Web Information Retrieval

Lecture 6

Vector Space Model

Recap of the last lecture

- Parametric and field searches
  - Zones in documents
- Scoring documents: zone weighting
  - Index support for scoring
- *tf*×*idf* and vector spaces

This lecture

- Vector space model
- Efficiency considerations
  - Nearest neighbors and approximations

Documents as vectors

- At the end of Lecture 5 we said:
- Each doc *j* can now be viewed as a vector of *tf*×*idf* values, one component for each term
- So we have a vector space
  - terms are axes
  - docs live in this space
  - even with stemming, may have 20,000+ dimensions

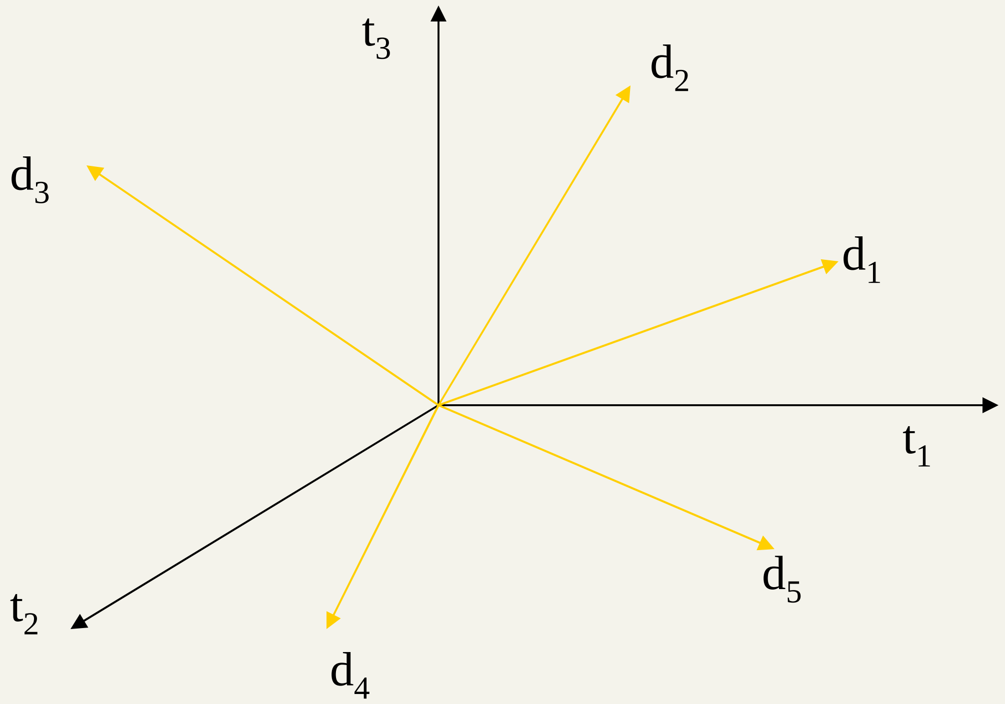
Example

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Brutus	3.0	8.3	0.0	1.0	0.0	0.0
Caesar	2.3	2.3	0.0	0.5	0.3	0.3
mercy	0.5	0.0	0.7	0.9	0.9	0.3

Why turn docs into vectors?

- First application: Query-by-example
  - Given a doc D, find others “like” it.
- Now that D is a vector, find vectors (docs) “near” it.

Intuition



Postulate: Documents that are "close together" in the vector space talk about the same things.

The vector space model

Query as vector:

- We regard query as short document
- We return the documents ranked by the closeness of their vectors to the query, also represented as a vector.

