



TF-IDF



David Kauchak

cs160

Fall 2009

adapted from:

<http://www.stanford.edu/class/cs276/handouts/lecture6-tfidf.ppt>

Administrative

- Homework 3 available soon
- Assignment 2 available soon
- Popular media article

Ranked retrieval

- Thus far, our queries have all been Boolean
 - Documents either match or don't
- Good for expert users with precise understanding of their needs and the collection
- Also good for applications: Applications can easily consume 1000s of results
 - Not good for the majority of users
 - Most users incapable of writing Boolean queries (or they are, but they think it's too much work)
- More importantly: most users don't want to wade through 1000s of results

Problem with Boolean search: feast or famine

- Boolean queries often result in either too few (=0) or too many (1000s) results.
- Query 1: “*standard user dlink 650*” → 200,000 hits
- Query 2: “*standard user dlink 650 no card found*”: 0 hits
- It takes skill to come up with a query that produces a manageable number of hits
- With a ranked list of documents it does not matter how large the retrieved set is

Scoring as the basis of ranked retrieval

- We wish to return in order the documents most likely to be useful to the searcher
- Assign a score that measures how well document and query “match”



Query-document matching scores

- We need a way of assigning a score to a query/document pair
- Besides whether or not a query (or query word) occurs in a document, what other indicators might be useful?
 - How many *times* the word occurs in the document
 - Where the word occurs
 - How “important” is the word – for example, **a** vs. **motorcycle**

Recall: Binary term-document incidence matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Each document is represented by a binary vector $\in \{0,1\}^{|V|}$

Term-document count matrices

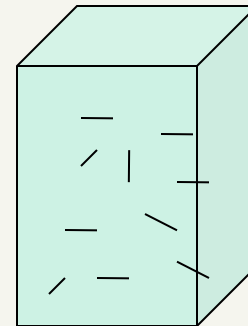
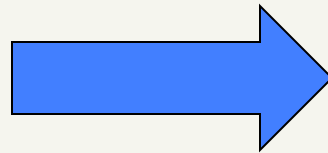
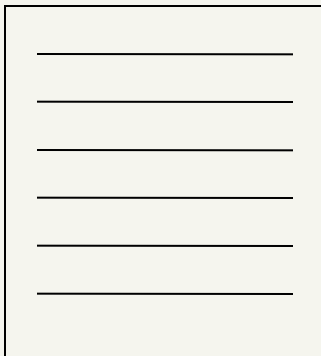
- Consider the number of occurrences of a term in a document:
 - Each document is a **count vector** in \mathbb{N}^v : a column below

	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

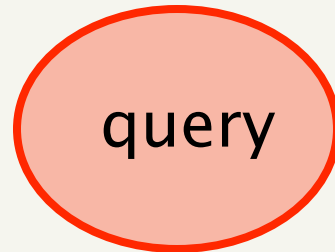
What information is lost with this representation?

Bag of words representation

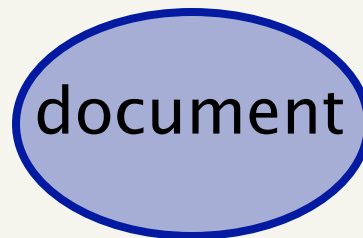
- Represent a document by the occurrence counts of each word
- **Ordering** of words is lost
- *John is quicker than Mary* and *Mary is quicker than John* have the same vectors



Boolean queries: another view



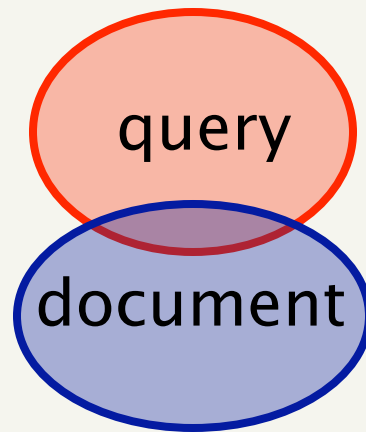
query



document

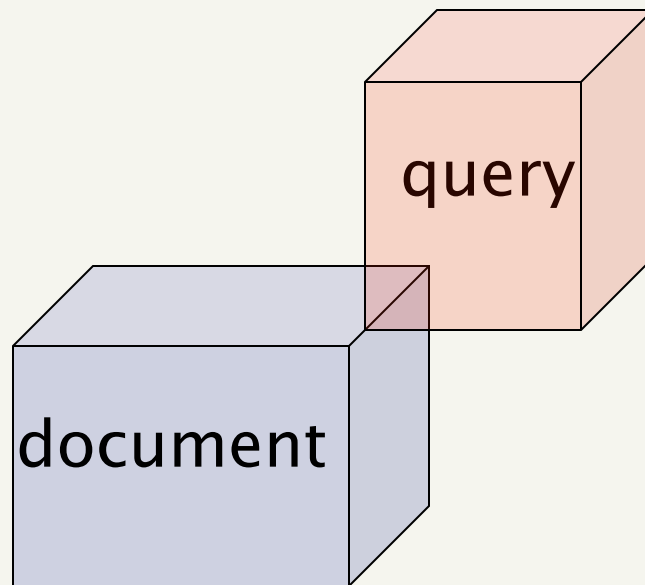
For the boolean representation, we can view a query/document as a set of words

