Lecture 2: The Term-Document Matrix

William Webber (william@williamwebber.com)

COMP90042, 2014, Semester 1, Lecture 1b

What we'll learn today

- How to calculate a "similarity" between two documents
- Therefore, given a document, how to find the most similar document to it in the corpus

Collection representation

- Each document is modelled as a "bag of words"
- That is, as a list of terms it contains, and the count of each term
- The whole collection could be modelled as a "list of bag of words"
- . . . but that fails to capture commonality of terms between documents (which is the core to the text analysis tools we'll be looking at)

The term-document matrix (TDM)

- An alternative, and fruitful, model is of a "term–document matrix" or TDM.
- Recall that a "matrix" is (in CS-speak) a two-dimensional array, with rows and columns
- In the TDM, rows represent documents, columns represent terms (in the collection vocabulary)
- Cell values are term frequency counts, or, more generally, "score" attached to a term for a document

Example TDM

doc1	Two for tea and tea for two
doc2	Tea for me and tea for you
doc3	You for me and me for you

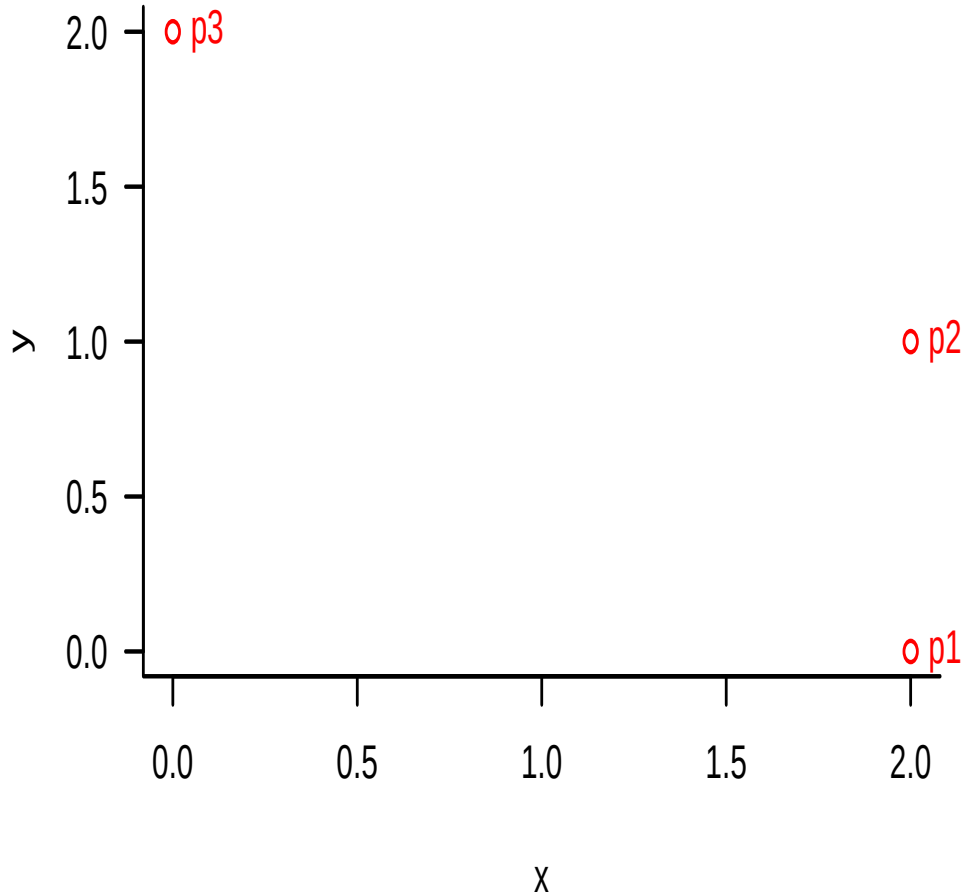
	two	tea	me	you
doc1	2	2	0	0
doc2	0	2	1	1
doc3	0	0	2	2

Geometrical interpretation of TDM

- The TDM is a handy conceptual representation (though, as we'll see later, not always a suitable implementation)
- But it also suggests a useful way of modelling and computing with documents
- That is, by modelling documents as *points (vectors) in multi-dimensional term space*

