Lecture 11
Information Retrieval, cont'd

Some slides due to Raghavan et al., via Dan Weld

Organization

- Exams still being graded
- Last time:
  - Basic intro to search
  - Boolean model
  - Intro to inverted index
  - Start of vector-space, tf-idf
- Today:
  - Finish vector-space and tf-idf
  - How to assess search quality
- Later:
  - Text handling and in-depth inverted index
  - Graph analysis (PageRank and HITS)
  - Crawler design (Mercator)
  - More search architecture

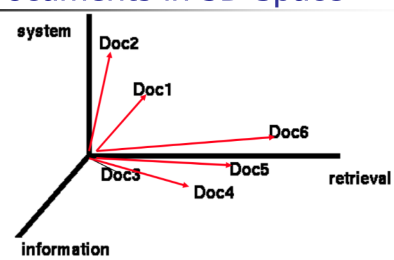
2

Documents as Vectors

- Each doc *j* can be viewed as a vector of *tf* values, one component for each term
- We thus have a *vector space*
  - Terms are axes
  - A doc is a point in the space
  - Space is hugely multidimensional. Can easily have 20,000+ dimensions

3

Documents in 3D Space



- One assumption: documents that are "close together" in space are also close in meaning

4

Vector Space Query Model

1. Treat a query as a short document
2. Sort documents by increasing distance (decreasing similarity) to the query document
3. Easy to compute, as both query & doc are vectors

- First used in Salton's SMART system (1970). Now used by almost every IR system

5

Vector Representation

- Docs & Queries are vectors
- Pos'n 1 corresponds to term 1. Pos'n *t* corresponds to term *t*
- Weight of term stored in each pos'n

$$D_i = w_{d_{i1}}, w_{d_{i2}}, \ldots, w_{d_{it}}$$

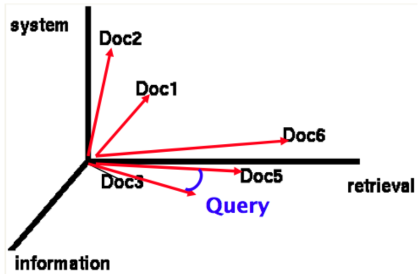
$$Q = w_{q1}, w_{q2}, \ldots, w_{qt}$$

$$w = 0 \text{ if a term is absent}$$

6

Documents in 3D Space

- Term weights indicate length of document vector along a dimension



7

Computing Weights

- Which word is more indicative of document similarity?
  - "Book" or "Rumplestiltskin"?
  - Need to consider **document frequency** - how often a word appears in doc collection
- Which doc is a better match for the query "kangaroo"?
  - One with a single mention of Kangaroos… or a doc that mentions it 10 times?
  - Need to consider **term frequency** - how many times the word appears in current document

8

TF x IDF

- "Term-Frequency" x "Inverse Document Frequency"

$$\boldsymbol{W_{ik} = tf_{ik} * \log(N / n_k)}$$

- T_k = term k in document D_i
- tf_{ik} = freq of term T_k in doc D_i
- idf_k = inverse doc freq of term T_k in C
  $idf_k = \log(\frac{N}{n_k})$
- N = total # docs in collection C
- N_k = # docs in C that contain T_k

9

Inverse Document Frequency

- IDF provides high values for rare words, low values for common words

$$\log\left(\frac{10000}{10000}\right) = 0$$

$$\log\left(\frac{10000}{5000}\right) = 0.301$$

$$\log\left(\frac{10000}{20}\right) = 2.698$$

$$\log\left(\frac{10000}{1}\right) = 4$$

10

TF-IDF normalization

- Normalize term weights
  - Longer docs not given more weight
  - Force all values within [0,1]

$$w_{ik} = \frac{tf_{ik} \log(N / n_k)}{\sqrt{\sum_{k=1}^{t} (tf_{ik})^2 [\log(N / n_k)]^2}}$$

11

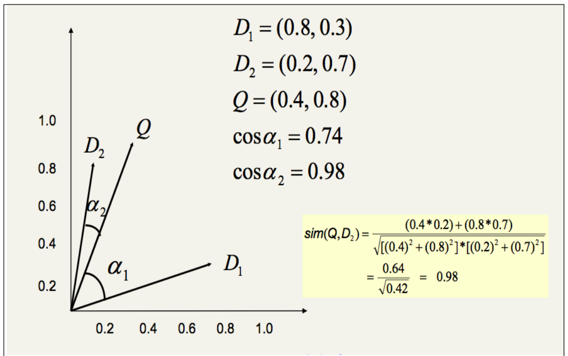
Vector space similarity

- Now, the similarity of two docs is:

$$Sim(Di, Dj) = \sum_{k=1}^{t} w_{ik} * w_{jk}$$

- Also called the cosine, or normalized inner product
  (normalization done when computing term weights)
- Recall that cosine:
  - Depends on two adjacent vector lengths
  - =1 when angle is zero (points are identical)
  - Smaller when angle is greater

12



13

Computing a similarity score

- Say we have query vector Q = (0.4,0.8)
- Also, document D2 = (0.2,0.7)
- What is the result of the similarity computation?

$$sim(Q,D_2) = \frac{(0.4*0.2)+(0.8*0.7)}{\sqrt{[(0.4)^2+(0.8)^2]*[(0.2)^2+(0.7)^2]}}$$

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

14

To Think About

- How does this ranking algorithm behave?
  - Make a set of hypothetical documents consisting of terms and their weights
  - Create some hypothetical queries
  - How are docs ranked, depending on weights of the terms and the queries' terms?

