Text Mining (2019)

Information Retrieval

Marco Kuhlmann
Department of Computer and Information Science

LINKÖPING UNIVERSITY

This lecture

- Introduction to information retrieval
- Index construction
- Ranked retrieval
- The vector space model
- Evaluation of information retrieval systems
- Introduction to the lab

Introduction to information retrieval

Information Retrieval

Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

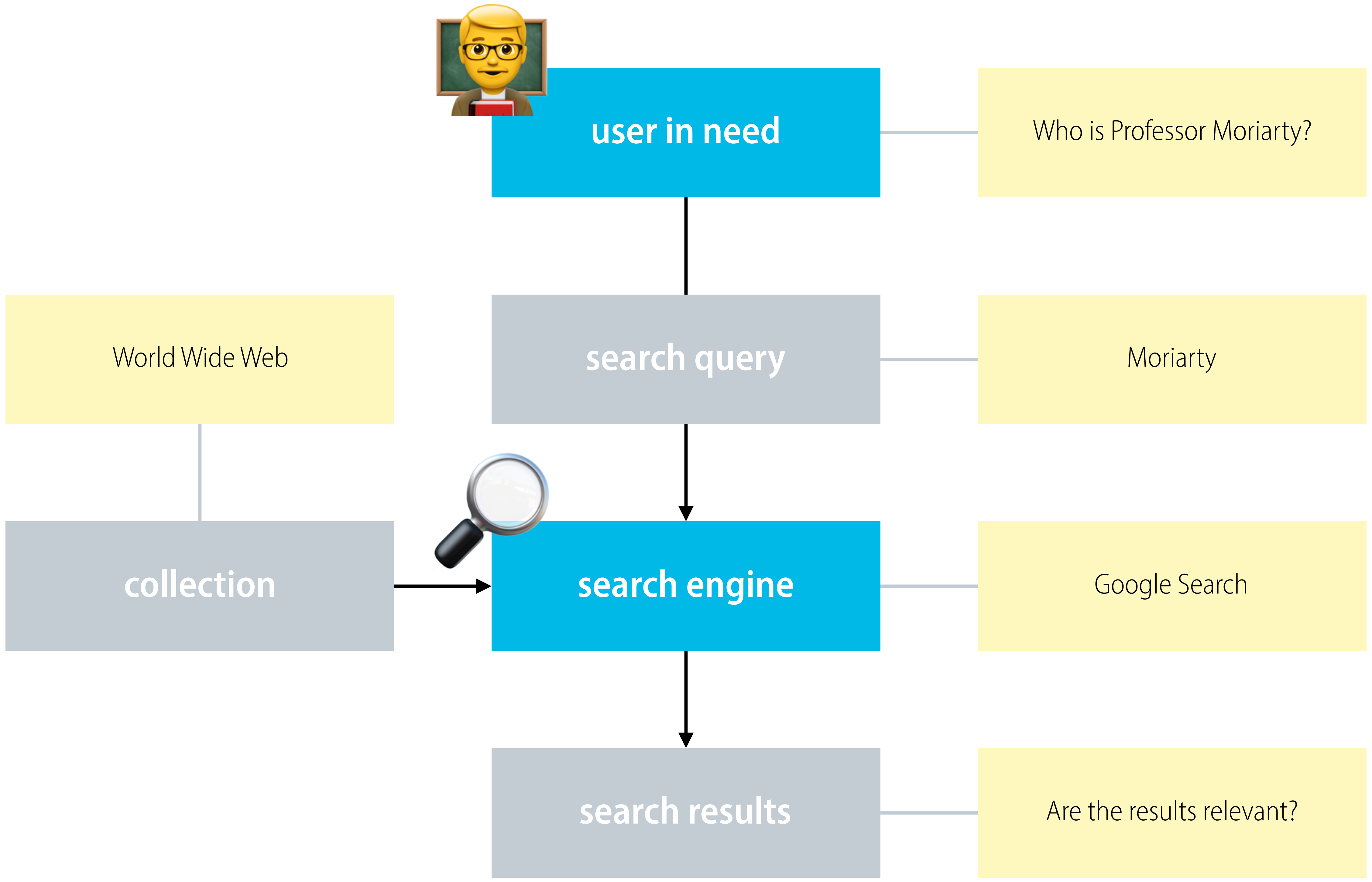
Manning, Raghavan, and Schütze (2008), p. 1

The classic search model

- To communicate her/his information need to an IR system, the user formulates a **search query**.
- The objective of the IR system is to find documents in the collection that ‘match’ the search query.
- A good IR system finds documents that are also **relevant** for the user’s information need.

  implications for evaluation

The classic search model



Adapted from Manning, Raghavan, and Schütze (2008)

Linear search

Which Sherlock Holmes short stories contain the word ‘Moriarty’?

- We could search through the texts exhaustively, for example by using a tool such as *grep*.
- But what if we want to only match those stories that also contain the word ‘Lestrade’?
- And what if, additionally, we want to exclude those stories that contain the word ‘Adair’?

Boolean retrieval

- The **Boolean retrieval model** is perhaps the simplest and historically the most widely used model in Information Retrieval.
- In this model, a query is a normal-form Boolean expression whose atoms correspond to **terms** (‘words’), and is evaluated on documents *d*. An atom *t* is true for *d* if and only if *t* occurs in *d*.

  Example: Moriarty AND Lestrade AND NOT Adair
- While modern IR uses other models, data structures, algorithms, and terminology from Boolean retrieval are still relevant.

Term–document matrix

In the **term–document matrix**,

- the rows correspond to search terms t
- the columns correspond to documents d
- a cell (t, d) is 1 if t occurs in d, and 0 otherwise

Term–document matrix

	Scandal in Bohemia	Final problem	Empty house	Norwood builder	Dancing men	Retired colourman
Adair	0	0	1	0	0	0
Adler	1	0	0	0	0	0
Lestrade	0	0	1	1	0	0
Moriarty	0	1	1	1	0	0

Moriarty AND Lestrade AND NOT Adair

	Scandal in Bohemia	Final problem	Empty house	Norwood builder	Dancing men	Retired colourman
Adair	0	0	1	0	0	0
Adler	1	0	0	0	0	0
Lestrade	0	0	1	1	0	0
Moriarty	0	1	1	1	0	0

011100 AND 001100 AND NOT 001000

Moriarty AND Lestrade AND NOT Adair

	Scandal in Bohemia	Final problem	Empty house	Norwood builder	Dancing men	Retired colourman
Adair	0	0	1	0	0	0
Adler	1	0	0	0	0	0
Lestrade	0	0	1	1	0	0
Moriarty	0	1	1	1	0	0

000100

Term–document matrices are sparse

The term–document matrix is not a practical data structure:

- Consider a medium-sized collection with 1,000,000 documents.
- Suppose that the search vocabulary has 500,000 distinct terms.
- This yields a matrix with 500,000,000,000 entries (62,5 GB).

Importantly though, almost all of the entries are zero!

