Multimedia Information Retrieval and Technology

Lecture 6. Weighting II

By : Fangyu Wu

Room: SD555

Xi'an Jiaotong-Liverpool University
西交利物浦大学

- Ranked retrieval
  a. Term frequency, document freq, collection freq
  b. idf weighting
  c. tf-idf weighting
- The vector space model for scoring

RECAP:Term frequency tf

The term frequency $\mathbf{tf}_{t,d}$ of term t in document d is defined as the number of times that t occurs in d.

The log frequency weight of term t in d is

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

Xi'an Jiaotong-Liverpool University
西交利物浦大学

RECAP Document frequency

Document frequency df_t, defined to be the number of documents in the collection that contain a term *t*.

We define the idf (**inverse document frequency)** of term *t* by

$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

Exercise

$idf_1 = \log_{10} \frac{10^7}{10^4} = 3.$

$idf_2 = \log_{10} \frac{10^7}{10^5} = 2$

The query "digital cameras"

The doc： "digital cameras and video cameras"

Assume the collection size is N = 10,000,000, Treat and as a stop word.

1) What is the raw tf for each term? The log frequency weight of each term?

2) Compute idf weight for each term.

	~~Doc1~~ Query				~~Doc2~~ doc	
	tf	wf	Df	idf	tf	wf
digital	1	1	10, 000	3	1	1
video	0	0	100, 000	2	1	1
cameras	1	1	50, 000	2.3	2	1.3

Solution

SOLUTION.

word	query tf	query wf	query df	query idf	query $q_i = \text{wf-idf}$	document tf	document wf	document $d_i = \text{normalized wf}$	$q_i \cdot d_i$
digital	1	1	10,000	3	3	1	1	0.52	1.56
video	0	0	100,000	2	0	1	1	0.52	0
cameras	1	1	50,000	2.3	2.3	2	1.3	0.68	1.56

Similarity score: $1.56 + 1.56 = 3.12$.

Normalized similarity score is also correct: $3.12/\text{length(query)} = 3.12/3.78 = 0.825$

- Ranked retrieval
  a. Term frequency, document freq, collection freq
  b. idf weighting
  c. tf-idf weighting
- The vector space model for scoring

tf-idf weighting

We now combine the definitions of term frequency (tf) and inverse document frequency(idf), to produce a composite weight for each term in each document.

The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t.$$

Best known weighting scheme in information retrieval

- Note: the "-" in tf-idf is a hyphen, not a minus sign!
- Alternative names: tf.idf, tf x idf

Xi'an Jiaotong-Liverpool University
西交利物浦大学

tf-idf weighting

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t.$$

- Increases with the number of occurrences within a document **(term frequency component)**

- Increases with the rarity of document frequency of the term in the collection. **(idf component)**

Xi'an Jiaotong-Liverpool University 西交利物浦大学

Exercise

$$\overbrace{(1+\log_{10} tf_{t,d})}^{tf}\, w_{t,d} \cdot \overbrace{\log_{10}\frac{N}{df_t}}^{idf}$$

Consider the table of term frequencies for 3 documents denoted Doc1, Doc2, Doc3. Compute the tf-idf weights for the term **car** for each document, using log frequency weight and the idf values from table below.

term frequencies	Doc1	Doc2	Doc3
car	100 3×1.65 = 4.85	10 2×1.65 = 3.3	10 2×1.65 = 3.3
auto	1	0	0
insurance	0	10	100
best	10	10	10

term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

Solution

	Doc1		Doc2		Doc3	
	tf	wf	tf	wf	tf	wf
Car	100	3	10	2	10	2
auto	1	1	0	0	0	0
insurance	0	0	10	2	100	3
best	10	2	10	2	10	2

Doc1： 3*1.65=4.95

Doc2、Doc3： 2*1.65=3.3

Solution

	Doc1				Doc2		Doc3	
	tf	wf	idf	tf-idf	tf	wf	tf	wf
Car	100	3	1.65		10	2	10	
auto	1	1	2.08		0	0	0	
insurance	0	0	1.62		10	2	100	
best	10	2	1.5		10	2	10	

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Score for a document with a given query

$$\text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

There are many variants

- How “tf” is computed (with/without logs)
- Whether the terms in the query are also weighted
- …

Exercise

$$\sum \overbrace{(1+\log_{10} tf_{t,d})}^{tf_{t,d}} w_{tf_t} \cdot idf_t$$

Query: "best car insurance"

Consider the table of term frequencies for 3 documents denoted Doc1, Doc2, Doc3, Using log term frequency weight and the idf values from table below, calculate the score for Doc 1, Doc 2 and Doc 3 on this Query respectively.

