Vector model of Information Retrieval

Dan Smith

This week's padlet

https://padlet.com/danjsmith756/ir_4

Overview

- Boolean retrieval
- Documents as vectors
- Term weighting
- Similarity measures

why a term-document matrix is not suitable

INDEXES

Initial view: term-document matrix

- A simple view of a collection of documents is as a matrix with
  - a column for each document
  - a row for each term
- Each cell is either
  - 1 if the document contains the term
  - 0 if it does not

Example for this lecture

- The first 5 paragraphs of *Alice Through the Looking-Glass*:

  One thing was certain, that the WHITE kitten had had nothing to do with it…
  The way Dinah washed her children's faces was this…
  But the black kitten had been finished with earlier in the afternoon…
  'Oh, you wicked little thing!' cried Alice, catching up the kitten…
  'Do you know what to-morrow is, Kitty?' Alice began…

Term-document matrix problems

- The matrix is sparse: 5 documents, 237 terms

– d1	49
– d2	61
– d3	64
– d4	96
– d5	89

- 1185 cells, 359 not blank
  - the proportion of blank cells rises rapidly as the collection grows

	looking	…	rolled	upon	floor	yards	unwound	…
d1	1		0	0	0	0	0	
d2	1		0	0	0	0	0	
d3	1		0	0	0	0	0	
d4	1		0	0	0	0	0	
d5	1		1	1	1	1	1	

… a better approach

- The inverted index structure provides the same information

  looking [1, 2, 3, 4, 5]
  rolled [5]
  upon [5]
  yards [5]
  floor [5]
  unwound [5]

- Grows as a function of the collection size
  - typically 6-7% of the collection size

BOOLEAN RETRIEVAL

Boolean queries

- Terms linked by Boolean operators AND, OR, NOT
  - black AND yarn
  - kitten OR cat
  - looking NOT glass
- These can be combined
  - white OR black AND yarn NOT wash

Boolean retrieval with inverted lists

- Basic operation is to merge two lists
  - AND add document to result list if there's a match
  - OR union the lists
  - NOT remove document from result list if there's a match

Boolean retrieval summary

- Often either give very many results or none
- Sensitive to small variations in vocabulary and query specification
- Good where
  - users are experts and can formulate precise queries
  - recall (finding *all* relevant documents) is important
- Disadvantages
  - most users are not experts in query formulation
  - most users only need a few relevant results to satisfy their information need

Boolean retrieval conclusion

- User perspective
  - Not useful for most search tasks
  - Useful as an advanced search feature
- Developer perspective
  - easy to understand
  - includes key functionality for search query processing
  - a good stepping-stone

TERMS AND WEIGHTING

Terms and weighting

1. Count term frequency
   - documents with more shared terms are more similar
2. Weight terms according to classificatory power
   - words found in every document are poor classifiers
   - words found in a single document are poor classifiers
3. Normalise to account for different length documents
   - Otherwise long documents will dominate

Term-document count matrices

- Consider the number of occurrences of a term in a document:
  - Bag of words model
  - Document is a vector in the document-term matrix - a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Bag of words view: issue

- The document
  - *The bird is on the wing*

is indistinguishable from
  - *The wing is on the bird*

Term frequency *tf*

- Long documents are favoured because they're more likely to contain query terms
- Can fix this in principle by normalizing for document length

Normalised term frequency

P_j	no. of terms in document j
tf_{ij}	normalised frequency
$freq_{i,j}$	count of term i in document j

$$tf_{ij} = \frac{freq_{i,j}}{P_j}$$

Weighting should depend on the term overall

- Which of these tells you more about a document?
  - 10 occurrences of *hernia*?
  - 10 occurrences of *the*?
- Would like to reduce the weights of common terms
  - But what is "common"?
- Could look at collection frequency (*cf*)
  - The total number of occurrences of the term in the entire collection of documents

Document frequency

- But document frequency (*df*) may be better:
- *df* = number of docs in the corpus containing the term

Word	cf	df
ferrari	10422	17
insurance	10440	3997

- Document or collection frequency weighting is only possible in known (static) collection.
- So how do we make use of *df* ?