Boolean queries: another view



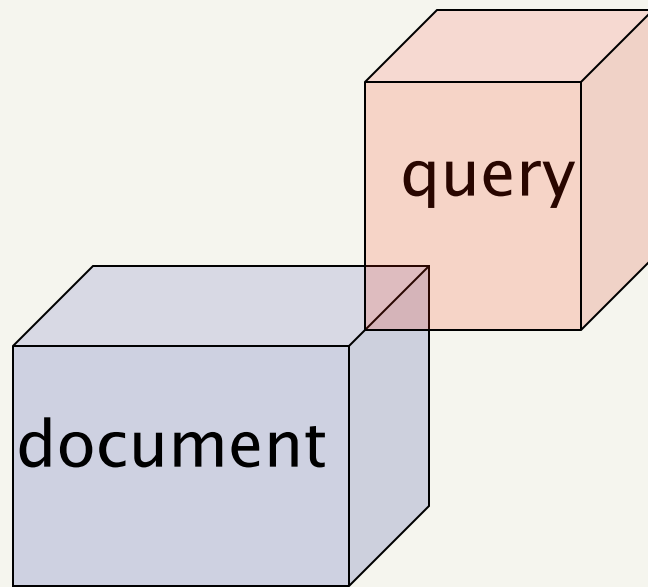
We want to return those documents where there is an overlap, i.e. intersection between the two sets

Bag of words



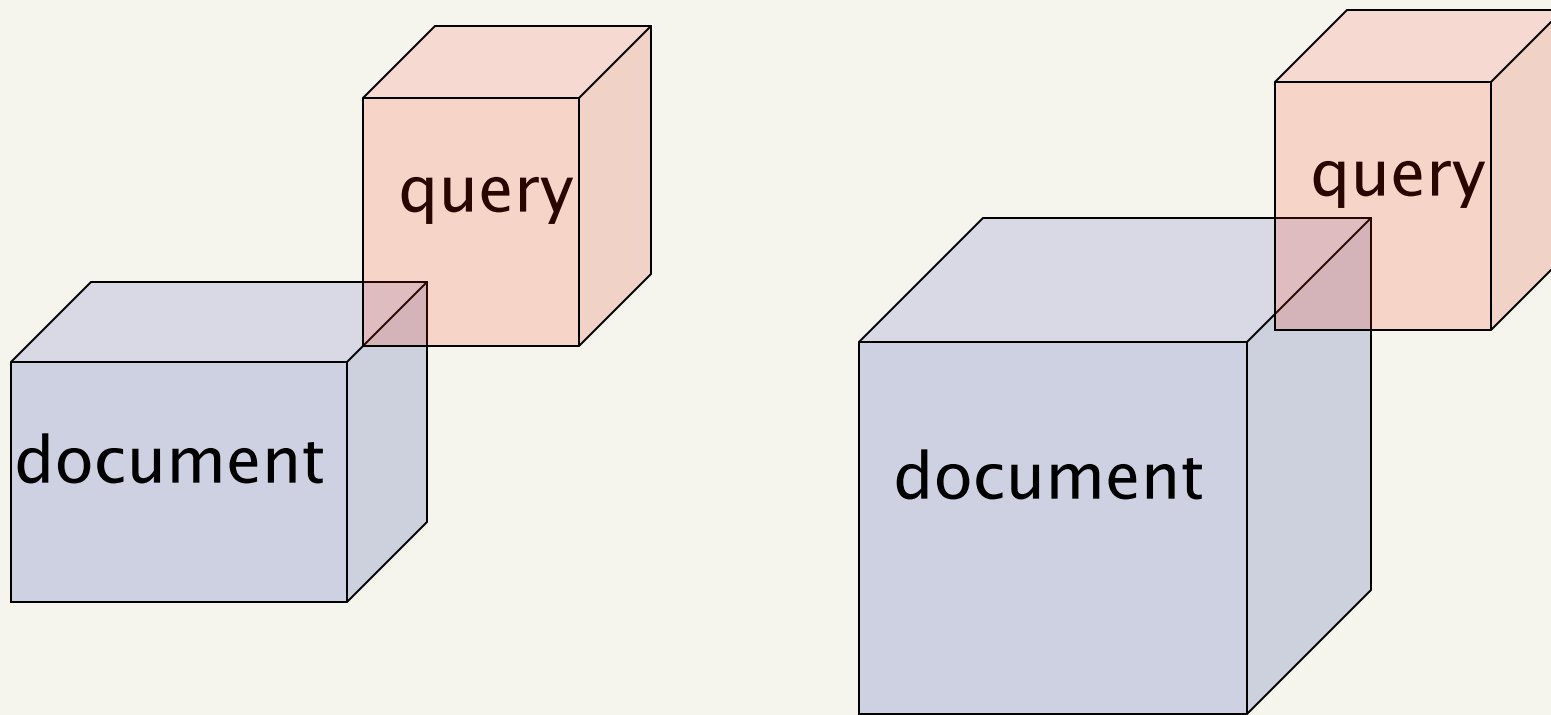
What is the notion of “intersection” for the bag of words model?

