Monsoon 2020

9 - Vector Space Model (contd)

I n f o r m a t i o n R e t r i e v a l

by

Dr. Rajendra Prasath

Indian Institute of Information Technology, Sri City, Chittoor

Sri City – 517 646, Andhra Pradesh, India

Lecture – 09

Topics Covered So Far

- Bi-Word Index
- Wild Card Queries
- Permuterm Index
- K-gram Index (k = 2 □ Bigram Index)
- Spell Correction
- Term Weighting
- Vector Space Models

Now: Vector Space Model

Recap: Overview

- Why Ranked Retrieval?
- Term Frequency
- Term Weighting
- TF-IDF Weighting
- The Vector Space Model

Recap: Ranked Retrieval

- Our Queries have all been Boolean
  - Documents either match or don’t
- Good for expert users with precise understanding of their needs and of the collection.
- Also good for applications: Applications can easily consume 1000s of results.
- Not good for the majority of users
- Most users don’t want to wade through 1000s of results.
- This is particularly true of web search.

Scoring as the basis of ranked retrieval

- Rank documents such that more relevant documents higher than less relevant document

- How do we do follow?
  - Accomplish a ranking of the documents in the collection with respect to a query?

- Assign a score to each query-document pair, say in [0, 1]
- This score measures how well document and query "**match**"

Query – Docs matching scores

- How do we compute the score of a query-document pair?

- Let’s start with a one-term query.
- If the query term does not occur in the document: score should be 0.
- The more frequent the query term in the document, the higher the score
- We will look at a number of alternatives for doing this.

Term Frequency (tf)

- The term frequency $tf_{t,d}$ of term t in document d:
  - The number of times that t occurs in d

- Use tf to compute query-doc. match scores
- Raw term frequency is not what we want

- A document with tf = 10 occurrences of the term is more relevant than a document
  with tf = 1 occurrence of the term
- But not 10 times more relevant
- Relevance does not increase proportionally with term frequency

Exercise

- Compute Jaccard matching score & TF matching score for the following query-document pairs

q: [information on cars]

d: “all you’ve ever wanted to know about cars”

- q: [information on cars]

  d: “information on trucks, information on planes, information on trains”

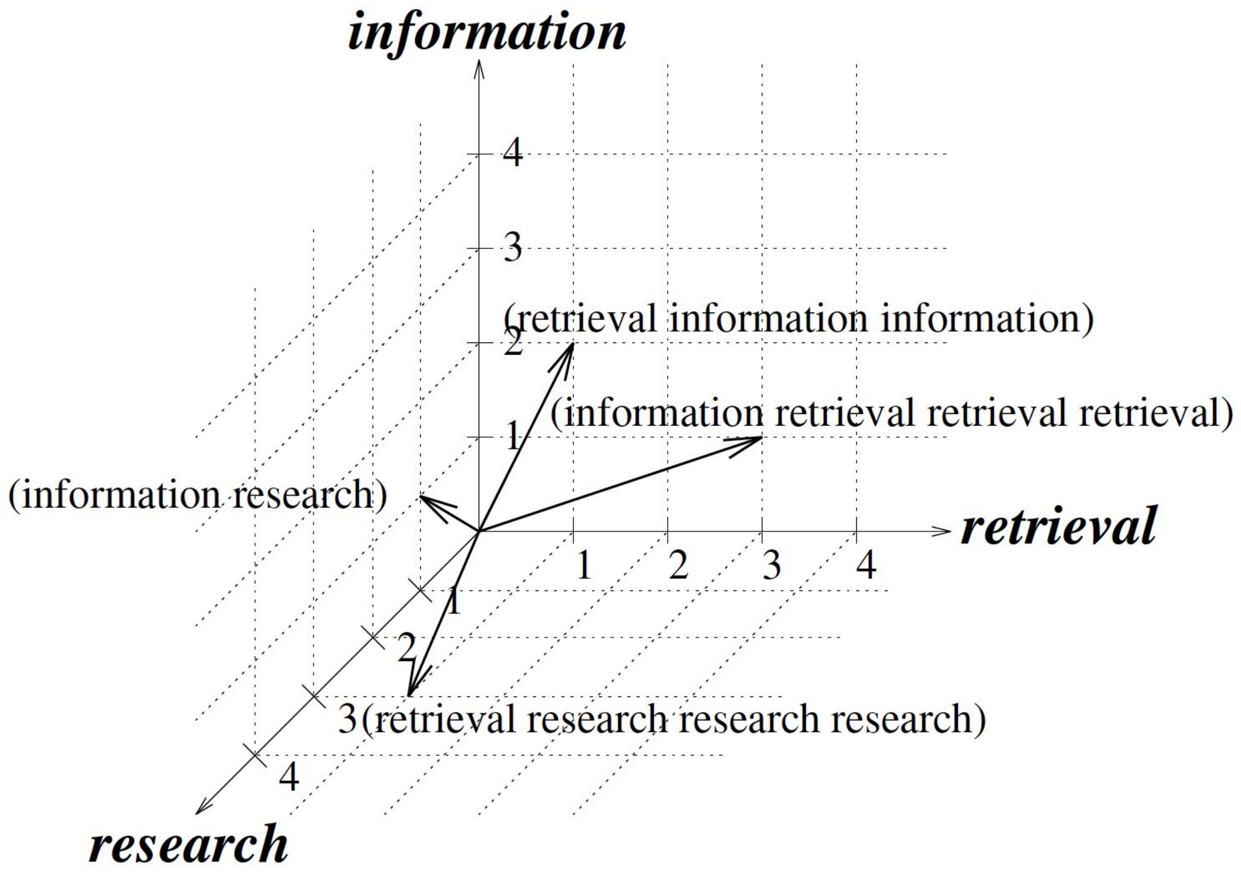
- q: [red cars and red trucks]

  d: “cops stop red cars more often”

Vector Space Model

Consider Three Words Model

"information retrieval research"


information
retrieval
research
4
3
2
1
1
2
3
4
1
2
3
4
(retrieval information information)
(information retrieval retrieval retrieval)
(information research)
(retrieval research research research)

Term Frequency Factor

- What is Term Frequency Factor?
  - The function of the term frequency used to compute a term’s importance

  - Some commonly used factors are:
    - Raw TF factor
    - Logarithmic TF factor
    - Binary TF factor
    - Augmented TF factor
    - Okapi’s TF factor

10

Measure of Closeness of Vectors

- **Measure the closeness between two vectors**
- Two texts are semantically related if they share some vocabulary
  - More Vocabulary they share, the stronger is the relationship
- This implies that the measure of closeness increases with the number of words matches between two texts
- If matching terms are important then vectors should be considered closer to each other