Weighting term frequency: *tf*

- What is the relative importance of
  - 0 vs. 1 occurrence of a term in a doc
  - 1 vs. 2 occurrences
  - 2 vs. 3 occurrences …
- Unclear: while it seems that more is better, a lot isn't proportionally better than a few
  - Can just use raw *tf*
  - Another option is:

$$wf_{t,d} = 0 \text{ if } tf_{t,d} = 0,\ 1 + \log tf_{t,d} \text{ otherwise}$$

Inverse Document Frequency

- Not all terms are equally important so weight each by the number of documents it occurs in:

N	no. of documents in collection
n_i	no. of documents containing term i

$$idf_i = \log_{10} \frac{N}{n_i}$$

Term Frequency - Inverse Document Frequency

- Measure weights terms according to
  - how common they are in the collection
  - how common they are in the document
- Performs well in most situations

$$w_{ij} = tf_{ij} \times idf_i$$

similarity is the inverse of distance

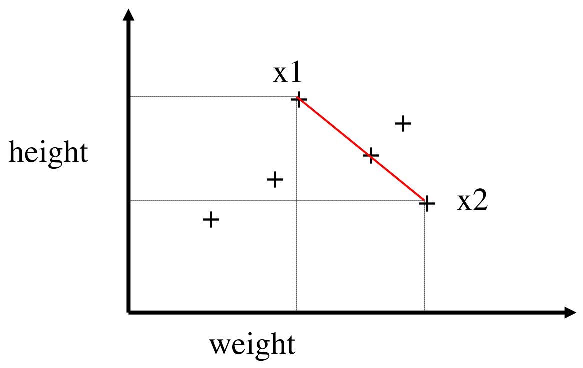
SIMILARITY MEASURES

Distance measurement

- Several measures possible:
  - Euclidian distance
    - straight line between points
  - City block
    - sum of distances between points along each dimension
  - Cosine distance
    - cosine of angle between vectors

