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

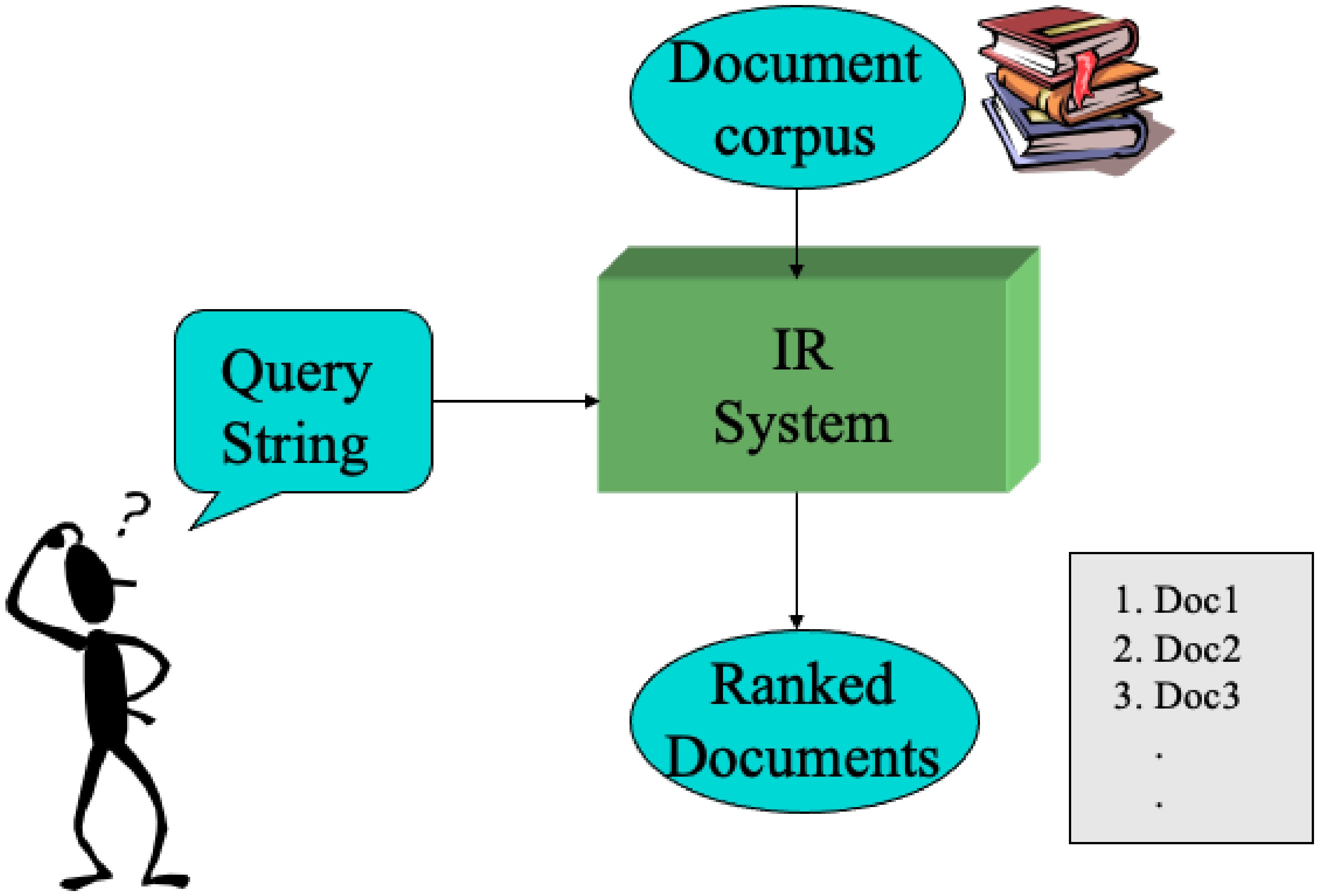
cataldo.musto@uniba.it

Credits: Marco Polignano

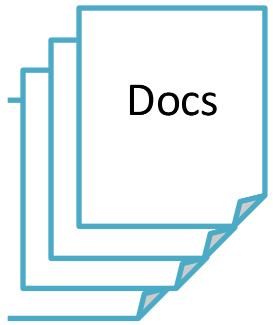
Recap

SEARCH ENGINE

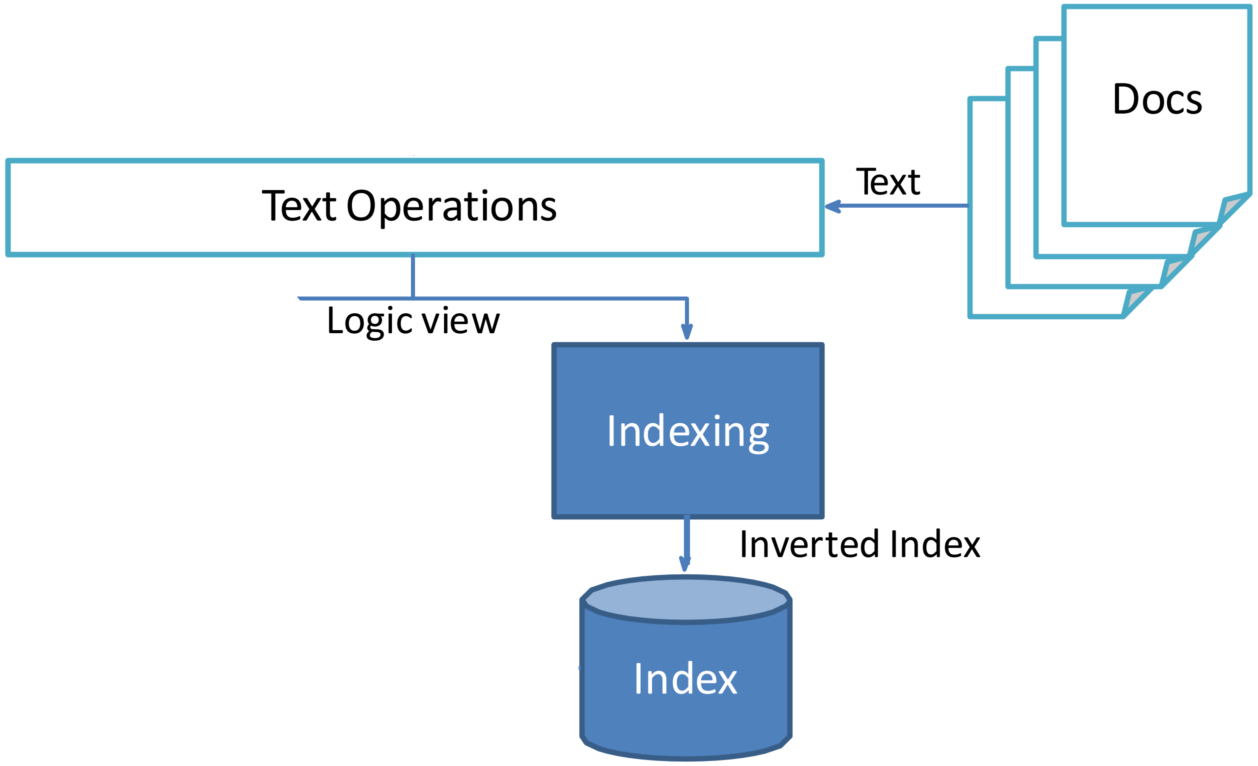
Information Retrieval Process (recap)



