Information Access with Apache Lucene - Part 1

Metodi per il Ritrovamento dell'Informazione

Laurea Triennale in Informatica

Università degli Studi di Bari Aldo Moro

Prof. Cataldo Musto

cataldo.musto@uniba.it

Credits: Marco Polignano

Code Repository & Requirements

Code repository

https://github.com/swapUniba/MRI_2024_25

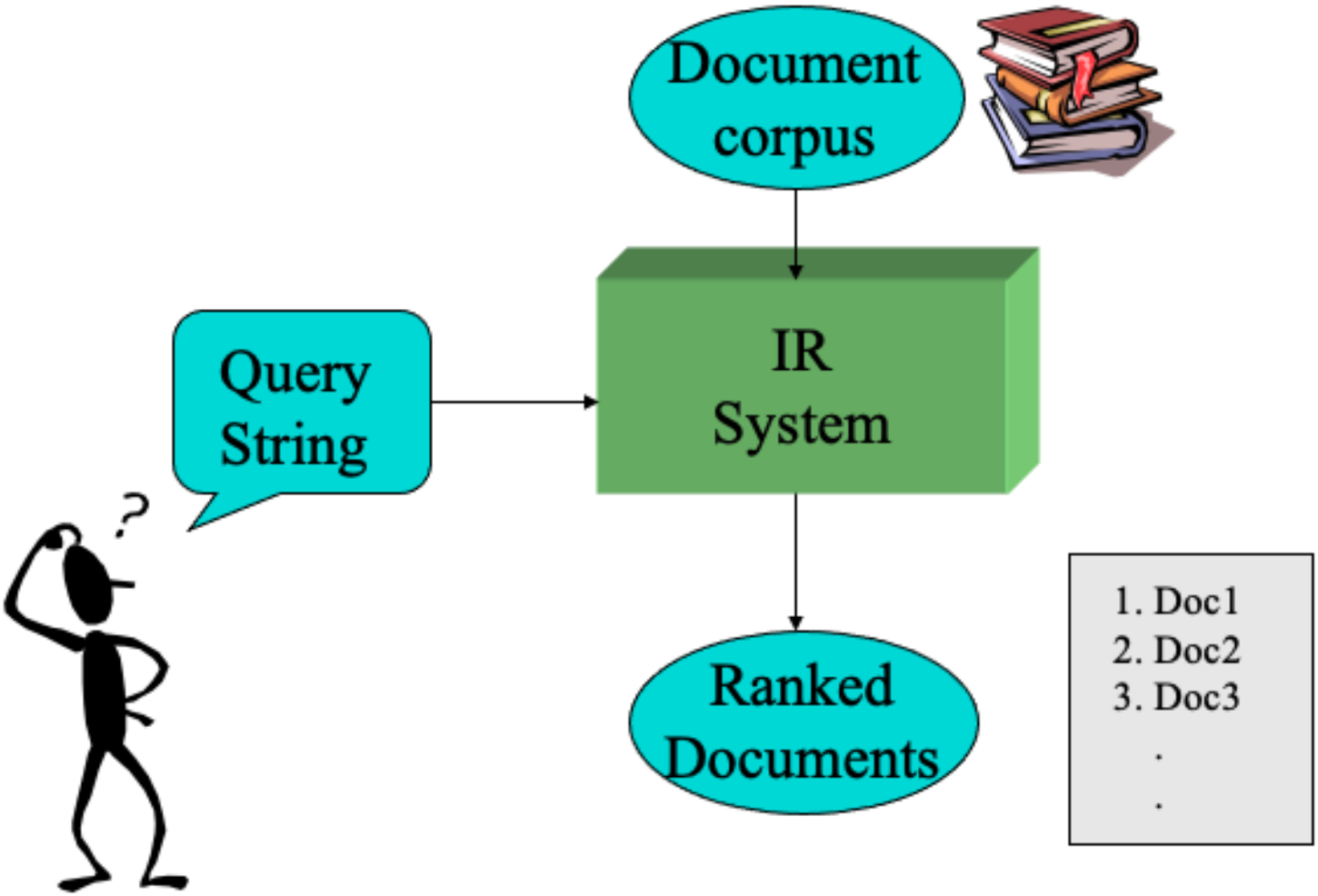
Requirements

- Java SDK 1.8+ https://www.java.com/en/download/
- IDE: NetBeans, IntelliJ, Eclipse, …
- **Maven:** https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