	Doc1 (7.95)	Doc2 (9.54)	Doc3 (11.16)
car	100 $3\times1.65 = 4.95$	10 $2\times1.65 = 3.3$	10 $2\times1.65 = 3.3$
auto	1	0	0
insurance	0 $0\times1.62 = 0$	10 $2\times1.62 = 3.24$	100 $3\times1.62 = 4.86$
best	10 $2\times1.5 = 3$	10 $2\times1.5 = 3$	10 $2\times1.5 = 3$

term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

$Q,Doc1=3*1.65+0+2*1.5=7.95$

Exercise

How does the base of the logarithm in (6.7) affect the score calculation in (6.9)? How does the base of the logarithm affect the relative scores of two documents on a given query?

$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

$$\mathrm{Score}(q,d) = \sum_{t \in q \cap d} \mathrm{tf.idf}_{t,d}$$

SOLUTION.

6.5 For any base b>0, $\text{idf}_t = \log_b(N/df_t) = (\log_b 10) * (\log_{10}(N/df_t)) = c * (\log(N/df_t))$

where c is a constant.

$\text{tf-idf}_{t,d,b} = \text{tf}_{t,d} * idf_t = tf_{t,d} * c * (\log(N/df_t)) = c * \text{tf-idf}_{t,d}$

$\text{Score}(q,d,b) = \sum_{t \in q} \text{tf-idf}_{t,d,b} = c * \sum_{t \in q} \text{tf-idf}_{t,d}$

So changing the base changes the score by a factor $c = (\log_b 10)$

The relative scoring of documents remains unaffected by changing the base.

- Ranked retrieval
    a. Term frequency, document freq, collection freq
    b. idf weighting
    c. tf-idf weighting
- The vector space model for scoring

RECAP:
Term-document incidence matrices

	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

1 if play contains word, 0 otherwise

RECAP:
Term-document count matrices

Consider the number of occurrences of a term in a document:

Each document is a count vector : 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

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 $\in R^{|V|}$

Documents as vectors

- So we have a |V|-dimensional real valued vector space!

Terms are axis of the space

- Documents are points or vectors in this space

Very high-dimensional: tens of millions of dimensions when you apply this to a web search engine.

- These are very sparse vectors - most entries are zero.

Vector space model

Vector space model: The representation of a set of documents as vectors in a common vector space.

Vector space model is fundamental to **a host of IR operations** including scoring documents on a query, document classification and document clustering.

Vector space model

We denote by $\vec{d}$ the vector derived from document d, with one component in the vector for each dictionary term.

The set of documents in a collection then may be viewed as a set of vectors in a vector space, in which there is one axis for each term.

Queries as vectors

A far more compelling reason to represent documents as vectors, we can also view a query as a vector.

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

Queries as vectors

.

How:

rank more relevant documents higher than less relevant documents

Formalizing vector space proximity

First cut: distance between two points

(= distance between the end points of the two vectors)

