



BUILDING TEST COLLECTIONS

Reusable Test Collections

- Document Collection
- Topics (sample of information needs)
- Relevance judgments (qrels)

How can we get it?

- For web search, companies apply their own studies to assess the performance of their search engine.
- Web-search performance is monitored by:
 - Traffic
 - User clicks and session logs
 - Labelling results for selected users' queries
- Academia (or lab settings):
 - Someone goes out and builds them (expensive)
 - As a byproduct of large scale evaluations (collaborative effort)
- IR **Evaluation Campaigns** are created for this reason

IR Evaluation Campaigns

- IR test collections are provided for scientific communities to develop better IR methods.
- Collections and queries are provided, relevance judgements are built during the campaign.
- TREC = Text REtrieval Conference <http://trec.nist.gov/>
 - Main IR evaluation campaign, sponsored by NIST (US gov).
 - Series of annual evaluations, started in 1992.
- Other evaluation campaigns
 - CLEF: European version (since 2000)
 - NTCIR: Asian version (since 1999)
 - FIRE: Indian version (since 2008)

TREC Tracks and Tasks

- TREC (and other campaigns) are formed of a set of **tracks**, each track is about (one or more) search **tasks**.
 - Each track/task is about searching a set of documents of given genre and domain.
- Examples
 - TREC Web track
 - TREC Medical track
 - TREC Legal track → CLEF-IP track → NTCIR patent mining track
 - TREC Microblog track
 - Adhoc search task
 - Filtering task

TREC Collection

- A set of hundreds of thousands or millions of docs
 - 1B in case of web search (TREC ClueWeb09)
- The typical format of a document:

```
<DOC>
<DOCNO> 1234 </DOCNO>
<TEXT>
    This is the document.
    Multilines of plain text.
</TEXT>
</DOC>
```

TREC Topic

- **Topic: a statement of information need**
- Multiple topics (~50) developed (mostly) **at NIST** for a collection.
- Developed by experts and associated with additional details.
 - Title: the query text
 - Description: description of what is meant by the query.
 - Narrative: what should be considered relevant.

`<num>189</num>`

`<title>Health and Computer Terminals</title>`

`<desc>Is it hazardous to the health of individuals to work with computer terminals on a daily basis?</desc>`

`<narr>Relevant documents would contain any information that expands on any physical disorder/problems that may be associated with the daily working with computer terminals. Such things as carpel tunnel, cataracts, and fatigue have been said to be associated, but how widespread are these or other problems and what is being done to alleviate any health problems</narr>`

Relevance Judgments

- For each topic, set of relevant docs is required to be known for an effective evaluation!
- **Exhaustive assessment** is usually impractical
 - TREC usually has 50 topics
 - Collection usually has >1 million documents
- **Random sampling** won't work
 - If relevant docs are rare, none may be found!
- **IR systems can help** focus the sample (**Pooling**)
 - Each system finds some relevant documents
 - Different systems find different relevant documents
 - Together, enough systems will find most of them

Pooled Assessment Methodology

1. Systems submit top **1000** documents per topic
 2. Top **100** documents from each are *manually* judged
 - Single pool, duplicates removed, arbitrary order
 - Judged by the person who developed the topic
 3. Treat unevaluated documents as **not** relevant
 4. Compute MAP (or others) down to **1000** documents
- To make pooling work:
- Good number of participating systems
 - Systems must do reasonably well
 - Systems must be different (not all “do the same thing”)

Example

In one of TREC tracks, 3 teams T_1 , T_2 , and T_3 have participated and they were asked to retrieve up to 15 documents per query. In reality (with exhaustive judgments), a query Q has 9 relevant documents in the collection: A, B, C, D, E, F, G, H, and I.

The submitted ranked lists are as follows:

Rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T_1	A	M	Y	R	K	L	B	Z	E	N	D	C	W		
T_2	Y	A	J	R	N	Z	M	C	G	B	X	P	D	K	W
T_3	G	B	Y	K	E	A	Z	L	N	C	H	K	W	X	

We constructed the judging pools for Q using only the top 5 documents of each of the submitted ranked lists.

What is the Average Precision of each of the 3 teams?