Modern Vector Space Models

- The length of the sub-vector in dimension - i is used to represent the importance or the weigh of word – i in a text
- Words that are absent in a text get a weight – 0 (zero)
- Apply **Vector Inner Product** measure between two vectors:
- This vector inner product increases:
  - # words match between two texts
  - Importance of the matching terms

Finding closeness between texts

- Given two texts in T dimensional vector space:

$$\vec{P} = (p_1, p_2, \ldots, p_T) \text{ and } \vec{Q} = (q_1, q_2, \ldots, q_T)$$

- The inner product between these two vectors:

$$\vec{P} \cdot \vec{Q} = \sum_{i=1}^{T} \sum_{j=1}^{T} p_i \times \vec{u_i} \cdot q_j \times \vec{u_j}$$

- Vectors u_i and u_j are unit vectors in dimensions *i and j* (Here $u_i \cdot u_j = 0$, if $i \neq j$ - orthogonal)
- Vector Similarity: Closeness between two texts

$$similarity(\vec{P}, \vec{Q}) = \sum_{i=1}^{T} p_i \times q_i$$

Recap: Exercise – Ex08

- Consider a collection of n documents
- Let n be sufficiently large (at least 100 docs)
- Find two lists:
  - The most frequency words and
  - The least frequent words
- Form k (=10) queries each with exactly 3-words taken from above lists (at least one from each)
- Compute Similarity between each query and and documents

Inverse Document Frequency

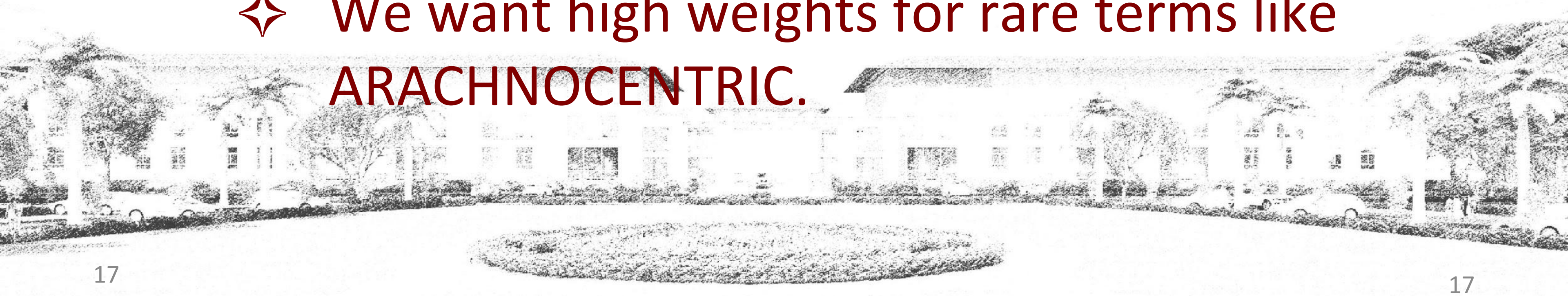
- Using the TF factors to estimate the term importance does not suffice
- Why?
  - Consider common words that occur with very high frequency across numerous articles.
  - Such words are not very informative
  - A match between a query and a document on words like “put” or “the’ does not mean much in terms of the semantic relationship between the query and the document.

Frequency in document vs. Frequency in collection

- In addition, to term frequency (the frequency of the term in the document) . . .
- . . .we also want to use the frequency of the term in the collection for weighting and ranking.

Desired weight for rare terms

- Rare terms are more informative than frequent terms.
- 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.
  - We want high weights for rare terms like ARACHNOCENTRIC.



Desired weight for frequent terms

- Frequent terms are less informative than rare terms.
- Consider a term in the query that is frequent in the collection (e.g., GOOD, INCREASE, LINE).
- A document containing this term is more likely to be relevant than a document that doesn't . . .
- . . . but words like GOOD, INCREASE and LINE are not sure indicators of relevance.
- For frequent terms like GOOD, INCREASE and LINE, we want positive weights . . .
- . . . but lower weights than for rare terms.

Document Frequency

- We want high weights for rare terms like ARACHNOCENTRIC.
- We want low (positive) weights for frequent words like GOOD, INCREASE and LINE.
- We will use document frequency to factor this into computing the matching score.
- The document frequency is the number of documents in the collection that the term occurs in.

19

IDF weight

- ✧ dft is the document frequency, the number of documents that t occurs in.
- ✧ df_t is an inverse measure of the informativeness of term t.

$$\mathrm{idf}_t = \log_{10} \frac{N}{\mathrm{df}_t}$$

- ✧ We define the idf weight of term t as follows: (N is the number of documents in the collection.)
- ✧ idft is a measure of the informativeness of the term.
- ✧ [log N/df_t] instead of [N/df_t] to “dampen” the effect of IDF
- ✧ Note that we use the log transformation for both term frequency and document frequency.

Examples for IDF

- Compute idf_t using the formula:

$$\text{idf}_t = \log_{10} \frac{1{,}000{,}000}{\text{df}_t}$$

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Effect of IDF on ranking

- idf affects the ranking of documents for queries with at least two terms.
- For example, in the query “arachnocentric line”, idf weighting increases the relative weight of ARACHNOCENTRIC and decreases the relative weight of LINE.
- idf has little effect on ranking for one-term queries.

Collection Frequency vs. Document Frequency

word	collection frequency	document frequency
INSURANCE	10440	3997
TRY	10422	8760

- Collection frequency of t: number of tokens of t in the collection
  - Document frequency of t: number of documents t occurs in
- Why these numbers?
- Which word is a better search term (and should get a higher weight)?
- This example suggests that df (and idf) is better for weighting than cf (and “icf”).

23

TF-IDF weighting

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

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

  - tf-weight
  - idf-weight
  - 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

Summary: TF-IDF

- Assign a tf-idf weight for each term t in each document d:

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

  - The tf-idf weight . . .
- . . . increases with the number of occurrences within a document. (term frequency)
- . . . increases with the rarity of the term in the collection. (inverse document frequency)

Exercise: Term, collection and document frequency

Quantity	Symbol	Definition
term frequency	$tf_{t,d}$	number of occurrences of t in d
document frequency	df_t	number of documents in the collection that t occurs in
collection frequency	cf_t	total number of occurrences of t in the collection

- Relationship between df and cf?
- Relationship between tf and cf?
- Relationship between tf and df?

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	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 as a binary vector ∈ $\{0, 1\}^{|V|}$

Count matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
. . .						

- Each document is now represented as a count vector ∈ N|V|.