Bag of words



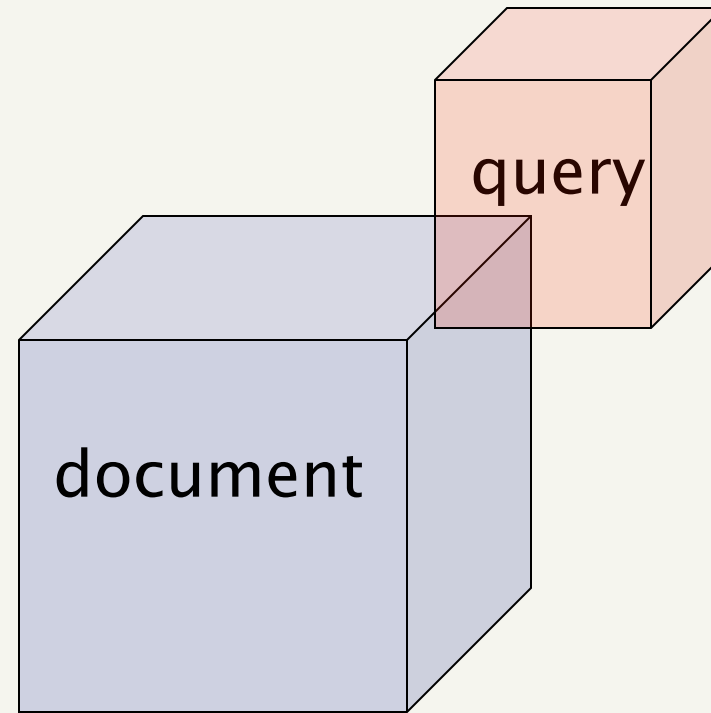
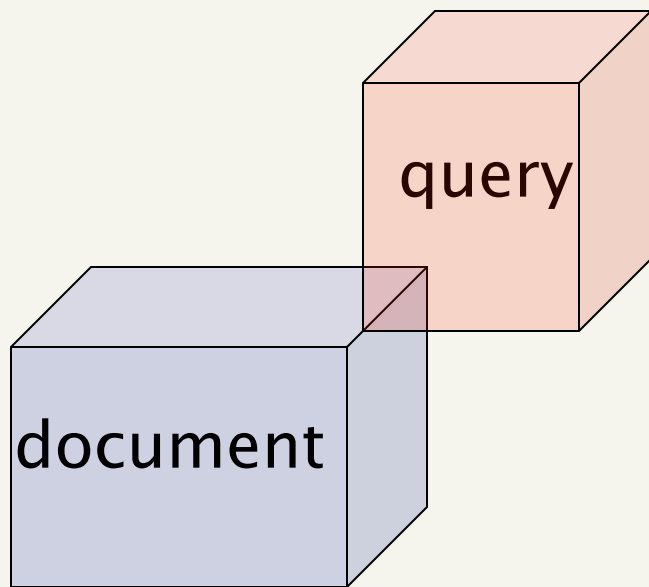
Want to take into account term frequency

Some things to be careful of...



Say I take the document and simply append it to itself. What happens to the overlap?

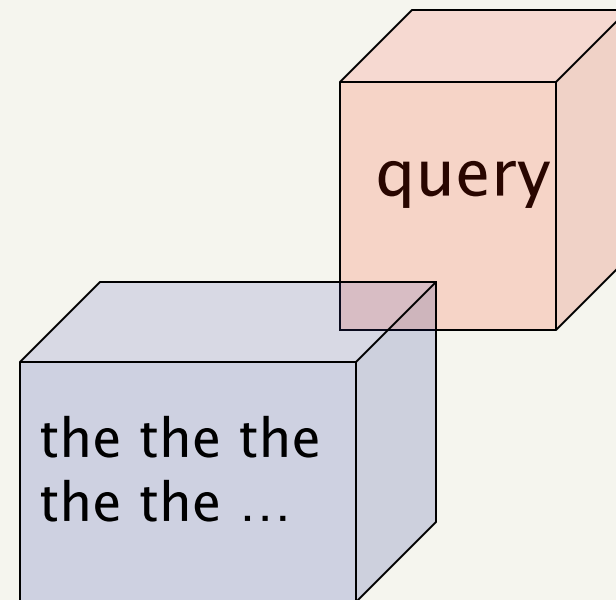
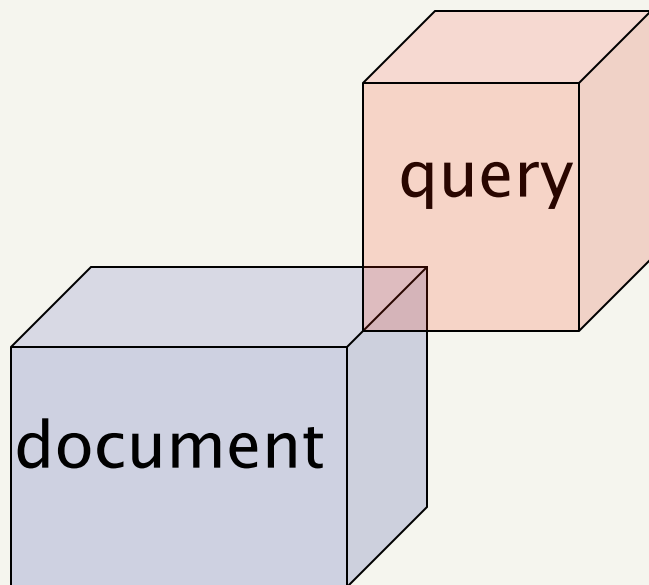
Some things to be careful of...



What is the issue?