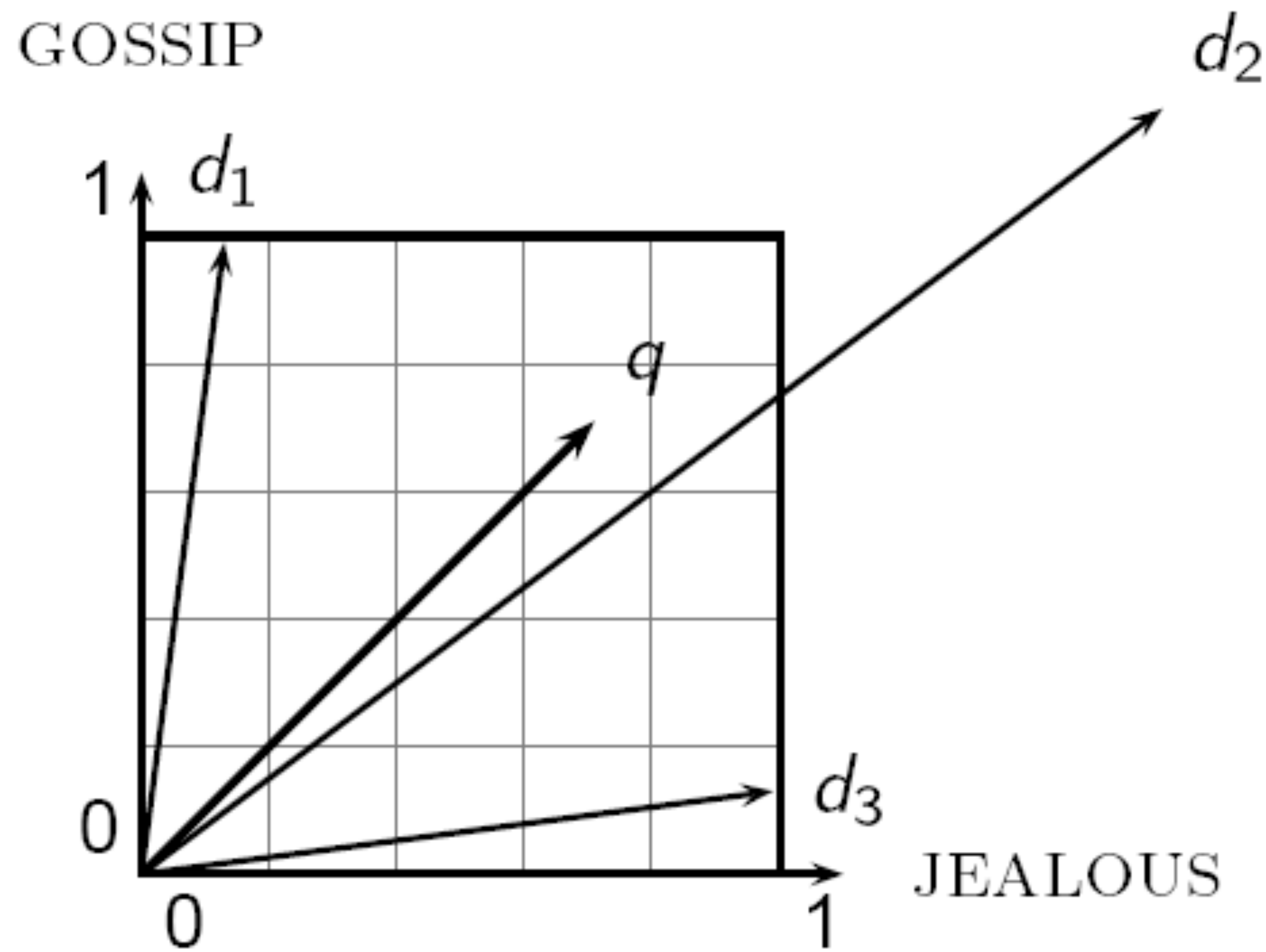
Euclidean distance?

Euclidean distance is a bad idea . . .

. . . because Euclidean distance is large for vectors of different lengths.

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

- 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.