Binary → count → weight matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	5.25	3.18	0.0	0.0	0.0	0.35
BRUTUS	1.21	6.10	0.0	1.0	0.0	0.0
CAESAR	8.59	2.54	0.0	1.51	0.25	0.0
CALPURNIA	0.0	1.54	0.0	0.0	0.0	0.0
CLEOPATRA	2.85	0.0	0.0	0.0	0.0	0.0
MERCY	1.51	0.0	1.90	0.12	5.25	0.88
WORSER	1.37	0.0	0.11	4.15	0.25	1.95
. . .						

- Each document is now represented as a real-valued vector of tf idf weights ∈ R|V|.

Documents as vectors

- Each document is now represented as a real-valued vector of tf-idf weights $\in R|V|$.
- So we have a |V|-dimensional real-valued vector space.
- Terms are axes of the space.
- Documents are points or vectors in this space.
- Very high-dimensional: tens of millions of dimensions when you apply this to web search engines
- Each vector is very sparse - most entries are zero.

Queries as vectors

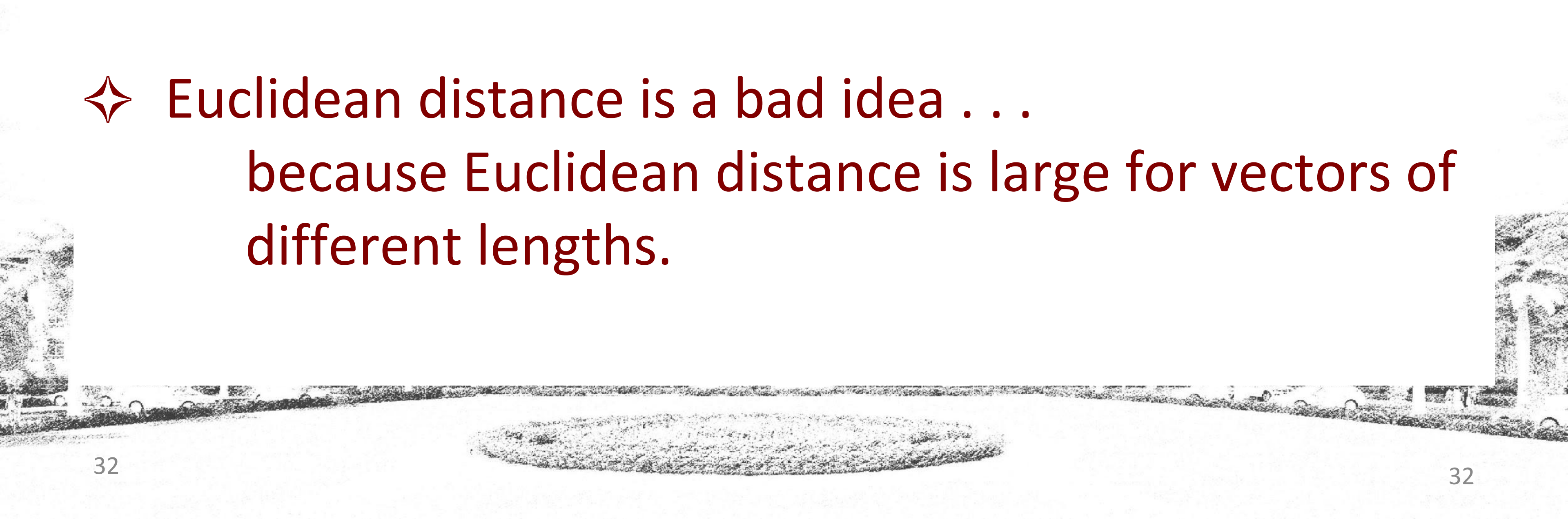
- Key idea 1: do the same for queries: represent them as vectors in the high-dimensional space
- Key idea 2: Rank documents according to their proximity to the query
  - proximity = similarity
  - proximity ≈ negative distance
- Recall: We're doing this because we want to get away from the you're-either-in-or-out, feast-or-famine Boolean model.
- Instead: rank relevant documents higher than non-relevant documents

How do we formalize vector space similarity?

- First cut: (negative) distance between two points
  (= distance between the endpoints of the two vectors)

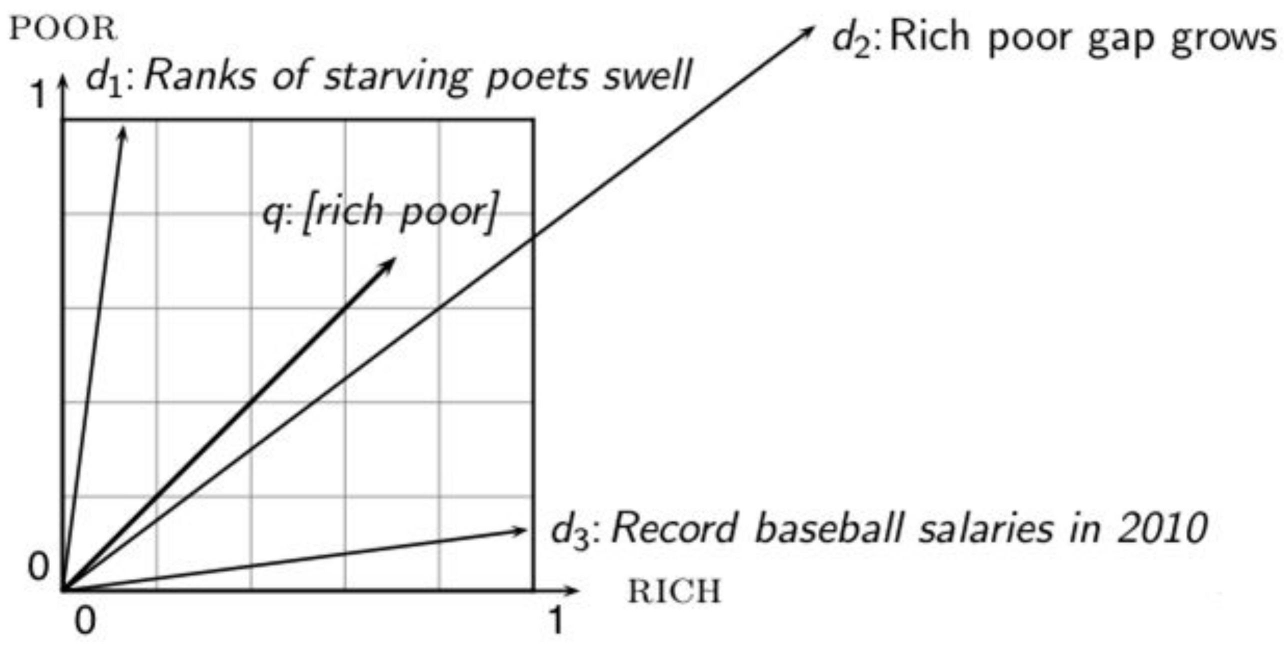
- Euclidean distance?