Desiderata for proximity

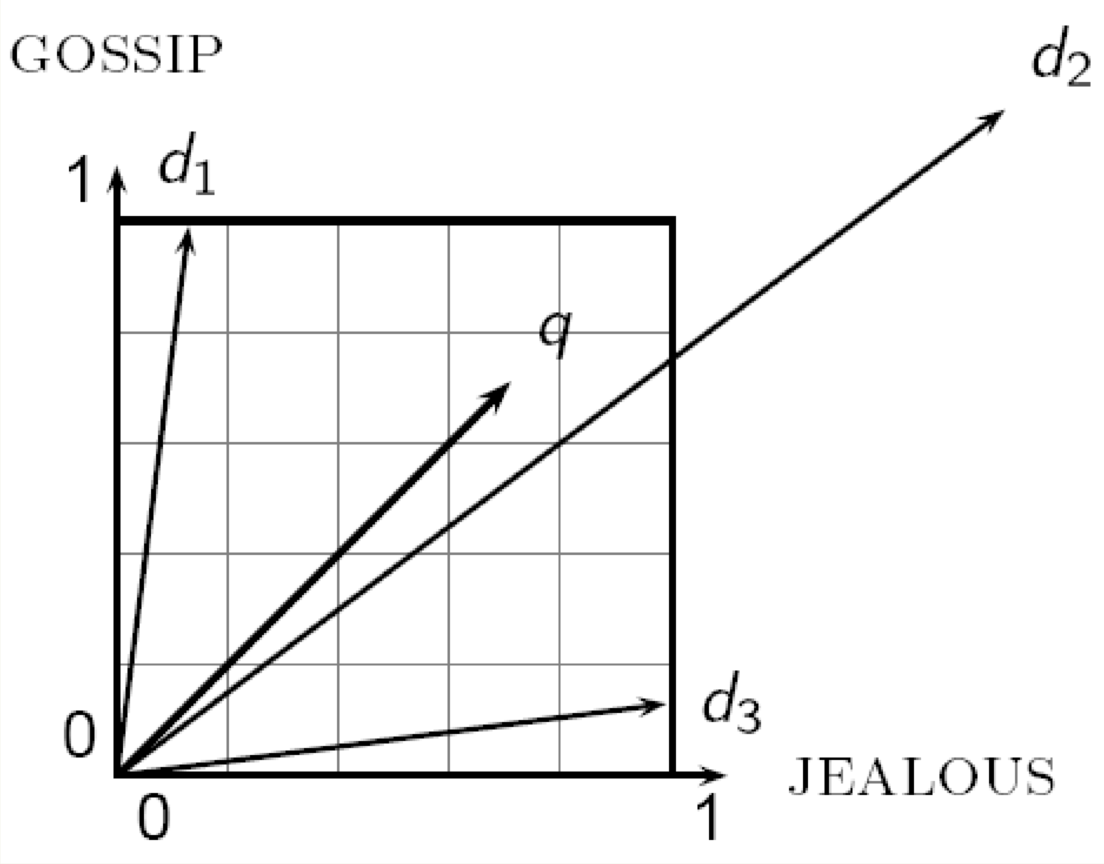
- If d_1 is near d_2, then d_2 is near d_1.
- If d_1 near d_2, and d_2 near d_3, then d_1 is not far from d_3.
- No doc is closer to d than d itself.

First cut

- Distance between d_1 and d_2 is the length of the vector $|d_1 - d_2|$.
  - Euclidean distance
- Why is this not a great idea?
- We still haven’t dealt with the issue of length normalization
- However, we can implicitly normalize by looking at *angles* instead

Why distance is a bad idea

The Euclidean distance between $\vec{q}$ and $\vec{d}_2$ is large even though the distribution of terms in the query $\vec{q}$ and the distribution of terms in the document $\vec{d}_2$ are very similar.



