



THE UNIVERSITY
of EDINBURGH

Text Technologies for Data Science

INFR11145

Ranked IR

Instructor:
Walid Magdy

8-Oct-2025

1

Lecture Objectives

- Learn about Ranked IR
 - TFIDF
 - VSM
 - SMART notation
- Implement:
 - TFIDF



2

Boolean Retrieval

- Thus far, our queries have all been Boolean.
 - Documents either: “match” or “no match”.
- Good for expert users with precise understanding of their needs and the collection.
 - Patent search uses sophisticated sets of Boolean queries and check hundreds of search results
(car OR vehicle) AND (motor OR engine) AND NOT (cooler)
- 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 for web search
 - Question: What is the most unused web-search feature?

Walid Magdy, TTDS 2025/2026



3

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
- Criteria:
 - Top ranked documents are the most likely to satisfy user's query
 - Score is based on how well documents match a query
Score(d, q)

Walid Magdy, TTDS 2025/2026



4

Old Example

- Find documents matching query {ink wink}
 - Load inverted lists for each query word
 - Merge two postings lists → **Linear merge**

- Apply function for matches

- Boolean: exist / not exist = 0 or 1
- Ranked: $f(tf, df, length, \dots) = 0 \rightarrow 1$

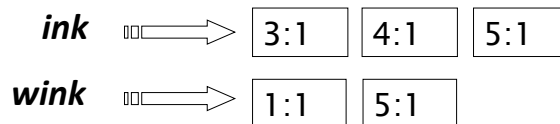
Matches

1: $f(0,1)$

3: $f(1,0)$

4: $f(1,0)$

5: $f(1,1)$



Function example: Jaccard coefficient

- a commonly used measure of overlap of two sets A and B
- $jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$
 - D1:** He likes to wink, he likes to drink
 - D2:** He likes to drink, and drink, and drink
- $jaccard(A, A) = 1$
- $jaccard(A, B) = 0$, if $A \cap B = 0$
- Example:
 - $D1 \cup D2 = \{\text{he, likes, to, wink, and, drink}\}$
 - $D1 \cap D2 = \{\text{he, likes, to, drink}\}$
 - $jaccard(D1, D2) = \frac{4}{6} = 0.6667$

Jaccard coefficient: Issues




- Does not consider **term frequency** (how many times a term occurs in a document)
- It treats all terms equally!
 - How about **rare terms** in a collection? more informative than frequent terms.
 - *He likes to drink*, shall “to” == “drink”?
- Needs more sophisticated way of **length** normalization
 - $|D1| = 3$, $|D2| = 1000$!
 - $D1 \rightarrow Q$, $D2 \rightarrow D$

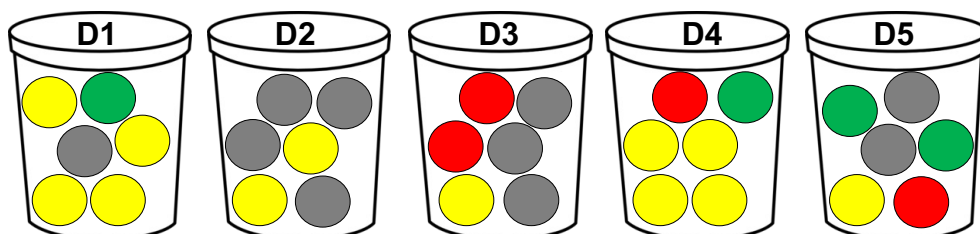
Walid Magdy, TTDS 2025/2026



7

Should terms be treated the same?

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



Walid Magdy, TTDS 2025/2026



8

TFIDF

- **TFIDF:**
Term Frequency, Inverse Document Frequency
- **$tf(t,d)$:**
number of times term t appeared in document d
 - As $tf(t,d) \uparrow \rightarrow$ importance of t in $d \uparrow$
 - Document about IR, contains “retrieval” more than others
- **$df(t)$:**
number of documents term t appeared in
 - As $df(d) \uparrow \rightarrow$ importance if t in a collection \downarrow
 - “the” appears in many document \rightarrow not important
 - “FT” is not important word in financial times articles

DF, CF, & IDF

- **DF \neq CF** (collection frequency)
 - **$cf(t)$** = total number of occurrences of term t in a collection
 - **$df(t) \leq N$** (N : number of documents in a collection)
 - **$cf(t)$** can be $\geq N$
- **DF** is more commonly used in IR than **CF**
 - **CF** is still used
- **$idf(t)$:** inverse of **$df(t)$**
 - As $idf(t) \uparrow \rightarrow$ rare term \rightarrow importance \uparrow
 - **$idf(t)$** \rightarrow measure of the informativeness of t

