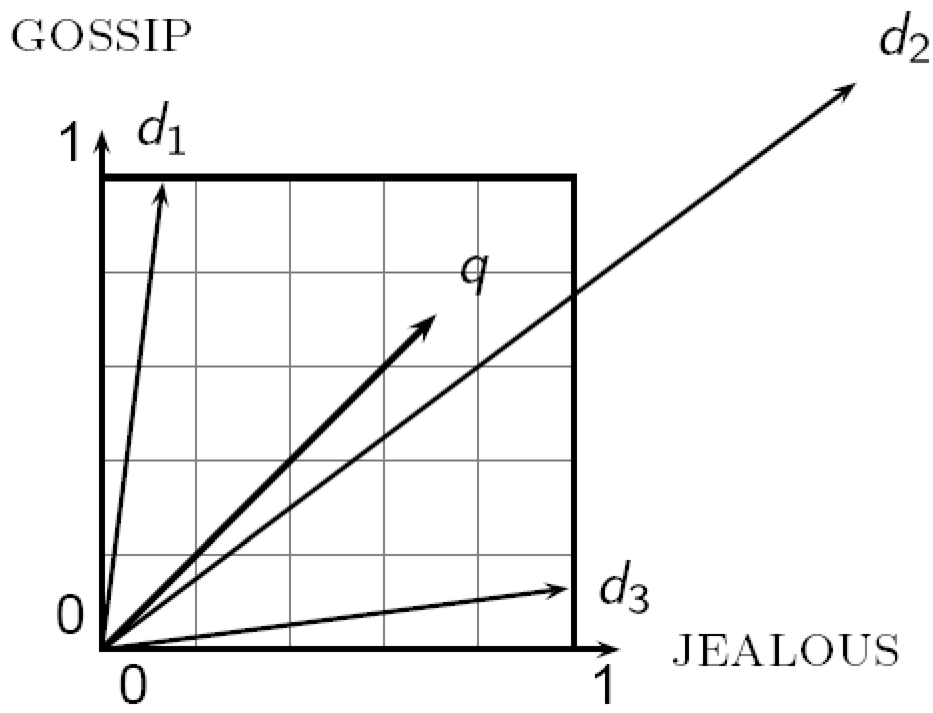
Cosine Similarity

How to define a good similarity measure?

- Euclidean distance
  - $dist(q,d) = \sqrt{\sum_{t \in V}[tf(t,q)idf(t) - tf(t,d)idf(t)]^2}$
  - Longer documents will be penalized by the extra words
  - We care more about how these two vectors are overlapped

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.