- Euclidean distance is a bad idea . . .
  because Euclidean distance is large for vectors of different lengths.



32

Why distance is a bad idea


POOR
1
0
0
1
RICH
d_1: Ranks of starving poets swell
d_2: Rich poor gap grows
q: [rich poor]
d_3: Record baseball salaries in 2010

- The Euclidean distance of $\vec{q}$ and $\vec{d_2}$ is large although the distribution of terms in the query q and the distribution of terms in the d2 are very similar.
- Questions about basic vector space setup?

Use angle instead of distance

- Rank documents according to angle with query
- Thought experiment:

  take a document d and append it to itself

  Call this document d′

  d′ is twice as long as d

- “Semantically” d and d′ have the same content.
- The angle between the two documents is 0, corresponding to maximal similarity . . .
- . . . even though the Euclidean distance between the two documents can be quite large.

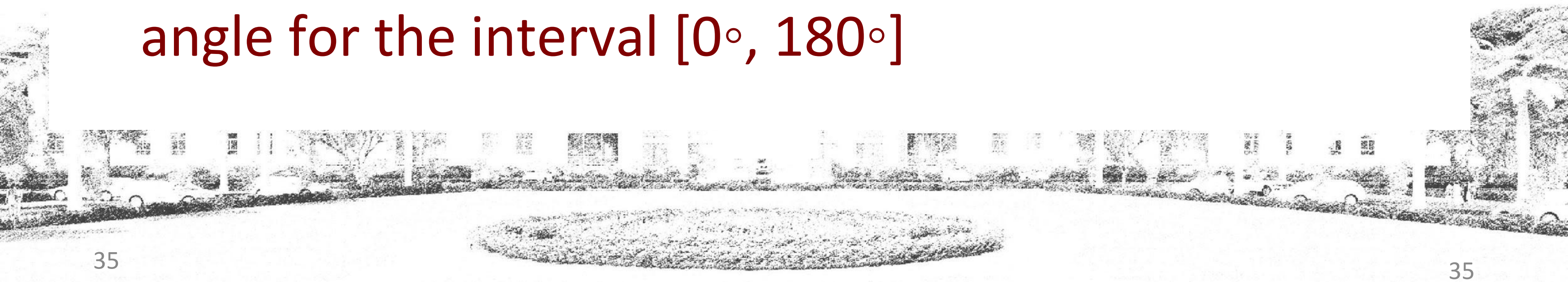
From angles to cosines

The following two notions are equivalent:

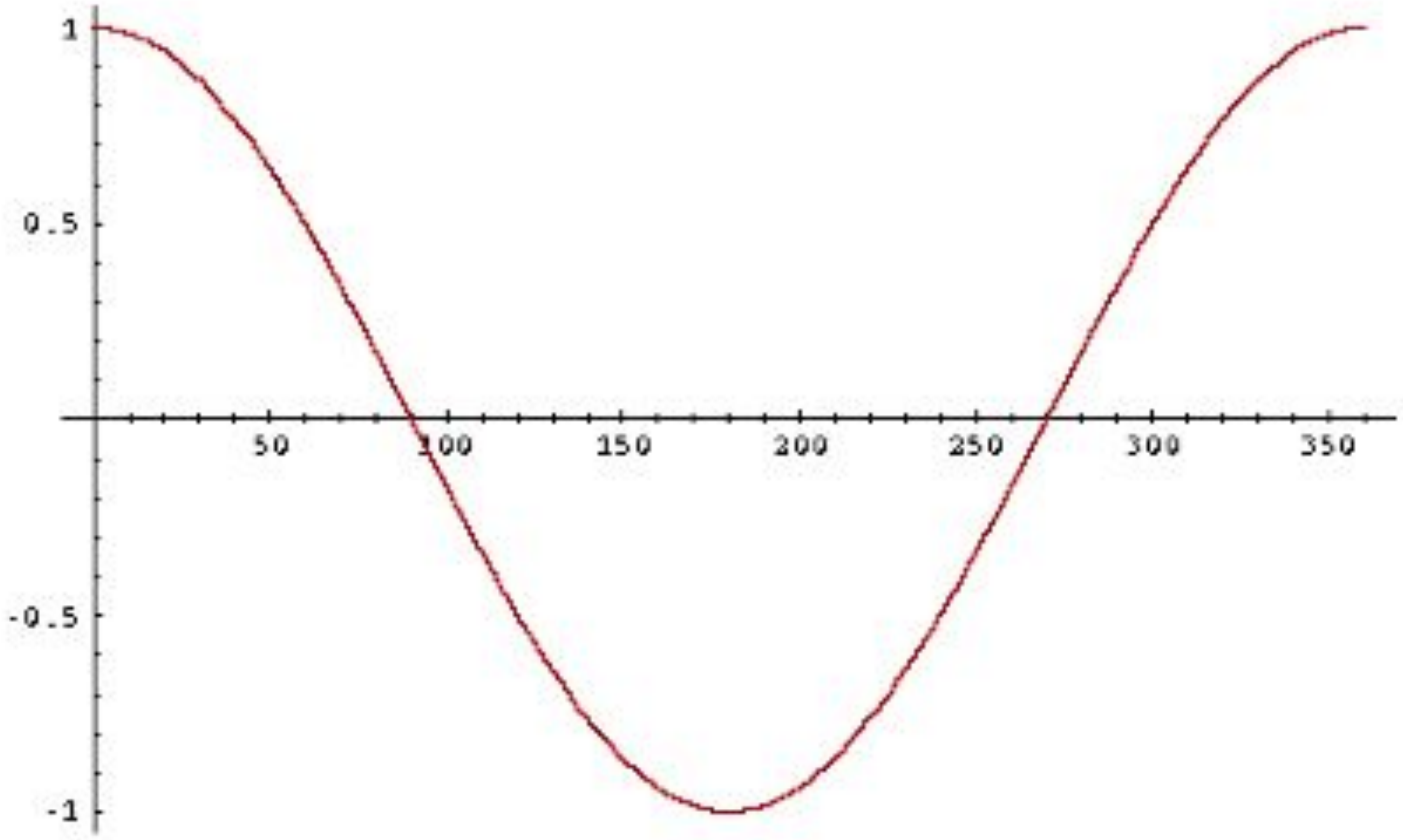
✧ Rank documents according to the angle between query and document in decreasing order

✧ Rank documents according to cosine(query,document) in increasing order

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



Cosine



Length normalization

- How do we compute the cosine?
- A vector can be (length-) normalized by dividing each of its components by its length – here we use the L2 norm:

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

- This maps vectors onto the unit sphere . . .
- . . . since after normalization: $||x||_2 = \sqrt{\sum_i x_i^2} = 1.0$
- As a result, longer docs and shorter docs have weights of the same order of magnitude.
- Effect on the two documents d and d′ (d appended to itself) from earlier slide: they have identical vectors after length-normalization.

