

Search Engine Technologies

03 – Search Engine Frameworks

Philipp Schaer

2024-11-07 – Cologne, Germany
<https://ir.web.th-koeln.de>



Technology
Arts Sciences
TH Köln



Registered teams (so far...)

- Premium Chayas
- The Golden Retrievers
- GirlsinSTEM
- Team Gecko 2.0
- TH Most Wanted!
- Red Hot Suchmaschinentechnologie
- blackbox
- The Unsearchables
- SeekSquadron
- SearchSisters
- CC Chaos

TREC – Topics

- A typical topic file

<topic number="794" type="single">

<query> pet therapy </query>

<description> How are pets or animals used in therapy?
what are the benefits? **</description>**

<narrative> Relevant documents must contain information about pet therapy programs, circumstances in which pet therapy is used, the degree of success of therapy, the degree of success of therapy, the degree of success of therapy, regulations governing it. **</narrative>**

</topic>

Animal-assisted therapy


From Wikipedia, the free encyclopedia

The neutrality of this article is disputed . Relevant discussion may be found on the talk page . Please do not remove this message until conditions to do so are met. (<i>March 2024</i>) (Learn how and when to remove this message) <div><div>Template:POV</div></div> A major contributor to this article appears to have a close connection with its subject . It may require cleanup to comply with Wikipedia's content policies, particularly neutral point of view . Please discuss further on the talk page . (<i>March 2024</i>) (Learn how and when to remove this message)	Read Edit View history Tools ▾
---	--

Relevant

🌐 24 languages ▾

Animal-assisted therapy (AAT) is an alternative or complementary type of therapy that includes the use of animals in a treatment. The goal of this animal-assisted intervention is to improve a patient's social, emotional, or cognitive functioning. Studies have documented some positive effects of the therapy on subjective self-rating scales and on objective physiological measures such as blood pressure and hormone levels.


Dogs are common in animal-assisted therapy.

qrels – example

topic-id	iter	document-id	relevance
301	0	FR940202-2-00150	1
301	0	CR93E-10505	0
301	0	CR93E-1282	1
302	0	CR93E-10071	0
302	0	CR93E-10276	0
302	0	CR93E-10279	2



needed!,
but ignored

These relevance values are the key and ground truth for our experiments.

TREC_EVAL

TREC_EVAL is **THE standard evaluation toolkit** for TREC runs

- Remember, in TREC, retrieval results are called “runs”
- allows a common foundation for evaluating runs
- is open source and therefore completely reusable and open for verification
- allows the extension by own evaluation methods
- is actually used :-) (both by science and industry)
- is used via the Unix shell or the Windows command line

TREC_EVAL command line syntax

- `./trec_eval [-h] [-q] {-m measure}* trec_rel_file trec_top_file`
- where
 - **trec_eval**: name of the program
 - -h: allows the output of a help text
 - -q: is a parameter that allows to get results for each topic
 - -m: allows the selection of individual result types (e.g. map, bpref, or **all_trec**)
 - **trec_rel_file**: is the qrel-file containing the relevance ratings
 - **trec_run_file**: is the file containing the runs, i.e., the ranked retrieval results

Only the parameters in bold are mandatory, the others are optional

```
[1648][schaer@lapad33:~/aiw-suma/it3-set/solr-trec]$ trec_eval qrel.txt run.txt
```

runid	all	0
num_q	all	1
num_ret	all	10
num_rel	all	5
num_rel_ret	all	5

Quiz time! What is the Precision?