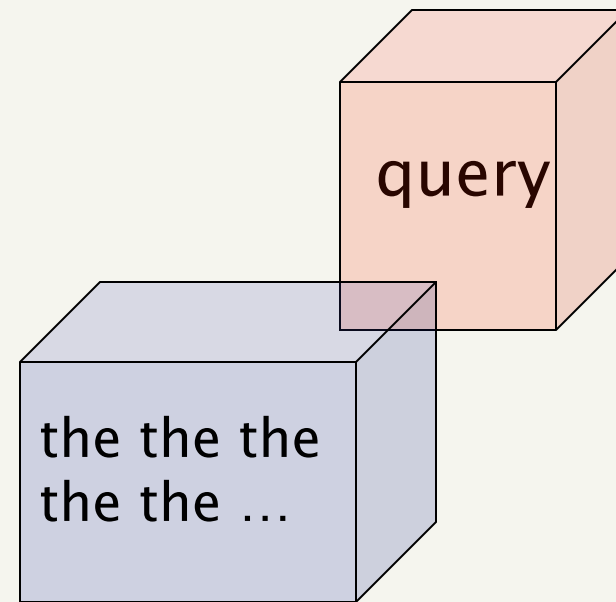
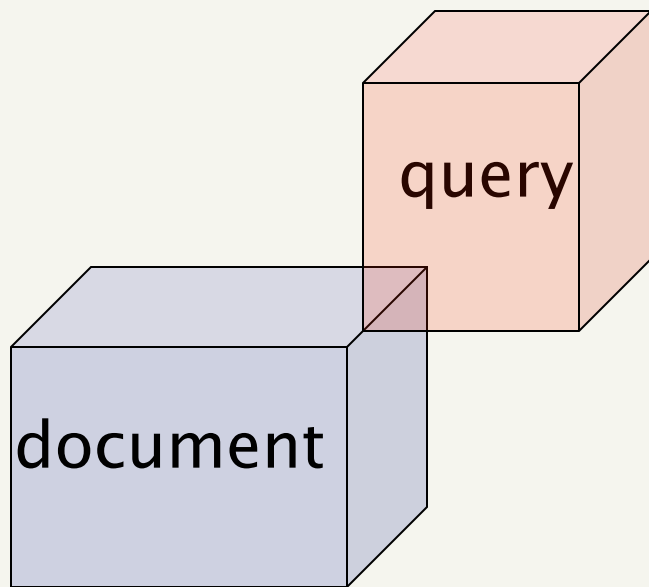
Need some notion of the length of a document

Some things to be careful of...



What about a document that contains only frequent words, e.g. **the**?

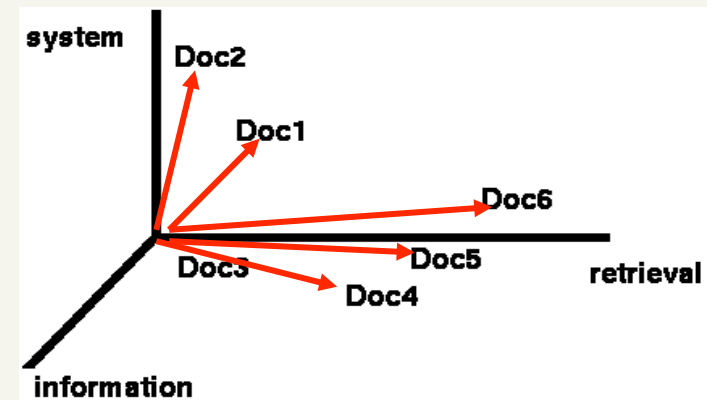
Some things to be careful of...



Need some notion of the importance of words

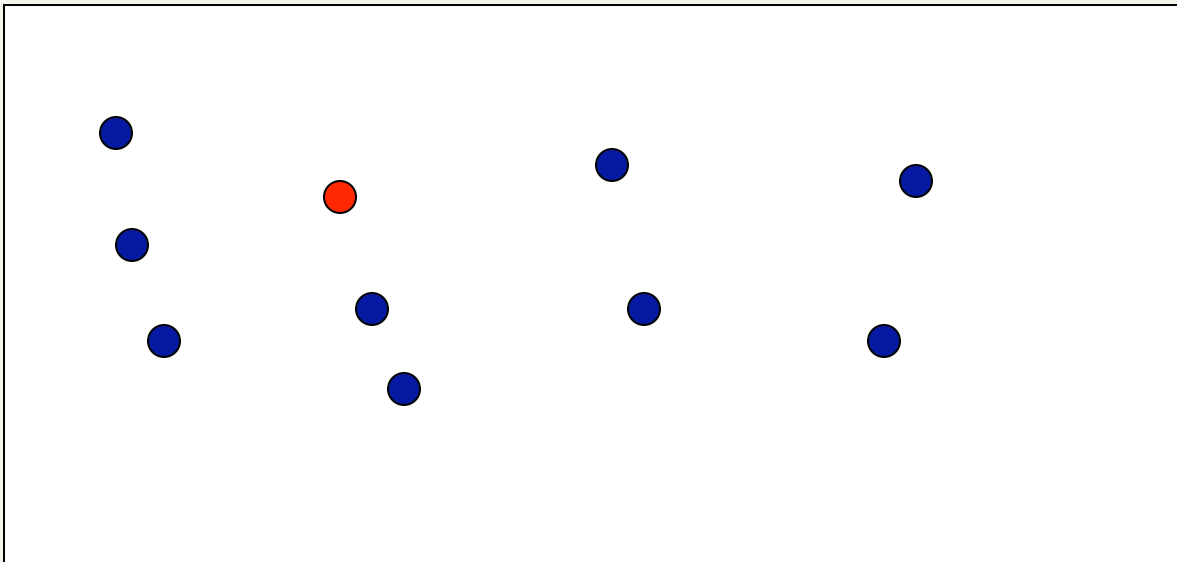
Documents as vectors

- We have a $|V|$ -dimensional vector space
- Terms are axes of the space
- Documents are points or vectors in this space
- Very high-dimensional: hundreds of millions of dimensions when you apply this to a web search engine
- This is a very sparse vector - most entries are zero



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



How should we rank documents?