Example due to Chris Manning

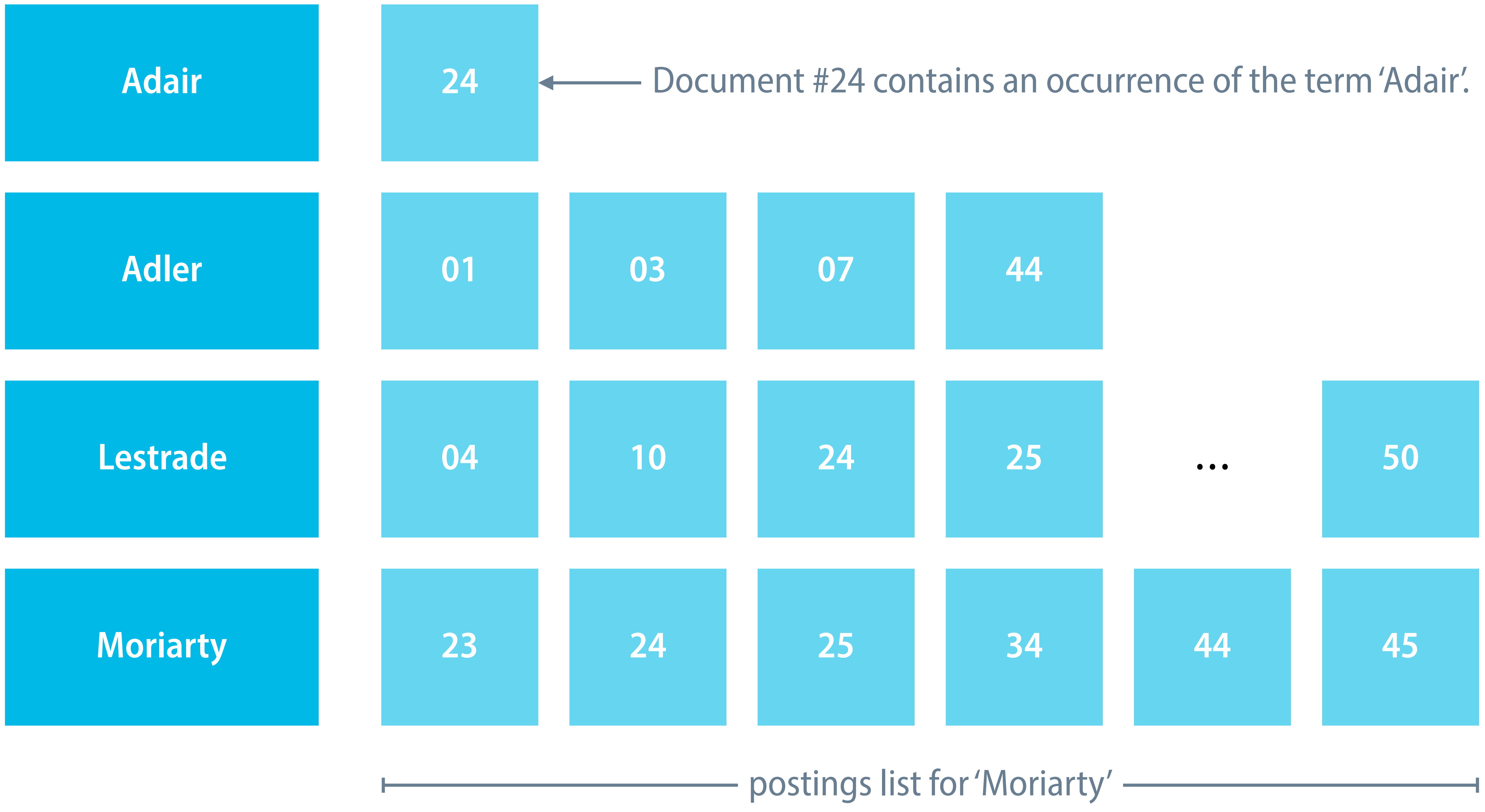
Inverted index

The **inverted index** is a key–value mapping in which

- the keys are search terms *t*
- the values are sorted lists of document identifiers (ids)
- the list for *t* identifies those documents *d* that contain *t*

The lists of document ids are traditionally called **postings lists**.

Inverted index

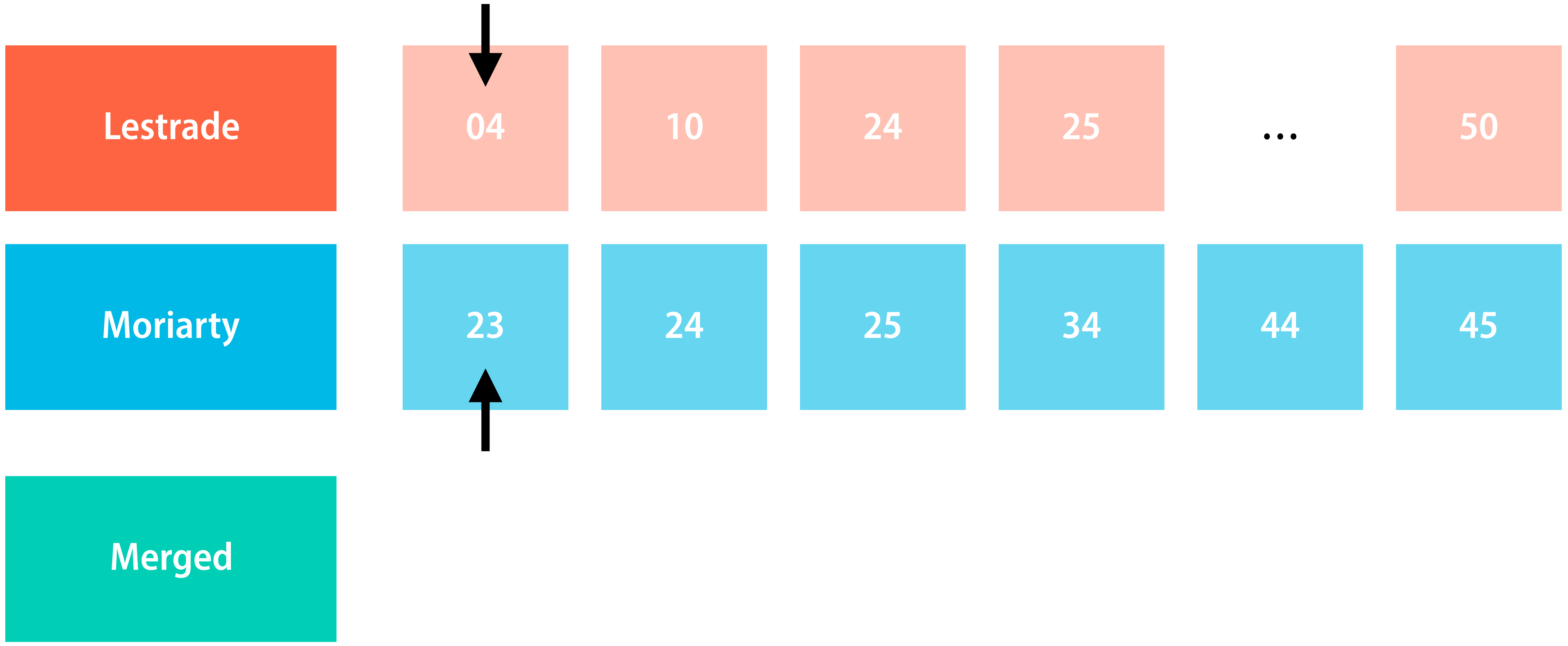

Adair
24
Document #24 contains an occurrence of the term 'Adair'.
Adler
01
03
07
44
Lestrade
04
10
24
25
...
50
Moriarty
23
24
25
34
44
45
postings list for 'Moriarty'

Boolean query processing

- Given an inverted index as the basic data structure, Boolean queries can be processed efficiently.
- Crucial in the processing of Boolean queries are algorithms for 'merging' the results from simpler queries.