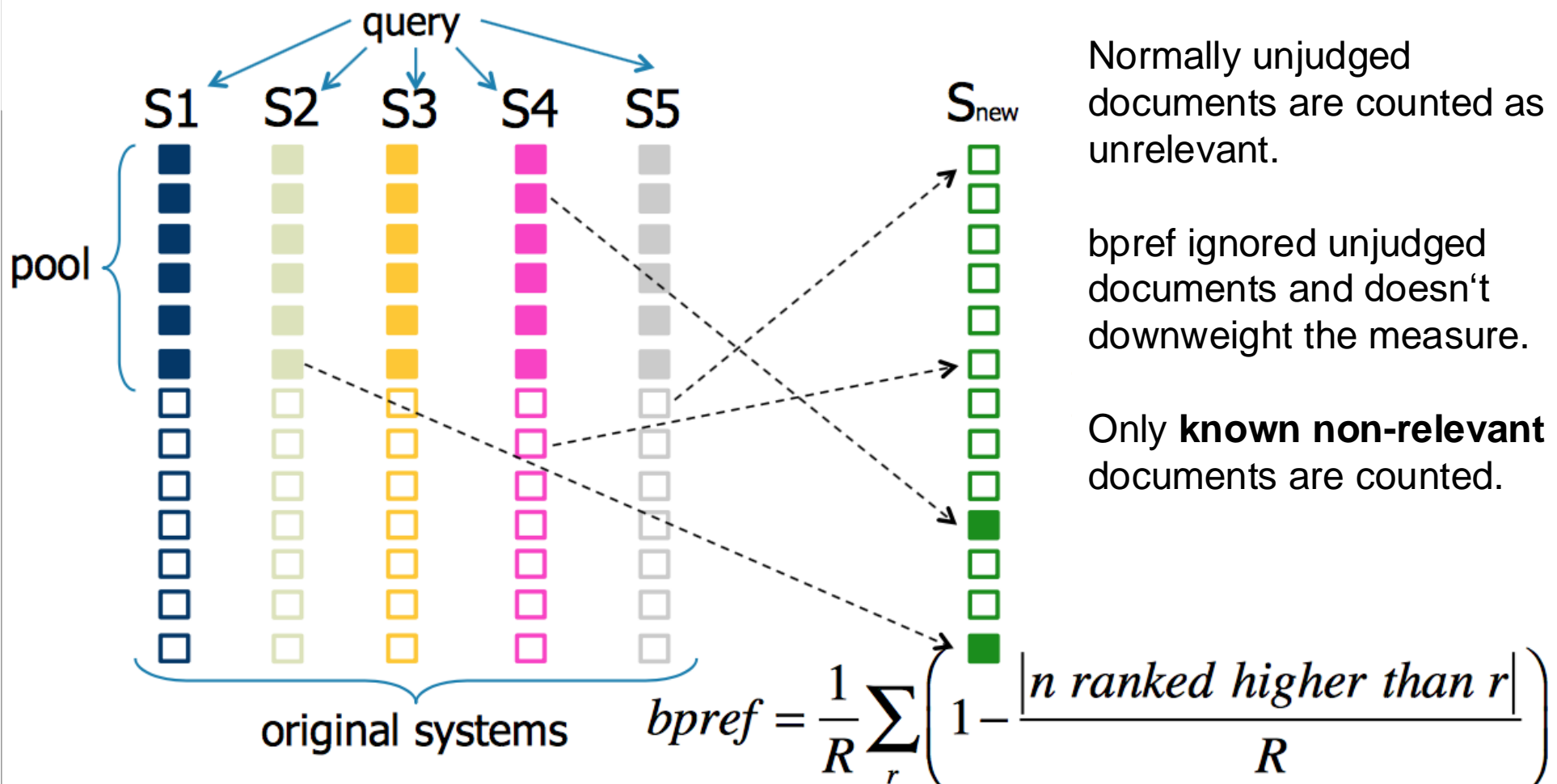
map	all	0.5444
gm_map	all	0.5444
Rprec	all	0.4000
bpref	all	0.4800
recip_rank	all	0.5000
iprec_at_recall_0.00	all	0.6667
iprec_at_recall_0.10	all	0.6667
iprec_at_recall_0.20	all	0.6667
iprec_at_recall_0.30	all	0.6667
iprec_at_recall_0.40	all	0.6667
iprec_at_recall_0.50	all	0.5556
iprec_at_recall_0.60	all	0.5556
iprec_at_recall_0.70	all	0.5556

TREC_EVAL output

- num_q number of topics worked on in the runs
- num_rel total number of relevant documents
- num_rel_ret number of returned relevant documents
- map Mean Average Precision
- R-prec R-Precision
- bpref Binary Preference
- recip_rank Reciprocal Rank
- P5 Precision at 5
- P10 Precision at 10
- ndcg Normalized Discounted Cumulative Gain

new, and
interesting!