Matrix representation for points in 2-d space

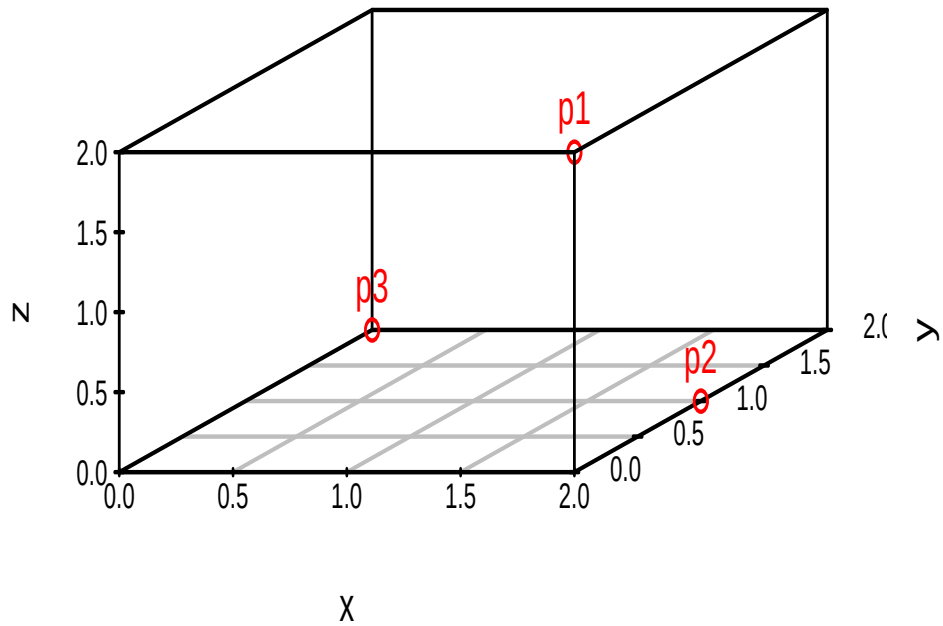
Point	x	y
p1	2	0
p2	2	1
p3	0	2



We can represent points in (abstract) 2d space using a matrix

Matrix representation for points in 3-d space

Point	x	y	z
p1	2	0	2
p2	2	1	0
p3	0	2	0



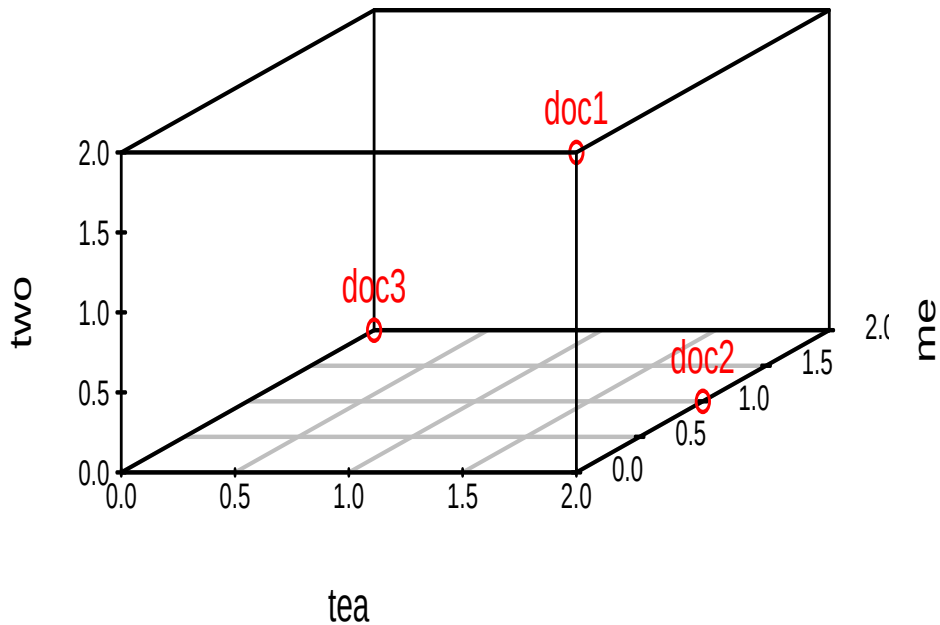
And similarly for points in 3d space (and higher dimensional space, too, though it gets tricky to draw)