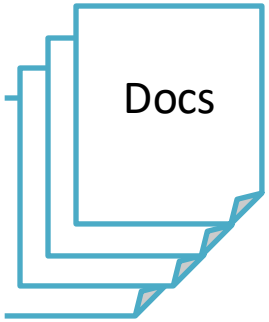
Recap

SEARCH ENGINE

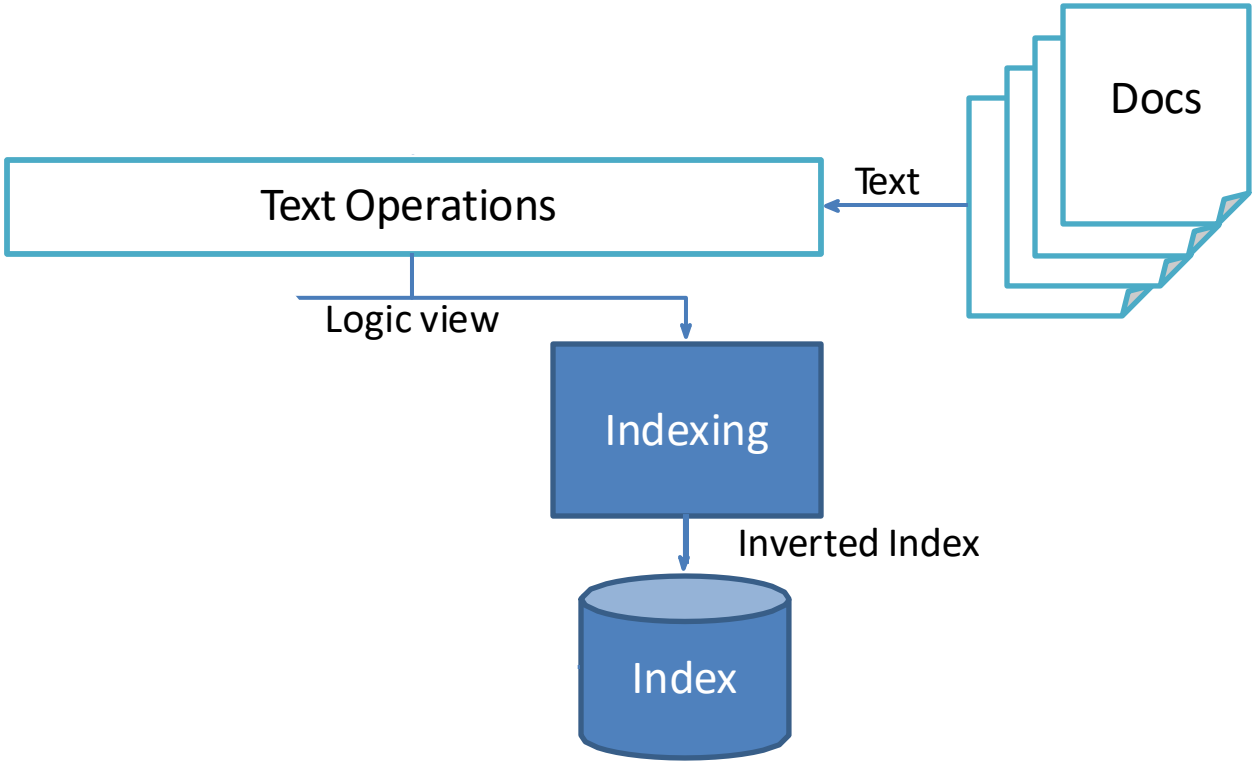
Information Retrieval Process (recap)