Cosine similarity between query and document

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

- 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.
- $|\vec{q}|$ and $|\vec{d}|$ are the lengths of $\vec{q}$ and $\vec{d}$.
- This is the cosine similarity of $\vec{q}$ and $\vec{d}$. . . . . . . or, equivalently,
- the cosine of the angle between $\vec{q}$ and $\vec{d}$.

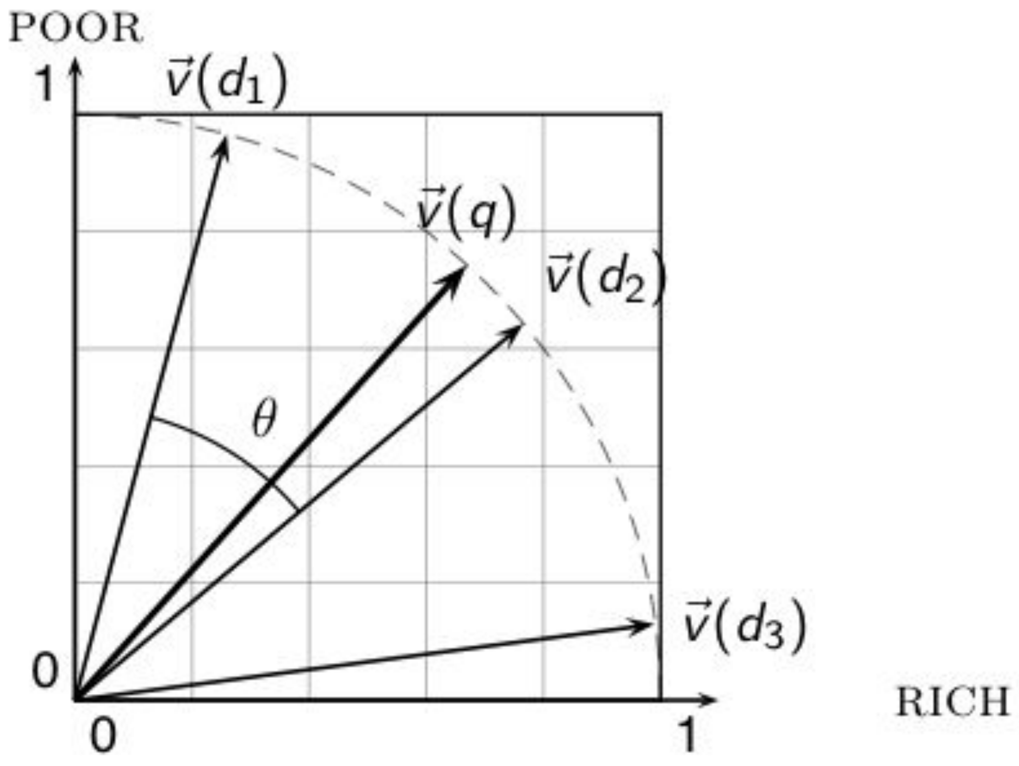
Cosine for normalized vectors

- For normalized vectors, the cosine is equivalent to the dot product or scalar product.

$$\cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} = \sum_i q_i \cdot d_i$$

- (if $\vec{q}$ and $\vec{d}$ are length-normalized).

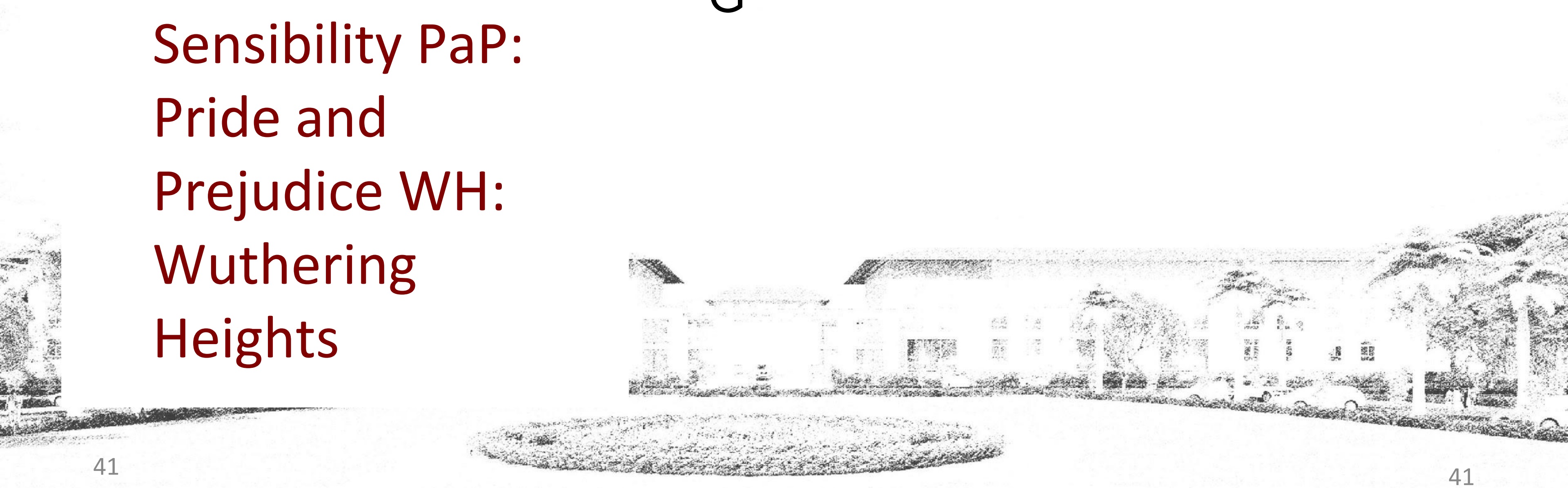
Cosine similarity illustrated



Cosine: Example

- term frequencies (counts)
- How similar are these novels? SaS: Sense and Sensibility PaP: Pride and Prejudice WH: Wuthering Heights

term	SaS	PaP	WH
AFFECTION	115	58	20
JEALOUS	10	7	11
GOSSIP	2	0	6
WUTHERING	0	0	38



Cosine: Example

term frequencies (counts)

term	SaS	PaP	WH
AFFECTION	115	58	20
JEALOUS	10	7	11
GOSSIP	2	0	6
WUTHERING	0	0	38

log frequency weighting

term	SaS	PaP	WH
AFFECTION	3.06	2.76	2.30
JEALOUS	2.0	1.85	2.04
GOSSIP	1.30	0	1.78
WUTHERING	0	0	2.58

(To simplify this example, we don't do idf weighting.)