Documents in term space

- We can transfer the same model to modelling documents as terms
- Each term (of the $|T|$ terms in the vocabulary) becomes a dimension
- The document's position in the dimension of term t is determined by $s_{d,t}$ (score of term t in document d – for now, just the term frequency)
- Each document becomes a point in $|T|$-d term space

Documents in term space

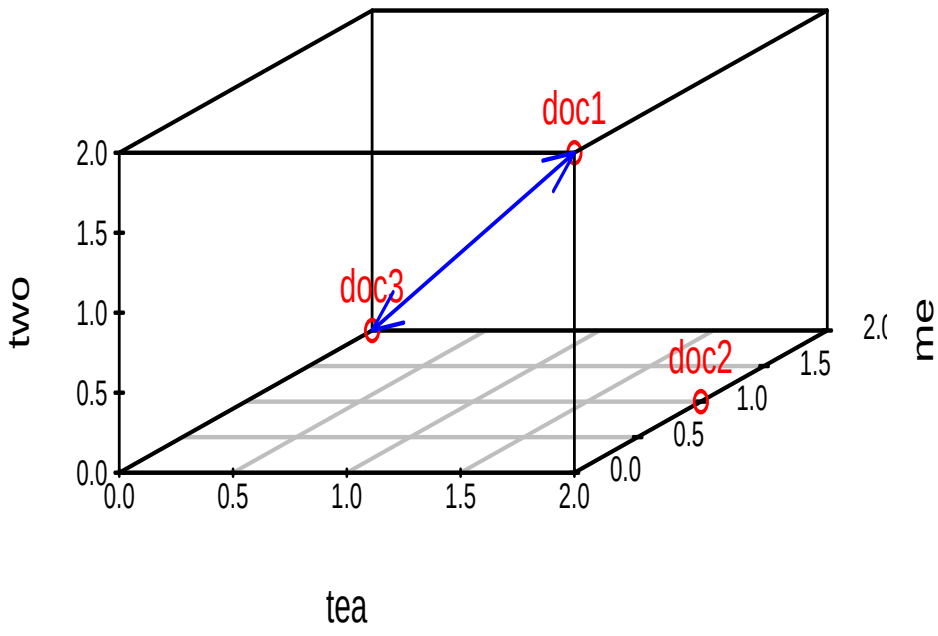
Point	tea	me	two
doc1	2	0	2
doc2	2	1	0
doc3	0	2	0



Like so

Document similarity as (inverse) distance

$$\begin{aligned}\text{sim}(\text{doc1}, \text{doc2}) &= -\sqrt{(2^2 + 2^2 + 2^2)} \\ &= -\sqrt{12}\end{aligned}$$



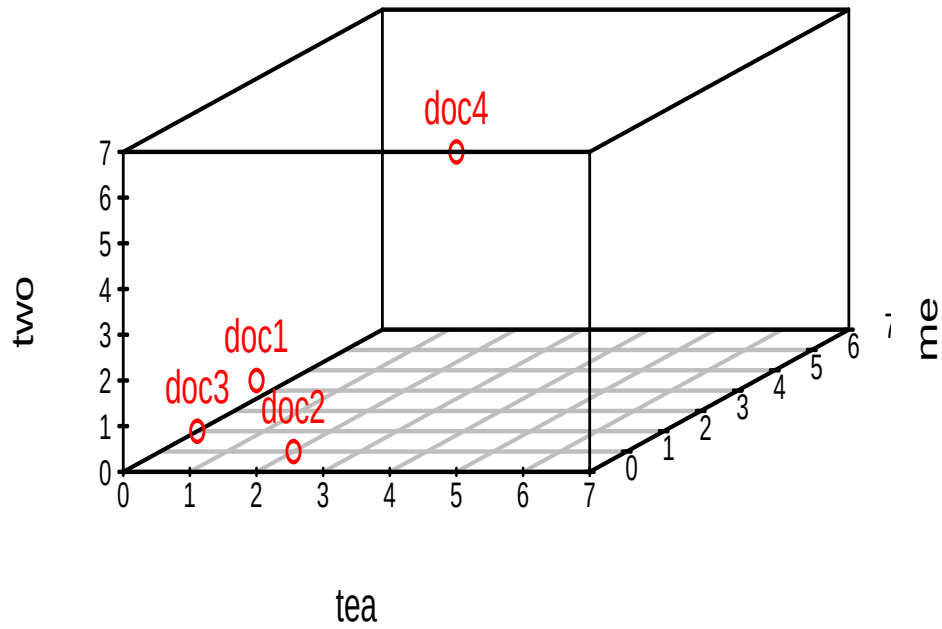
- We can then answer "how similar are two documents?" by "how close are they in term space?" (e.g. by Euclidean distance)
- Note that the computation extends trivially to multiple dimensions (though again it's difficult to visualize)