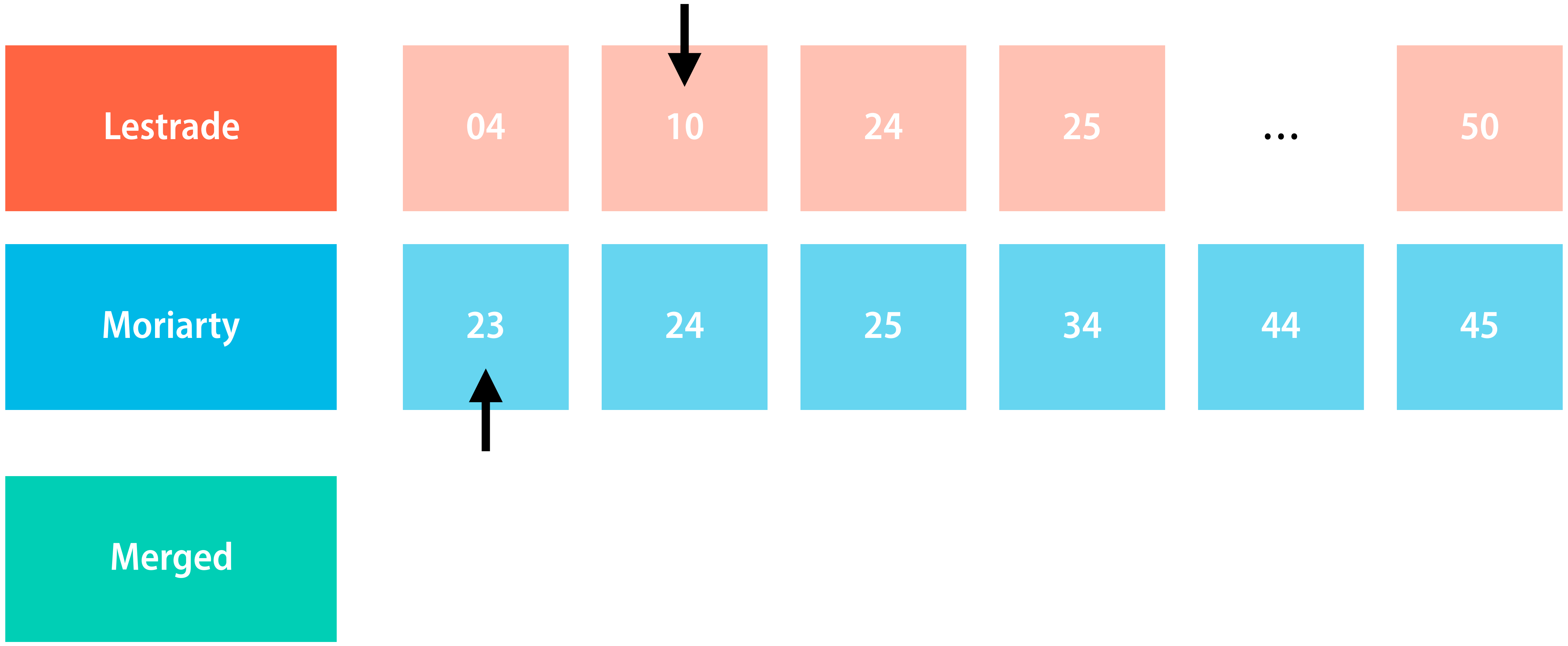
Processing an AND query

Lestrade AND Moriarty



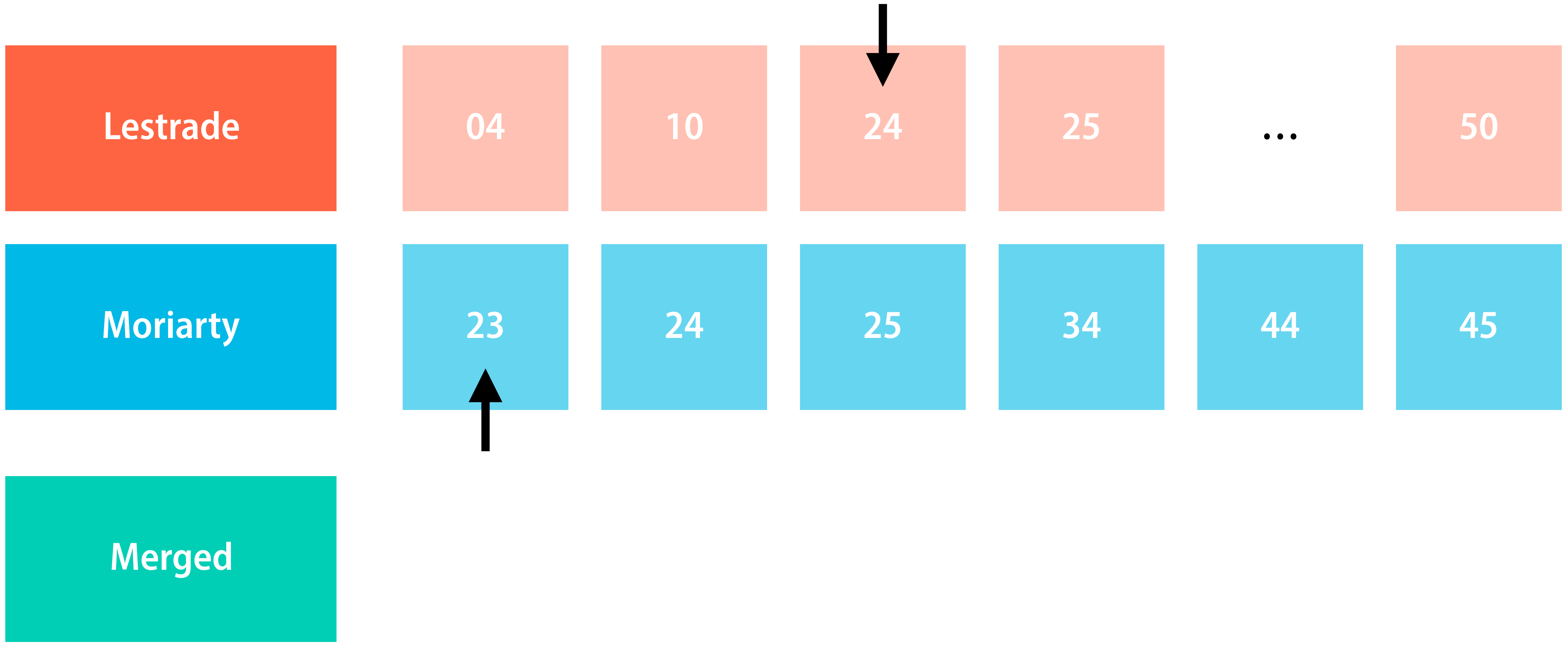
Processing an AND query

Lestrade AND Moriarty



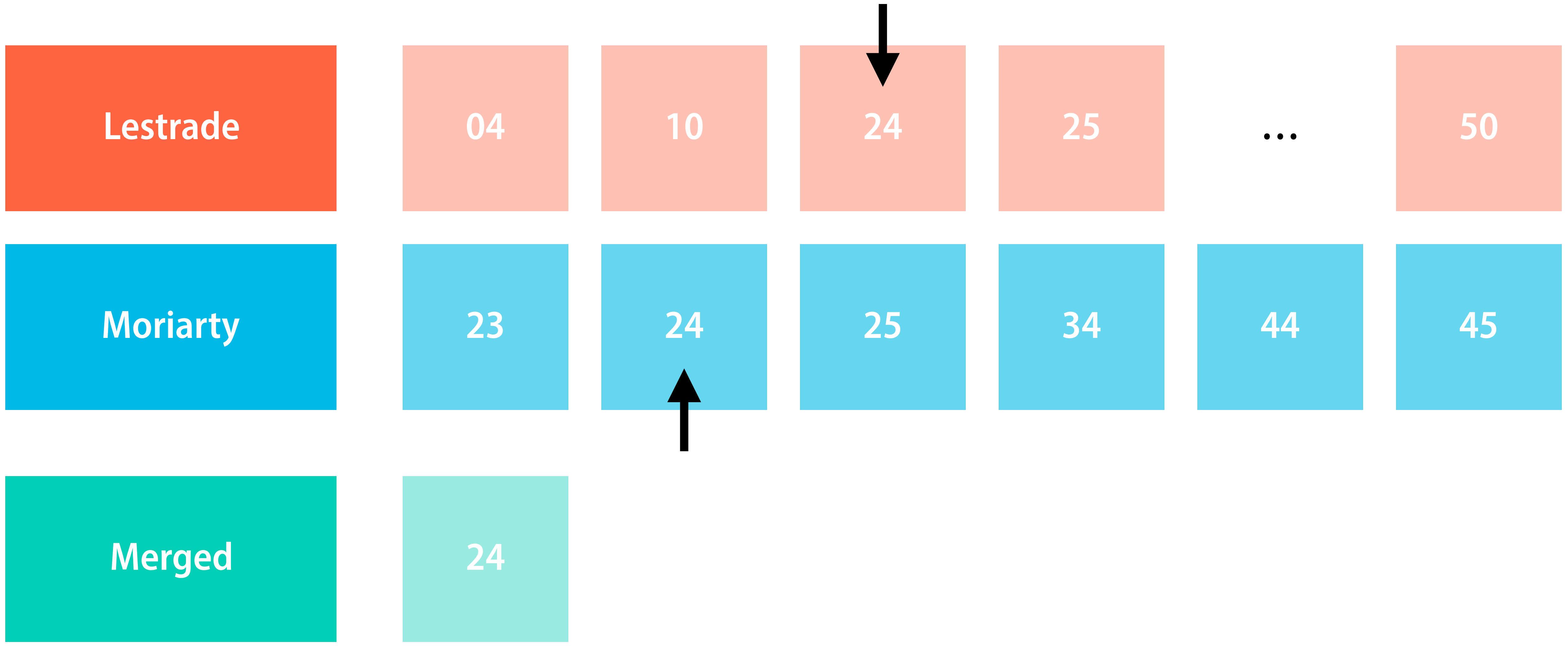
Processing an AND query

Lestrade AND Moriarty



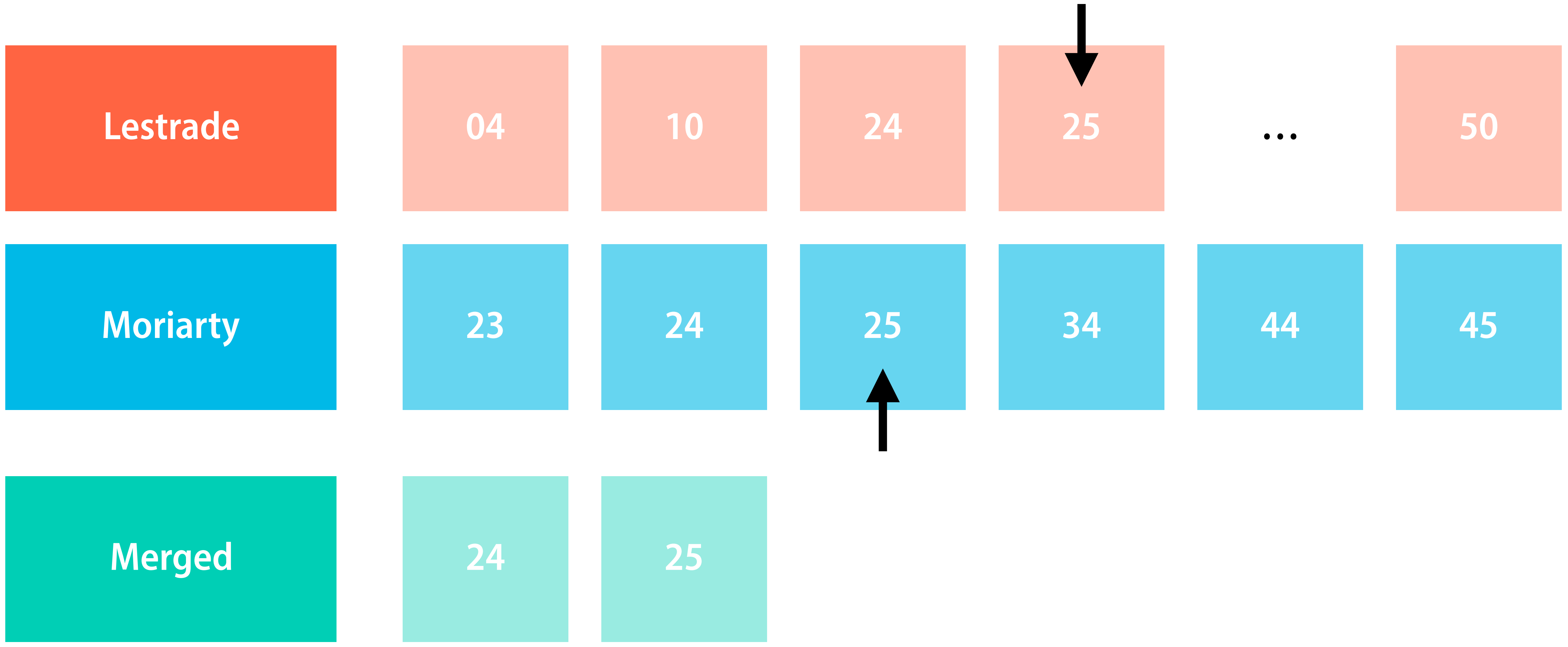
Processing an AND query

Lestrade AND Moriarty



Processing an AND query

Lestrade AND Moriarty



Important concepts

- search query
- relevance
- Boolean retrieval
- term
- term–document matrix
- inverted index

This lecture

- Introduction to information retrieval
- Index construction
- Ranked retrieval
- The vector space model
- Evaluation of information retrieval systems
- Introduction to the lab

Index construction

Index construction

The major steps in index construction:

1. Collect the documents to be indexed.
2. Tokenize the text.
3. Do linguistic preprocessing of tokens.
4. Index the documents that each term occurs in.