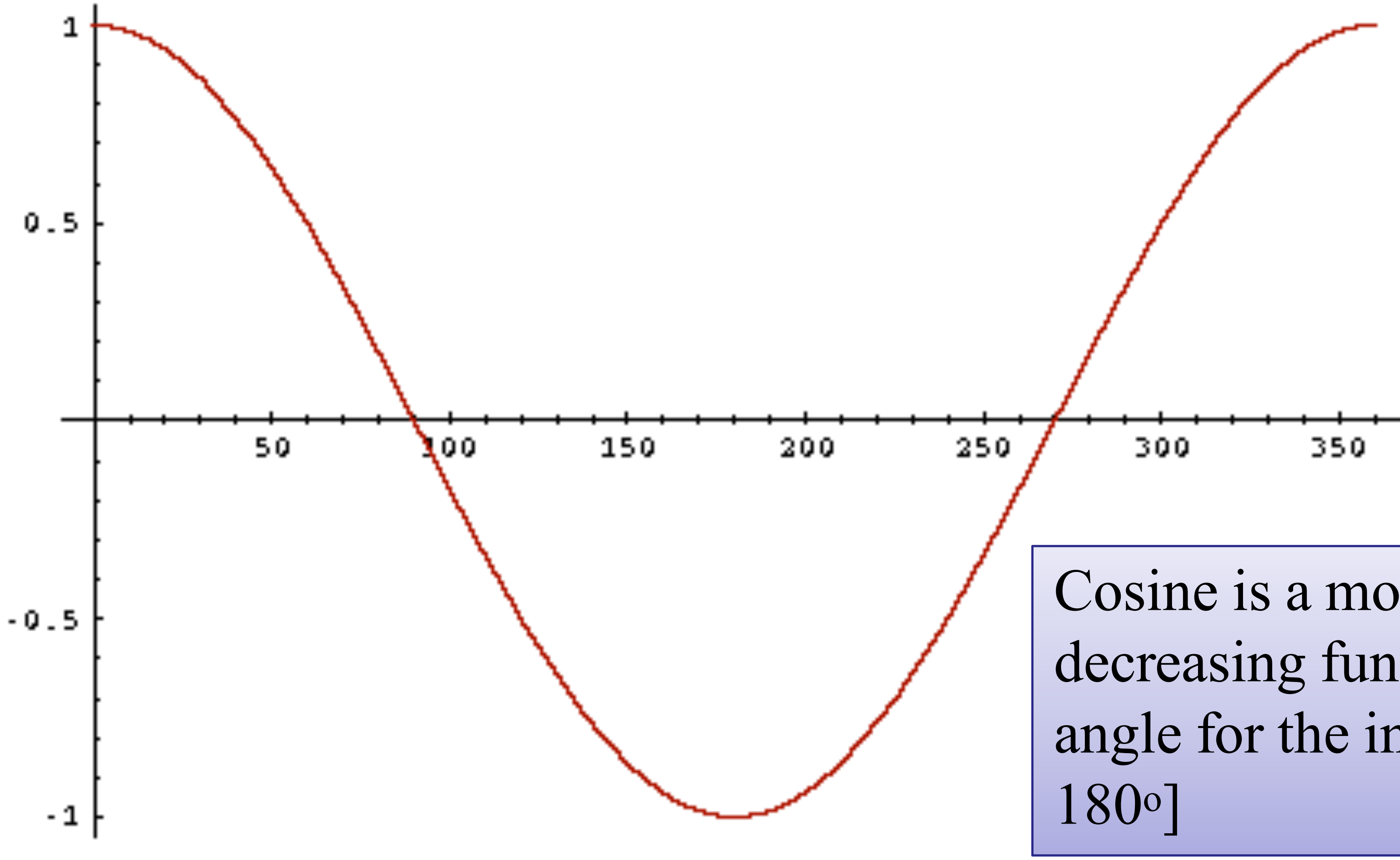
From angles to cosines

The following two notions are equivalent.

-- Rank documents in decreasing order of the angle between query and document

-- Rank documents in increasing order of cosine(query,document)

From angles to cosines



Cosine is a monotonically decreasing function of the angle for the interval [0^o, 180^o]

But how should we be computing cosines?

Cosines Similarity

- Compute the **cosine similarity** of their vector representations:

$$similarity(\vec{d}_1, \vec{d}_2) = \frac{\vec{d}_1 \cdot \vec{d}_2}{|\vec{d}_1||\vec{d}_2|}$$

- Cosine similarity = dot product of length-normalized vectors.

Cosine(query,document) Similarity

The dot product (also known as inner product) $\vec{x} \cdot \vec{y}$ of two vectors is defined as $\sum_{i=1}^{M} x_i y_i$.

Dot product

$$cos(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}||\vec{d}|}$$

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Length normalization

Let $\vec{d}$ denote the document vector for *d*, with *M* components $d_1 ... d_M$.