27

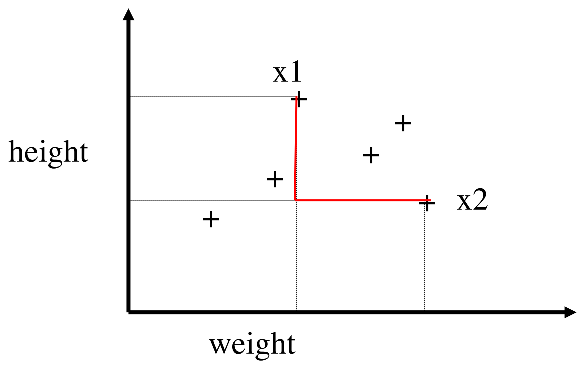
Euclidian distance



$$d(x1,x2) = |x1 - x2| = \sqrt{\sum_{i=1}^{N} (x1_i - x2_i)^2}$$

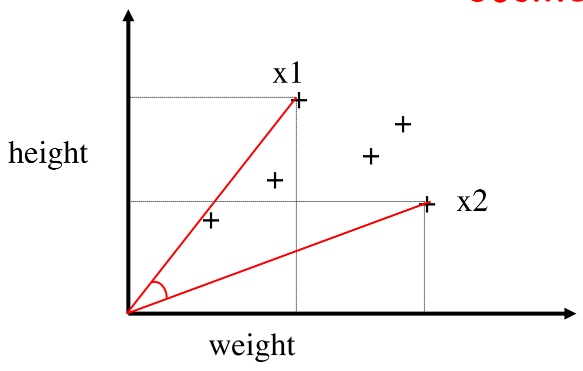
28

City block distance



$$d(x1,x2) = \sum_{i=1}^{N} |x1_i - x2_i|$$

Cosine distance



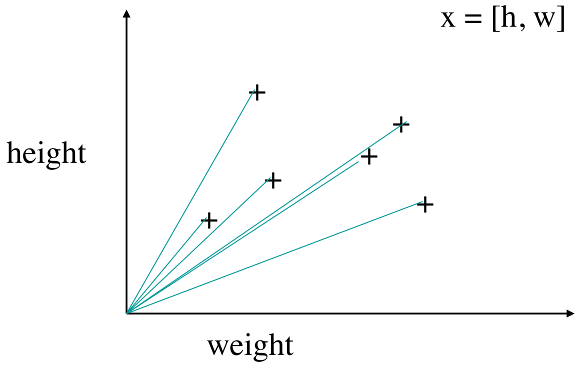
DOCUMENTS AS VECTORS

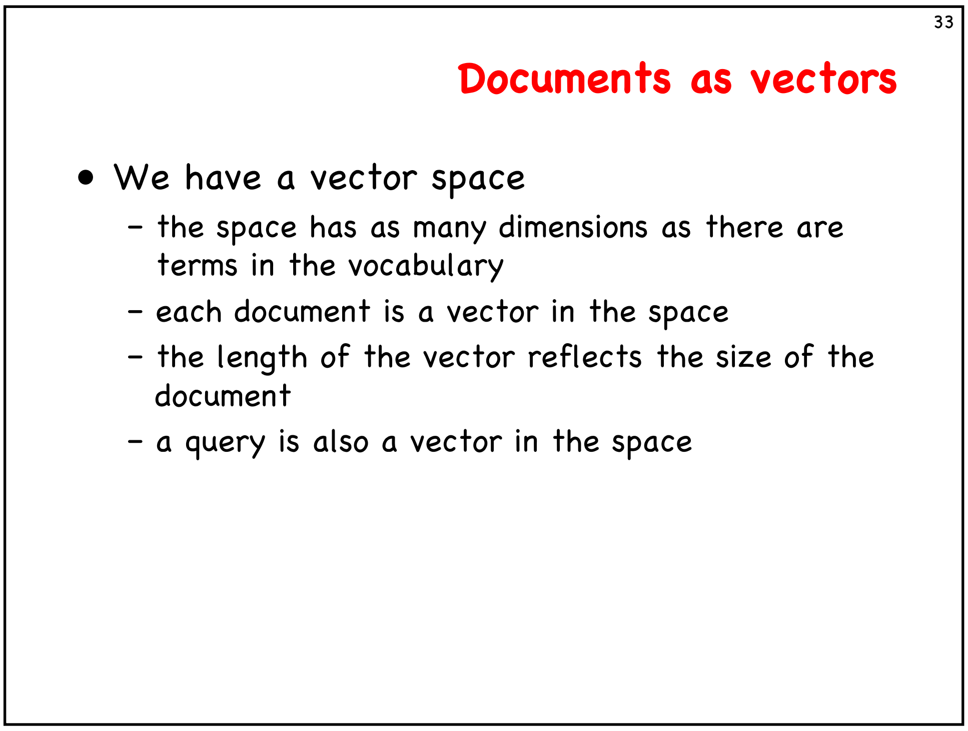
Vector spaces

- We can describe an object as a set of *independent* measurements
- This can be represented as a vector $X = [x_1 \dots x_N]$
- If the measurements are the dimensions of an N-dimensional space, the vector X is a point in that space

Simple example

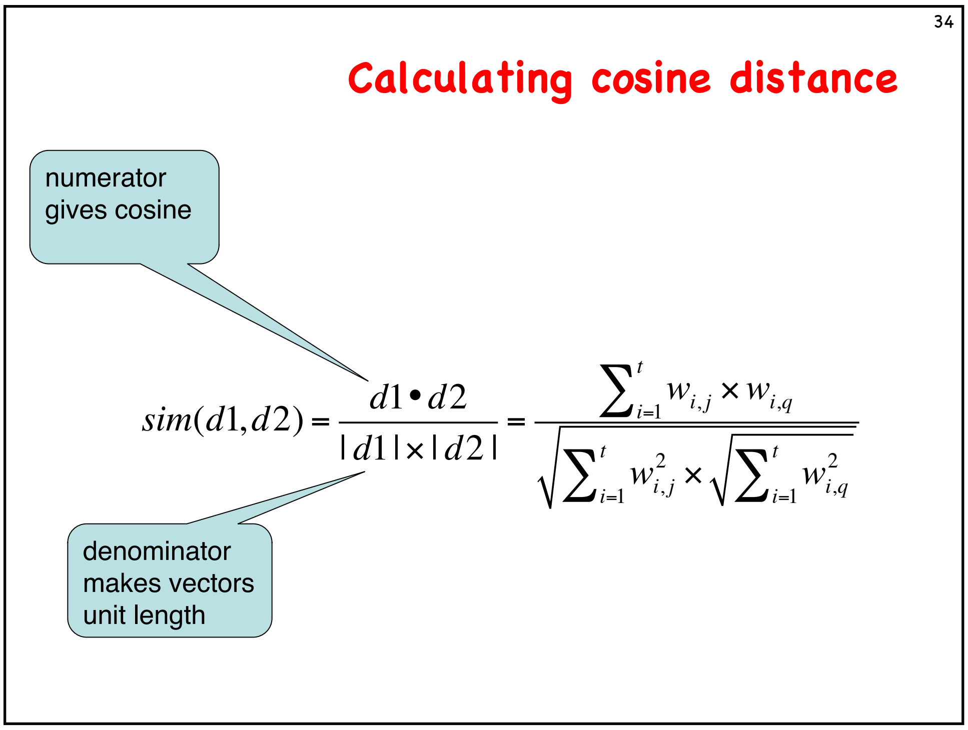
Each point is a vector
x = [h, w]