DF vs CF

he	drink	ink	likes	pink	think	wink	
2	1	0	2	0	0	1	← D1: He likes to wink, he likes to drink
1	3	0	1	0	0	0	← D2: He likes to drink, and drink, and drink
1	1	1	1	0	1	0	← D3: The thing he likes to drink is ink
1	1	1	1	1	0	0	← D4: The ink he likes to drink is pink
1	1	1	1	1	0	1	← D5: He likes to wink, and drink pink ink
5	5	3	5	2	1	2	DF
6	7	3	6	2	1	2	CF

Walid Magdy, TTDS 2025/2026



11

IDF: formula

$$idf(t) = \log_{10}\left(\frac{N}{df(t)}\right)$$

- Log scale used to dampen the effect of IDF

- Suppose $N = 1$ million \rightarrow

term	df(t)	idf(t)
calpurnia	1	6
animal	100	4
sky	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Walid Magdy, TTDS 2025/2026



12

TFIDF term weighting

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

- Combines TF and IDF to find the weight of terms

$$w_{t,d} = (1 + \log_{10} tf(t, d)) \times \log_{10} \left(\frac{N}{df(t)} \right)$$

- For a query q and document d , retrieval score $f(q,d)$:

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

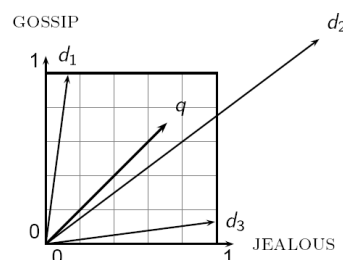
Document/Term vectors with tfidf

	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

→ Vector Space Model

Vector Space Model

- Documents and Queries are presented as vectors
- $\text{Match}(Q, D) = \text{Distance between vectors}$
- Example: $Q = \text{Gossip Jealous}$
- Euclidean Distance?
Distance between the endpoints of the two vectors
- Large for vectors of diff. lengths
- 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



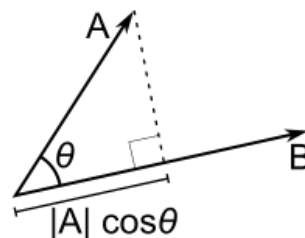
Walid Magdy, TTDS 2025/2026



15

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 cosine (query, document)
- Cosine of angle = projection of one vector on the other



Walid Magdy, TTDS 2025/2026



16

Length Normalization

- A vector can be 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

Example

- $D1 = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} \rightarrow \|\vec{D1}\|_2 = \sqrt{1 + 9 + 4} = 3.74$
- $D1_{normalized} = \begin{bmatrix} 0.267 \\ 0.802 \\ 0.535 \end{bmatrix}$
- $D2 = \begin{bmatrix} 3 \\ 9 \\ 6 \end{bmatrix} \rightarrow \|\vec{D1}\|_2 = \sqrt{9 + 81 + 36} = 11.25$
- $D2_{normalized} = \begin{bmatrix} 0.267 \\ 0.802 \\ 0.535 \end{bmatrix}$

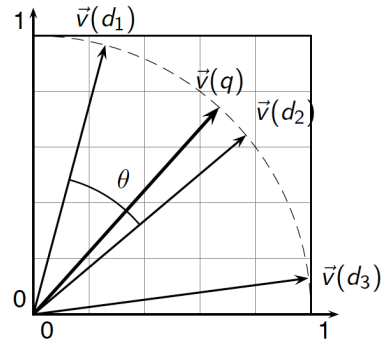
Cosine “Similarity” (Query, Document)

- \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 document
- 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:

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



Walid Magdy, TTDS 2025/2026



THE UNIVERSITY
of EDINBURGH

19

Algorithm

COSINESCORE(q)

- 1 *float* Scores[N] = 0
- 2 *float* 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] += $w_{t,d} \times w_{t,q}$
- 7 Read the array Length
- 8 **for each** d
- 9 **do** Scores[d] = Scores[d] / Length[d]
- 10 **return** Top K components of Scores[]

Walid Magdy, TTDS 2025/2026



THE UNIVERSITY
of EDINBURGH

20

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

- 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.Itc*

Walid Magdy, TTDS 2025/2026



21

For Lab and CW

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

“OR” operator, then:

$$Score(q, d) = \sum_{t \in q \cap d} (1 + \log_{10} tf(t, d)) \times \log_{10} \left(\frac{N}{df(t)} \right)$$

Walid Magdy, TTDS 2025/2026



22

Summary of Steps:

- 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

Retrieval Output

- For a query q_1 , the output would be a list of documents ranked according to the $score(q_1, d)$
- Possible output format:

1,	710,	0.9234
1,	213,	0.7678
1,	103,	0.6761
1,	13,	0.6556
1,	501,	0.4301

Query id document id score

Resources

- Text book 1: Intro to IR, Chapter 6.2 → 6.4
- Text book 2: IR in Practice, Chapter 7

- Lab 3