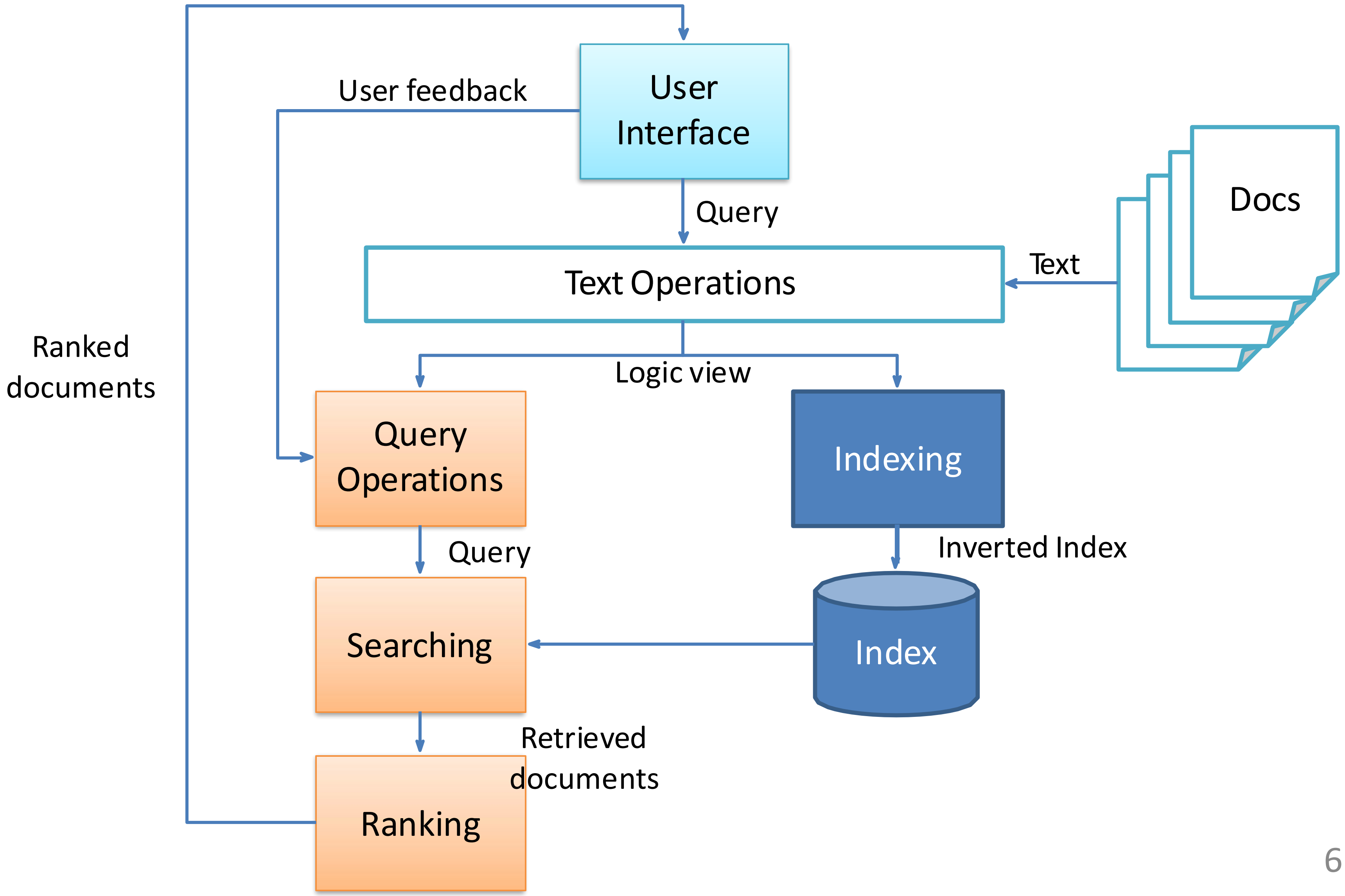
Information Retrieval Process (recap)



Information Retrieval Process (recap)



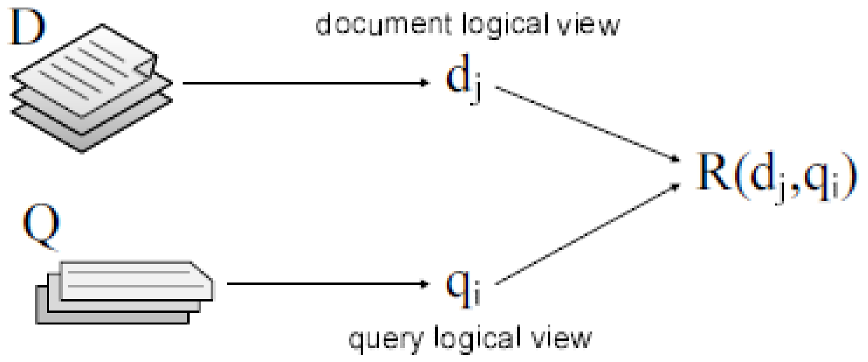
Information Retrieval Process (recap)



Information Retrieval Model

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

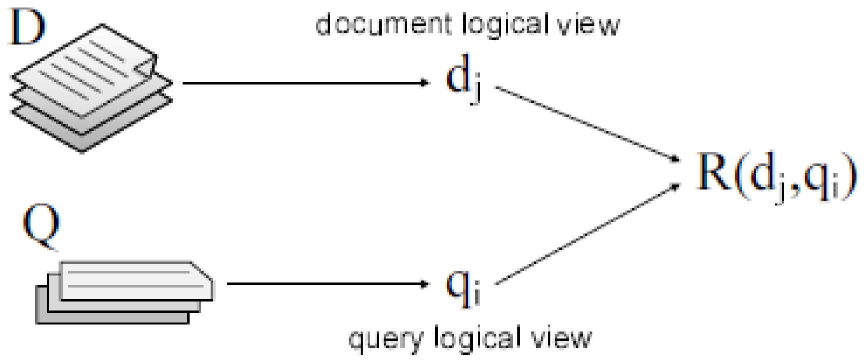
- D: document
- Q: query
- F: query/document representation function
- R(q_i, d_j): ranking function



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.

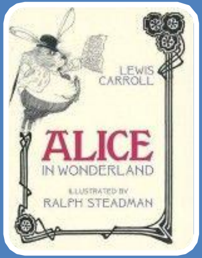
Bag-of-words representation

Document/query as unordered collection of words

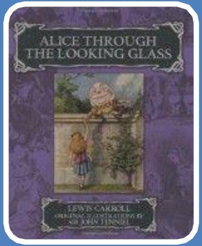
John likes to watch movies. Mary likes too. John also likes to watch football games.

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

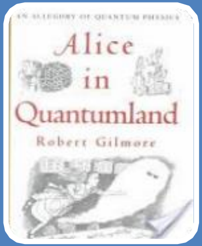
Term-Document matrix



- ‘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.

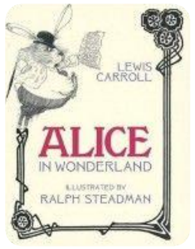


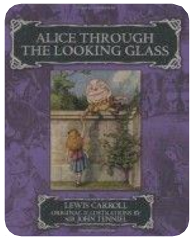
- Alice watched the White King as he slowly struggled up from bar to bar, till at last she said, ‘Why, you’ll be hours and hours getting to the table, at that rate.
- Alice looked round eagerly, and found that it was the Red Queen. ‘She’s grown a good deal!’ was her first remark.

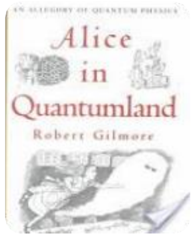


- In the pool of light was a billiards table, with two figures moving around it. Alice walked toward them, and as she approached they turned to look at her.
- Alice lay back, and closed her eyes. There was the Red Queen again, with that incessant grin. Or was it the Cheshire cat's grin?