42

Cosine: Example

log frequency weighting

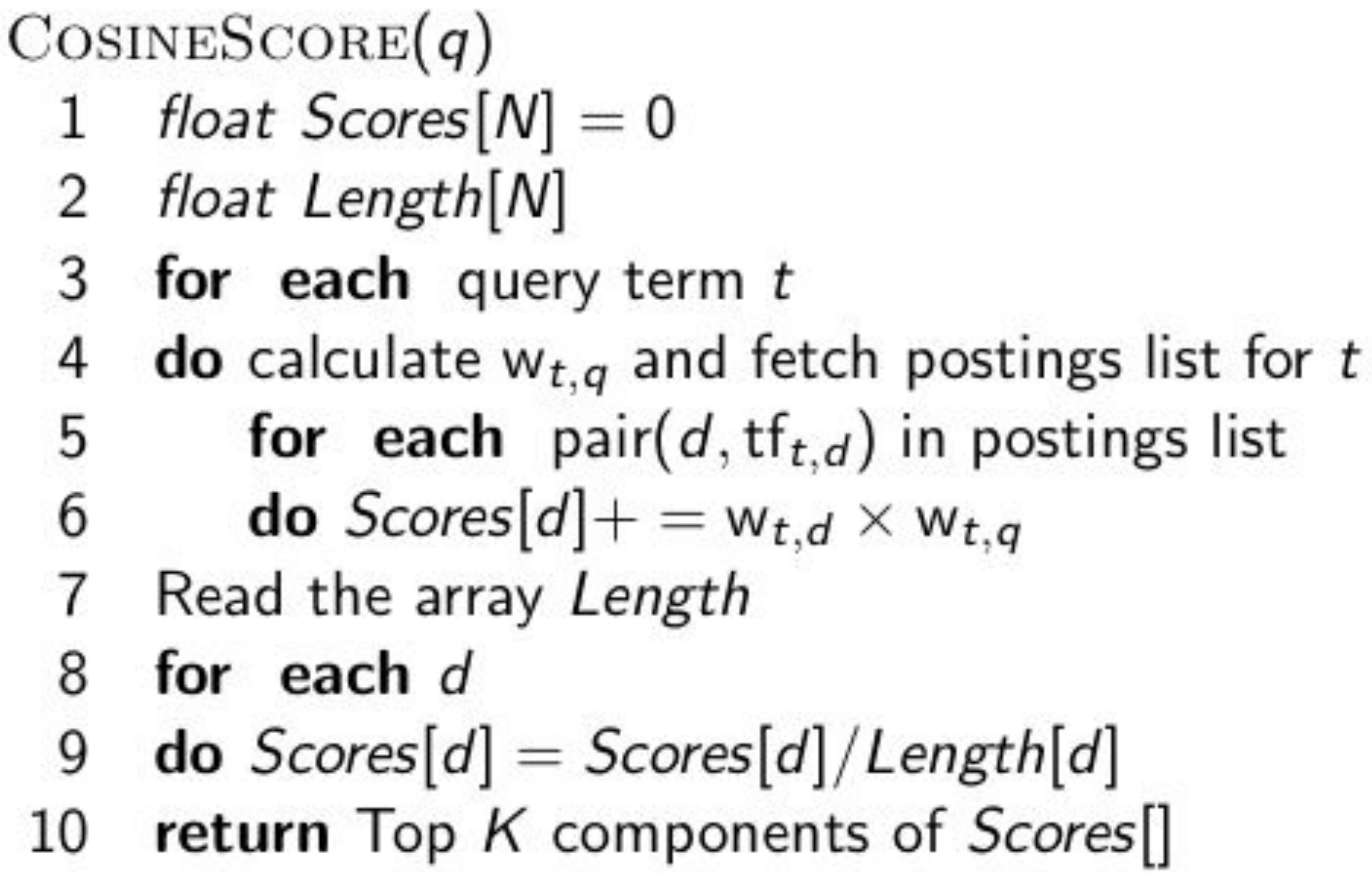
term	SaS	PaP	WH
AFFECTION	3.06	2.76	2.30
JEALOUS	2.0	1.85	2.04
GOSSIP	1.30	0	1.78
WUTHERING	0	0	2.58

log frequency weighting & cosine normalization

term	SaS	PaP	WH
AFFECTION	0.789	0.832	0.524
JEALOUS	0.515	0.555	0.465
GOSSIP	0.335	0.0	0.405
WUTHERING	0.0	0.0	0.588

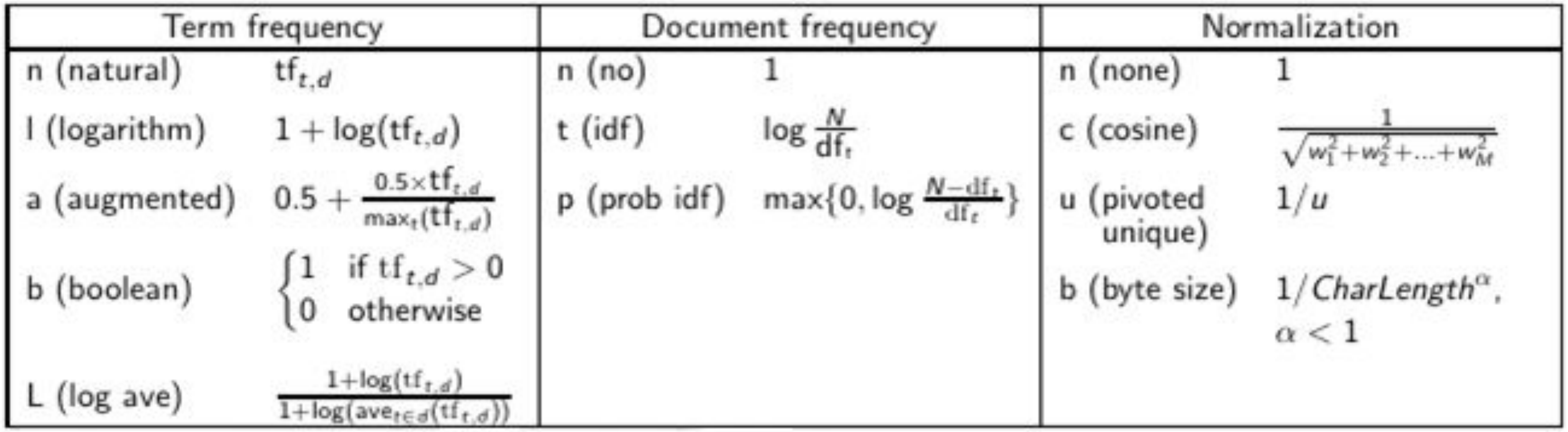
- cos(SaS,PaP) ≈ $0.789 * 0.832 + 0.515 * 0.555 + 0.335 * 0.0 + 0.0 * 0.0 \approx 0.94$
- cos(SaS,WH) ≈ 0.79
- cos(PaP,WH) ≈ 0.69
- Why do we have cos(SaS,PaP) > cos(SAS,WH)?