The Euclidean length of $\vec{d}$ is defined as

$$|\vec{d}| = \sqrt{\sum_{i=1}^{M} {d_i}^2}$$

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Length normalization

A vector can be (length-) normalized by dividing each of its components by its length:

$$\frac{\vec{d}}{|\vec{d}|}$$

Dividing a vector by its Euclidean length makes it **a 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.

Long and short documents now have comparable weights.

Unit Vectors (length-normalized)

	Doc1	Doc2	Doc3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

▶ **Figure 6.9** Table of tf values for Exercise 6.10.

	Doc1	Doc2	Doc3
car	0.88	0.09	0.58
auto	0.10	0.71	0
insurance	0	0.71	0.70
best	0.46	0	0.41

▶ **Figure 6.11** Euclidean normalized tf values for documents in Figure 6.9.

Exercise

Consider the table of term frequencies for 3 documents denoted Doc1, Doc2, Doc3. Write out the vector for $\vec{Doc_1}$ with tf-idf weight. Using log term frequency weight and the idf values from tables below. Then calculate the unit vector for $\vec{Doc_1}$.

	Doc1	Doc2	Doc3
car	100	10	10
auto	1	0	0
insurance	0	10	100
best	10	10	10

term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

(4.95, 2.08, 0, 3)

Exercise

Consider the table of term frequencies for 3 documents denoted Doc1, Doc2, Doc3. The idf values for each term is listed in Table 2. Compute the Euclidean normalized document vectors (tf-idf weights) for Doc2, (each vector has four components, one for each of the four terms).

	Doc1	Doc2	Doc3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

Solution

Doc1=(0.897,0.125,0,0.423),

Doc2=(0.076,0.786,0.613,0)

Doc3=(0.595,0,0.706,0.383)

	D1	D2	D3
car	0.897	0.076	0.595
Auto	0.125	0.786	0
insurance	0	0.613	0.706
best	0.423	0	0.383