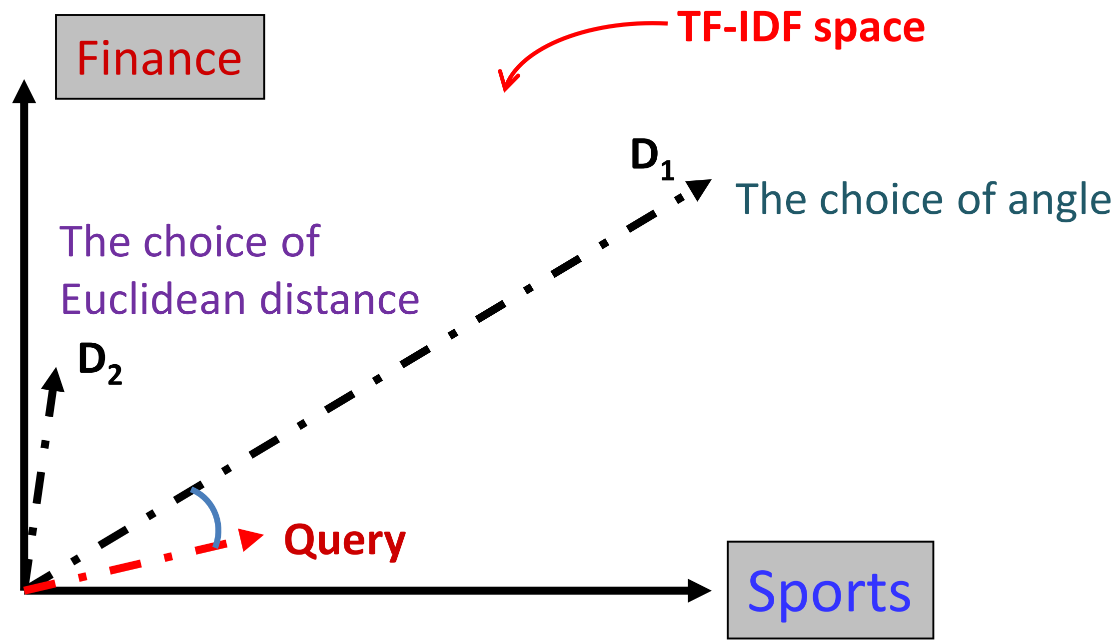
Use angle instead of distance

- 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 distance to angle

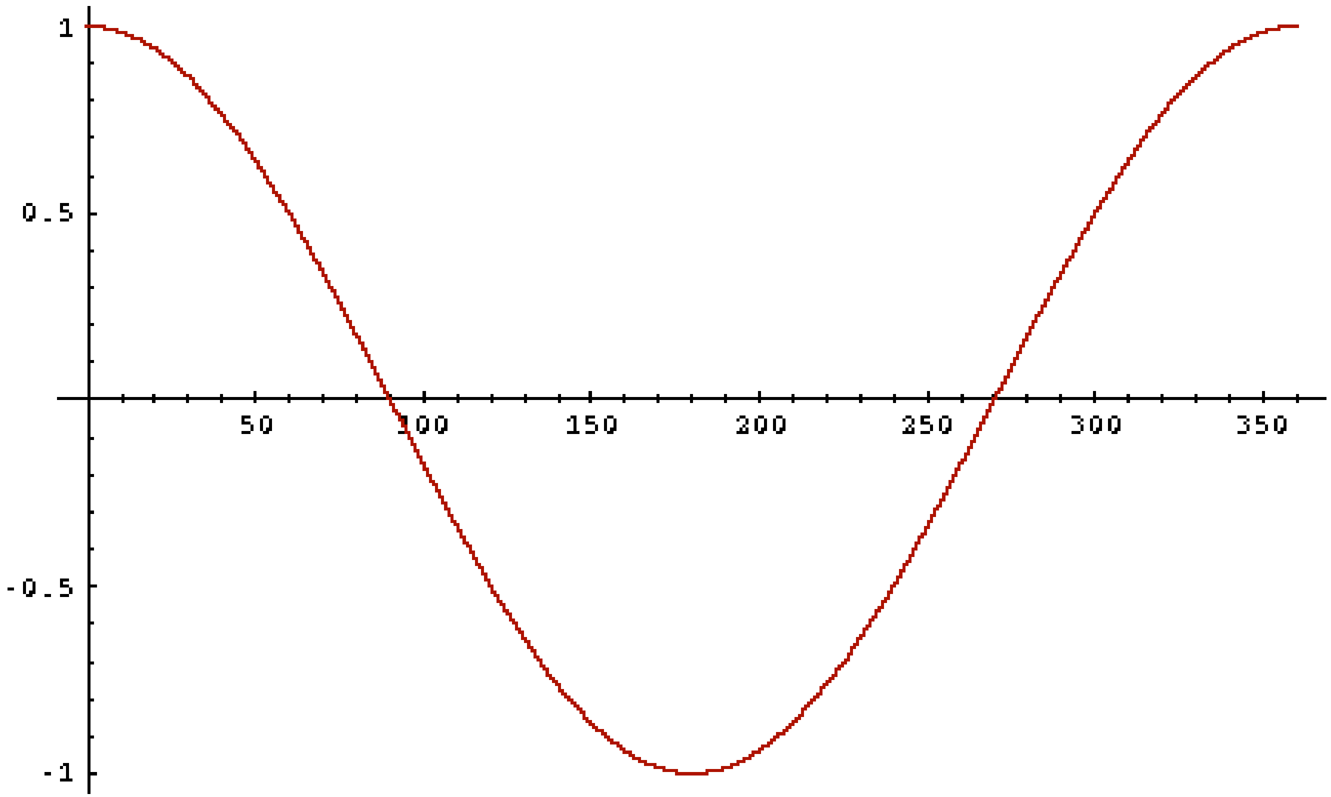
- Angle: how vectors are overlapped
  - Cosine similarity – projection of one vector onto another



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)
- Cosine is a monotonically decreasing function for the interval $[0^{o}, 180^{o}]$

From angles to cosines



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

From angles to cosines

- Cosine of 0 degrees = 1
- Cosine of 90 degree = 0
- Cosine of 180 degree = -1

Cosine of angles varies between -1 to 1.