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

دورة "استرجاع المعلومات" باللغة العربية - صيف ٢٠٢١

Information Retrieval – Summer 2021



4. Ranked Retrieval I (Vector Space Model & BM25)

Tamer Elsayed
Qatar University

Today's Roadmap

- Simple scoring
- TF-IDF ranking
- Vector Space Model
- BM25 ranking



Boolean 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.
- **Not good for the majority of users.**
 - Most incapable of writing Boolean queries.
 - Most don't want to go through 1000s of results.
 - This is particularly true of web search.

Ranked Retrieval

- Typical queries: free text queries
- Results are “ranked” with respect to a query
- Large result sets are not an issue
 - We just show the top k (≈ 10) results
 - We don't overwhelm the user

How?

- Top ranked documents are the most likely to satisfy user's query.
- Assign a score – say in $[0, 1]$ – to each document.
- **Score (d, q)** measures how well doc d matches a query q .

Scoring Example: Jaccard coefficient

- Commonly-used measure of overlap of two sets A and B
 - What are the 2 sets in our context?

$$jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- $jaccard(A, A) = 1$ and $jaccard(A, B) = 0$ if $A \cap B = \emptyset$
- A and B don't have to be of the same size.
- Always assigns a number between 0 and 1.

Is it a good scoring function?

Issues With Jaccard for Scoring

Term frequency?

- Doesn't consider **term frequency** (how many times a term occurs in a document)

Term importance?

- It treats all terms equally!
 - How about **rare terms** in a collection? more informative than frequent terms.

Length?

- Needs more sophisticated way of **length normalization**





12



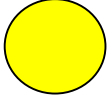
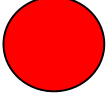
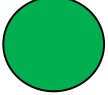
For the query "the arab world" and the document "fifa world cup in arab country", what is Jaccard similarity (after removing stop words)?

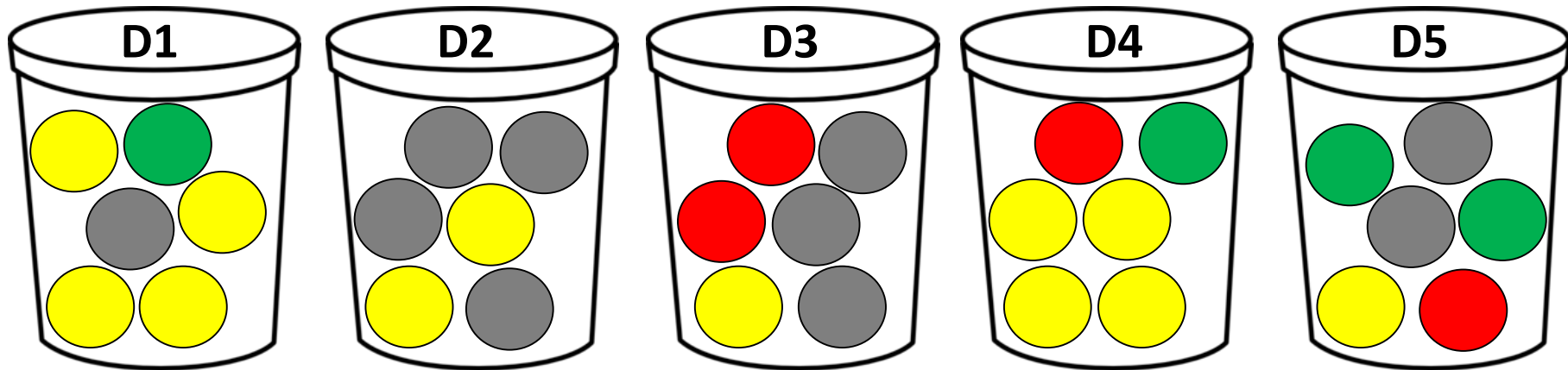
- $2/3$
- $2/6$
- $2/7$
- $2/10$
- $3/6$
- $3/7$
- $3/10$



TF-IDF RANKING

Example

- Collection of 5 documents (balls = terms)
- Query   
- Which is the least relevant document?
- Which is the most relevant document?