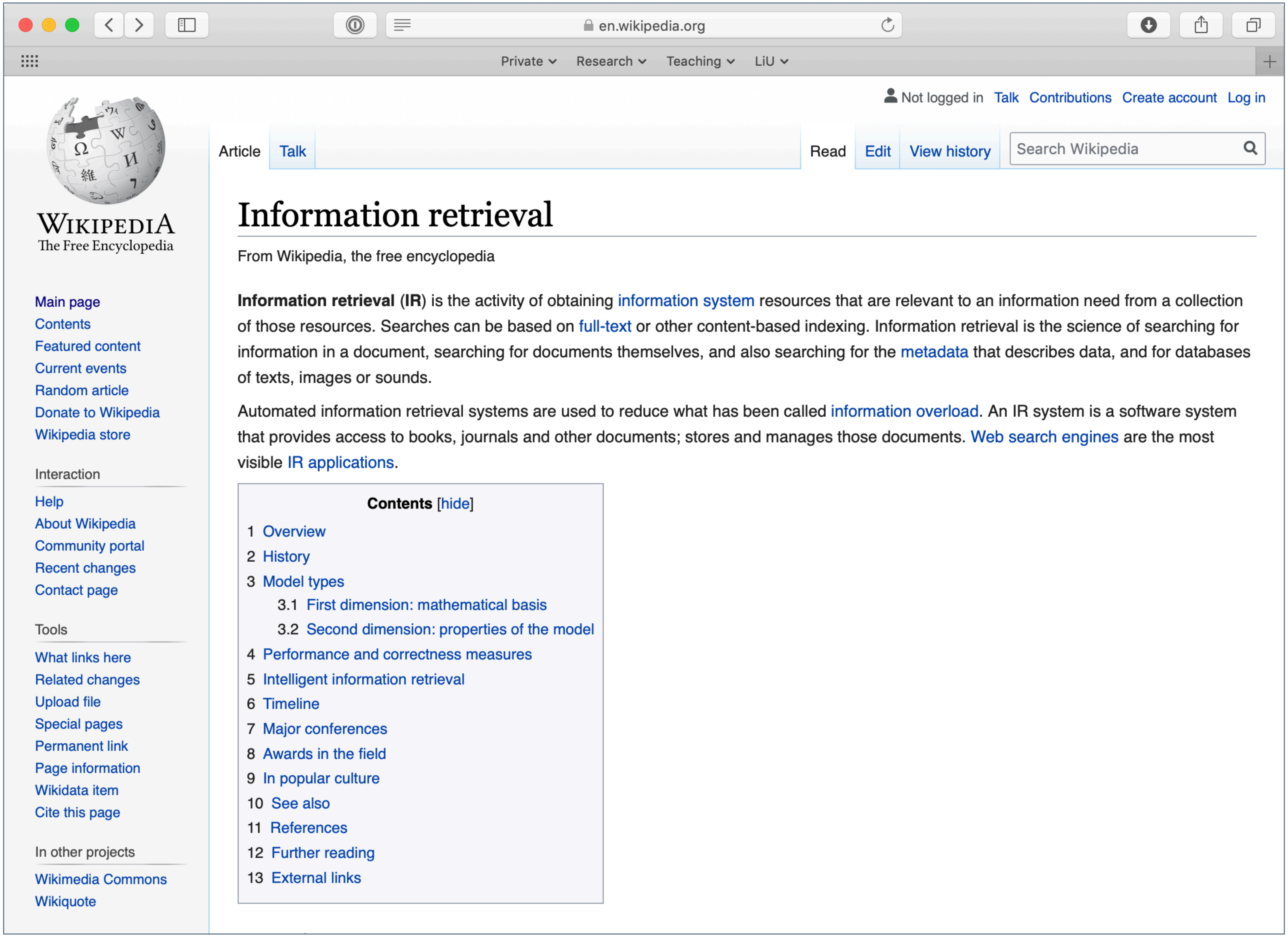
Manning, Raghavan, and Schütze (2008), p. 6 ff.

Web scraping

- **Web scraping** is to automatically extract data from websites.
- Scraping a web page involves fetching it (often using a web crawler), parsing it, and extracting data from it.

  BeautifulSoup, Scrapy
- Web scraping may violate the terms of use of some websites, and may also constitute copyright infringement.

  Counter measures include the blocking of the scraper's IP address!



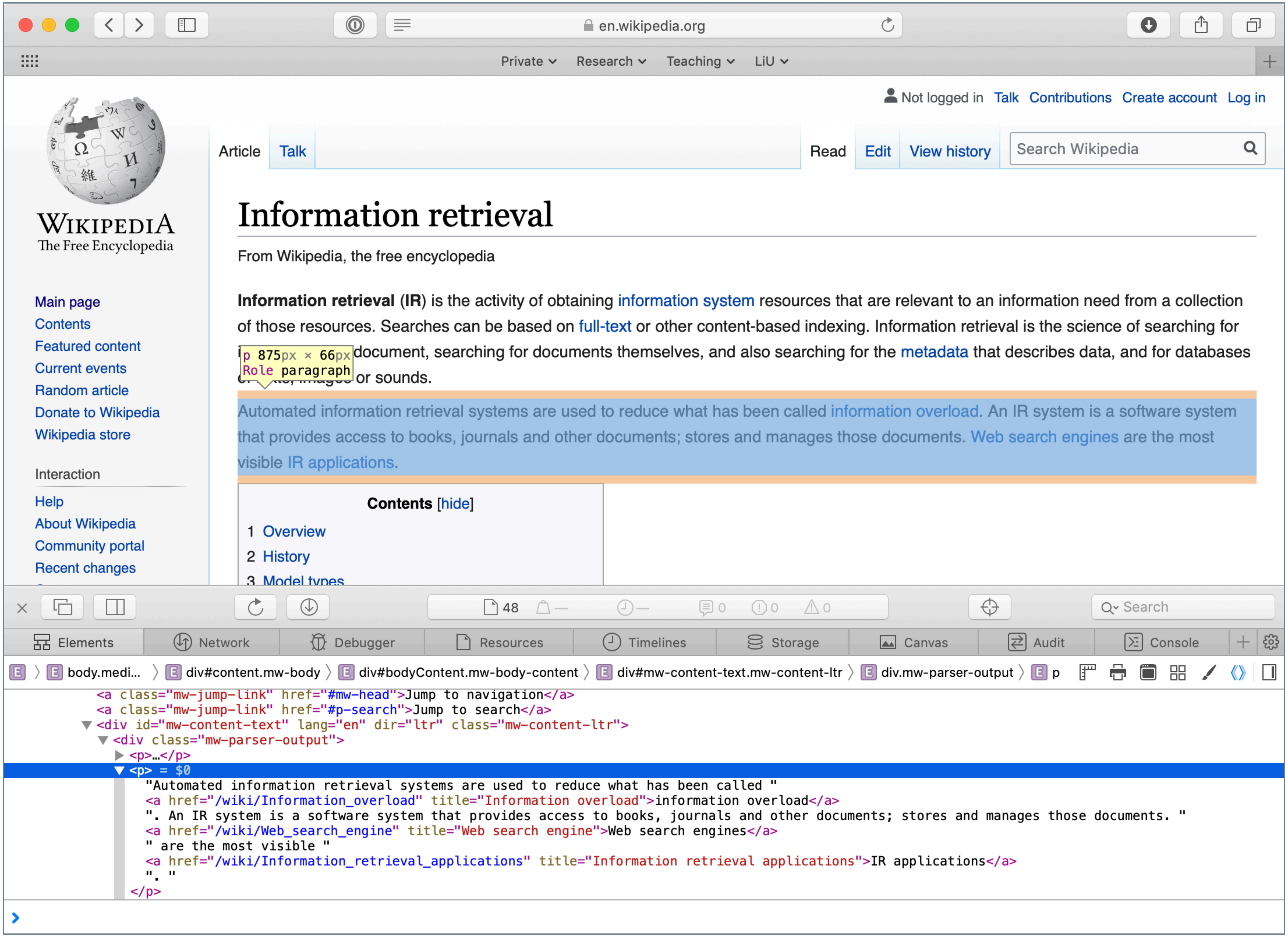
Information retrieval

From Wikipedia, the free encyclopedia

Information retrieval (**IR**) is the activity of obtaining information system resources that are relevant to an information need from a collection of those resources. Searches can be based on full-text or other content-based indexing. Information retrieval is the science of searching for information in a document, searching for documents themselves, and also searching for the metadata that describes data, and for databases of texts, images or sounds.