15

Summary: Vector Spaces

- User's query treated as short document
- Query is in same space as docs
- Easy to measure a doc's dist. to query
- Obvious extension from simple Boolean world

16

Assessing Quality

- You've built a ranker. How do you know it's any good?
- Relevance has been studied for a long time
  - Many contributing factors
  - People disagree on what's relevant
- Retrieval/assessment models differ
  - Binary relevance vs sorted relevance
  - Query-relevance vs user-relevance

17

Data!

- Large hand-marked query/result tuples form the "answer key" for the ranker
- TREC is an annual conference, also publishes data
- Different tracks this year:
  - Blog track studies information-seeking
  - Chemical IR, Legal IR
  - Entity-extraction

18

Formats

- Query, URL, Relevant?
  - "Britney"
  - http://www.britneyspears.com
  - Yes!

  - "Britney"
  - http://www.eecs.umich.edu/~michjc/
  - No!

19

Formats

- Query, URLs, Ranking
  - "Britney"
  - URL_0, URL_1, URL_2, …
  - 3, 9, 1, 2, 10, …

20

Evaluating Search Ranking

- Precision/Recall Curves
- Kendall's Tau

21

Precision and recall

- Precision: fraction of retrieved docs that are relevant = relevant/retrieved
- Recall: fraction of relevant docs that are retrieved = retrieved/relevant

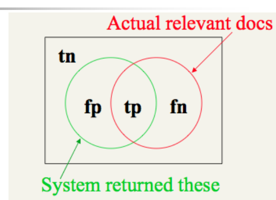
	Relevant	Not Relevant
Retrieved	Tp	Fp
Not retrieved	Fn	Tn

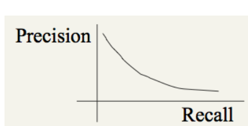
- Precision P = tp/(tp+fp)
- Recall R = tp/(tp+fn)

22

More P&R

- Precision:
  - % of selected items that are correct
- Recall:
  - % of correct items that are selected
- P/R curve shows tradeoff





23

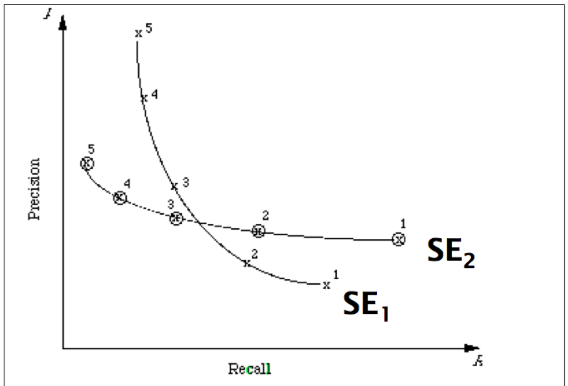
More P&R

- Generally trade precision vs recall
  - How to get a system with high recall?
- Recall is a non-decreasing function of the # of docs retrieved
  - Precision usually decreases in good system
- Drawbacks
  - Binary relevance
  - Need human judgments
  - Must average over large corpus
  - Alternatively, skewed by corpus/author selection

24

P-R Curves

- Imagine you have a total ordering on documents
  - To generate P&R, just choose a threshold
  - Can return any # of results, ordered by sim
- By choosing various levels of recall (adjusting the threshold), you can produce a *precision-recall curve*



Other Measures

- Precision at fixed recall
  - Critical for Web Search
- Kendall's Tau for comparing sorts

Kendall's Tau

- Imagine we have a real ordering of documents, not just binary judgments
- The correct document ordering is:
  - 1, 2, 3, 4
- Search Engine A outputs:
  - 1, 2, 4, 3
- Search Engine B outputs:
  - 4, 3, 1, 2
- Intuitively, A is better. How do we capture this numerically?

Measuring Rank Correlation

- Kendall's Tau has some nice properties:
  - If agreement between 2 ranks is perfect, then KT = 1
  - If disagreement is perfect, then KT = -1
  - If rankings are uncorrelated, then KT = 0 on average
- Intuition: Compute fraction of pairwise orderings that are consistent

Kendall's Tau

 # pairs that agree

 # pairs that disagree

$$\tau = \frac{n_c - n_d}{\frac{1}{2}n(n-1)}$$

total # pairs

- The unnormalized version is called Kendall's Tau Distance
- It's also called *bubble-sort distance*

Try it out

- Correct ordering:
  - 1, 2, 3, 4
- Search Engine A:
  - 1, 2, 4, 3

$$\tau = \frac{5-1}{\frac{1}{2}4(4-1)} = \frac{4}{6} = 0.666$$

- Search Engine B:
  - 4, 3, 2, 1

$$\tau = \frac{0-6}{\frac{1}{2}4(4-1)} = \frac{-6}{6} = -1$$

31

Ranking Assessment

- Requires lots of hand-judged data
- Precision & Recall
  - Usually trade off each other
  - With a ranker, can generate PR curve
  - Requires relevant/not-relevant judgments
- Kendall's Tau
  - Measures correlation between two rankings
  - +1 if perfect agreement; -1 disagreement
  - Measure "fraction of pairs in agreement"

32

Thought Experiment

- If you're Google, getting data is easy
- What if you're a search startup? How can you evaluate your ranker without a popular site?

33