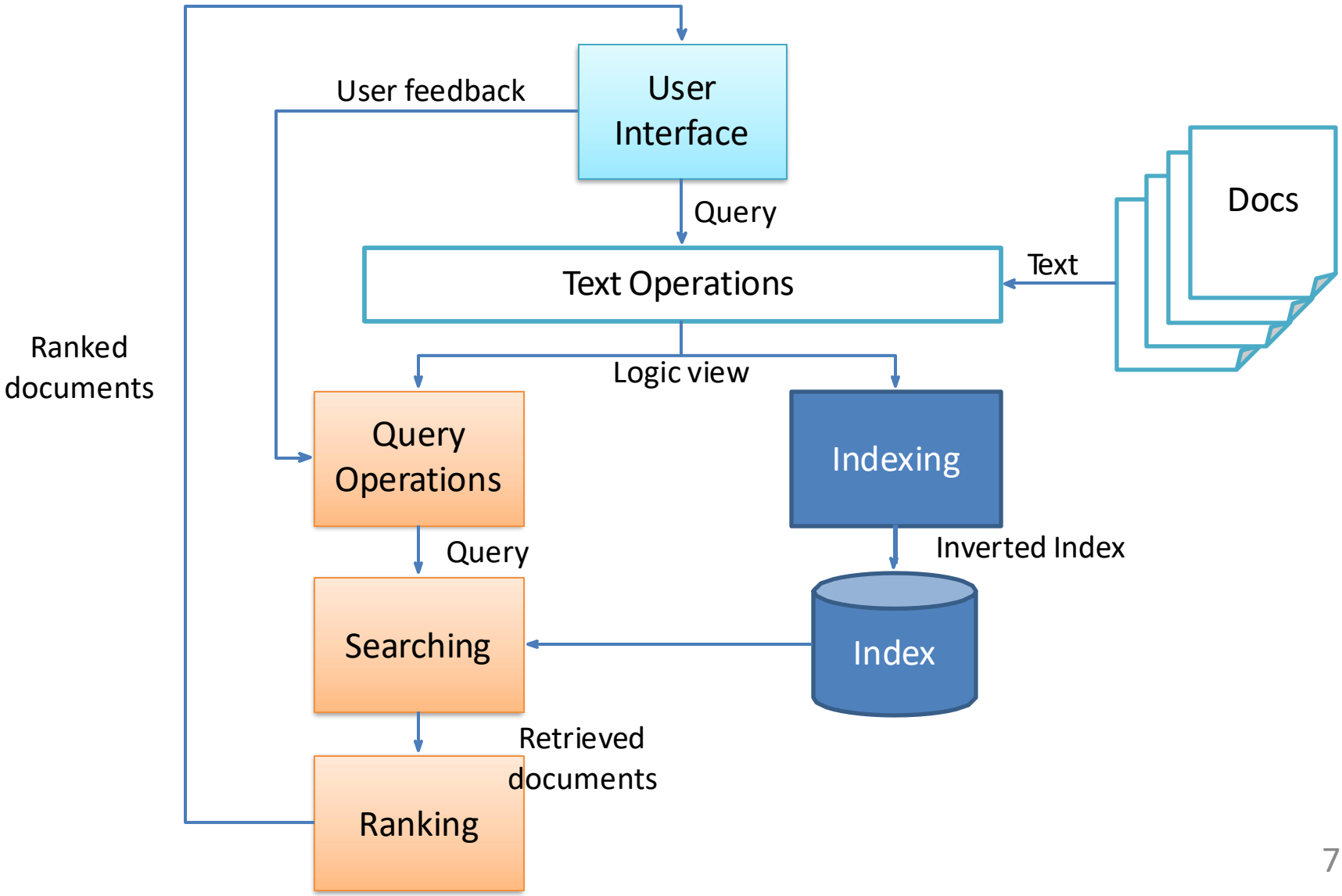
Information Retrieval Process (recap)



Information Retrieval Process (recap)



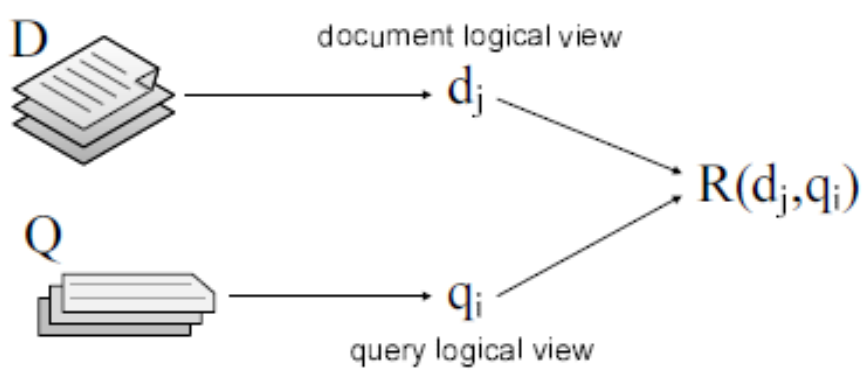
Information Retrieval Process (recap)