Use angle instead of distance

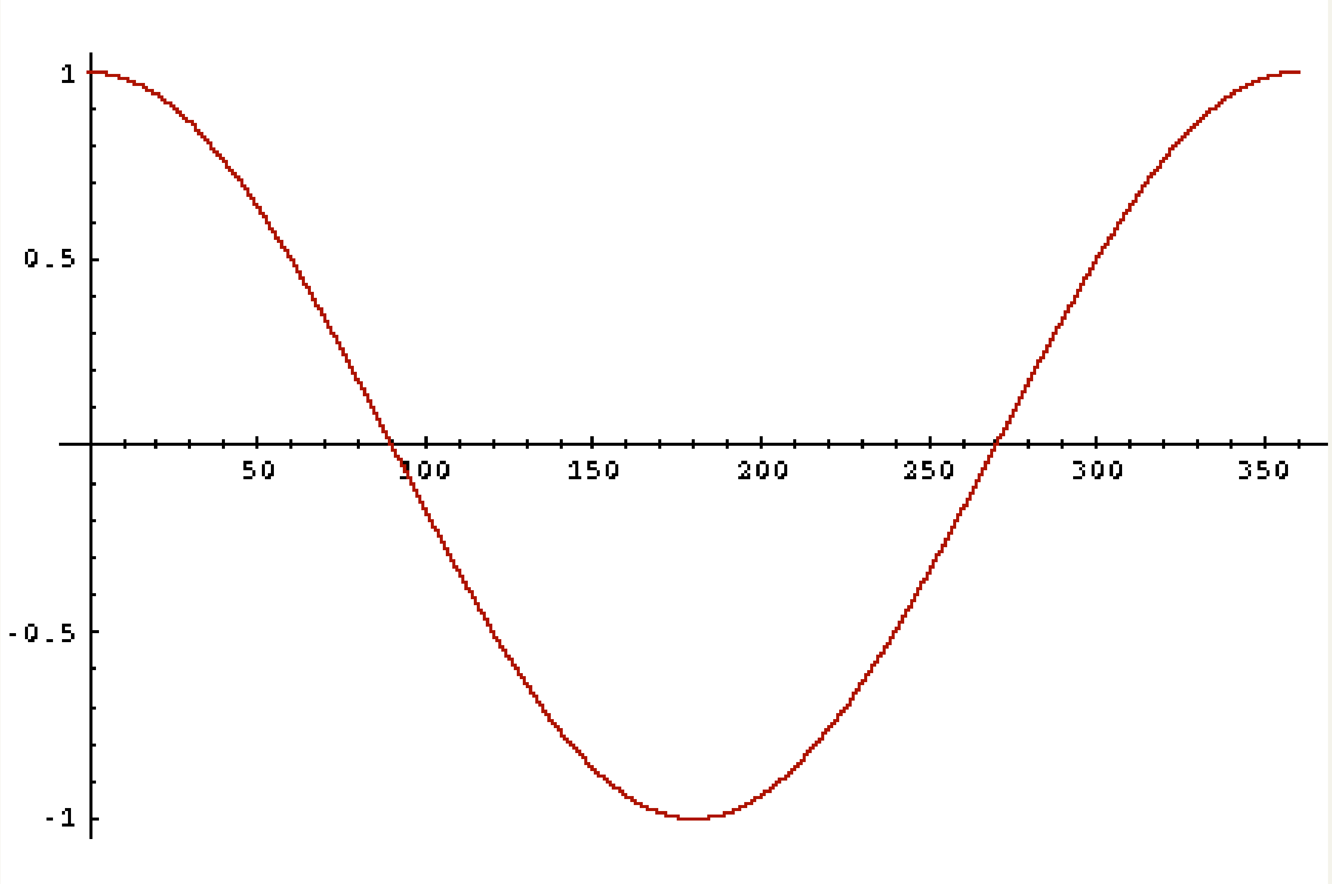
- Thought experiment: take a document *d* and append it to itself. Call this document *d*′.
- "Semantically" d and d′ have the same content
- The Euclidean distance between the two documents can be quite large
- The angle between the two documents is 0, corresponding to maximal similarity.

- Key idea: Rank documents according to angle with query.