Automated information retrieval systems are used to reduce what has been called information overload. An IR system is a software system that provides access to books, journals and other documents; stores and manages those documents. Web search engines are the most visible IR applications.

Contents [hide]



Information retrieval

From Wikipedia, the free encyclopedia

Information retrieval (**IR**) is the activity of obtaining information system resources that are relevant to an information need from a collection of those resources. Searches can be based on full-text or other content-based indexing. Information retrieval is the science of searching for i… document, searching for documents themselves, and also searching for the metadata that describes data, and for databases o… or sounds.

Automated information retrieval systems are used to reduce what has been called information overload. An IR system is a software system that provides access to books, journals and other documents; stores and manages those documents. Web search engines are the most visible IR applications.

Contents [hide]

1 Overview
2 History
3 Model types

```
<a class="mw-jump-link" href="#mw-head">Jump to navigation</a>
<a class="mw-jump-link" href="#p-search">Jump to search</a>
▼<div id="mw-content-text" lang="en" dir="ltr" class="mw-content-ltr">
  ▼<div class="mw-parser-output">
    ▶<p>…</p>
    ▼<p> = $0
        "Automated information retrieval systems are used to reduce what has been called "
        <a href="/wiki/Information_overload" title="Information overload">information overload</a>
        ". An IR system is a software system that provides access to books, journals and other documents; stores and manages those documents. "
        <a href="/wiki/Web_search_engine" title="Web search engine">Web search engines</a>
        " are the most visible "
        <a href="/wiki/Information_retrieval_applications" title="Information retrieval applications">IR applications</a>
        ". "
      </p>
```

Tokenization

```
raw = "Apple is looking at buying U.K. startup for $1 billion."

# tokenize raw text based on whitespace
for token in raw.split():
    print(token)

# tokenize using spaCy
import spacy
nlp = spacy.load("en_core_web_sm")
for token in nlp(raw):
    print(token.text)
```

Tokenization

Before tokenization

An IR system is a software system that provides access to books, journals and other documents; stores and manages those documents. Web search engines are the most visible IR applications.

After tokenization

An IR system is a software system that provides access to books , journals and other documents ; stores and manages those documents . Web search engines are the most visible IR applications .

Stop words

- A **stop word** is a word that is frequent but does not contribute much value for the application in question.

  standard examples: *a*, *the*, *and*

- Stop words are application-specific – there is no single universal list of stop words, and not all applications use such lists.

  Stop word removal may even be disadvantageous!

Stop word removal

Before stop word removal

An IR system is a software system that provides access to books , journals and other documents ; stores and manages those documents . Web search engines are the most visible IR applications .

After removing stop words and punctuation

IR system software system provides access books journals documents stores manages documents Web search engines visible IR applications

Lexemes and lemmas

- The term **lexeme** refers to a set of word forms that all share the same fundamental meaning.

  word forms *run*, *runs*, *ran*, *running* – lexeme RUN

- The term **lemma** refers to the particular word form that is chosen, by convention, to represent a given lexeme.

  what you would put into a lexicon

Lemmatization

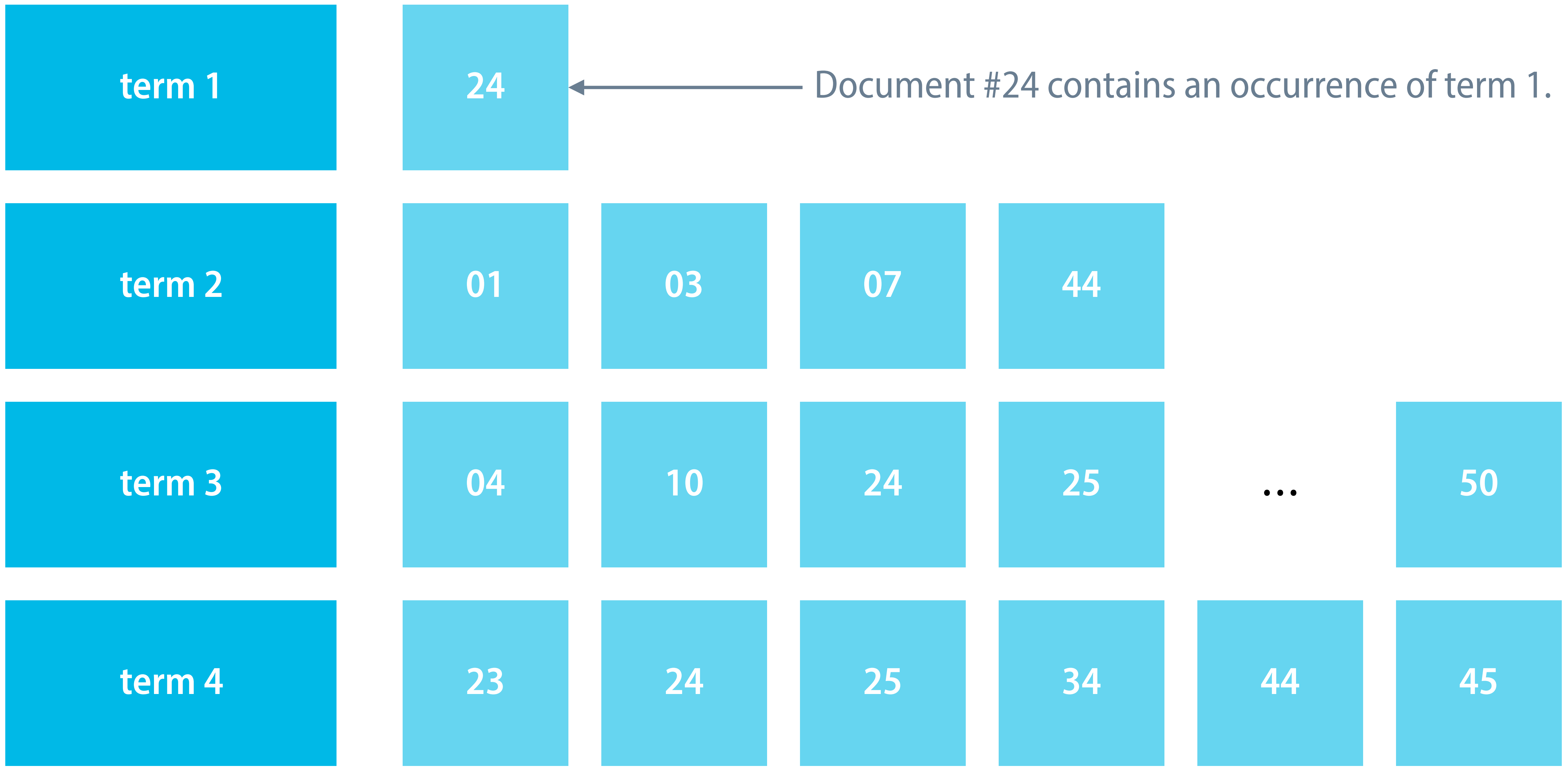
Before lemmatization

IR system software system
provides access books journals
documents stores manages
documents Web search
engines visible IR applications

After lemmatization

IR system software system
provide access book
journal document store
manage document web search
engine visible IR application

Inverted index



Important concepts

- web scraping
- tokenization
- stop words
- lemmatization

This lecture

- Introduction to information retrieval
- Index construction
- Ranked retrieval
- The vector space model
- Evaluation of information retrieval systems
- Introduction to the lab

Ranked retrieval

Reminder: Boolean retrieval

- A Boolean query is a normal-form Boolean expression whose atoms correspond to terms, and is evaluated on documents d. An atom t is true for d if and only if t occurs in d.

  Example: Moriarty AND Lestrade AND NOT Adair

- A system based on the Boolean retrieval model returns exactly those documents for which the query evaluates to TRUE – thus, a document either matches the query, or it does not.

Problems with Boolean retrieval

- Not many users are capable of writing high-quality Boolean queries, and many find the process too time-consuming.
- **Feast or famine:** Boolean queries tend to return either too many results, or no results at all.
- Intuitively, whether or not a document ‘matches’ a search query is not a Boolean property, but is gradual in nature.

Ranked retrieval

- A **ranked retrieval system** assigns scores to documents based on *how well* they match a given search query.

  There are many possible ways of scoring.

- Based on the score, a ranked retrieval system can return a list of the top documents in the collection with respect to the query.

Term weighting

- The score of a document d with respect to a query q is the sum of the weights of all terms t that occur in both d and q.

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

- Any specific way to assign weights to terms is called a **term weighting scheme**.

Term frequency

- Consider the query ‘Moriarty’.
- Several Sherlock Holmes stories contain the term ‘Moriarty’. We would like to rank those stories that contain many occurrences higher than stories that contain few occurrences.
- The number of times a term t occurs in a document d is called the **term frequency** of t in d, and is denoted by $\mathrm{tf}(t, d)$.

  absolute frequency, count

Background information about Moriarty

Moriarty's first and only appearance occurred in *The Adventure of the Final Problem* [no. 23] […].