Information Retrieval Model

<D, Q, F, R(q_i, d_j)>

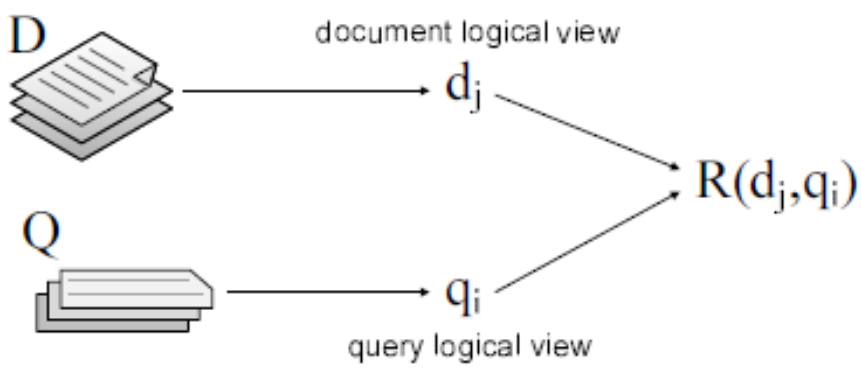
- D: document
- Q: query
- F: query/document representation function
- R(q_i, d_j): ranking function



Information Retrieval Model

<D, Q, F, $R(q_i, d_j)$>

- D: document
- Q: query
- F: query/document representation function ?
- $R(q_i, d_j)$: ranking function



Bag-of-words representation

Document/query as unordered collection of words

John likes to watch movies. Mary likes too. John also likes to watch football games.

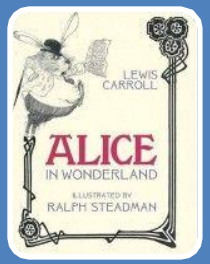
Bag-of-words representation

Document/query as unordered collection of words

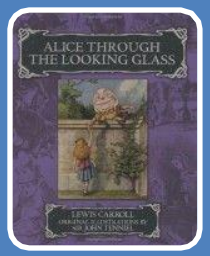
John likes to watch movies. Mary likes too. John also likes to watch football games.

{"John": 1, "likes": 2, "to": 3, "watch": 4, "movies": 5, "also": 6, "football": 7, "games": 8, "Mary": 9, "too": 10}

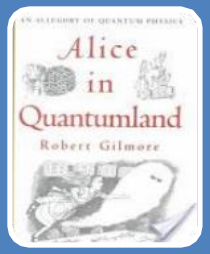
Term-Document matrix



- ‘It’s a friend of mine — a Cheshire Cat,’ said Alice: ‘allow me to introduce it.’
- ‘It’s the oldest rule in the book,’ said the King. ‘Then it ought to be Number One,’ said Alice.

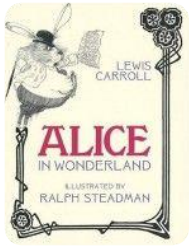


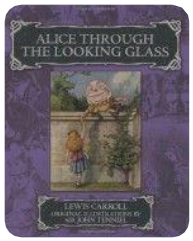
- Alice watched the White King as he slowly struggled up from bar to bar, till at last she said, ‘Why, you’ll be hours and hours getting to the table, at that rate.
- Alice looked round eagerly, and found that it was the Red Queen. ‘She’s grown a good deal!’ was her first remark.

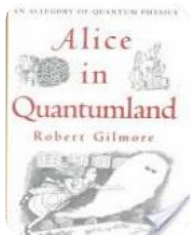


- In the pool of light was a billiards table, with two figures moving around it. Alice walked toward them, and as she approached they turned to look at her.
- Alice lay back, and closed her eyes. There was the Red Queen again, with that incessant grin. Or was it the Cheshire cat's grin?

Term-Document matrix







	D1	D2	D3
Cheshire Cat	1	0	1
Alice	2	2	2
book	1	0	0
King	1	1	0
table	0	1	1
Queen	0	1	1
grin	0	0	2

Term-Document matrix

	D1	D2	D3
Cheshire Cat	1	0	1
Alice	2	2	2
book	1	0	0
King	1	1	0
table	0	1	1
Queen	0	1	1
grin	0	0	2

Query: **Alice AND Queen**

Term-Document matrix

	D1	D2	D3	Q
Cheshire Cat	1	0	1	0
Alice	2	2	2	1
book	1	0	0	0
King	1	1	0	0
table	0	1	1	0
Queen	0	1	1	1
grin	0	0	2	0

Query: **Alice AND Queen**

Term-Document matrix

	D1	D2	D3	Q
Cheshire Cat	1	0	1	0
Alice	2	**2**	**2**	1
book	1	0	0	0
King	1	1	0	0
table	0	1	1	0
Queen	0	**1**	**1**	1
grin	0	0	2	0