From angles to cosines

- The following two notions are equivalent.
  - Rank documents in <u>decreasing</u> order of the angle between query and document
  - Rank documents in <u>increasing</u> order of cosine(query,document)
- Cosine is a monotonically decreasing function for the interval of interest [0^o, 90^o]

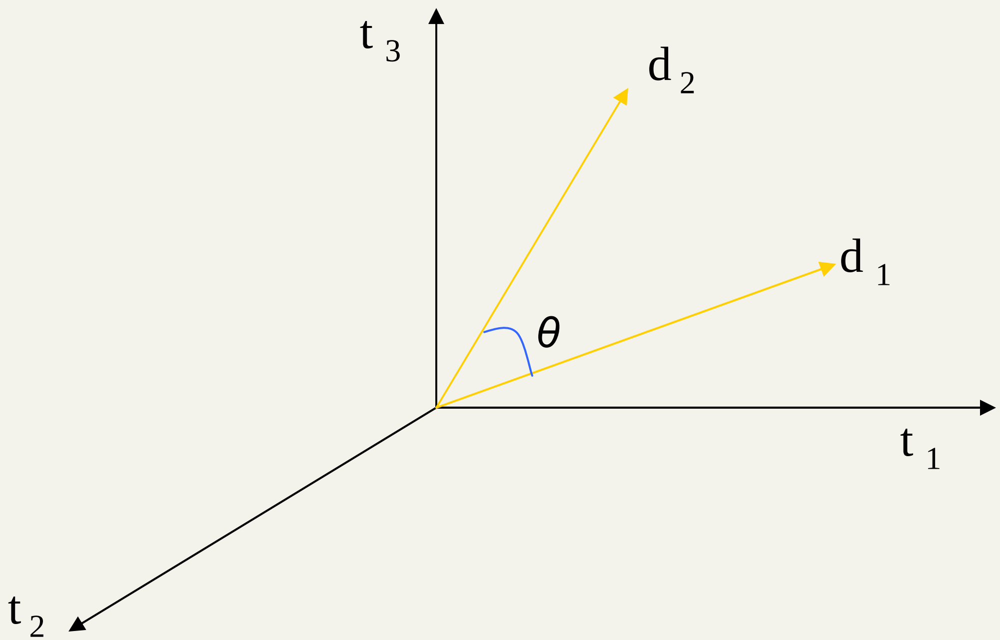
From angles to cosines



- But how – *and why* – should we be computing cosines?

Cosine similarity

- Distance between vectors d_1 and d_2 *captured* by the cosine of the angle x between them.
- Note – this is *similarity*, not distance



Cosine similarity

- A vector can be *normalized* (given a length of 1) by dividing each of its components by its length – here we use the L_2 norm

$$\|\mathbf{x}\|_2 = |x| = \sqrt{\sum_i x_i^2}$$

- This maps vectors onto the unit sphere:
- Then, $$\left|\vec{d}_j\right| = \sqrt{\sum_{i=1}^{M} w_{i,j}} = 1$$
- Longer documents don't get more weight

Cosine similarity

$$sim(d_j, d_k) = \cos(d_j, d_k) = \frac{\vec{d}_j \cdot \vec{d}_k}{\left|\vec{d}_j\right|\left|\vec{d}_k\right|} = \frac{\sum_{i=1}^{M} w_{i,j} w_{i,k}}{\sqrt{\sum_{i=1}^{M} w_{i,j}^2}\sqrt{\sum_{i=1}^{M} w_{i,k}^2}}$$

- Cosine of angle between two vectors
- The denominator involves the lengths of the vectors.