Holmes mentions Moriarty reminiscently in five other stories: *The Adventure of the Empty House* [no. 24], *The Adventure of the Norwood Builder* [no. 25], *The Adventure of the Missing Three-Quarter* [no. 34], *The Adventure of the Illustrious Client* [no. 45], and *His Last Bow* [no. 44].

Source: Wikipedia (numbering by me)

Term frequency

Moriarty

expected	23	24	25	34	44	45
retrieved	23	24	25	34	44	45
# Moriarty	20	15	1	1	1	1

A problem with term frequency

- Is a document with 20 occurrences of 'Moriarty' really 20 times more relevant than a document with 1 occurrence?
- Intuitively, relevance is not a linear function of term frequency.
- A standard solution to this problem is to down-scale frequency using the log function. This is called **log-frequency weighting.**

$$\mathrm{weight}(t,d) = \begin{cases} 1 + \log_{10}\mathrm{tf}(t,d) & \text{if } \mathrm{tf}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Another problem with term frequency

Moriarty Holmes

expected	23	24	25	34	44	45
retrieved	28	36	48	22	25	52
# Moriarty	0	0	0	0	1	0
# Holmes	87	80	72	68	64	63

Document frequency

- Consider the query ‘Moriarty Holmes’.
- All Sherlock Holmes stories contain many occurrences of the term ‘Holmes’. We would like to rank those few stories that also contain the term ‘Moriarty’ higher than all of them.
- To implement this idea, we can let the weight of a term grow proportionally to the inverse of the fraction of documents in the collection which contain that term.

Inverse document frequency

- Let N be the total number of documents in the collection.
- The number of documents that contain a term t is called the **document frequency** of t, and is denoted by $\mathrm{df}(t)$.
- The multiplicative inverse of the document frequency is called the **inverse document frequency**, and is denoted by $\mathrm{idf}(t)$:

$$\mathrm{idf}(t) = \log \frac{N}{\mathrm{df}(t)}$$

Exercise: Inverse document frequency

Compute the idf values.

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

Suppose that $N = 1{,}000{,}000$.

term t	df(t)	idf(t)
moriarty	1	
poison	10	
exhaust	100	
turn	1,000	
free	10,000	
first	100,000	
the	1,000,000	

Term frequency–inverse document frequency

The **tf–idf weight** of a term t in a document d is defined as

$$\text{tf–idf}(t,d) \;=\; \text{tf}(t,d)\cdot\log\frac{N}{\text{df}(t)}$$

where N denotes the number of documents in the collection.

Variations of the tf–idf weighting scheme

In scikit-learn, the tf–idf weight is computed as

$$\text{tf–idf}(t, d) \;=\; \text{tf}(t, d) \cdot \left(\log \frac{1 + N}{1 + \text{df}(t)} + 1 \right)$$

where N denotes the number of documents in the collection.

Important concepts

- ranked retrieval
- term weighting scheme
- term frequency
- document frequency
- inverse document frequency
- tf–idf term weighting

This lecture

- Introduction to information retrieval
- Index construction
- Ranked retrieval
- The vector space model
- Evaluation of information retrieval systems
- Introduction to the lab

The vector space model

Reminder: Term–document matrix

	Scandal in Bohemia	Final problem	Empty house	Norwood builder	Dancing men	Retired colourman
Adair	0	0	1	0	0	0
Adler	1	0	0	0	0	0
Lestrade	0	0	1	1	0	0
Moriarty	0	1	1	1	0	0

Document representations

- **Documents as sets of terms**

  In Boolean retrieval, the only relevant information is whether or not a term is present in a document.

- **Documents as bags of terms**

  In ranked retrieval based on term frequency, the only relevant information is how often a term is present in a document.

  bag = multiset = set with counts

Term–document matrix with term frequency values

	Scandal in Bohemia	Final problem	Empty house	Norwood builder	Dancing men	Retired colourman
Adair	0	0	14	0	0	0
Adler	13	0	0	0	0	0
Lestrade	0	0	10	51	0	0
Moriarty	0	20	15	1	0	0

Term–document matrix with tf–idf values

	Scandal in Bohemia	Final problem	Empty house	Norwood builder	Dancing men	Retired colourman
Adair	0,0000	0,0000	0,0692	0,0000	0,0000	0,0000
Adler	0,0531	0,0000	0,0000	0,0000	0,0000	0,0000
Lestrade	0,0000	0,0000	0,0291	0,1424	0,0000	0,0000
Moriarty	0,0000	0,0845	0,0528	0,0034	0,0000	0,0000

The vector space model – idea 1

Represent documents as vectors in a high-dimensional space:

- The dimensions (axes) of the space correspond to the terms in the **vocabulary** (potentially relevant terms).

  could be set of all words in the collection, set of most frequent words, …

- The values of the vector components depend on the term weighting scheme: Boolean values, counts, tf–idf values, …

  in scikit-learn: CountVectorizer, TfidfVectorizer

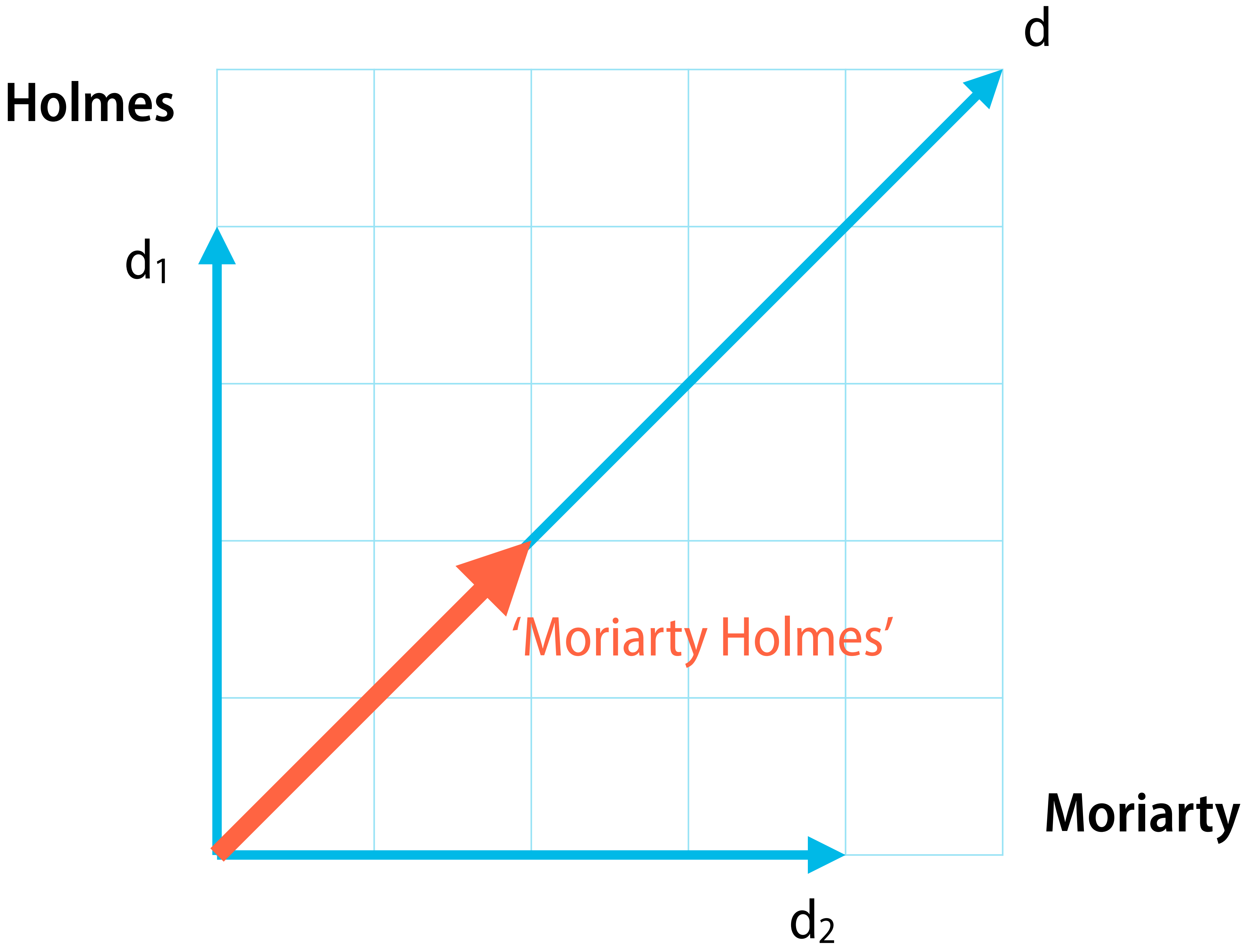
The vector space model – idea 2

To rank documents in the vector space model,

- we represent the query as a vector in the same space as the documents in the collection
- we compute the score of a candidate document as the **similarity** between its document vector and the query vector

  similarity = proximity in the vector space

A problem with Euclidean distance



Example adapted from Chris Manning

From distance to angles

- Euclidean distance is unable to capture similarity of term distributions, as it also varies with the length of the vectors.

  Vectors with similar distributions can have very different lengths.