Query: **Alice AND Queen**

Result: D2, D3

This is the representation that we adopt for information retrieval tasks

Inverted Index

Index

Page numbers in **bold face** refer to key term definitions
Page numbers in *italics* refer to images or diagrams
Page numbers followed by a "t" indicate a table

Inverted Index

Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

Inverted Index

Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

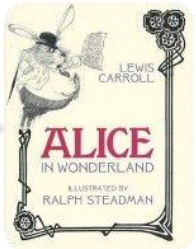
document:occurrences

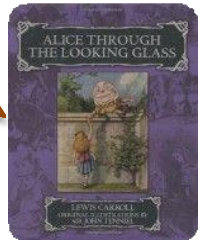
Inverted Index

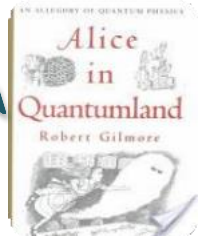
Dictionary	Posting List		
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

Inverted Index

Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

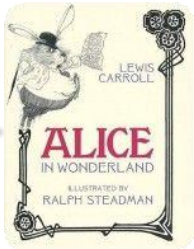


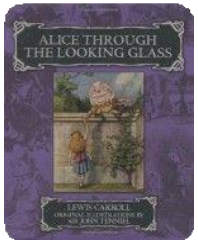


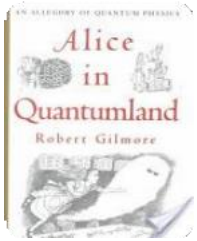


Inverted Index

Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

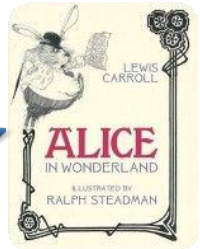


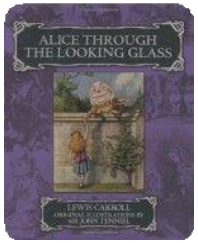


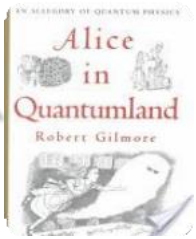


Inverted Index

Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

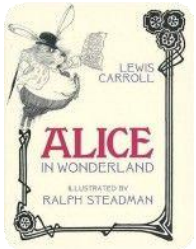


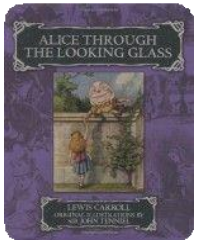


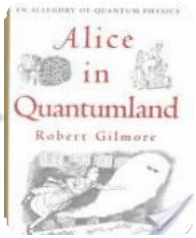


Inverted Index

Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

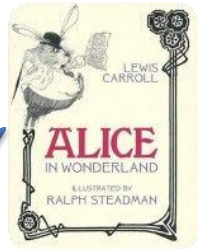


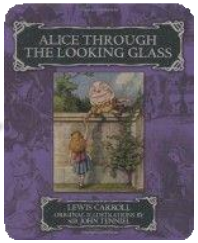


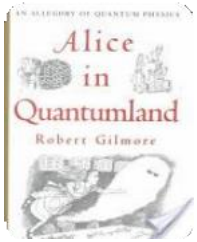


Inverted Index

Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

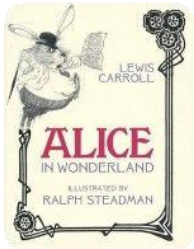


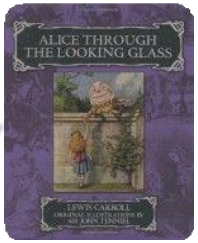


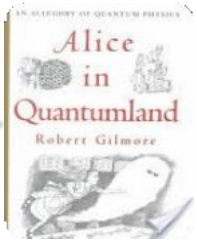


Inverted Index

Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

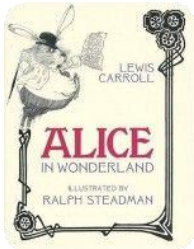


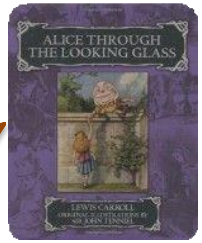


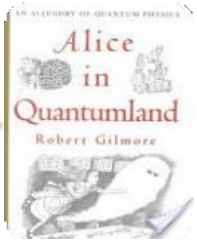


Inverted Index

Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			







Inverted Index

Dictionary	Posting List		
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

In more advanced models, we also take into account **the precise position** of the token in the document.