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

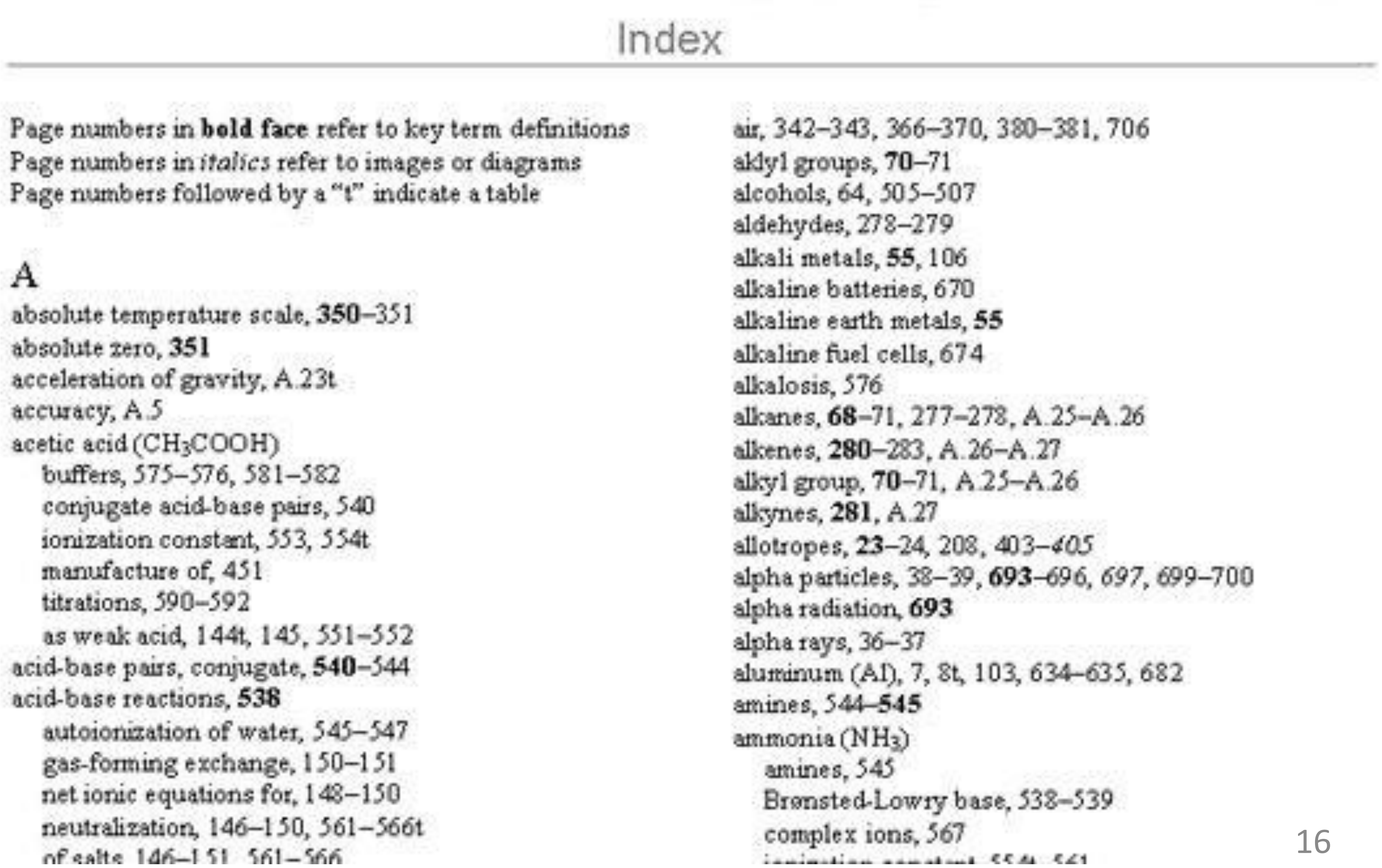
	D1	D2	D3	Q
Cheshire Cat	1	0	1	0
Alice	2	**2**	**2**	1
book	1	0	0	0
King	1	1	0	0
table	0	1	1	0
Queen	0	**1**	**1**	1
grin	0	0	2	0

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

A

absolute temperature scale, **350**–351
absolute zero, **351**
acceleration of gravity, A.23t
accuracy, A.5
acetic acid (CH_3COOH)
  buffers, 575–576, 581–582
  conjugate acid-base pairs, 540
  ionization constant, 553, 554t
  manufacture of, 451
  titrations, 590–592
  as weak acid, 144t, 145, 551–552
acid-base pairs, conjugate, **540**–544
acid-base reactions, **538**
  autoionization of water, 545–547
  gas-forming exchange, 150–151
  net ionic equations for, 148–150
  neutralization, 146–150, 561–566t
  of salts, 146–151, 561–566

air, 342–343, 366–370, 380–381, 706
aklyl groups, **70**–71
alcohols, 64, 505–507
aldehydes, 278–279
alkali metals, **55**, 106
alkaline batteries, 670
alkaline earth metals, **55**
alkaline fuel cells, 674
alkalosis, 576
alkanes, **68**–71, 277–278, A.25–A.26
alkenes, **280**–283, A.26–A.27
alkyl group, **70**–71, A.25–A.26
alkynes, **281**, A.27
allotropes, **23**–24, 208, 403–*405*
alpha particles, 38–39, **693**–*696*, *697*, 699–700
alpha radiation, **693**
alpha rays, 36–37
aluminum (Al), 7, 8t, 103, 634–635, 682
amines, 544–**545**
ammonia (NH_3)
  amines, 545
  Brønsted-Lowry base, 538–539
  complex ions, 567
  ionization constant, 554t, 561

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	
…			

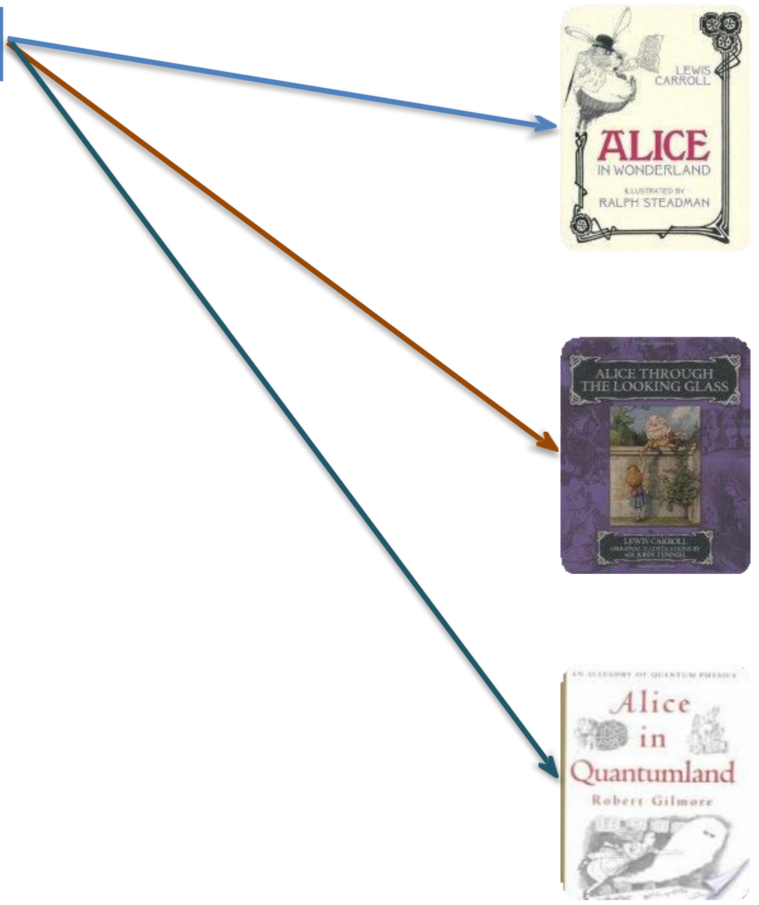
document:occurrences

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	
…			

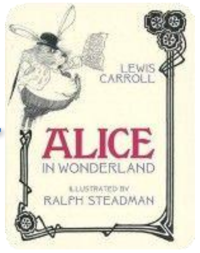
Inverted Index

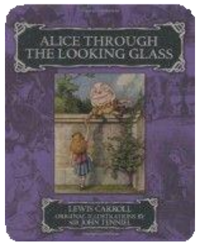
Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