Computing the cosine score

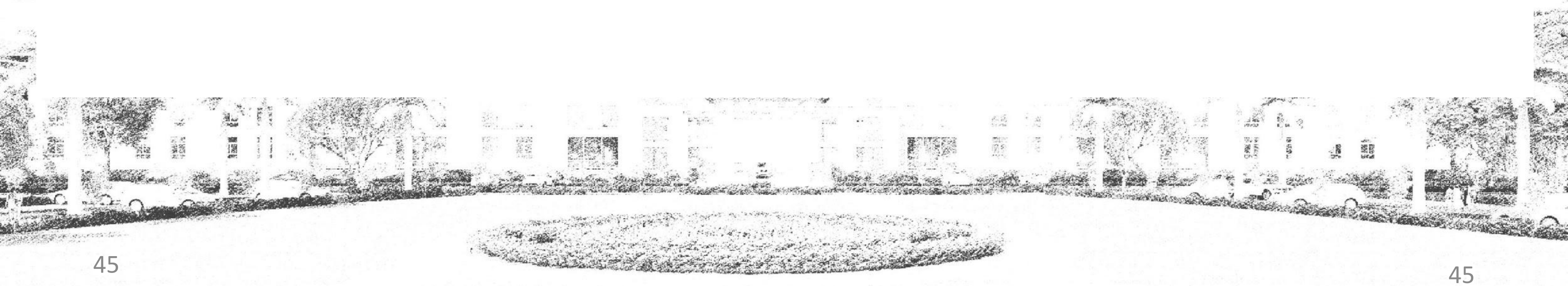


```
CosineScore($q$)
float $Scores[N] = 0$
float $Length[N]$
for each query term $t$
do calculate $w_{t,q}$ and fetch postings list for $t$
    for each pair($d$, $tf_{t,d}$) in postings list
    do $Scores[d] += w_{t,d} \times w_{t,q}$
Read the array $Length$
for each $d$
do $Scores[d] = Scores[d]/Length[d]$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

Components of tf-idf weighting



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



tf-idf example

- We often use different weightings for queries and documents.

- Notation: ddd.qqq
- Example: lnc.ltn

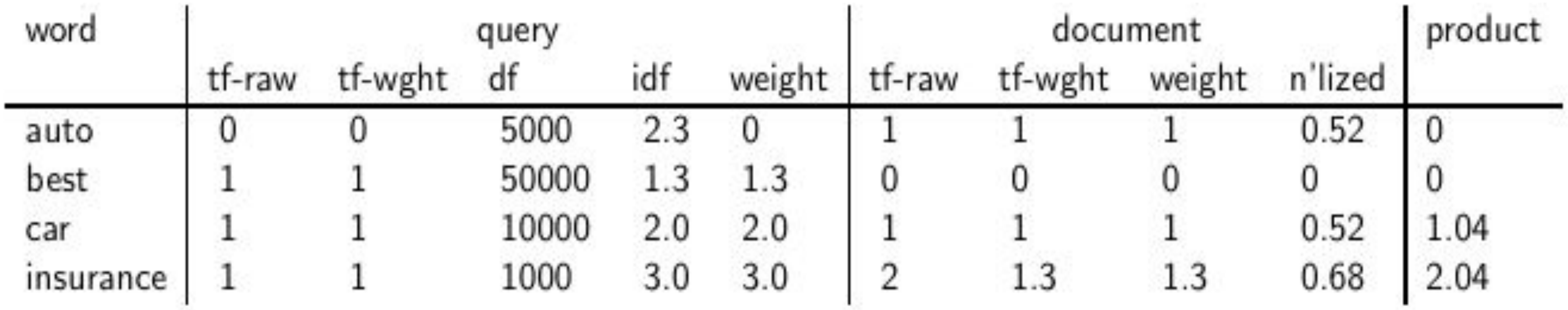
- document: logarithmic tf, no df weighting, cosine normalization

- query: logarithmic tf, idf, no normalization

- Isn’t it bad to not idf-weight the document?
- Example query: “best car insurance”
- Example document: “car insurance auto insurance”

TF-IDF Example:

Query: “best car insurance”. Document: “car insurance auto insurance”



word	query					document				product
	tf-raw	tf-wght	df	idf	weight	tf-raw	tf-wght	weight	n'lized	
auto	0	0	5000	2.3	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0	0	0	0	0
car	1	1	10000	2.0	2.0	1	1	1	0.52	1.04
insurance	1	1	1000	3.0	3.0	2	1.3	1.3	0.68	2.04

Key to columns: tf-raw: raw (unweighted) term frequency, tf-weight: logarithmically weighted term frequency, df: document frequency, idf: inverse document frequency, weight: the final weight of the term in the query or document, n’lized: document weights after cosine normalization, product: the product of final query weight and final document weight

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

$1/1.92 \approx 0.52$