Inverted Index

Dictionary	Posting List		
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

Position List		
14, 49	1, 40	18, 34
33		

Term positions in D1

Inverted Index: query processing

Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

AND (intersection ∩)

<Alice>∩<King>

Inverted Index: query processing

Dictionary	
Alice	D1:2 D2:2 D3:2
…	
book	D1:1
…	
Cheshire Cat	D1:1 D3:1
…	
grin	D3:2
…	
King	D1:1 D2:1
…	
Queen	D2:1 D3:1
…	
table	D2:1 D3:1
…	

AND (intersection ∩)

D1:2 D2:2 D3:2

<Alice>∩<King>

D1:1 D2:1

Inverted Index: query processing

Dictionary	
Alice	D1:2 D2:2 D3:2
…	
book	D1:1
…	
Cheshire Cat	D1:1 D3:1
…	
grin	D3:2
…	
King	D1:1 D2:1
…	
Queen	D2:1 D3:1
…	
table	D2:1 D3:1
…	

AND (intersection ∩)

D1:2 D2:2 D3:2

<Alice>∩<King>

D1:1 D2:1

Alice AND King -> (D1, D2)

Inverted Index: query processing

Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

OR (union ∪)

<grin> ∪ <King>

Inverted Index: query processing

Dictionary	
Alice	D1:2 D2:2 D3:2
…	
book	D1:1
…	
Cheshire Cat	D1:1 D3:1
…	
grin	D3:2
…	
King	D1:1 D2:1
…	
Queen	D2:1 D3:1
…	
table	D2:1 D3:1
…	

OR (union ∪)

D3:2

<grin> ∪ <King>

D1:1 D2:1

grin OR King -> (D1, D2, D3)

Inverted Index: query processing

Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

NOT (complement \)

<Alice> \ <grin>

Inverted Index: query processing

Dictionary	
Alice	D1:2 D2:2 D3:2
…	
book	D1:1
…	
Cheshire Cat	D1:1 D3:1
…	
grin	D3:2
…	
King	D1:1 D2:1
…	
Queen	D2:1 D3:1
…	
table	D2:1 D3:1
…	

NOT (complement \)

D1:2 D2:2 D3:2

<Alice> \ <grin>

D3:2

Alice NOT grin -> (D1, D2)

Term-weight

- Measures the term relevance in a document
  - component value in the document representation
  - TF
    - TF (term frequency): term occurrences in the document

Term-weight

- Measures the term relevance in a document
  - component value in the document representation
  - TF
    - TF (term frequency): term occurrences in the document

Problem: popular (and not significant) terms, i.e., articles, adverbs, common nouns, etc. **are way too important. <u>Their importance should be lowered.</u>**

Term-weight

- Measures the term relevance in a document
  - component value in the document representation
  - TF*IDF
    - TF (term frequency): term occurrences in the document
    - IDF (inverse document frequency): inverse to the number of documents in which the term occurs

$$tf * idf(t,d) = tf(t,d) * \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

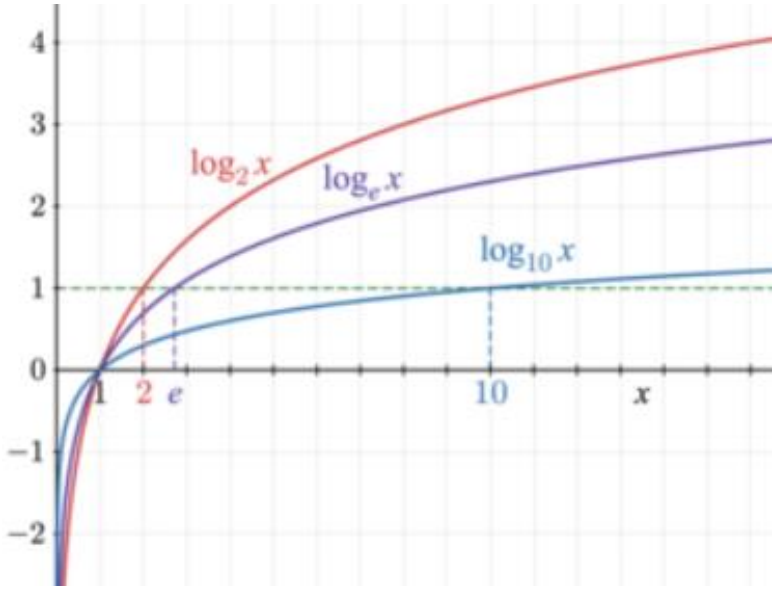
Term-weight

- Measures the term relevance in a document
  - component value in the document representation
  - TF*IDF
    - TF (term frequency): term occurrences in the document
    - IDF (inverse document frequency): inverse to the number of documents in which the term occurs