bpref – unjudged documents



MRR: Mean Reciprocal Rank

- General assumption: Users look at results from the top; gets annoyed pretty fast; stops once they found the first relevant; doesn't care about the rest
- MRR puts the focus on the first relevant document
- Applicable with sparse judgements or assuming users are satisfied with one relevant document

Mean over all queries

$$MRR(Q) = \frac{1}{|Q|} * \sum_{q \in Q} \frac{1}{\text{FirstRank}(q)}$$

Reciprocal Rank

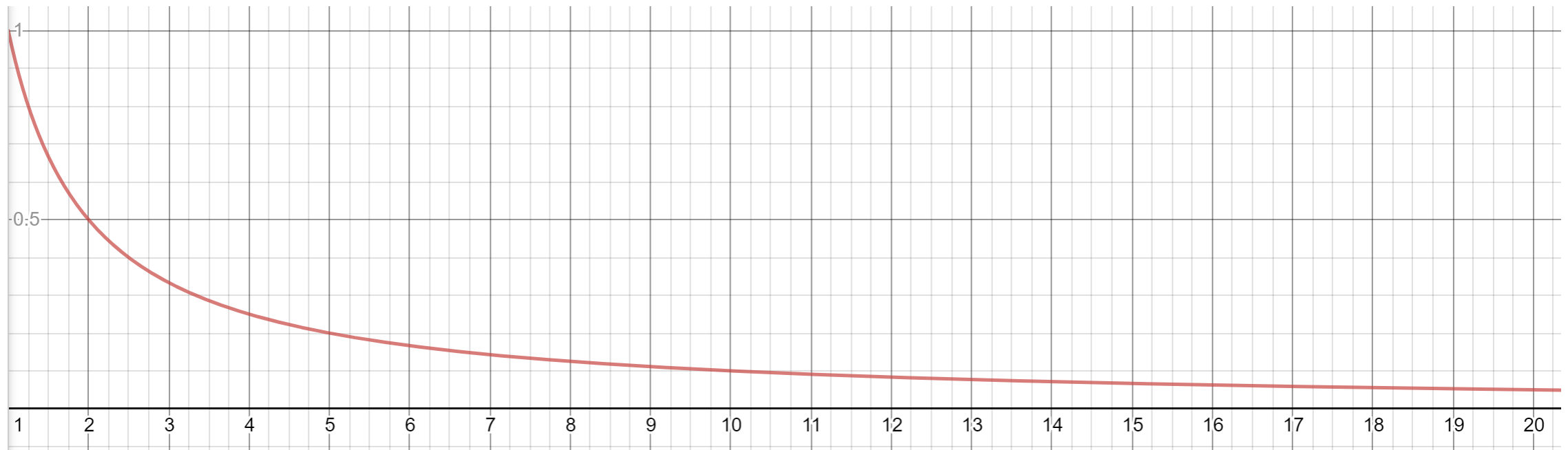
Q Query set

$|Q|$ Number of queries

$\text{FirstRank}(q)$
Returns the Rank of the first relevant doc for 1 query

MRR: The Reciprocal Rank

- Reciprocal Rank: $\frac{1}{x}$
- Very strong emphasis on the first position



* x is plotted continuously, but in MRR x is discrete with the position in step size of 1

Graded Relevance

- Previous metrics all use binary relevance labels
 - Simple enough or too simple?
- Major problem: Of course, there can be a difference in the importance of relevance
 - Binary labels cannot distinguish
- Graded relevance allows to assign different values of relevance
 - Can be floating point or fixed set of classes for manual annotation
 - Fixed set of classes for manual annotation
 - Floating point can be used when relevance inferred from logs

Common Graded TREC Relevance Labels

- **[3] Perfectly relevant**

- Document is dedicated to the query, it is worthy of being a top result in a search engine.

- **[2] Highly relevant**

- The content of this document provides substantial information on the query.

- **[1] Relevant**

- Document provides some information relevant to the query, which may be minimal.

