Community Questions

- Question: Does this definition of community require that the hypothesis be true
- Can someone force a page into a community.
- Can community be computed on the fly as part of a search focusing algorithm (e.g., here is a relevant document, show me its community)?
  - Both speed and unility an issue here
- Does every web page have a community --
  - “... Every man's death diminishes me, for I am involved in mankind...”

Algorithm

- Id a candidate community
  - Just get a couple of sources
  - Crawl bi-directionally to depth N
  - Count links on each page to other pages found in crawl (A)
  - Count links to pages not found in crawl (B)
  - If (B>A) the remove page from community
  - Repeat A,B counting on pages until stable
  - Repeat crawl a couple of times

Results

- Simple disjunctions characterize the communities found
  - Error rates of about 1% (is this any good on the web)?
  - Communities had about 200 members
    - Why are they all about the same size, is te significant of anything?
  - Surprises?
    - Some community members are a long way from sources
    - Rosiland Franklin?
    -