Term-Document Count Matrix

- Each document as a count (frequency) vector in \mathbb{N}^v

	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

Bag-of-Words Model

- Doesn't consider the ordering of words in a document
 - John is quicker than Mary*
 - Mary is quicker than John*
- Same vectors!**

1. Frequent Terms in a Document

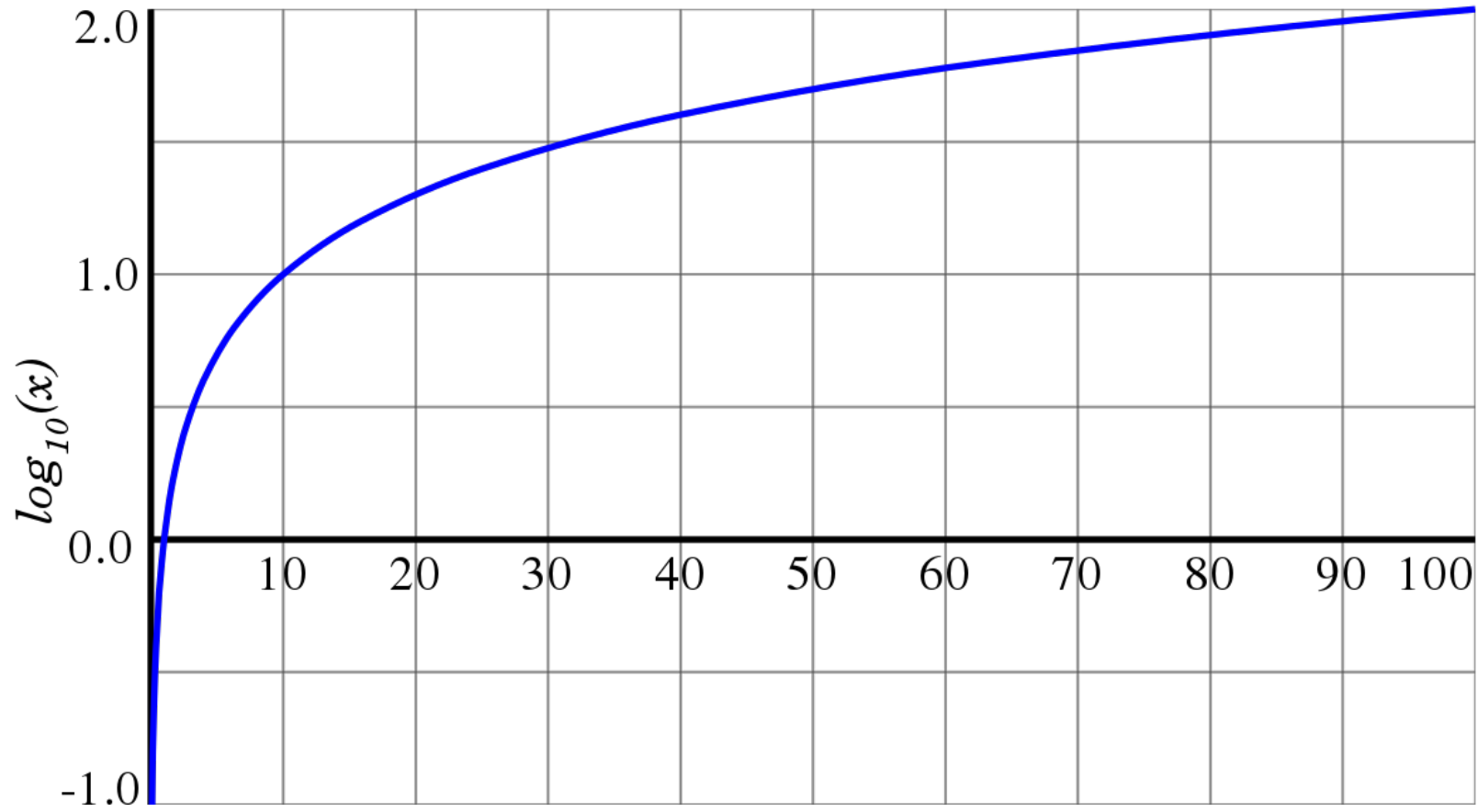
Term Frequency

- $tf_{t,d}$: the number of times that term t occurs in doc d .
- We want to use tf when computing query-document match scores. But how?

○ Raw term frequency?