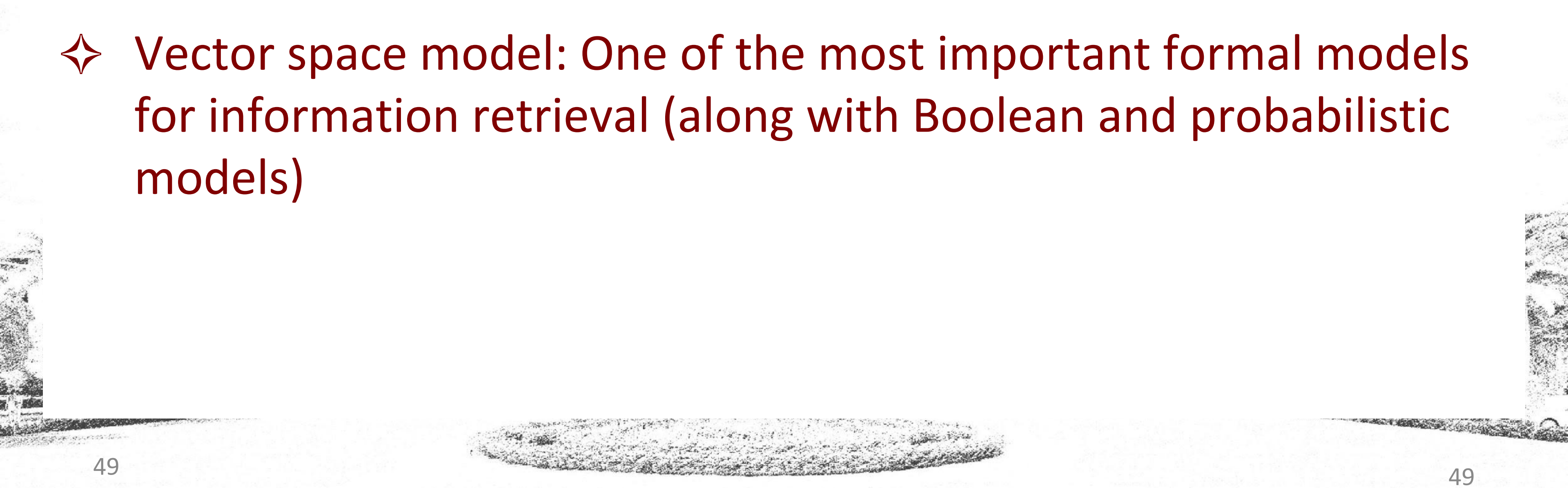
$1.3/1.92 \approx 0.68$ Final similarity score between query and document: $\sum i \; wq_i \cdot wd_i = 0 + 0 + 1.04 + 2.04 = 3.08$ **Questions?**

Summary: Ranked retrieval in the vector space model

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

Take-away today

- Ranking search results: why it is important (as opposed to just presenting a set of unordered Boolean results)
- Term frequency: This is a key ingredient for ranking.
- TF-IDF ranking: best known traditional ranking scheme
- Vector space model: One of the most important formal models for information retrieval (along with Boolean and probabilistic models)



Summary

In this class, we focused on:

(a) Words / Terms / Lexical Units

(b) Preparing Term – Document matrix

(c) Boolean Retrieval

(d) Inverted Index Construction

  i. Computational Cost

  ii. Managing Bigger Collections

  iii. How much storage is required?

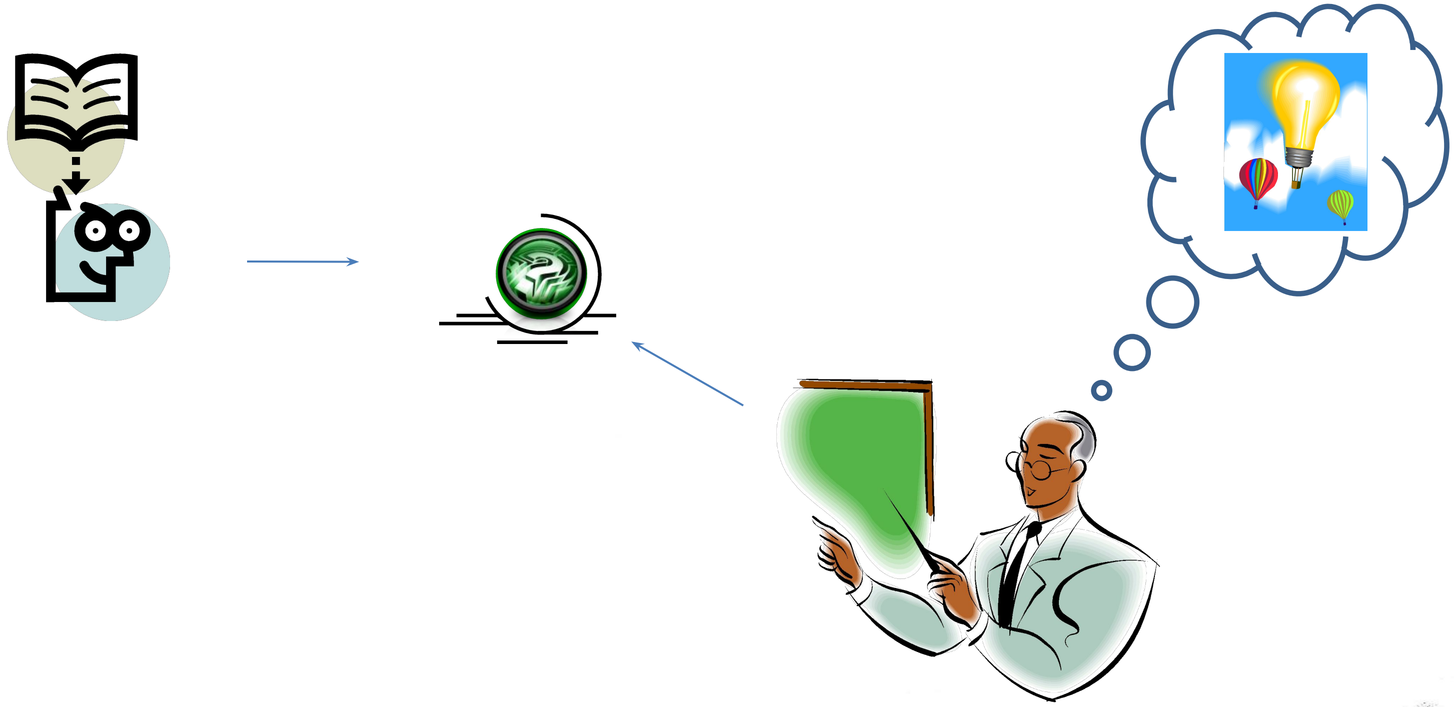
  iv. Boolean Queries: Exact match

Acknowledgements

Thanks to ALL RESEARCHERS:

1. Introduction to Information Retrieval Manning, Raghavan and Schutze, Cambridge University Press, 2008.
2. Search Engines Information Retrieval in Practice W. Bruce Croft, D. Metzler, T. Strohman, Pearson, 2009.
3. Information Retrieval Implementing and Evaluating Search Engines Stefan Büttcher, Charles L. A. Clarke and Gordon V. Cormack, MIT Press, 2010.
4. Modern Information Retrieval Baeza-Yates and Ribeiro-Neto, Addison Wesley, 1999.
5. Many Authors who contributed to SIGIR / WWW / KDD / ECIR / CIKM / WSDM and other top tier conferences
6. Prof. Mandar Mitra, Indian Statistical Institute, Kolkatata (https://www.isical.ac.in/~mandar/)

Thanks ...



... Questions ???