- **[0] Irrelevant**

- Document does not provide any useful information about the query

nDCG: normal. Discounted Cumulative Gain

- General assumption: Users take for each document the relevance grade and position into account, normalize by best possible ranking per query
- nDCG compares actual results with maximum per query
- Relevance is graded
- nDCG@10 most commonly used in modern offline web search evaluation

$$DCG(D) = \sum_{d \in D, i=1} \frac{rel(d)}{\log_2(i + 1)}$$

$$nDCG(Q) = \frac{1}{|Q|} * \sum_{q \in Q} \frac{DCG(q)}{DCG(sorted(rel(q)))}$$

Q	Query set
$ Q $	Number of queries
D	Single doc result list
$rel(d)$	Relevance grade for single query-doc pair
$rel(q)$	List of all relevance grades for a query
$sorted()$	Return graded document by desc. relevance

nDCG: A closer Look

$$DCG(D) = \sum_{d \in D, i=1} \frac{rel(d)}{\log_2(i + 1)}$$

Gain (relevance value, commonly 0 -> 3)

Position Discounting

$$nDCG(Q) = \frac{1}{|Q|} * \sum_{q \in Q} \frac{DCG(q)}{DCG(sorted(rel(q)))}$$

Actual Results

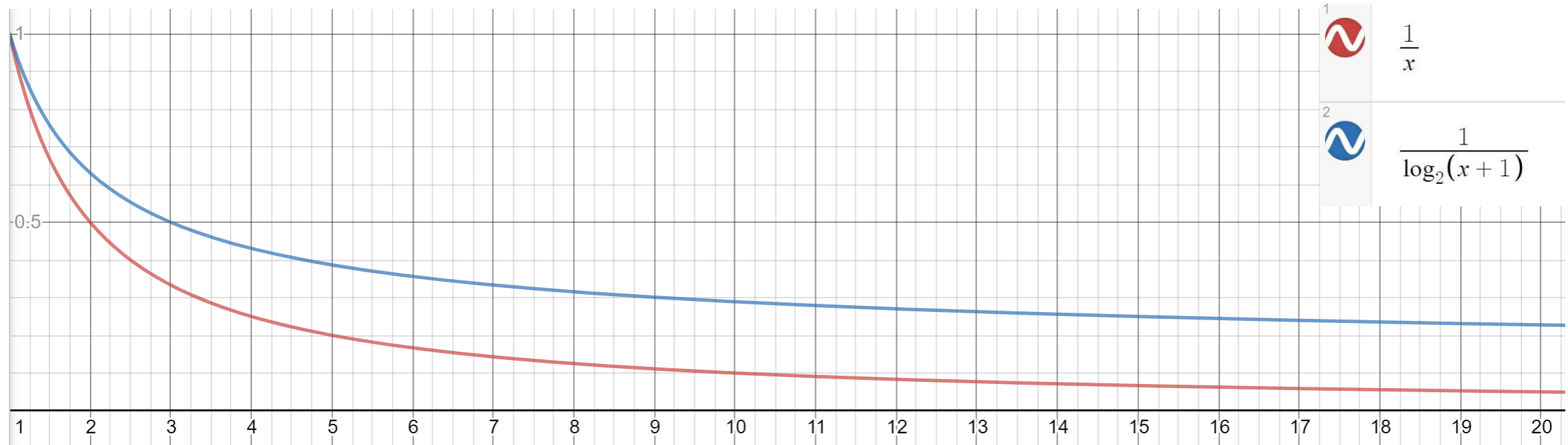
Best possible sorting (ground truth)