- Intuitively, we should instead rank documents based on the *angle* between document and query vector.

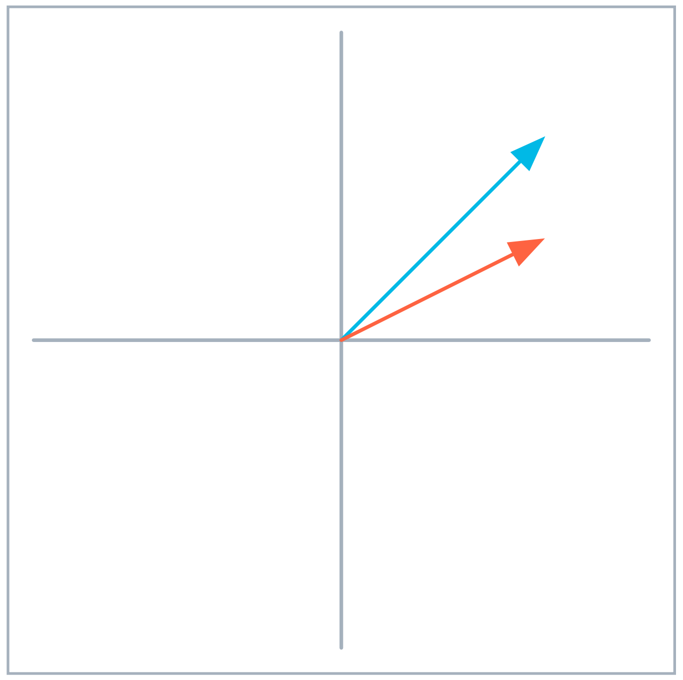
- It turns out that using angle instead of distance also has computational benefits.

The dot product

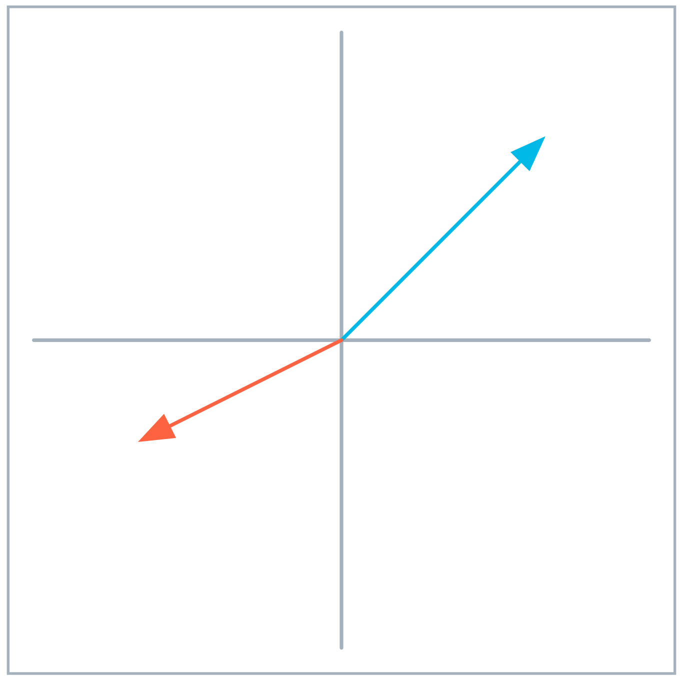
v_1	v_2	w_1	w_2
+2	+2	+2	+1

v_1	v_2	w_1	w_2
+2	+2	−2	−1

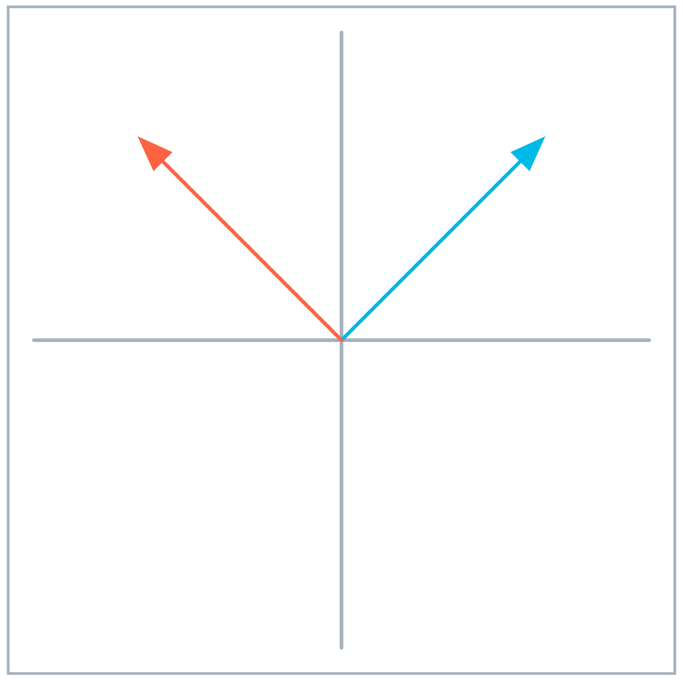
v_1	v_2	w_1	w_2
+2	+2	−2	+2



$\boldsymbol{v} \cdot \boldsymbol{w} = +6$



$\boldsymbol{v} \cdot \boldsymbol{w} = -6$



$\boldsymbol{v} \cdot \boldsymbol{w} = \pm 0$

Cosine similarity

- Like Euclidean distance, the dot product is sensitive to length. To fix this, we can normalize each vector to unit length.

- This length-normalised dot product is **cosine similarity:**

$$\cos(\boldsymbol{v}, \boldsymbol{w}) = \frac{\boldsymbol{v}}{|\boldsymbol{v}|} \cdot \frac{\boldsymbol{w}}{|\boldsymbol{w}|} = \frac{\boldsymbol{v} \cdot \boldsymbol{w}}{|\boldsymbol{v}||\boldsymbol{w}|} = \frac{\sum_{i=1}^{d} v_i w_i}{\sqrt{\sum_{i=1}^{d} v_i^2}\sqrt{\sum_{i=1}^{d} w_i^2}}$$

- Cosine similarity ranges from –1 (opposite) to +1 (identical).

Computational properties of cosine similarity

- To compute the cosine similarity between two vectors, we only need to consider the non-zero dimensions.

  Recall that term–document matrices are very sparse.

- Like other operations based on matrix multiplication, cosine similarity can be computed efficiently on modern GPUs.

Important concepts

- Euclidean distance
- dot product
- cosine similarity

This lecture

- Introduction to information retrieval
- Index construction
- Ranked retrieval
- The vector space model
- Evaluation of information retrieval systems
- Introduction to the lab

Evaluation of information retrieval systems

Evaluation of IR systems

To evaluate an IR system we need:

- a document collection
- a collection of queries
- a gold-standard relevance judgement

Gold-standard relevance judgement

query	document 1	document 2	document 3
505	✔	✔	✘
506	✘	✘	✘
507	✔	✘	✘
508	✘	✘	✔
509	✔	✔	✘
510	✔	✔	✔
511	✘	✘	✘

Sample guideline given to assessors

<num> Number: 508

<title> hair loss is a symptom of what diseases

<desc> Description:

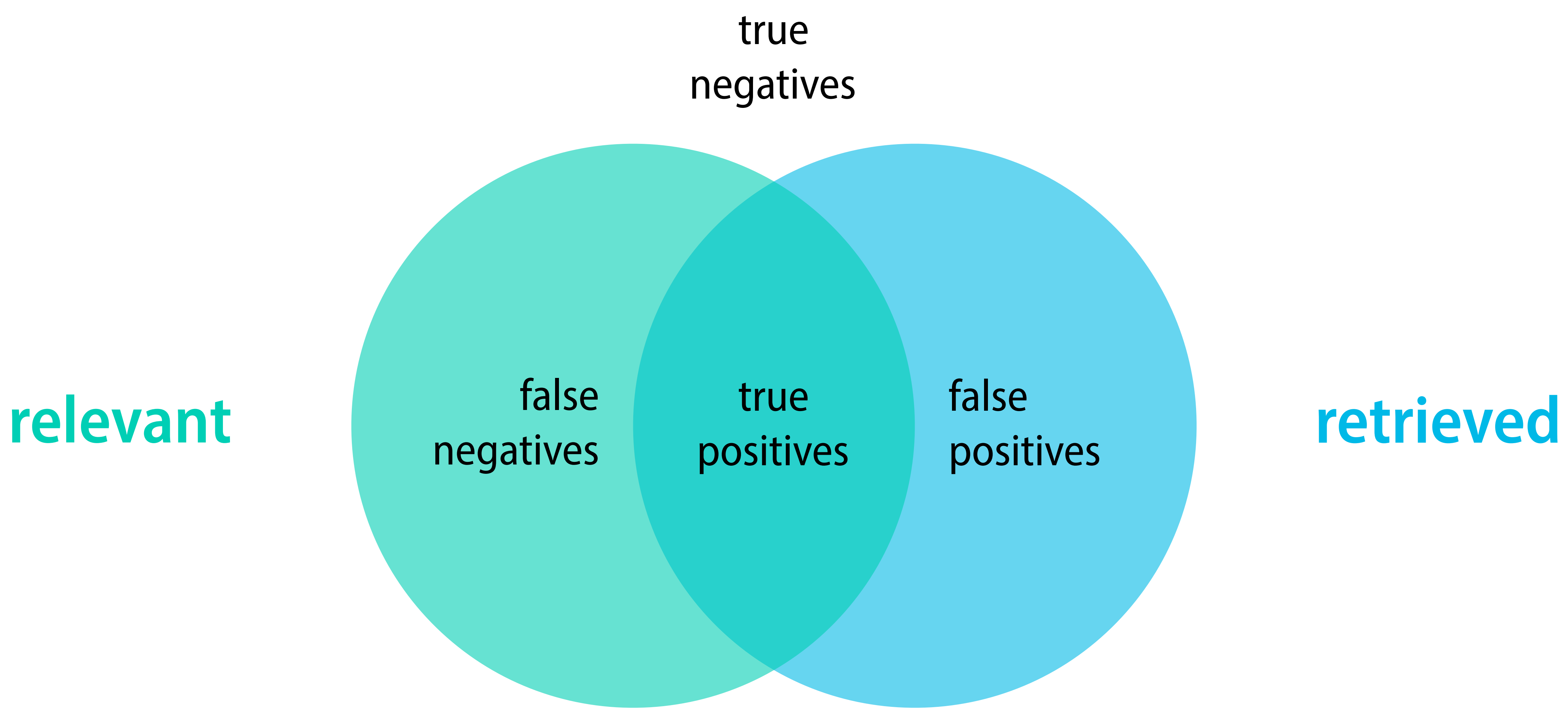
Find diseases for which hair loss is a symptom.