Document length bias

- Measuring literal distance between documents in term space has problem:
- Documents with lots of terms will be further from origin . . .
- Documents with few terms closer to it . . .
- So we'll find all short documents relatively similar . . .
- Even if they're unrelated

“Long” documents

Point	tea	me	two
doc1	2	0	2
doc2	2	1	0
doc3	0	2	0
doc4	5	0	7



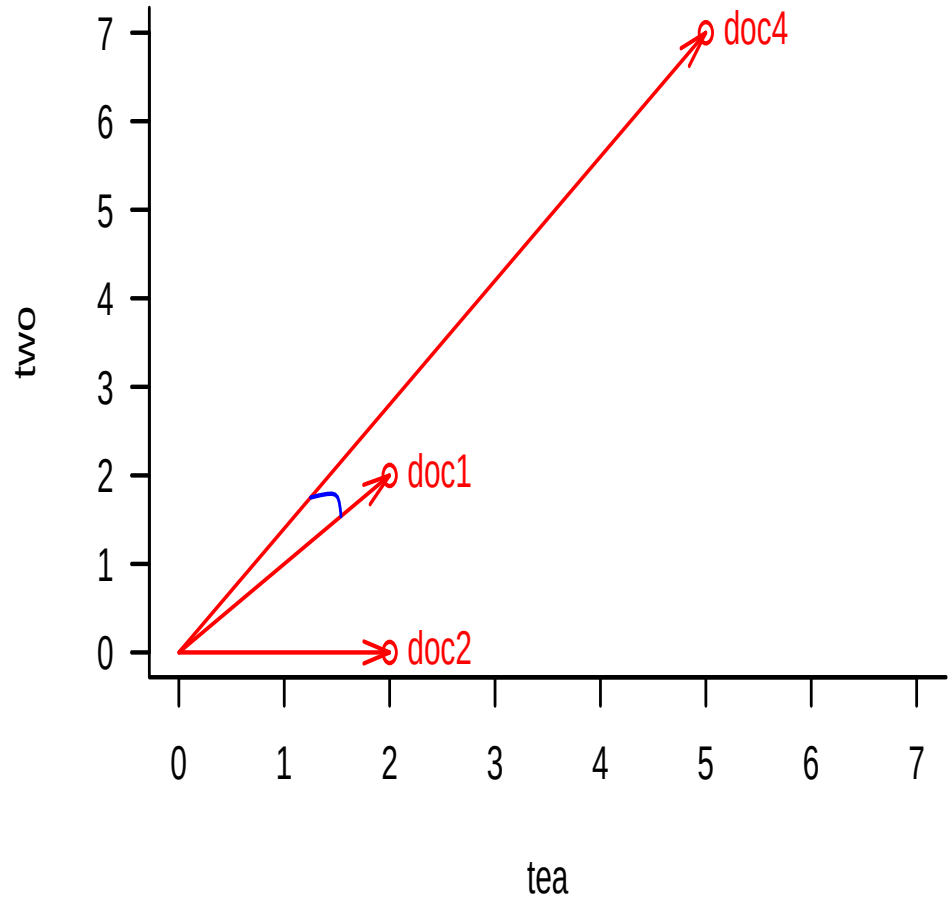
- Doc4, like Doc2, is all about “tea” and “two”.
- But because it is longer, it is in a space by itself.