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

Log-Frequency Weighting



Log-Frequency Weighting

- The log frequency weight of term t in d is

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

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

- **Score for a document-query pair**: sum over terms t in both q and d :

$$\textit{score}(q, d) = \sum_{t \in q \cap d} w_{t,d}$$

- The score is 0 if none of the query terms is present in the document.

2. Informative Terms in a Collection

- Rare terms are more informative than frequent terms
 - Recall stop words
- We want a high weight for rare terms.
- **Collection Frequency cf_t ?**
 - number of occurrences of term t in the collection
- **Document Frequency df_t ?**
 - the number of documents that contain t
 - **inverse measure of the informativeness of t**
 - $df_t \leq N$

Inverse Documents Frequency, *idf*

- *idf* (inverse document frequency) of t :

$$idf_t = \log_{10}\left(\frac{N}{df_t}\right)$$

- $\log(N/df_t)$ instead of N/df_t to “dampen” the effect of *idf*.

- 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

tf.idf Term Weighting

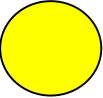
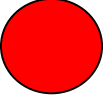
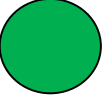
- The tf-idf weight of a term is the product of its *tf* weight and its *idf* weight.

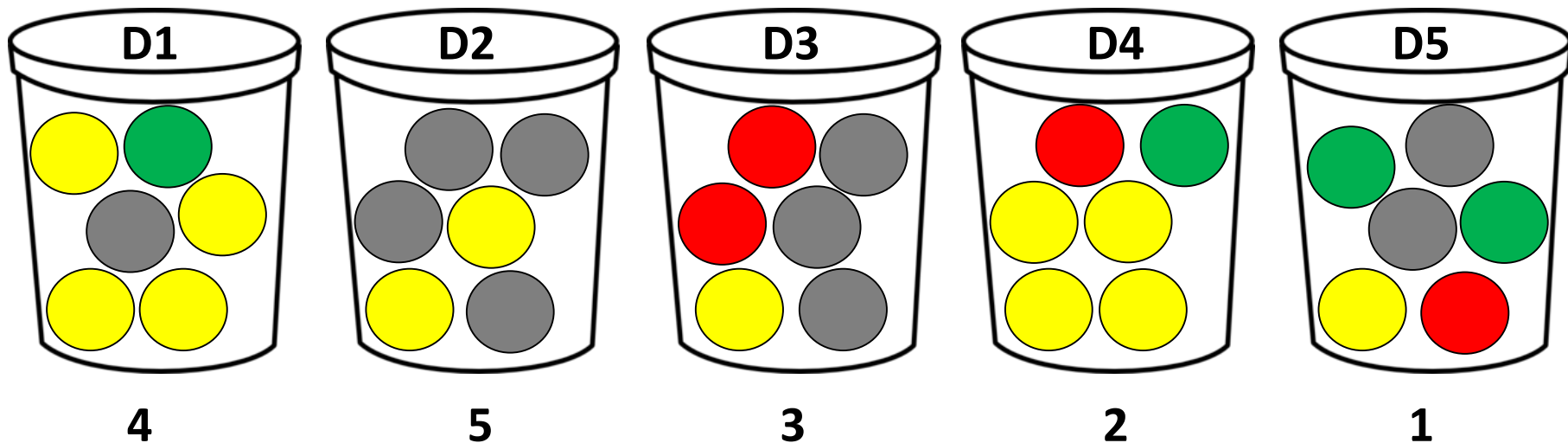
$$w_{t,d} = (1 + \log_{10} tf_{t,d}) \times idf_t$$

- One of the best-known weighting scheme in IR
 - Increases with the number of occurrences within a document
 - Increases with the rarity of the term in the collection

$$score(q, d) = \sum_{t \in q \cap d} w_{t,d}$$

Back to our example ...

- Collection of 5 documents (balls = terms)
- Query    the destructive storm
- Which is the least relevant document?
- Which is the most relevant document?







13



..... is a relation between a term and a collection

- Term frequency
- Document frequency
- Collection frequency

A rare term has high IDF.

- Yes
- No

A high tf-idf of a term indicates the importance of a term in ...