$|V|$ dimensional space

Formalizing vector space proximity

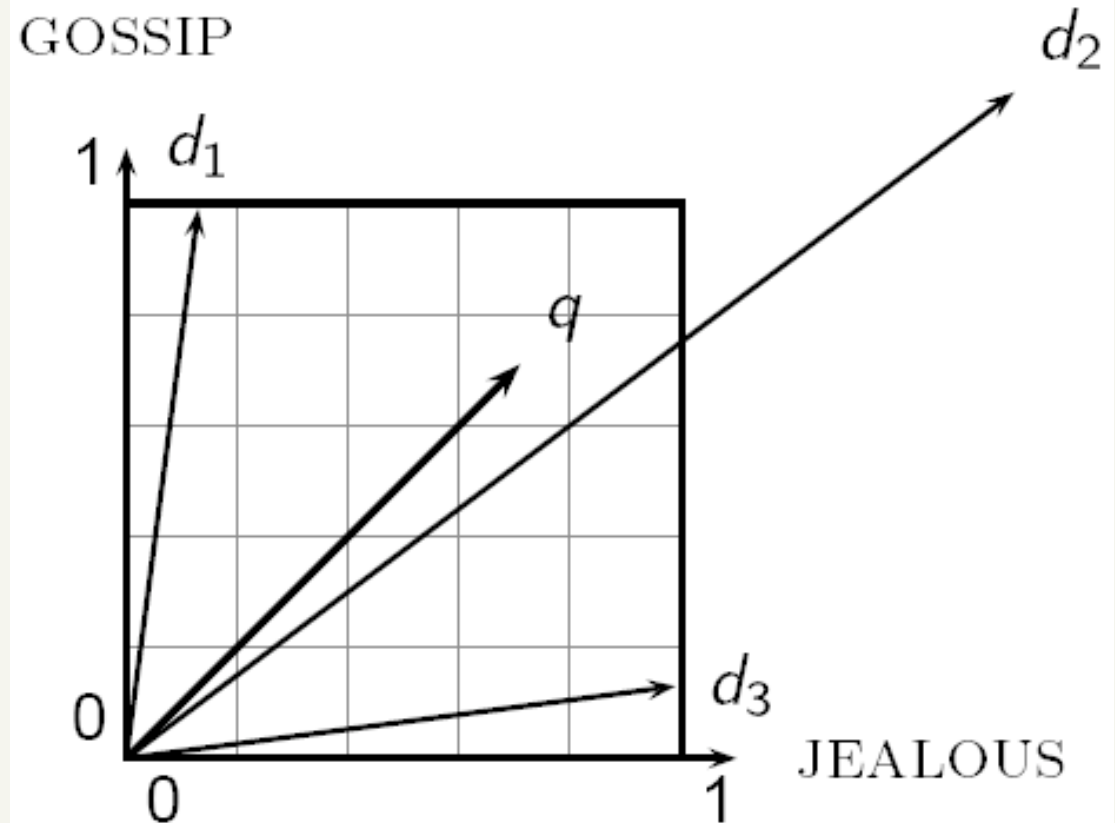
- We have points in a $|V|$ dimensional space
- How can we measure the proximity of documents in this space?

- First cut: distance between two points
- Euclidean distance?

Why distance is a bad idea

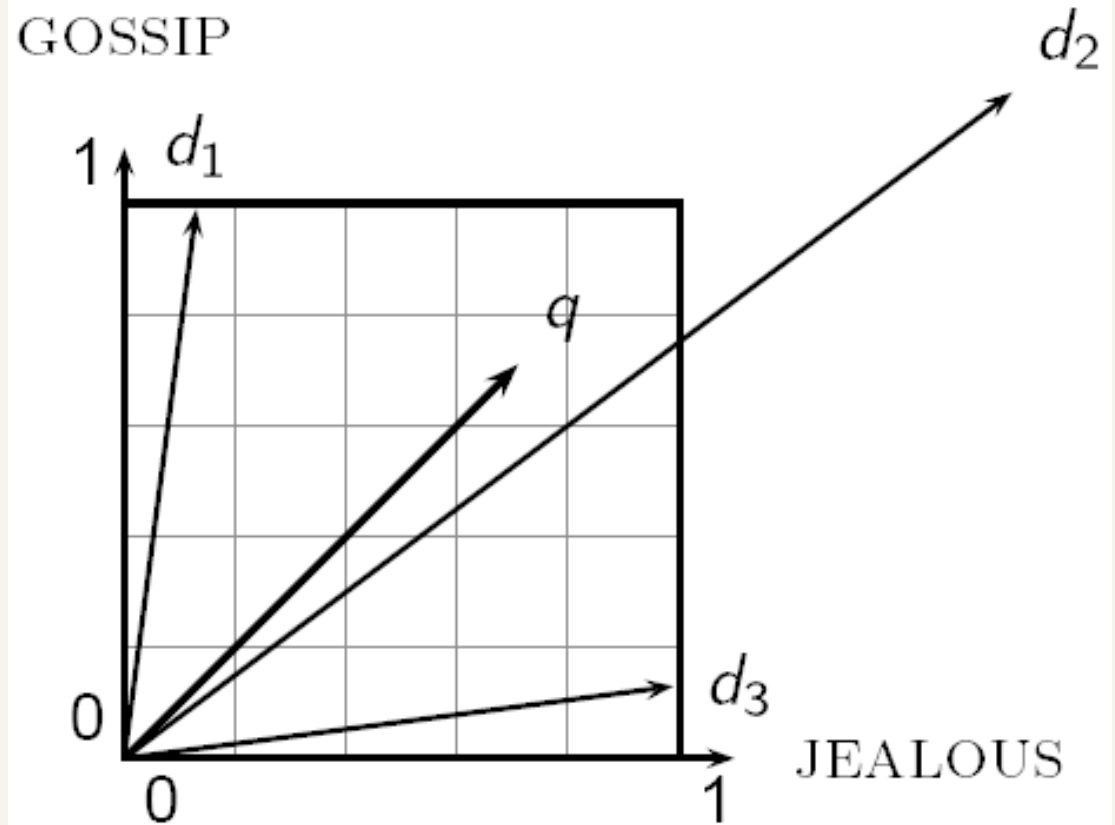
Which document is closer using Euclidian distance?