Angular distance

- To avoid ("normalize" for) the length issue, we instead treat the documents as "vectors"
- (that is, as lines from the origin to their locations in term space)
- and then measure their similarity by the angle between the vectors

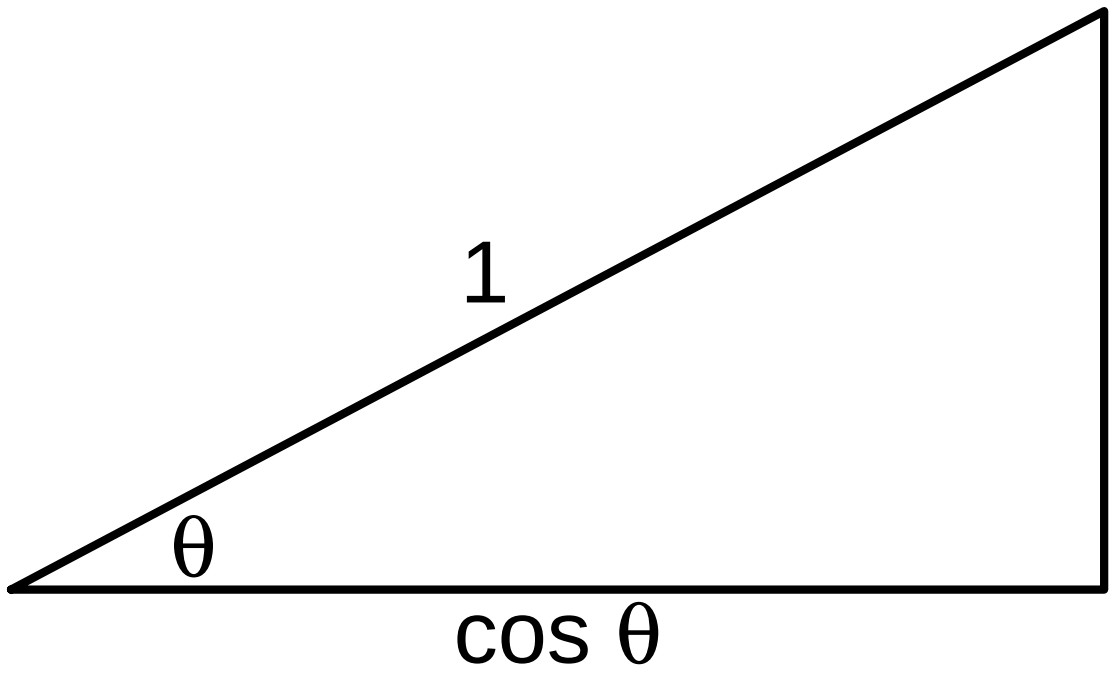
Angular distance

Point	tea	two
doc1	2	2
doc2	2	0
doc4	5	7



- We see that Doc1 and Doc4 are indeed similar

Cosine distance



- In fact, what we measure is the cosine of the angle.
- Larger cosine = closer together (a similarity measure)
- All term values positive, so cosine always between $[0, 1]$
- Also, there's a simple trick for calculating cosine easily

Calculating cosine distance

If we have two vectors, A and B, then the cosine between them is:

$$\cos(A, B) = \frac{A \bullet B}{|A| \cdot |B|} \tag{1}$$

where $A \bullet B$ is the "dot product" (explained next), and $|A|$ and $|B|$ are the length of the vectors. In fact, if we normalize the vectors so that they have *unit length* (i.e. they're 1 unit long), as $a = A/|A|$ and $b = B/|B|$, then:

$$\cos(A, B) = a \bullet b \tag{2}$$

Dot product

If we have two vectors (unit or otherwise):

$$a = \langle a_1, a_2, \cdots, a_n \rangle \tag{3}$$
$$b = \langle b_1, b_2, \cdots, b_n \rangle \tag{4}$$

then their dot product is defined as:

$$a \bullet b = a_1 \cdot b_1 + a_2 \cdot b_2 + \cdots + a_n \cdot b_n \tag{5}$$

That is, it is just the sum of the termwise multiplication between elements.