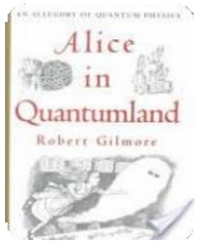


Inverted Index

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	
…			

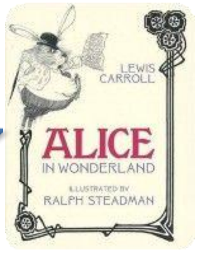


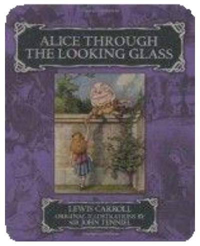


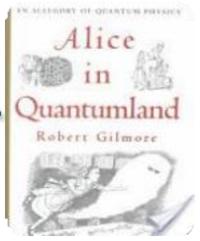


Inverted Index

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	
…			

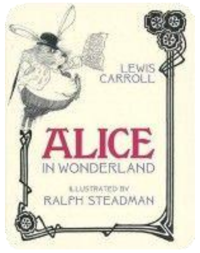


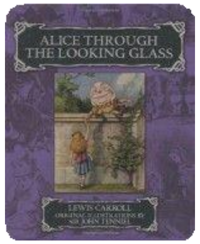


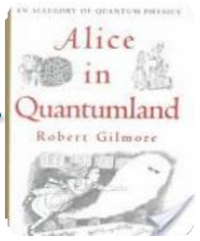


Inverted Index

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	
…			

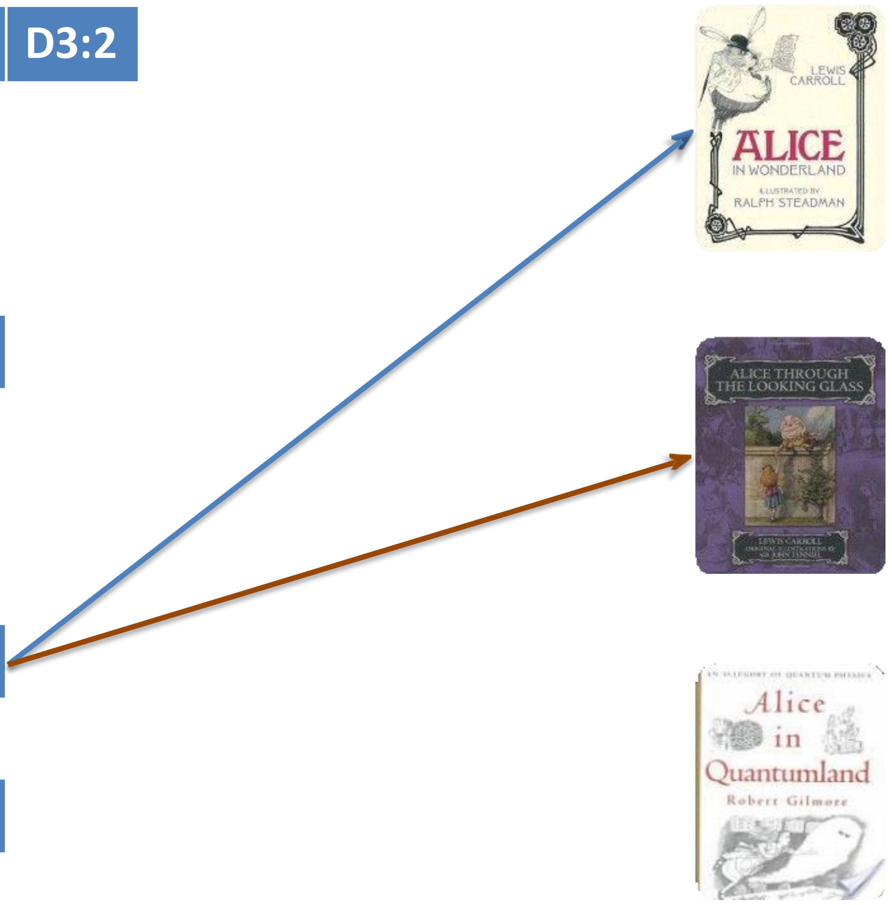






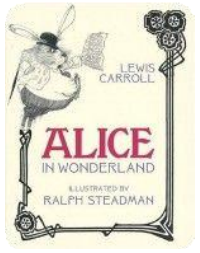
Inverted Index

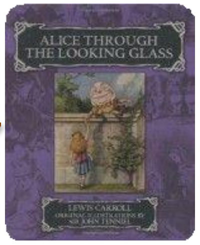
Dictionary			
Alice	D1:2	D2:2	D3:2
…			
book	D1:1		
…			
Cheshire Cat	D1:1	D3:1	
…			
grin	D3:2		
…			
King	D1:1	D2:1	
…			
Queen	D2:1	D3:1	
…			
table	D2:1	D3:1	
…			