<narr> Narrative:

A document is relevant if it positively connects the loss of head hair in humans with a specific disease. In this context, “thinning hair” and “hair loss” are synonymous. Loss of body and/or facial hair is irrelevant, as is hair loss caused by drug therapy.

Sample query from TREC

Precision and recall for Boolean retrieval



$$P = \frac{|\text{relevant} \cap \text{retrieved}|}{|\text{retrieved}|} \qquad R = \frac{|\text{relevant} \cap \text{retrieved}|}{|\text{relevant}|}$$

F1-measure

A good system should balance between precision and recall.
The **F1-measure** is the harmonic mean of the two values:

$$\text{F1} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Evaluation of ranked retrieval

- Intuition: A system for ranked retrieval is good if the relevant documents rank high and the irrelevant documents rank low.
- We can generalize the evaluation of Boolean retrieval to ranked retrieval by computing precision and recall at different ranks.
- In practice, recall is hard to evaluate, so evaluation focuses on precision.

  Remember that recall requires us to know all relevant documents.

rank	document	relevant?
1	191	✓
2	153	✓
3	28	✓
4	198	✓
5	61	✓
6	174	✗
7	178	✗
8	145	✗
9	183	✗
10	172	✗

best

rank	document	relevant?
1	191	✓
2	174	✗
3	153	✓
4	178	✗
5	28	✓
6	198	✓
7	145	✗
8	61	✓
9	183	✗
10	172	✗

worse

Mean Average Precision (MAP)

- For each query, compute the precision up to each rank where a relevant document was returned.

  up to a fixed maximal rank, say $k = 100$

- Take the average of the query-specific precision values.

- Take the average of the query-specific averages, for all queries in the collection used for the evaluation.

  macro-averaging: each query counts equally

rank	document	relevant	precision @ rank
1	191	✓	1/1
2	153	✓	2/2
3	28	✓	3/3
4	198	✓	4/4
5	61	✓	5/5
6	174	✗	
7	178	✗	
8	145	✗	
9	183	✗	
10	172	✗	

average precision
for this query:

$$\frac{\frac{1}{1}+\frac{2}{2}+\frac{3}{3}+\frac{4}{4}+\frac{5}{5}}{5} = 1.00$$

rank	document	relevant	precision @ rank
1	191	✓	1/1
2	174	✗	
3	153	✓	2/3
4	178	✗	
5	28	✓	3/5
6	198	✓	4/6
7	145	✗	
8	61	✓	5/8
9	183	✗	
10	172	✗	

average precision for this query:

$$\frac{\frac{1}{1}+\frac{2}{3}+\frac{3}{5}+\frac{4}{6}+\frac{5}{8}}{5} \approx 0.71$$

Important concepts

- precision, recall, F1
- mean average precision (MAP)

This lecture

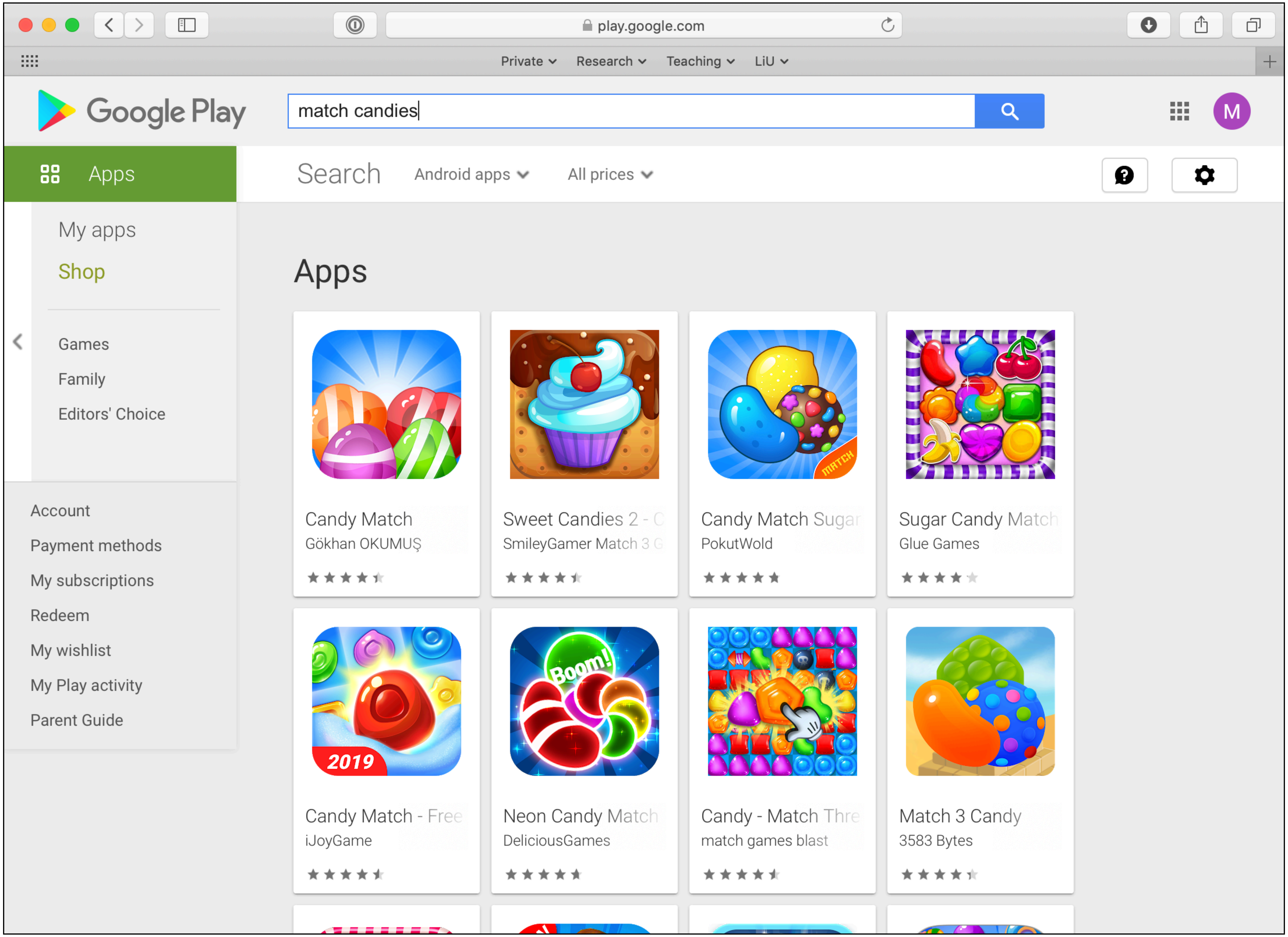
- Introduction to information retrieval
- Index construction
- Ranked retrieval
- The vector space model
- Evaluation of information retrieval systems
- Introduction to the lab

Introduction to the lab

Description of lab 1

- Your task in this lab is to implement the core of a minimalistic search engine for apps from the Google Play Store.
- More specifically, you will implement ranked search over a collection of app descriptions scraped from the Store.

  tf–idf vectorization, cosine similarity
- You are allowed to use a full set of Python libraries, including pandas, spaCy, and scikit-learn.


play.google.com
Private
Research
Teaching
LiU
Google Play
match candies
M
Apps
Search
Android apps
All prices
My apps
Shop
Games
Family
Editors' Choice
Account
Payment methods
My subscriptions
Redeem
My wishlist
My Play activity
Parent Guide
Apps
Candy Match
Gökhan OKUMUŞ
Sweet Candies 2 - C
SmileyGamer Match 3 G
Candy Match Sugar
PokutWold
Sugar Candy Match
Glue Games
2019
Candy Match - Free
iJoyGame
Boom!
Neon Candy Match
DeliciousGames
Candy - Match Thre
match games blast
Match 3 Candy
3583 Bytes