Number of documents in the collection

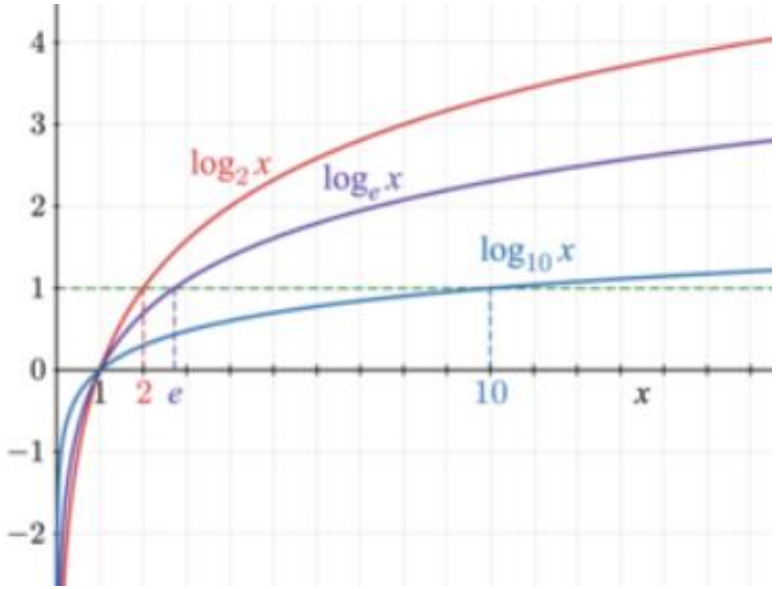
$$tf * idf(t,d) = tf(t,d) * \underbrace{\log \frac{|D|}{|\{d \in D : t \in d\}|}}_{\text{idf}}$$

Term-weight



$$tf * idf(t,d) = tf(t,d) * \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

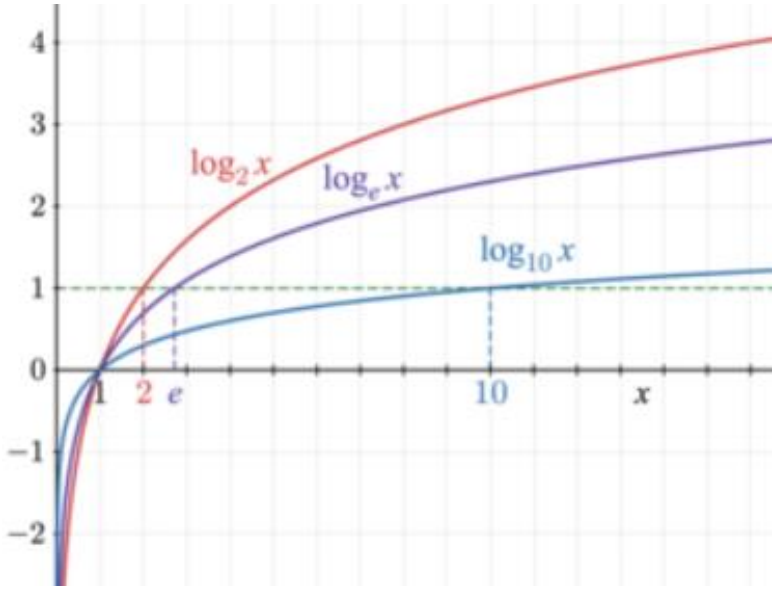
Term-weight



What happens if a term appears in many documents?

$$tf * idf(t,d) = tf(t,d) * \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Term-weight