Which do you think should be closer?



Issues with Euclidian distance

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.

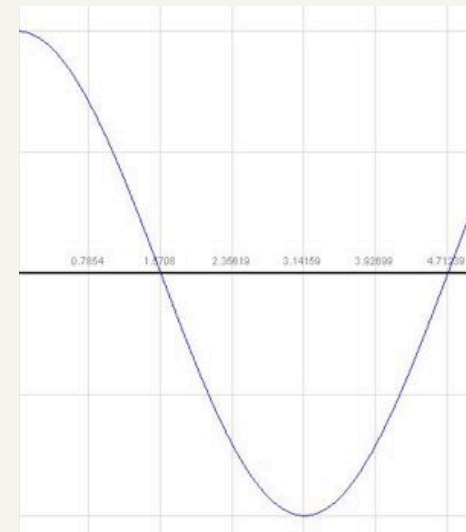


Use angle instead of distance

- 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
- Any other ideas?
- Rank documents according to angle with query

From angles to cosines

- Cosine is a monotonically decreasing function for the interval $[0^\circ, 180^\circ]$
- The following two notions are equivalent.
 - Rank documents in decreasing order of the angle between query and document
 - Rank documents in increasing order of $\text{cosine}(\text{query}, \text{document})$



cosine(query, document)

How do we calculate the cosine between two vectors?

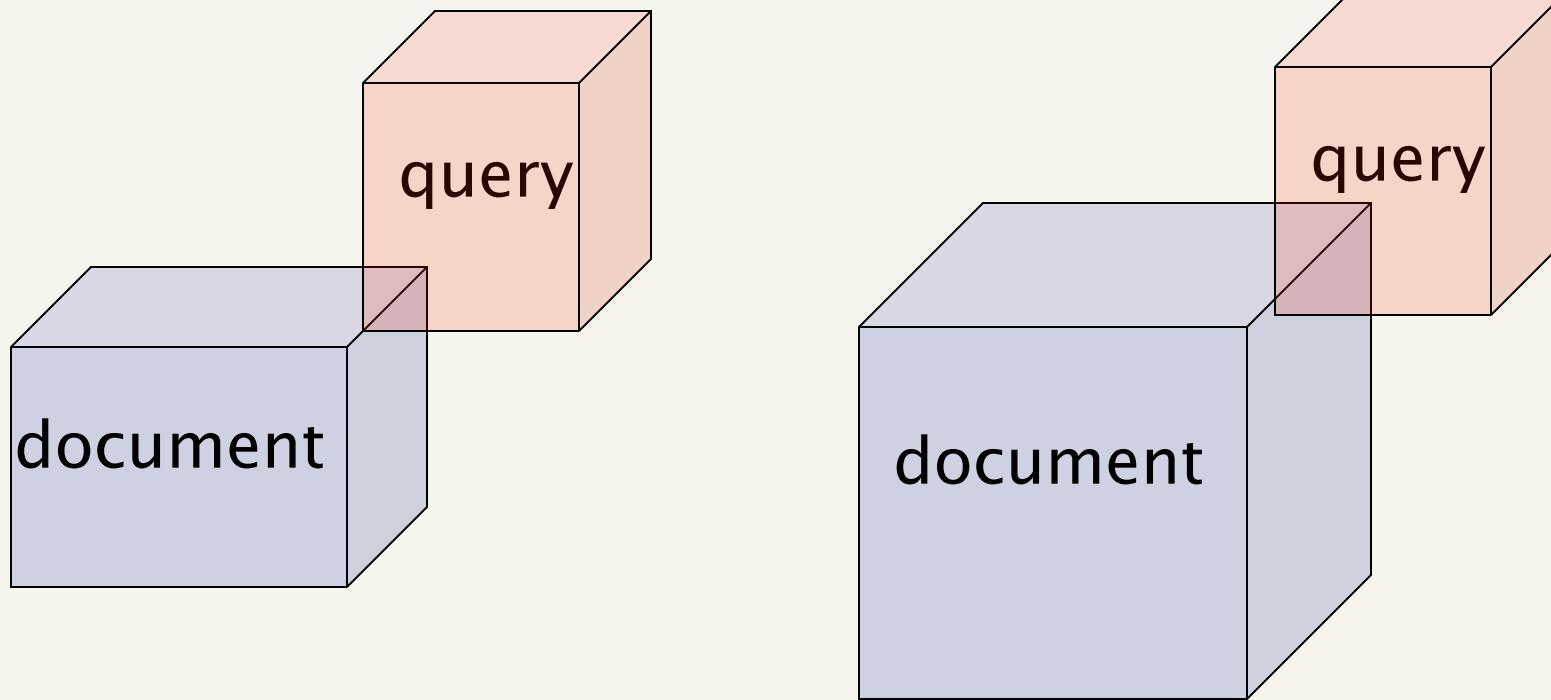
cosine(query, document)