Speeding up cosine computation

- All document vectors are stored normalized to unit length.
- Cosine distance can be quickly calculated as a summed series of multiplications
- Only dimensions in which *both* elements are non-zero need to be calculated

This (well, this whole line of geometric modelling) is known as the *vector space model*)

Down-weighting common terms

- Terms that occur in a lot of documents tend to be less discriminative than terms that appear in fewer
- For instance, the document “llama stories” is more distinctively about llamas than it is about stories
- But (perversely) frequent terms can have more impact upon similarity comparisons
- (they tend to “bend” the term space towards their dimension)

Inverse document frequency

- A commonly-used measure of a term's discriminative power is its *inverse document frequency* or IDF.
- If N is the number of documents in the corpus, and df_t is the number of documents that term t appears in ...
- Then (under one formulation) the IDF of t is defined as:

$$\text{idf}_t = \log \frac{N}{\text{df}_t} \tag{6}$$

TF*IDF

- The weight of a term's appearance in a document is frequently calculated by combining the terms frequency or TF in the document with its inverse document frequency, or IDF.

$$w_{t,d} = \mathrm{tf}_{d,t} * \mathrm{idf}_t \tag{7}$$

- This term–document score is known as TF*IDF, and is quite widely used.

Sub-linear TF weighting

- The TF term in TF*IDF can be the raw term frequency, $f_{d,t}$
- However, a term that occurs 20 times is not generally 20 times as important as a term that occurs once.
- Therefore, an alternative formulation of the TF component is:

$$\text{tf}_{d,t} = \log(1 + f_{d,t}) \tag{8}$$

- Note there are lots of variant formulations and combinations!
- Whatever formulation is used, the unit-length-normalized TF*IDF scores are the precomputed and stored, so that similarity comparison is just a dot product

Memory space requirements

- The full term–document matrix is very large.
- Even a small collection (by modern standards) might have 1 million documents and 500,000 “terms”
- This would require 500 billion elements in the matrix
- Or 2TB of memory if each entry was 4 bytes in size

Sparseness of matrix

- Most of the entries in this term–document matrix will be empty
- Because only a few terms appear in each document, and vice versa
- Storing all the empty cells is wasteful (especially since they contribute no value to the dot product similarity computation)
- Various "spare matrix" representations are possible
- . . . and these become highly specialized for query processing (see later lecture)

Comparing terms in document space

- So far, we have considered comparing documents by projecting them into term space
- But it is also possible to compare terms by projecting them into document space
- What would it mean for a two terms to be “close” when projected into document space?

Looking back and forward

Back

- We can interpet the TDM geometrically, by projecting documents into term space
- The distance between documents (or more specifically, the cosine of the angle between their vectors) is a measure of similarity
- TF*IDF is a common scoring method for the "weight" of a term in a document (the document's location in that term's dimension)
- Unit length normalization permits calculation to be performed as a simple vector dot product

Looking back and forward

Forward

- In the next lecture, we will look at how to perform queries in the vector space model (hint: treat the query as a pseudo-document, and find what [real] documents are close to it)
- After that, we will look at text clustering, in which we automatically group documents into "clusters" based upon their closeness in the term space

Further reading

- Chapter 6, "Scoring, term weighting & the vector space model"[1], of Manning, Raghavan, and Schutze, *Introduction to Information Retrieval*
- G Salton, A Wong, CS Yang, "A Vector Space Model for Automatic Indexing"[2], Communications of the ACM, 1975 (a classic early presentation of the vector space model).
- J Zobel, A Moffat, "Exploring the Similarity Space"[3], ACM SIGIR Forum, 1998 (for a survey of all the different combinations of TF, IDF, and normalizations then in use in similarity computation; writte by two now-professors in our department).

---

[1] `http://nlp.stanford.edu/IR-book/pdf/06vect.pdf`

[2] `http://ecommons.library.cornell.edu/bitstream/1813/6057/2/74-218.ps`

[3] `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18.6193&rep=rep1&type=pdf`