Question

Is it a problem for tf-idf that the cosine of the angle between vectors is negative between 90° and 180°?

A. Yes, because we don't know that to do with negative scores.
B. No, because a vector d_1 pointing in the opposite direction of a vector d_2 would just have the same content.
C. No, because we can always normalize by $(1+\cos(d_1, d_2)) / 2$ which is always between 0 and 1.
D. No, because the cosine of the angles between tf-idf vectors will always be non-negative.

Answer

- Correct Option is D
- Explanation
  - The cosine of the angles between tf-idf vectors will always be non-negative (i.e. are never more than 90° apart). This is because all the elements of any tf-idf vector are non-negative (i.e. they are in $\mathbf{R}^{n}_{+}$, where n is the number of unique terms), and no two vectors in the non-negative orthant are more than 90° apart.

Cosine similarity

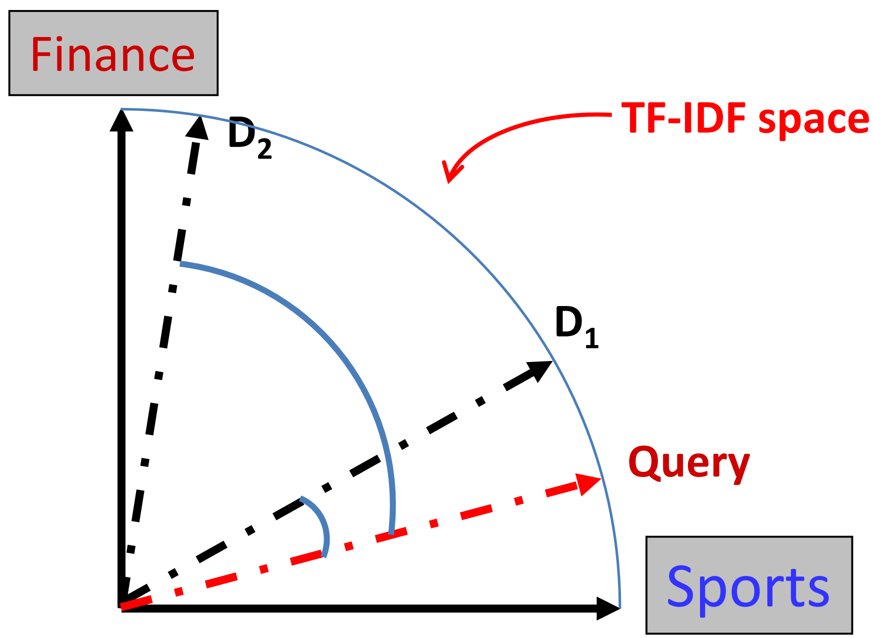
- Angle between two vectors

  TF-IDF vector

  - $cosine(V_q, V_d) = \frac{V_q \times V_d}{|V_q|_2 \times |V_d|_2} = \boxed{\frac{V_q}{|V_q|_2}} \times \frac{V_d}{|V_d|_2}$

  Unit vector

  - Document length normalized



cosine(query,document)

Can we ignore query length in this formula?