Documents as vectors

- We have a vector space
  - the space has as many dimensions as there are terms in the vocabulary
  - each document is a vector in the space
  - the length of the vector reflects the size of the document
  - a query is also a vector in the space



Calculating cosine distance

numerator gives cosine

$$sim(d1,d2)=\frac{d1\bullet d2}{|d1|\times|d2|}=\frac{\sum_{i=1}^{t} w_{i,j}\times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2}\times\sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

denominator makes vectors unit length

Cosine distance

dot product of x1, x2

product of lengths of x1, x2

sum of products of vector components

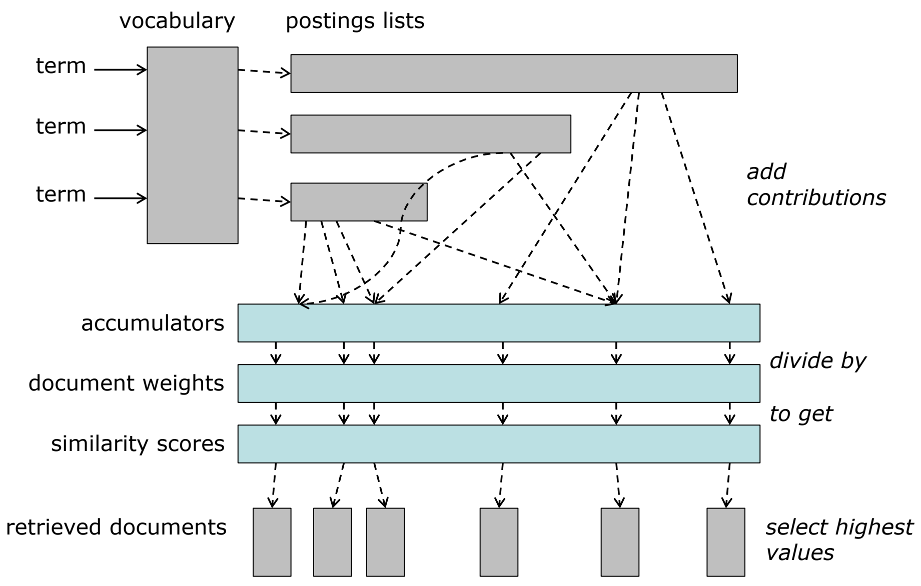
square root of sum of squared lengths of the components of x1, x2

$$sim(d1,d2) = \frac{d1 \bullet d2}{|d1| \times |d2|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

Cosine calculation pseudocode

```
cosineScore(q)
    global doclengths
    global postings
    float scores[]
    for each term t in q
         calculate wt,q
         fetch postings list for t
         for each pair(d, tft,d) in postings
                 scores[d] += wft,d × wt,q
                 for each d in doclengths
                     scores[d] = scores[d]/doclengths[d]
    return top K components of scores[]
```

Vector model schematic diagram



(after Zobel and Moffat (2006))

Resources

- Manning et al., *Intro to IR*, Chapter 6.2, 6.3 and 7.1
- Zobel J., Moffat A. (2006) Inverted files for text search engines. *ACM Comput. Surv.* 38(2), Article 6, doi>10.1145/1132956.1132959
- Lester, N., Moffat, A., Webber, W., Zobel, J. (2005). Space-limited ranked query evaluation using adaptive pruning. WISE 2005, 470-477