- a document
- the collection
- both



Today's Roadmap

- Simple scoring
- TF-IDF ranking
- Vector Space Model
- BM25 ranking



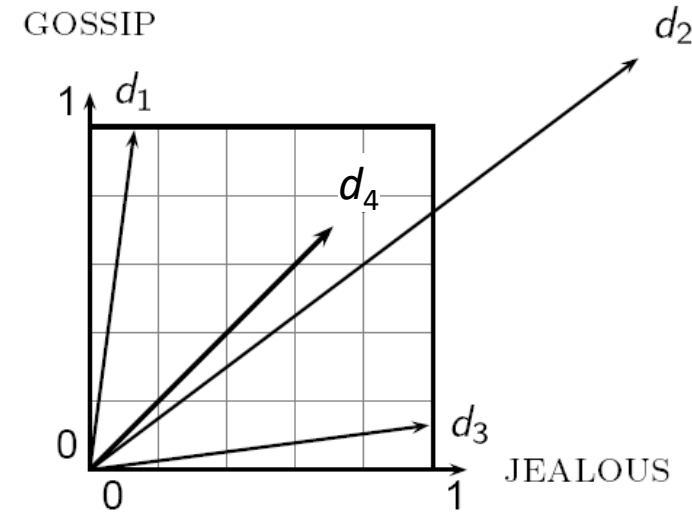
Binary → Count → Weight Matrix

	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 $\in \mathbb{R}^{|V|}$

Documents as Vectors

- $|V|$ -dimensional 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 a web search engine.
- These are very sparse vectors - most entries are zero.



Vector Space Model

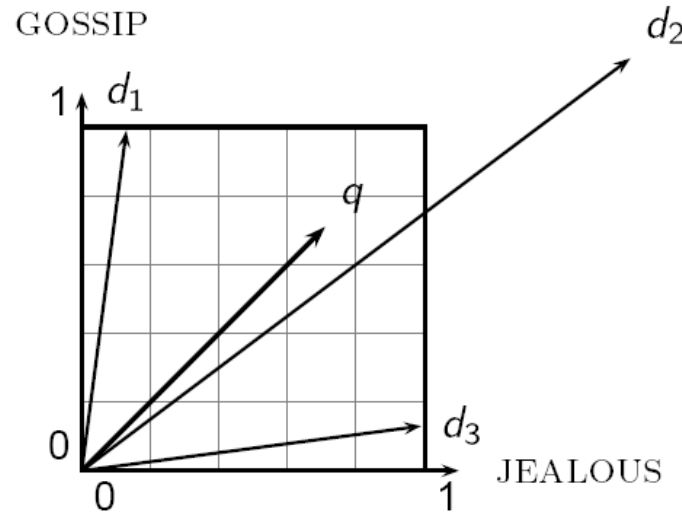
Queries as Vectors

- **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

How?

Euclidean Distance?

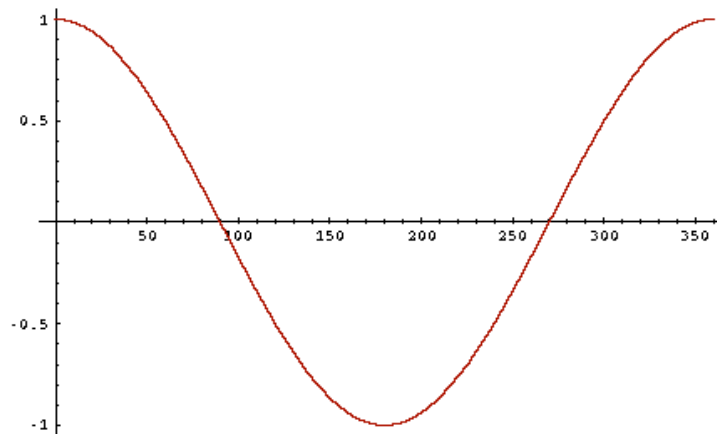
- Distance between the end points of the two vectors



- Large for vectors of different lengths.
- Thought experiment: take a document d and append it to itself. Call this document d' .
 - “Semantically” d and d' have the same content
 - Euclidean distance can be quite large

Angle Instead of Distance

- The angle between the two documents is 0, corresponding to maximal similarity.
- Key idea: Rank documents according to angle with query.
 - Rank documents in increasing order of the angle with query
 - Rank documents in decreasing order of $\cos(\text{query}, \text{document})$
- Cosine is a monotonically decreasing function for the interval $[0^\circ, 180^\circ]$



How to compute?