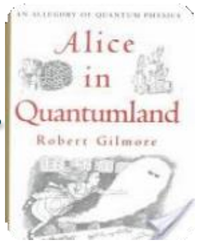


Inverted Index

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	
…			

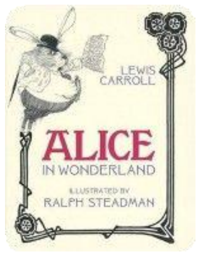


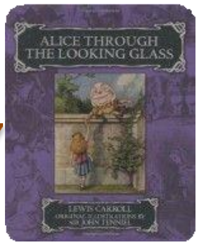


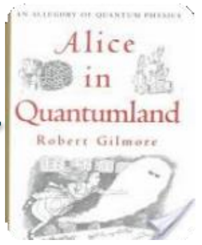


Inverted Index

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	
…			





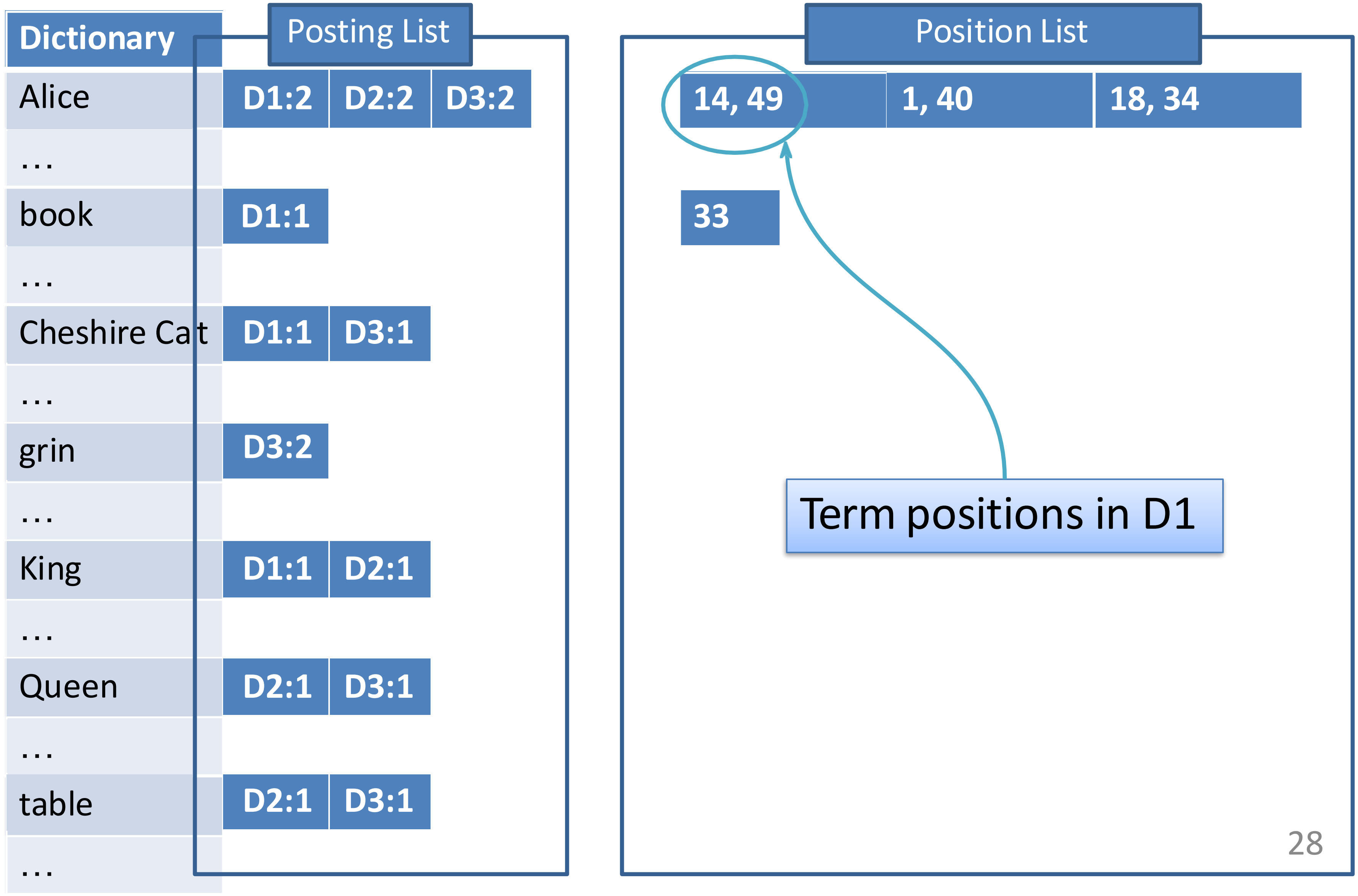


Inverted Index

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



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	
…			

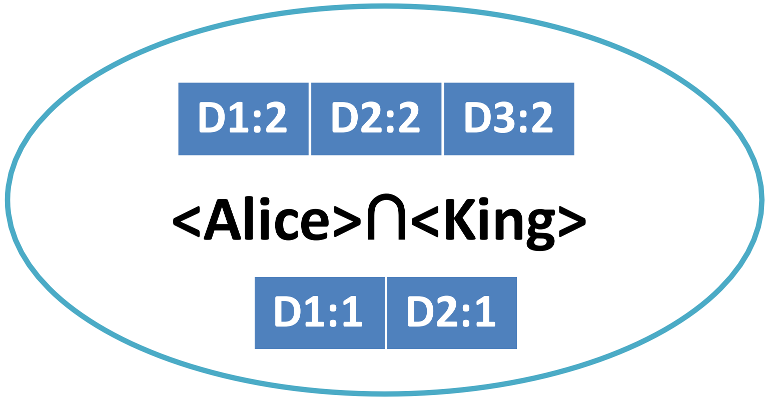
AND (intersection ∩)

<Alice>∩<King>

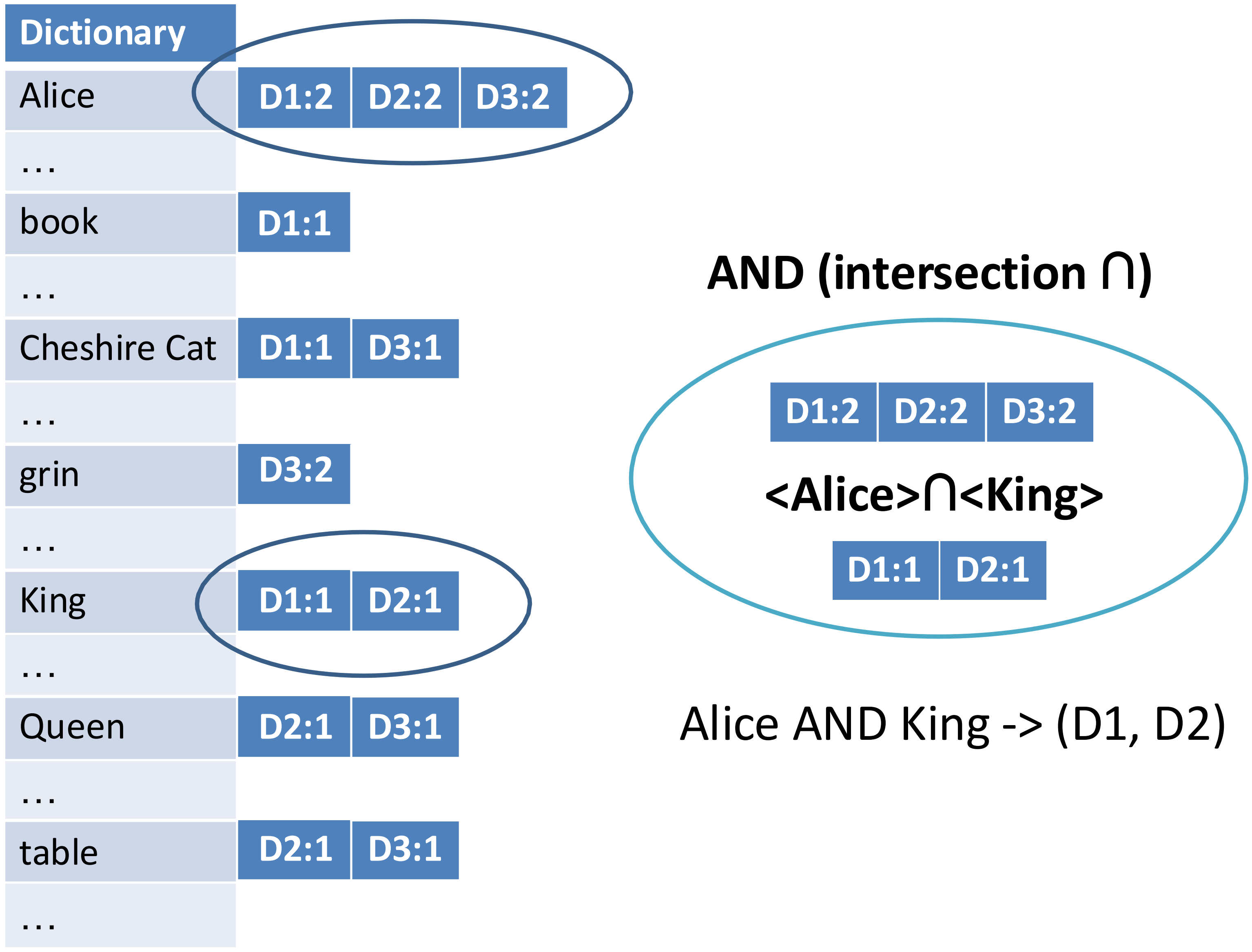
Inverted Index: query processing

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 ∩)



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 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	
…			