Cosine(query,document) Similarity

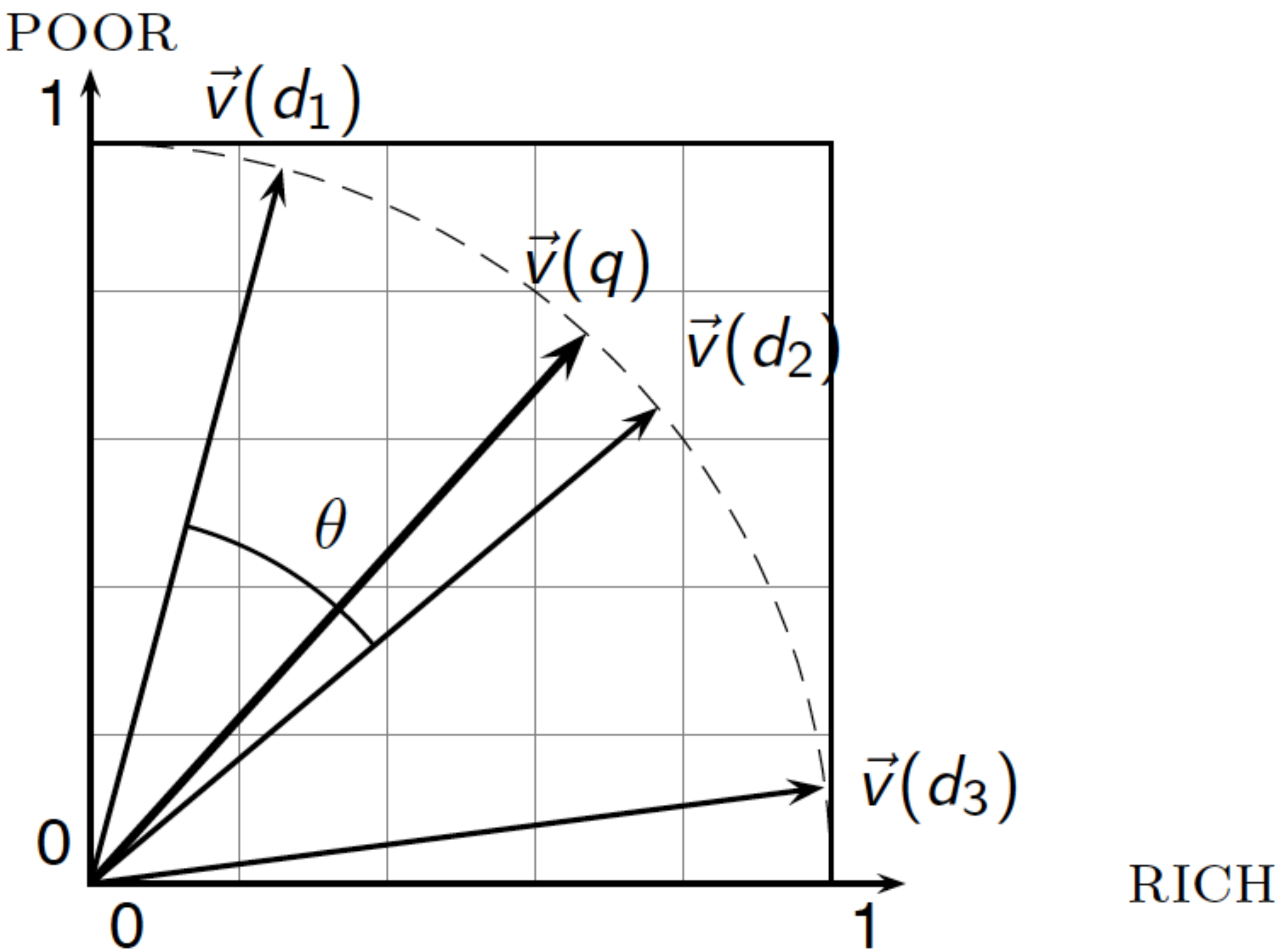
Unit vectors

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

q_i is the tf-idf weight of term i in the query;
d_i is the tf-idf weight of term i in the document.

Cosine similarity of $\vec{q}$ and $\vec{d}$ … or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

Cosine similarity illustrated



Example: Cosine similarity amongst 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
wuthering	0	0	38

Term frequencies (counts)

Note: To simplify this example, we don't do idf weighting.

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 length 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(SaS,PaP) ≈

$0.789 \times 0.832 + 0.515 \times 0.555 + 0.335 \times 0.0 + 0.0 \times 0.0$

≈ 0.94

cos(SaS,WH) ≈ 0.79

cos(PaP,WH) ≈ 0.69

Exercise

Compute the vector space similarity between the query "digital cameras" and the document "digital cameras and video cameras" by filling out the empty columns in Table 6.1. Assume the collection size is N = 10,000,000, logarithmic term weighting (wf columns) for query and document, idf weighting for the query only and cosine normalization for the document only. Treat and as a stop word. Enter term counts in the tf columns. What is the final similarity score?

	query					document			
word	tf	wf	df	idf	q_i = wf-idf	tf	wf	d_i = normalized wf	$q_i \cdot d_i$
digital			10,000						
video			100,000						
cameras			50,000						

▶ **Table 4** Cosine computation for Exercise 6.19.

Exercise

The query "digital cameras"

The doc： "digital cameras and video cameras"

Assume the collection size is N = 10,000,000, Treat and as a stop word.

of each

	Doc1					Doc2	
	tf	wf	Df	idf	tf-idf	tf	wf
digital			10, 000			0	0
video			100, 000			10	2
cameras			50, 000			10	2

	query					document			
word	tf	wf	df	idf	q_i = wf-idf	tf	wf	d_i = normalized wf	$q_i \cdot d_i$
digital			10,000						
video			100,000						
cameras			50,000						

▶ **Table 4** Cosine computation for Exercise 6.19.

Solution

SOLUTION.

	query					document			
word	tf	wf	df	idf	q_i = wf-idf	tf	wf	d_i = normalized wf	$q_i \cdot d_i$
digital	1	1	10,000	3	3	1	1	0.52	1.56
video	0	0	100,000	2	0	1	1	0.52	0
cameras	1	1	50,000	2.3	2.3	2	1.3	0.68	1.56

Similarity score: $1.56 + 1.56 = 3.12$.

Normalized similarity score is also correct: 3.12/length(query) = 3.12/3.78 = 0.825