3. 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 (on surface of unit hypersphere)
- 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

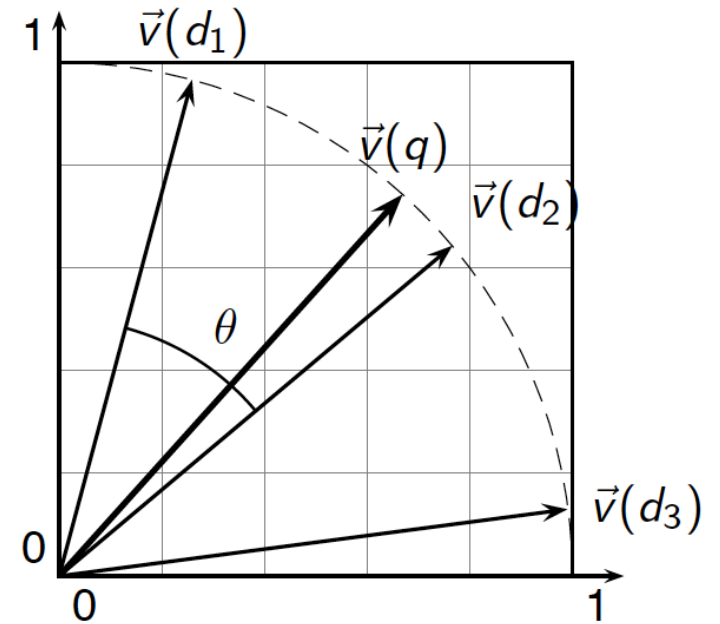
Cosine “Similarity” (Query, Document)

- \vec{q}_i is the [tf-idf] weight of term i in the query
- \vec{d}_i is the [tf-idf] weight of term i in the doc
- For normalized vectors:

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

- For non-normalized vectors:

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



Computing Cosine Scores?

COSINESCORE(q)

```
1  float Scores[N] = 0
2  Initialize Length[N]
3  for each query term  $t$ 
4  do calculate  $w_{t,q}$  and fetch postings list for  $t$ 
5      for each pair( $d, tf_{t,d}$ ) in postings list
6      do Scores[d] +=  $wf_{t,d} \times w_{t,q}$ 
7  Read the array Length[d]
8  for each  $d$ 
9  do Scores[d] = Scores[d] / Length[d]
10 return Top  $K$  components of Scores[]
```

TAAT
Query Processing

Variants of tf-idf Weighting

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)	$\frac{1 + \log(tf_{t,d})}{1 + \log(\text{ave}_{t \in d}(tf_{t,d}))}$				

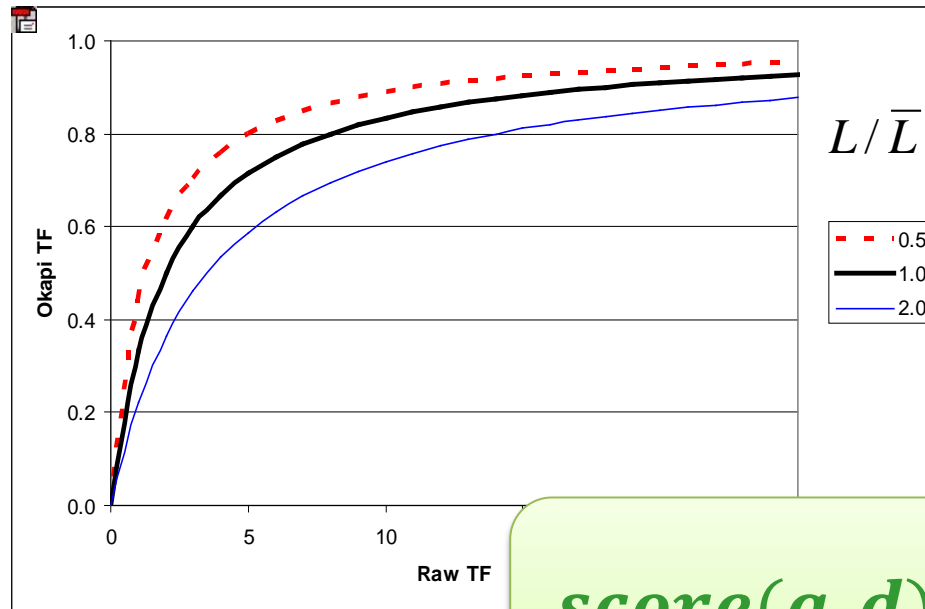
- Many search engines allow for **different weightings for queries vs. documents.**
- SMART Notation: use notation *ddd.qqq*, using the acronyms from the table
- A very standard weighting scheme is: Inc.ltc