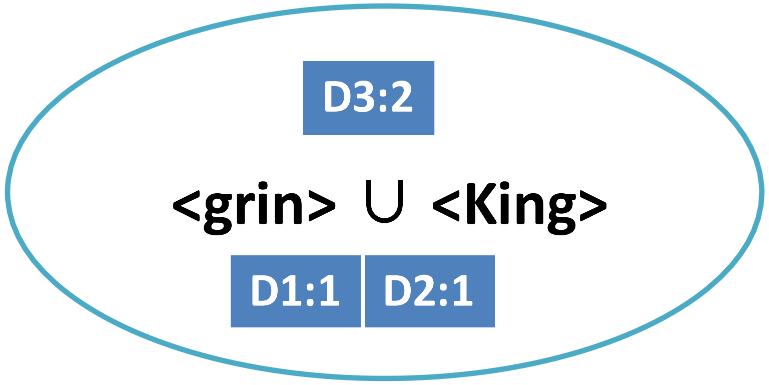
OR (union ∪)

<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 ∪)



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

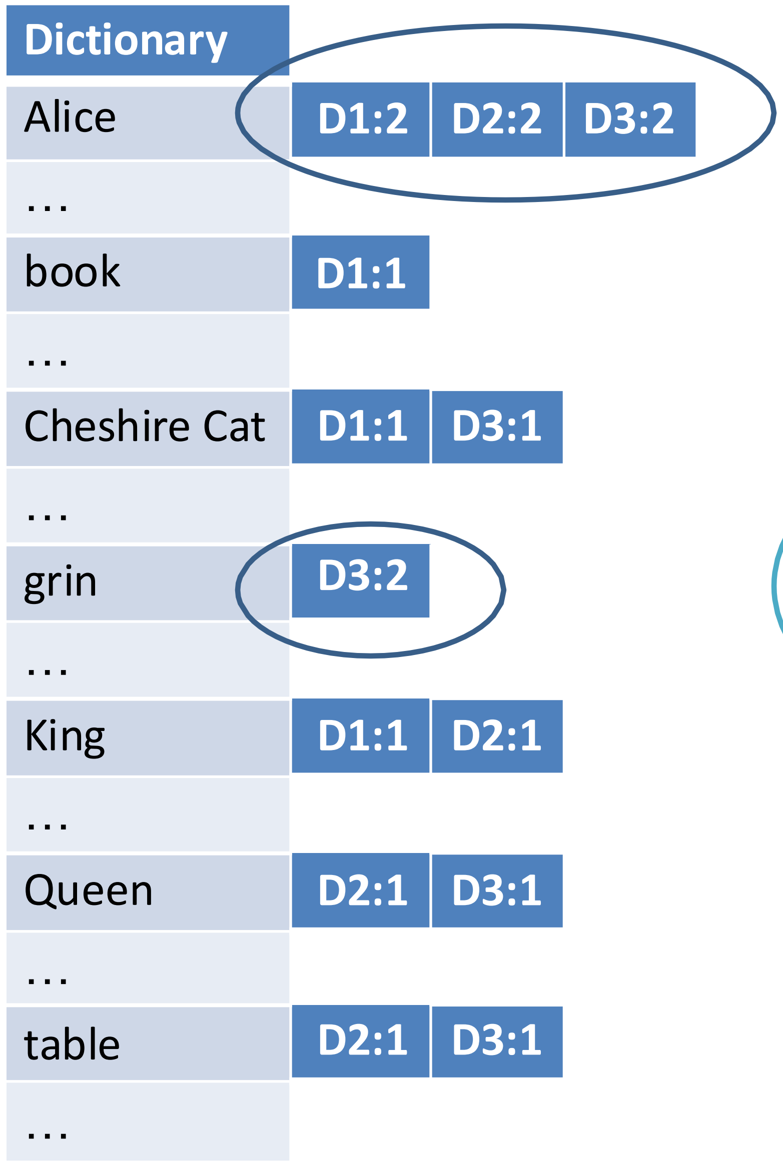
Inverted Index: query processing

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 \)

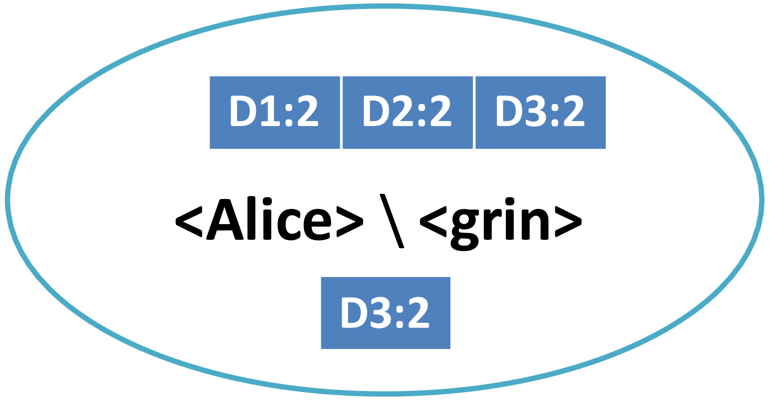
<Alice> \ <grin>

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 \\)



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

- Is there any problem with simple counting?

Term-weight

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

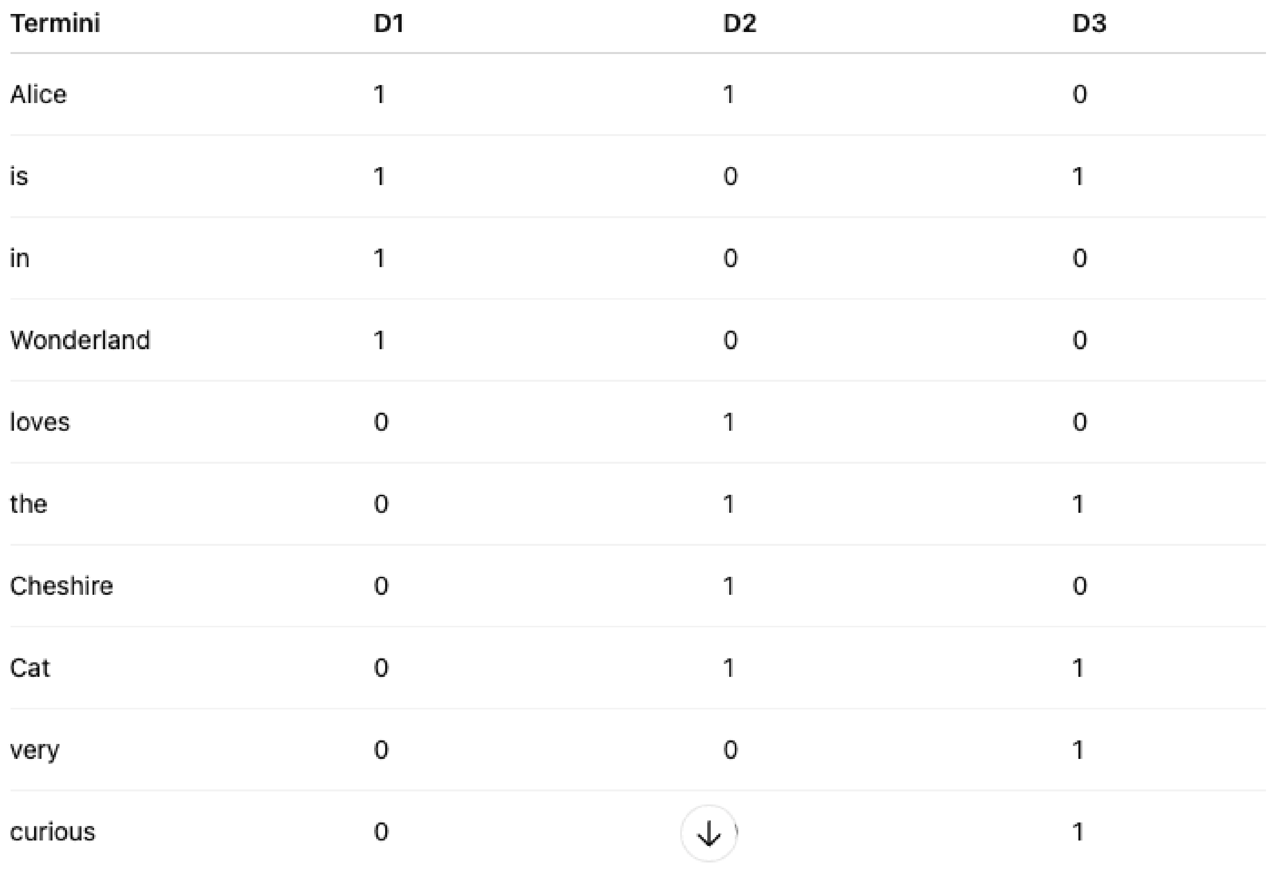
- Is there any problem with simple counting?
  - Yes, poorly informative words have high frequency