Dot product

Unit vectors

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

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

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

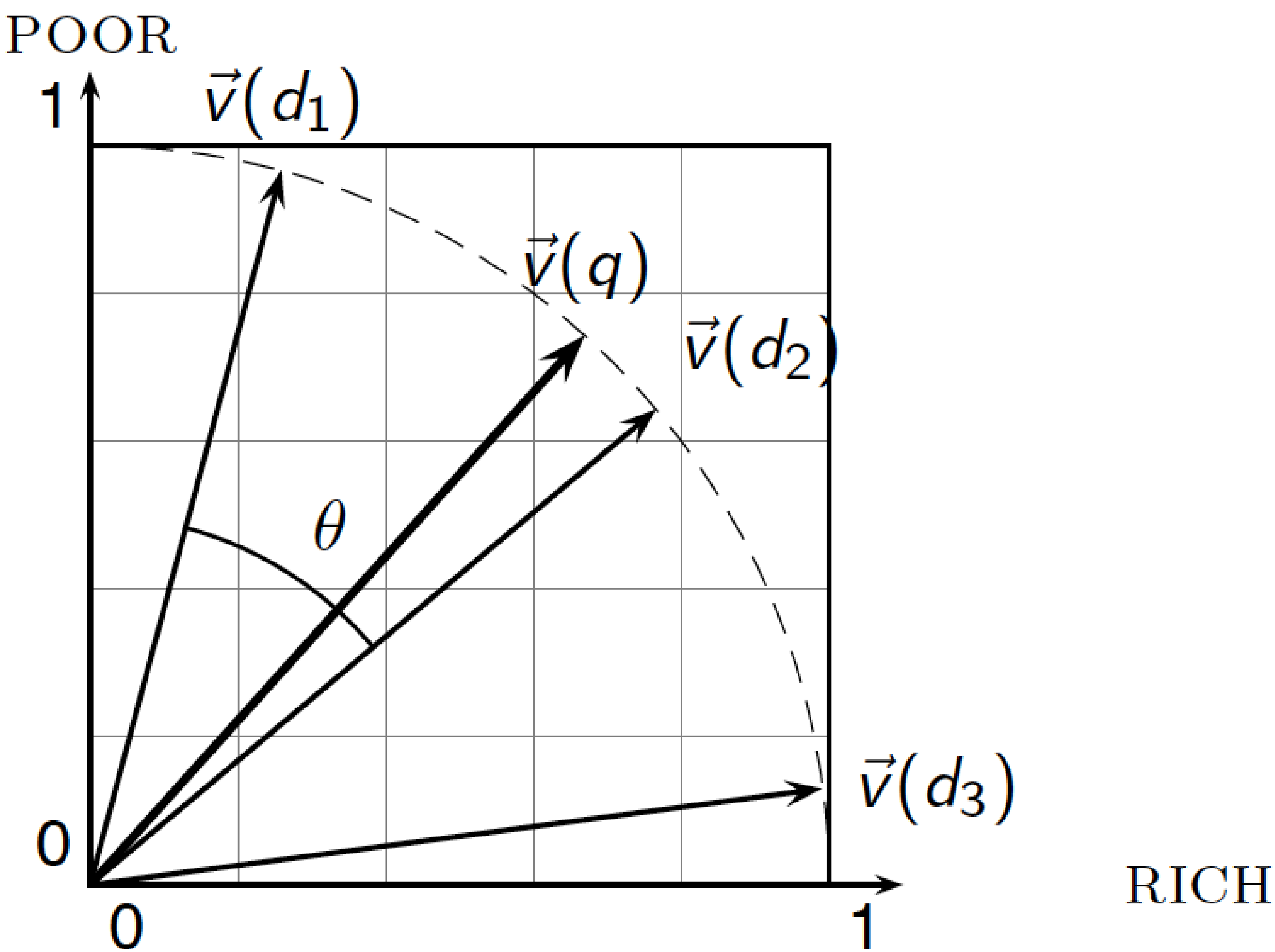
Cosine for length-normalized vectors

- For length-normalized vectors, cosine similarity is simply the dot product (or scalar product):

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

for q, d length-normalized.

Cosine similarity illustrated



Cosine similarity amongst 3 documents

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6
wuthering	0	0	38

Term frequencies (counts)

How similar are the novels

SaS: *Sense and Sensibility*

PaP: *Pride and Prejudice*

WH: *Wuthering Heights*?