Dot product

$$\cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} = \sum_{i=1}^{|V|} q_i d_i$$

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

Some things to be careful of...



Need some notion of the length of a document

Length normalization

- A vector can be (length-) normalized by dividing each of its components by its length – for this we use the L_2 norm:
$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$
- Dividing a vector by its L_2 norm makes it a unit (length) vector
- **What is a “unit vector” or “unit length vector”?**
- Effect on the two documents d and d' (d appended to itself) from earlier slide: they have identical vectors after length-normalization.

cosine(query, document)

Dot product

Unit vectors

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

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

Cosine similarity with 3 documents

How similar are
the novels:

SaS: *Sense and
Sensibility*

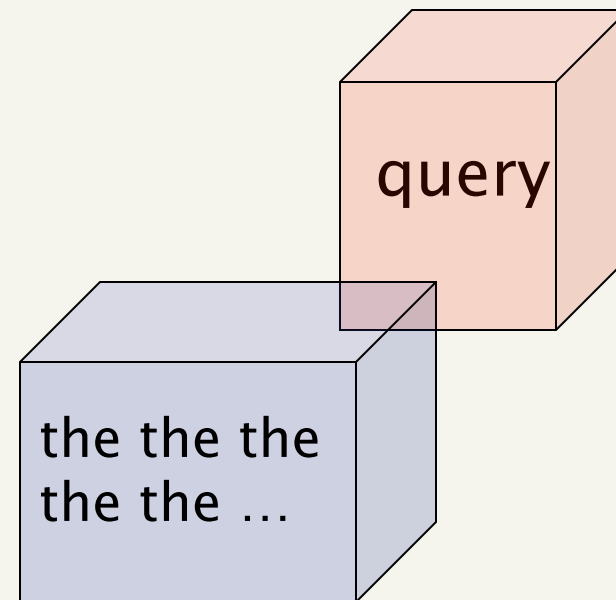
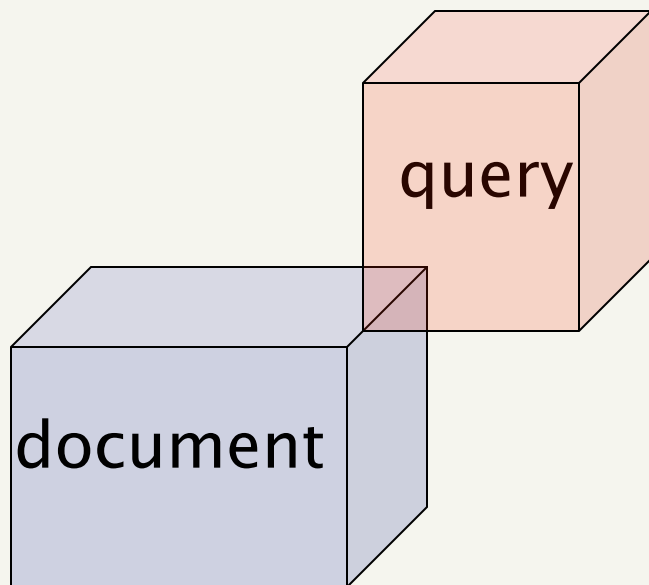
PaP: *Pride and
Prejudice*, and

WH: *Wuthering
Heights*?

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

Term frequencies (counts)

Some things to be careful of...



Need some notion of the importance of words

Term importance

- Rare terms are more informative than frequent terms
 - Recall stop words
- Consider a term in the query that is rare in the collection (e.g., *arachnocentric*)
- A document containing this term is very likely to be relevant to the query *arachnocentric*
- → We want a high weight for rare terms like *arachnocentric*
- Ideas?

Document frequency

- We will use document frequency (df) to capture this in the score
- Terms that occur in many documents are weighted less, since overlapping with these terms is very likely
 - In the extreme case, take a word like **the** that occurs in EVERY document
- Terms that occur in only a few documents are weighted more

Collection vs. Document frequency

- The collection frequency of t is the number of occurrences of t in the collection, counting multiple occurrences
- Example:

Word	Collection frequency	Document frequency
<i>insurance</i>	10440	3997
<i>try</i>	10422	8760

- Which word is a better search term (and should get a higher weight)?

Document frequency

- How does “importance” or “informativeness” relate to document frequency?