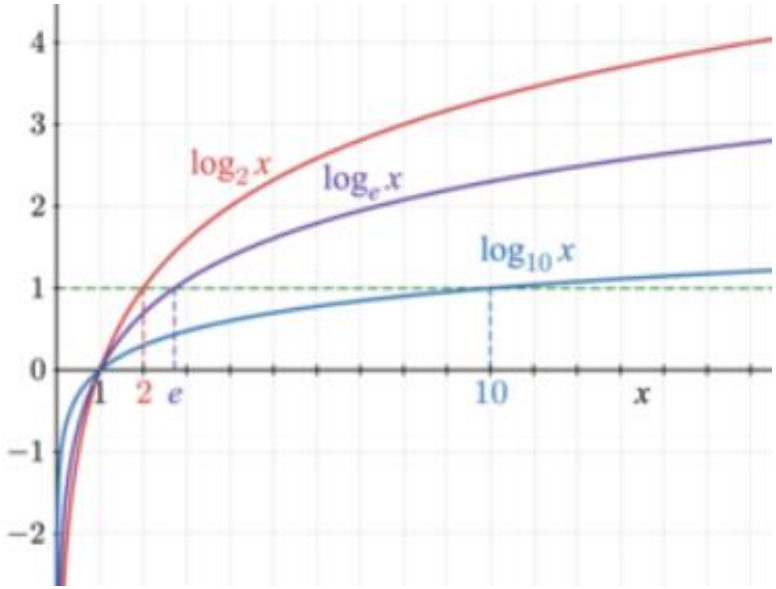


What happens if a term appears in many documents?

The ratio is close to 1, so the logarithm is close to 0

$$tf * idf(t,d) = tf(t,d) * \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Term-weight



What happens if a term appears in many documents?

The ratio is close to 1, so the logarithm is close to 0

Conversely, if a term appears in just a few document, **the ratio is high and the logarithm is high as well**

$$tf * idf(t,d) = tf(t,d) * \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

TF*IDF insights

- **increases proportionally** to the term frequency in the document
- **decreases** to the number of documents in which the term belongs
  - common words are generally more frequent in the collection
- **IDF** depends on the collection, **TF** on the document

Inverted index/TF*IDF

- TF: computed by term occurrences in the posting list
- IDF: computed by the posting list cardinality

Alice	D1:2	D2:2	D3:2

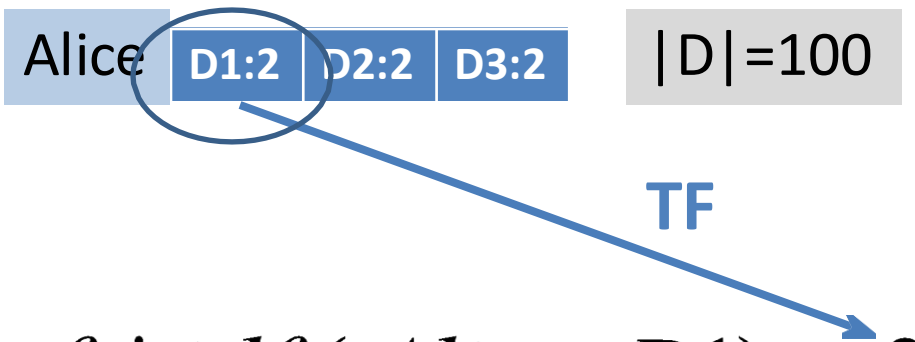
|D|=100

$$\log \frac{|D|}{|\{d \in D : t \in d\}|}$$

$$tf * idf(Alice, D1) =$$

Inverted index/TF*IDF

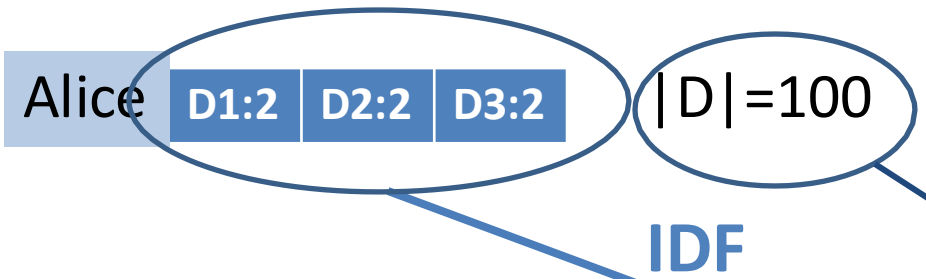
- TF: computed by term occurrences in the posting list
- IDF: computed by the posting list cardinality



$$tf * idf(Alice, D1) = 2 * \log \frac{100}{3}$$

Inverted index/TF*IDF

- TF: computed by term occurrences in the posting list
- IDF: computed by the posting list cardinality



$$tf * idf(Alice, D1) = 2 * \log \frac{100}{3}$$