A toy example

- D1: "Alice is in Wonderland"
- D2: "Alice loves the Cheshire Cat"
- D3: "The Cat is very curious"

A toy example

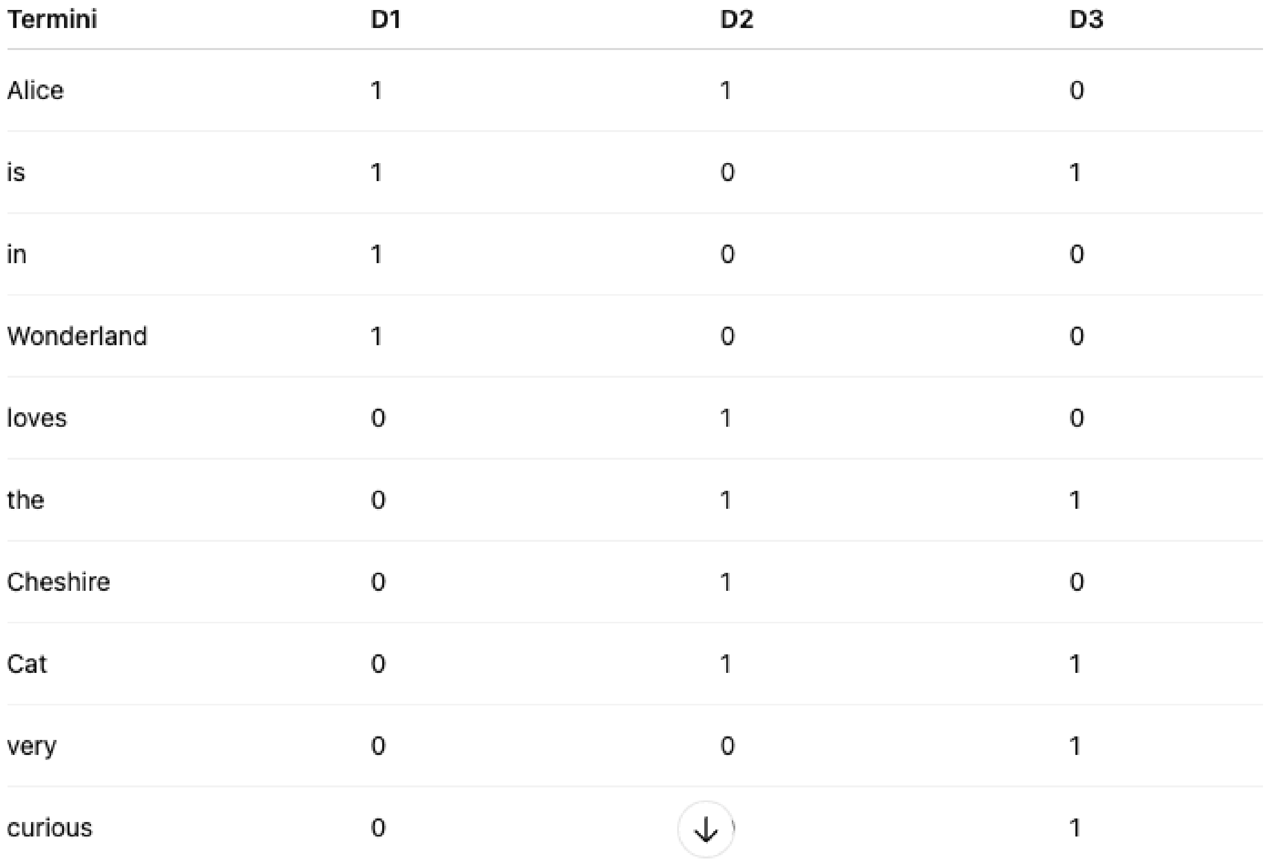
- D1: "Alice is in Wonderland"
- D2: "Alice loves the Cheshire Cat"
- D3: "The Cat is very curious"



Termini	D1	D2	D3
Alice	1	1	0
is	1	0	1
in	1	0	0
Wonderland	1	0	0
loves	0	1	0
the	0	1	1
Cheshire	0	1	0
Cat	0	1	1
very	0	0	1
curious	0	[illegible]	1

A toy example

- D1: "Alice is in Wonderland"
- D2: "Alice loves the Cheshire Cat"
- D3: "The Cat is very curious"



Termini	D1	D2	D3
Alice	1	1	0
is	1	0	1
in	1	0	0
Wonderland	1	0	0
loves	0	1	0
the	0	1	1
Cheshire	0	1	0
Cat	0	1	1
very	0	0	1
curious	0	[illegible]	1

It is hard to catch the real ‘meaning’ of a document.
Possible solutions: (a) remove noise; (b) give more importance to more relevant terms