Okapi BM25 Ranking Function

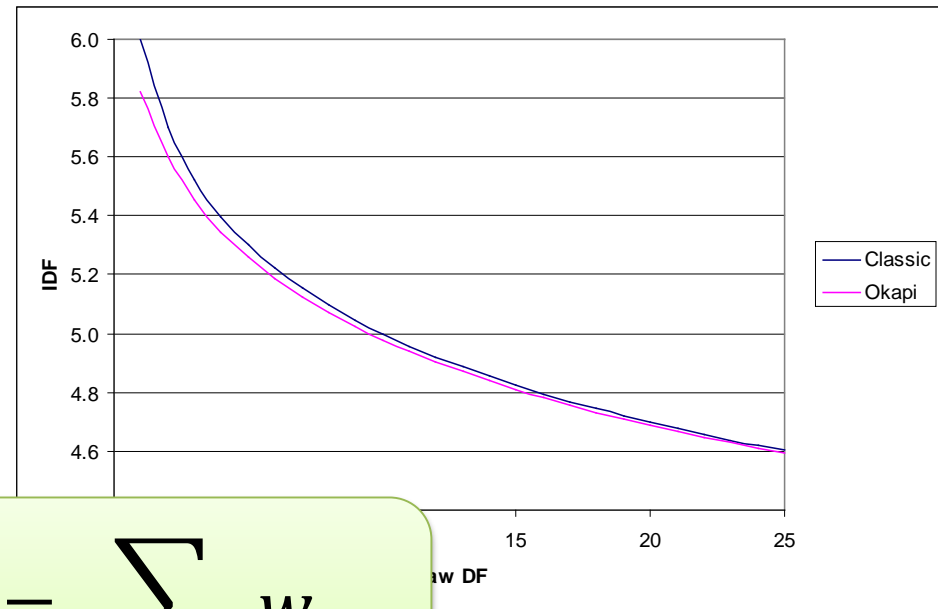
L_d : Length of d
 \bar{L} : average doc length
in collection

$$w_{t,d} = \frac{tf_{t,d}}{1.5 \frac{L_d}{\bar{L}} + tf_{t,d} + 0.5} * \log \left(\frac{N - df_t + 0.5}{df_t + 0.5} \right)$$

tf component



idf component



$$score(q, d) = \sum_{t \in q \cap d} w_{t,d}$$

Okapi BM25 Ranking Function

L_d : Length of d
 \bar{L} : average doc length
in collection
 k_1, b : parameters

$$w_{t,d} = \frac{(k_1 + 1) tf_{t,d}}{k_1((1 - b) + b \frac{L_d}{\bar{L}}) + tf_{t,d}} * \log \left(\frac{N - df_t + 0.5}{df_t + 0.5} \right)$$

$k_1 = 2, b = 0.75$

$$w_{t,d} = \frac{tf_{t,d}}{1.5 \frac{L_d}{\bar{L}} + tf_{t,d} + 0.5} * \log \left(\frac{N - df_t + 0.5}{df_t + 0.5} \right)$$

Summary – Vector Space Ranking

- Represent the query as a term-weighted (e.g., tf-idf) vector.
- Represent each document as a term-weighted (e.g., 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.





14



**In Vector Space Model,
are represented as vectors.**

**In Vector Space Model, the
dimensionality of the space
is**

**Cosine similarity can be
used to compute similarity
between (you can
choose multiple)**

- documents only
- queries only
- both documents and queries

- the vocabulary size
- the collection size
- the maximum length of documents in the collection

- a query and a document
- a query and another query
- a document and another document



*Can we “model the language”
to rank search results?*