Normalization

Normalized vectors

- For normalized vectors, the cosine is simply the dot product:

$$\cos(\vec{d}_j, \vec{d}_k) = \vec{d}_j \cdot \vec{d}_k$$

Cosine similarity exercises

- *Exercise: Rank the following by decreasing cosine similarity:*
  - Two docs that have only frequent words ***(the, a, an, of)*** in common.
  - Two docs that have no words in common.
  - Two docs that have many rare words in common ***(wingspan, tailfin).***

Exercise

- Euclidean distance between vectors:

$$\left| d_j - d_k \right| = \sqrt{\sum_{i=1}^{M} \left(d_{i,j} - d_{i,k} \right)^2}$$

- Show that, for normalized vectors, Euclidean distance gives the same proximity ordering as the cosine measure

Example

- **Docs:** Austen's *Sense and Sensibility*, *Pride and Prejudice*; Bronte's *Wuthering Heights*

	SaS	PaP	WH
affection	**115**	**58**	**20**
jealous	**10**	**7**	**11**
gossip	**2**	**0**	**6**

Example

- **Docs:** Austen's *Sense and Sensibility*, *Pride and Prejudice*; Bronte's *Wuthering Heights*

	SaS	PaP	WH
affection	**115**	**58**	**20**
jealous	**10**	**7**	**11**
gossip	**2**	**0**	**6**

	SaS	PaP	WH
affection	**0.996**	**0.993**	**0.847**
jealous	**0.087**	**0.120**	**0.466**
gossip	**0.017**	**0.000**	**0.254**

Example

- **Docs:** Austen's *Sense and Sensibility*, *Pride and Prejudice*; Bronte's *Wuthering Heights*

	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

	SaS	PaP	WH
affection	0.996	0.993	0.847
jealous	0.087	0.120	0.466
gossip	0.017	0.000	0.254

- cos(SAS, PAP) = .996 x .993 + .087 x .120 + .017 x 0.0 = 0.999
- cos(SAS, WH) = .996 x .847 + .087 x .466 + .017 x .254 = 0.889

Queries as vectors

- Key idea 1: Do the same for queries: represent them as vectors in the space
- Key idea 2: Rank documents according to their proximity to the query in this space
- proximity = similarity of vectors

Cosine(query,document)

Dot product

Unit vectors

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\bullet\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\vec{q}}{|\vec{q}|}\bullet\frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{M} q_i d_i}{\sqrt{\sum_{i=1}^{M} q_i^2}\sqrt{\sum_{i=1}^{M} d_i^2}}$$

$\cos(\vec{q}, \vec{d})$ is the cosine similarity of $\vec{q}$ and $\vec{d}$ or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

Summary: What's the real point of using vector spaces?

- Key: A user's query can be viewed as a (very) short document.
- Query becomes a vector in the same space as the docs.
- Can measure each doc's proximity to it.
- Natural measure of scores/ranking – no longer Boolean.
  - Queries are expressed as bags of words
- Other similarity measures: see http://www.lans.ece.utexas.edu/~strehl/diss/node52.html for a survey

Interaction: vectors and phrases

- Phrases don't fit naturally into the vector space world:
  - ***"hong kong" "new york"***
  - Positional indexes don't capture tf/idf information for ***"hong kong"***
- Biword indexes treat certain phrases as terms
  - For these, can pre-compute tf/idf.
- A hack: we cannot expect end-user formulating queries to know what phrases are indexed

Vectors and Boolean queries

- Vectors and Boolean queries really don’t work together very well
- We cannot express AND, OR, NOT, just by summing term frequencies

Vector spaces and other operators

- Vector space queries are apt for no-syntax, bag-of-words queries
  - Clean metaphor for similar-document queries
- Not a good combination with Boolean, positional query operators, phrase queries, …
- But …

Query language vs. scoring

- May allow user a certain query language, say
  - Freetext basic queries
  - Phrase, wildcard etc. in Advanced Queries.
- For scoring (oblivious to user) may use all of the above, e.g. for a freetext query
  - Highest-ranked hits have query as a phrase
  - Next, docs that have all query terms near each other
  - Then, docs that have some query terms, or all of them spread out, with tf x idf weights for scoring

Exercises

- How would you augment the inverted index built in lectures 1–3 to support cosine ranking computations?
- What information do we need to store?
- Walk through the steps of serving a query.
- *The math of the vector space model is quite straightforward, but being able to do cosine ranking efficiently at runtime is nontrivial*

Resources

- IIR Chapters 6.3, 7.3