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

Why do we have cos(SaS,PaP) > cos(SaS,WH)?

tf-idf example

Document: *car insurance auto insurance*
Query: *best car insurance*

Term	Query		Document
	tf-raw	df	tf-raw
Auto	0	5000	1
best	1	50000	0
car	1	10000	1
insurance	1	1000	2

Total documents = 1 million

tf-idf example

Document: *car insurance auto insurance*
Query: *best car insurance*

Term	Query					Document		Product
	tf-raw	tf-wt	df	idf	tf*idf	tf-raw	tf-wt	
auto	0	0	5000	2.3	0	1	1	0
best	1	1	50000	1.3	1.3	0	0	0
car	1	1	10000	2.0	2.0	1	1	2
insurance	1	1	1000	3.0	3.0	2	1.3	3.9

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

Note: We have not used query length here so the similarity score is not between 0 and 1

Score = (0+0+2+3.9) / (1.92) = 5.9/1.92 = 3.07

IDF can be multiplied to only one vector

- $(Tf_{best,\ q} * \mathbf{IDF_{best}}) * Tf_{best,\ d} + (Tf_{car,\ q} * \mathbf{IDF_{car}}) * Tf_{car,d}$

- $Tf_{best,\ q} * \mathbf{IDF_{best}} * Tf_{best,\ d} * \mathbf{IDF_{best}} + Tf_{car,\ q} * \mathbf{IDF_{car}} * Tf_{car,\ d} * \mathbf{IDF_{car}}$

$= Tf_{best,\ q} * Tf_{best,\ d} * (\mathbf{IDF_{best}})^2 + Tf_{car,\ q} * Tf_{car,\ d} * (\mathbf{IDF_{car}})^2$

IDF of a word is same in query and document so multiplying it with any one of the vectors is enough

Question

- What is *N*, the number of documents, for this example?
  (Hint: Note that the document frequency for "car" is 10,000 and its inverse document frequency is 2.0.)

A. 1 million
B. 10,000
C. 500,000
D. No idea...

Answer

- Option A is correct
- Explanation:
  - N=1 million because we see that, from the formula for idf, $\log_{10} N/10{,}000=2.0$ so we have $N=10{,}000\times 10^2=10{,}000\times 100=10^6$, which is 1 million.

Fast computation of cosine in retrieval

- $cosine\left(V_q, V_d\right) = V_q \times \frac{V_d}{|V_d|_2}$
  - $\left|V_q\right|_2$ would be the same for all candidate docs
  - Normalization of V_d can be done in index time
  - Only count $t \in q \cap d$
  - Score accumulator for each query term when intersecting postings from inverted index

Computing cosine scores

$\textsc{CosineScore}(q)$

$\mathit{float}\ \mathit{Scores}[N] = 0$
$\mathit{float}\ \mathit{Length}[N]$
for each query term t
do calculate $\mathrm{w}_{t,q}$ and fetch postings list for t
    for each pair$(d, \mathrm{tf}_{t,d})$ in postings list
    do $\mathit{Scores}[d] += \mathrm{w}_{t,d} \times \mathrm{w}_{t,q}$
Read the array *Length*
for each d
do $\mathit{Scores}[d] = \mathit{Scores}[d]/\mathit{Length}[d]$
return Top K components of $\mathit{Scores}[]$

Fast computation of cosine in retrieval

- Maintain a score accumulator for each doc when scanning the postings

Query = "info security"

$S(d,q)=g(t_1)+\ldots+g(t_n)$ [sum of TF of matched terms]

Info: (d1, 3), (d2, 4), (d3, 1), (d4, 5)

Security: (d2, 3), (d4, 1), (d5, 3)

Can be easily applied to TF-IDF weighting!

	Accumulators:	**d1**	**d2**	**d3**	**d4**	**d5**
info	(d1,3) =>	**3**	0	0	0	0
	(d2,4) =>	3	4	0	0	0
	(d3,1) =>	3	4	**1**	0	0
	(d4,5) =>	3	4	1	5	0
security	(d2,3) =>	3	**7**	1	5	0
	(d4,1) =>	3	7	1	**6**	0
	(d5,3) =>	3	7	1	6	**3**

Advantages of VS Model

- Empirically effective! (Top TREC performance)
- Intuitive
- Easy to implement
- Well-studied/Mostly evaluated
- The Smart system
  - Developed at Cornell: 1960-1999
  - Still widely used
- Warning: Many variants of TF-IDF!

Disadvantages of VS Model

- Assume term independence
- Assume query and document to be the same
- Lack of “predictive adequacy”
  - Arbitrary term weighting
  - Arbitrary similarity measure

What you should know

- Document ranking v.s. selection
- Basic idea of vector space model
- Two important heuristics in VS model
  - TF
  - IDF
- Similarity measure for VS model