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

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

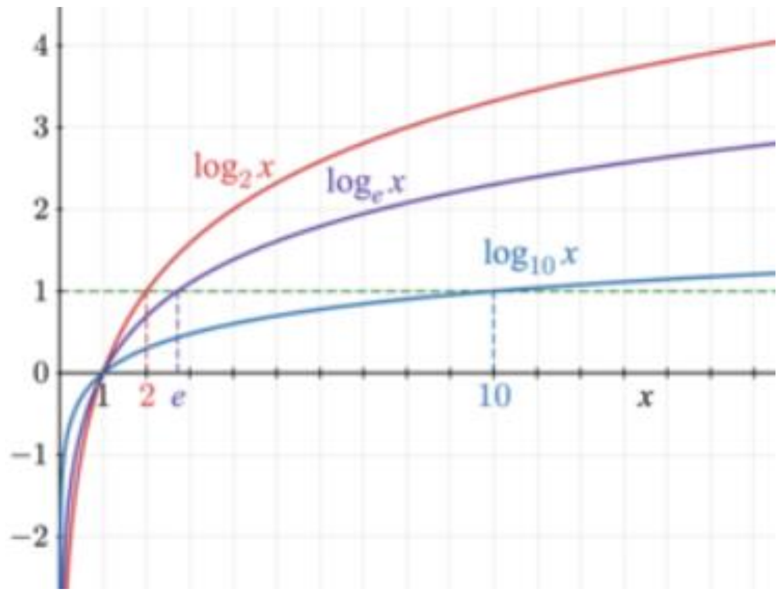
Term-weight

- 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

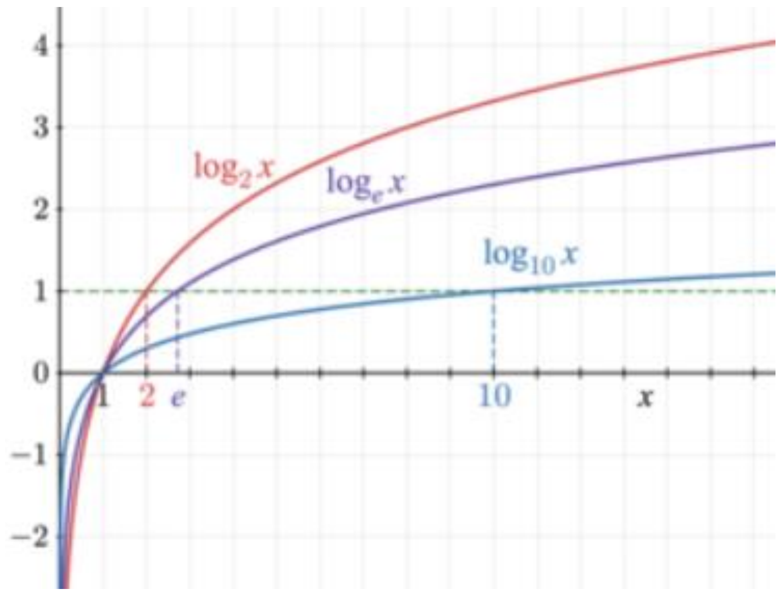
$$tf*idf(t,d) = tf(t,d) * \underbrace{\log \frac{|D|}{|\{d \in D : t \in d\}|}}_{\text{idf}}$$

Term-weight



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

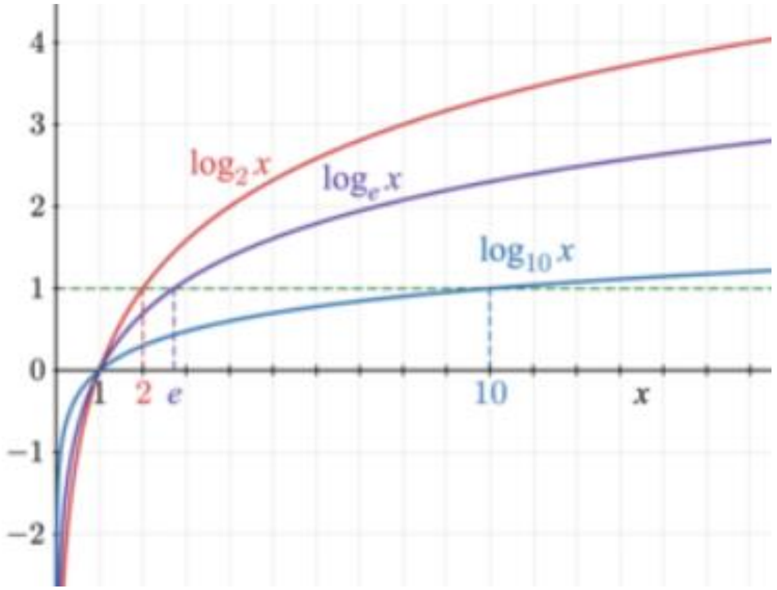
Term-weight



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

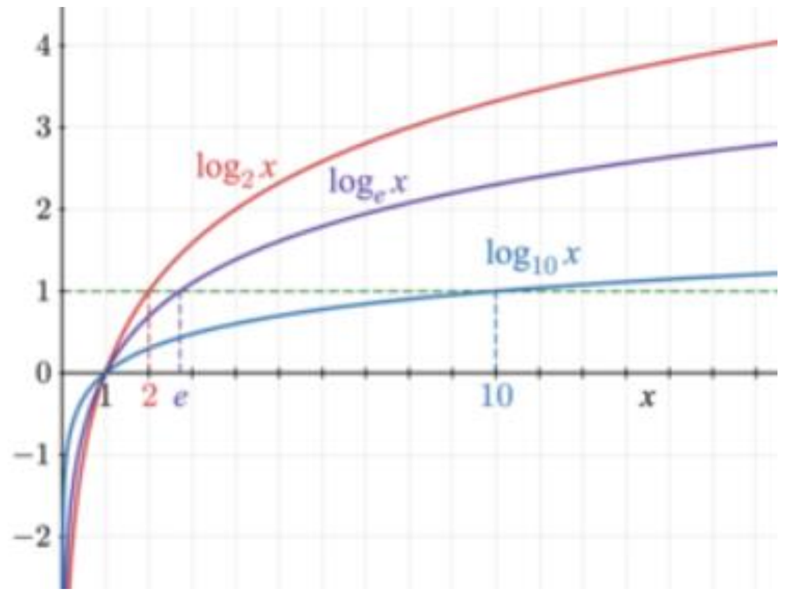


What happens if a term appears in many documents?

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

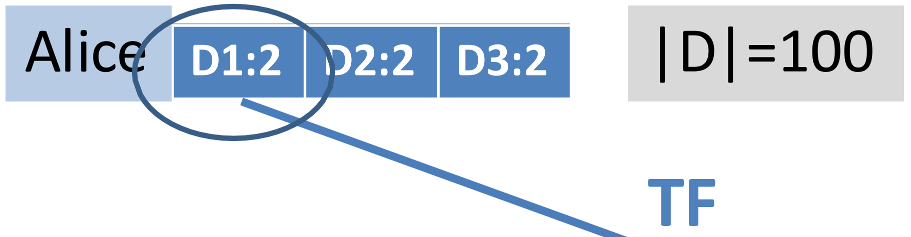
|D|=100

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

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

Inverted index/TF*IDF

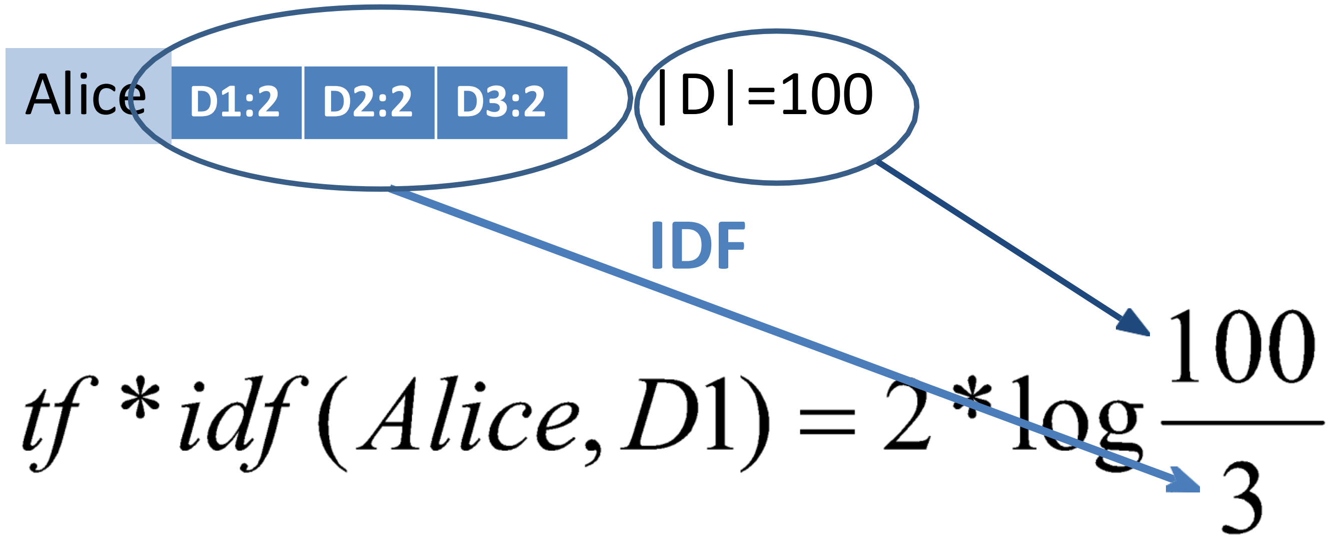
- TF: computed by term occurrences in the posting list
- IDF: computed by the posting list cardinality



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

Inverted index/TF*IDF

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



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