Word	Collection frequency	Document frequency
<i>insurance</i>	10440	3997
<i>try</i>	10422	8760

Inverse document frequency

- df_t is the document frequency of t : the number of documents that contain t
 - df is a measure of the informativeness of t
- We define the idf (inverse document frequency) of t by

$$idf_t = \log N/df_t$$

- We use $\log N/df_t$ instead of N/df_t to “dampen” the effect of idf

idf example, suppose $N= 1$ million

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

There is one idf value for each term t in a collection.

idf example, suppose $N= 1$ million

term	df_t	idf_t
calpurnia	1	
animal	100	
sunday	1,000	
fly	10,000	
under	100,000	
the	1,000,000	

What if we didn't use the log to dampen the weighting?

idf example, suppose $N= 1$ million

term	df_t	idf_t
calpurnia	1	1,000,000
animal	100	10,000
sunday	1,000	1,000
fly	10,000	100
under	100,000	10
the	1,000,000	1

What if we didn't use the log to dampen the weighting?

Putting it all together

- We have a notion of term frequency overlap
- We have a notion of term importance
- We have a similarity measure (cosine similarity)

- Can we put all of these together?
 - Define a weighting for each term
 - The tf-idf weight of a term is the product of its tf weight and its idf weight

$$w_{t,d} = \text{tf}_{t,d} \times \log N / \text{df}_t$$

tf-idf weighting

$$w_{t,d} = \text{tf}_{t,d} \times \log N / \text{df}_t$$

- Best known weighting scheme in information retrieval
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection
- Works surprisingly well!
- Works in many other application domains

Binary → count → weight matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Each document is now represented by a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$

We then calculate the similarity using cosine similarity with these vectors

Burstiness

- Take a rare word like *arachnocentric*
- What is the likelihood that *arachnocentric* occurs in a document?
- Given that you've seen it once, what is the likelihood that you'll see it again?
- Does this have any impact on our model?

Log-frequency weighting

- Want to reduce the effect of multiple occurrences of a term
- A document about “Clinton” will have “Clinton” occurring many times
- Rather than use the frequency, use the log of the frequency

$$w_{t,d} = \begin{cases} 1 + \log \text{tf}_{t,d}, & \text{if } \text{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- $0 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 1.3, 10 \rightarrow 2, 1000 \rightarrow 4, \text{ etc.}$

Cosine similarity with 3 documents

How similar are
the novels:

SaS: *Sense and
Sensibility*

PaP: *Pride and
Prejudice*, and

WH: *Wuthering
Heights*?

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

Term frequencies (counts)

3 documents example contd.

Log frequency weighting

term	SaS	PaP	WH
affection	3.06	2.76	2.30
jealous	2.00	1.85	2.04
gossip	1.30	0	1.78
wuthering	0	0	2.58

After normalization

term	SaS	PaP	WH
affection	0.789	0.832	0.524
jealous	0.515	0.555	0.465
gossip	0.335	0	0.405
wuthering	0	0	0.588

$$\cos(\text{SaS}, \text{PaP}) \approx$$

$$0.789 * 0.832 + 0.515 * 0.555 + 0.335 * 0.0 + 0.0 * 0.0 \\ \approx 0.94$$

$$\cos(\text{SaS}, \text{WH}) \approx 0.79$$

$$\cos(\text{PaP}, \text{WH}) \approx 0.69$$

tf-idf weighting has many variants

Term frequency		Document frequency		Normalization	
n (natural)	$tf_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(tf_{t,d})$	t (idf)	$\log \frac{N}{df_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2 + w_2^2 + \dots + w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times tf_{t,d}}{\max_t(tf_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N - df_t}{df_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^\alpha$, $\alpha < 1$
L (log ave)	$\frac{1 + \log(tf_{t,d})}{1 + \log(\text{ave}_{t \in d}(tf_{t,d}))}$				

Columns headed 'n' are acronyms for weight schemes.

Why is the base of the log in idf immaterial?

Weighting may differ in queries vs documents

- Many search engines allow for different weightings for queries vs documents
- To denote the combination in use in an engine, we use the notation `qqq.ddd` with the acronyms from the previous table
- Example: `ltn.ltc` means:
- Query: logarithmic tf (l in leftmost column), idf (t in second column), no normalization ...
- Document logarithmic tf, no idf and cosine normalization

Is this a bad idea?



tf-idf example: ltn.Inc (log idf none . log none cosine)

Document: *car insurance auto insurance*

Query: *best car insurance*

Term	Query					Document			Prod
	tf-raw	tf-wt	df	idf	wt	tf-raw	tf-wt	n'lized	
auto	0	0	5000	2.3	0	1			
best	1	1	50000	1.3	1.3	0			
car	1	1	10000	2.0	2.0	1			
insurance	1	1	1000	3.0	3.0	2			

$$\text{Doc length} = \sqrt{1^2 + 0^2 + 1^2 + 1^2} \approx 1.92$$

tf-idf example: Itn.Inc

Document: *car insurance auto insurance*

Query: *best car insurance*

Term	Query					Document			Prod
	tf-raw	tf-wt	df	idf	wt	tf-raw	tf-wt	n'lized	
auto	0	0	5000	2.3	0	1	1	0.52	0
best	1	1	50000	1.3	1.3	0	0	0	0
car	1	1	10000	2.0	2.0	1	1	0.52	1.04
insurance	1	1	1000	3.0	3.0	2	1.3	0.677	2.04

$$\text{Doc length} = \sqrt{1^2 + 0^2 + 1^2 + 1.3^2} \approx 1.92$$

$$\text{Score} = 0 + 0 + 1.04 + 2.04 = 3.08$$

Summary – vector space ranking

- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity score for the query vector and each document vector
- Rank documents with respect to the query by score
- Return the top K (e.g., $K = 10$) to the user