Mean over all queries

Q	Query set
$ Q $	Number of queries
D	Single doc result list
$rel(d)$	Relevance grade for single query-doc pair
$rel(q)$	List of all relevance grades for a query
$sorted()$	Return graded document by desc. relevance

nDCG: Position Discounting

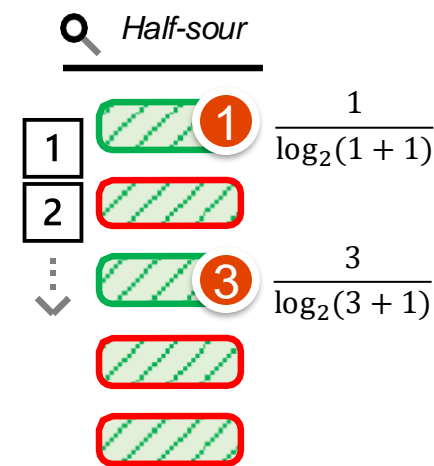
- Comparing the document position discount with reciprocal rank
 - Only for binary case $rel=1$
- nDCG discounts less than MRR



nDCG: An example

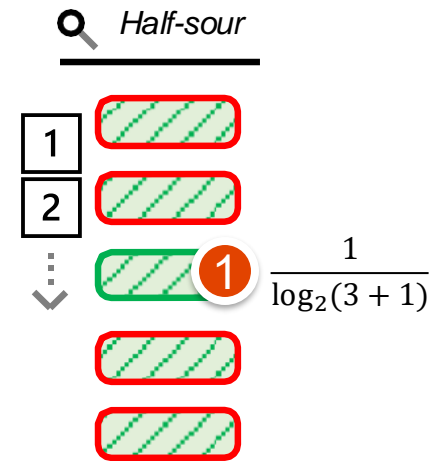
- Assuming two differently relevant docs (rel = 3 & 1)

- Ideal DCG = $\frac{3}{\log_2(1+1)} + \frac{1}{\log_2(2+1)} = 3.63$



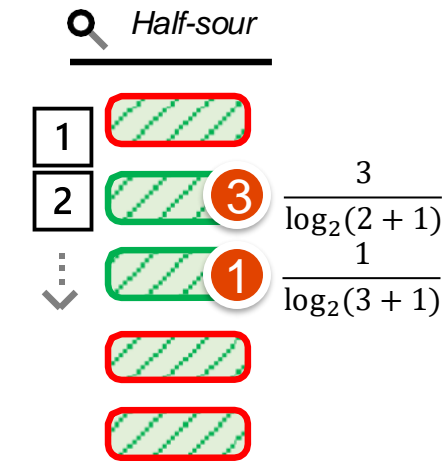
System A

$$\text{nDCG} = \frac{1 + 1.5}{3.63} = 0.69$$



System B

$$\text{nDCG} = \frac{0.5}{3.63} = 0.14$$



System C

$$\text{nDCG} = \frac{1.9 + 0.5}{3.63} = 0.66$$

TREC_EVAL format: qrels

- **qrels** encode the relevance judgements of the assessors
- TREC_EVAL is a simple text format
 - topic-id iter document-id relevance

where

- **topic-id:** is the topic identifier of a topic in the topics file
- **iter:** is a constant about the iteration (not used, but needed)
- **document-id:** is the document-id from our collection
- **relevance:** the relevance value of the document-id for the query-id
 - “-1”: not assessed; “0”: not relevant; “1-255”: relevant

TREC_EVAL format: run results

- Runs are the individual retrieval results that are now “running for evaluation”. Again a simple text format is used :
 - `topic-id Q0 document-id rank score Exp`

where

- **topic-id**: is the topic identifier from the topic files
- **iter** and **Exp**: are constant (e.g. 0) and are rarely used, but ... 😊
- **document-id**: is the document-id from our collection
- **rank**: position in the result list (starting by 0, is usually ignored)
- **score**: is the estimated relevance value given by the search engine (this is the ranking)

run results – example

topic-id	iter	document-id	rank	sim	run_id
301	Q0	FBIS4-50478	1	3.340779	baseline
301	Q0	FR940202-2-00150	104	2.129133	baseline
301	Q0	FBIS4-45552	105	2.127882	baseline
301	Q0	FBIS4-49075	119	2.112576	algoX
301	Q0	FBIS3-27288	499	1.655729	algoX
302	Q0	FR940126-2-00106	1	3.903381	baseline
302	Q0	FBIS3-60449	200	1.374640	algoX
302	Q0	FBIS3-60572	499	1.099626	algoX



needed!,
but ignored



Basics on current search engines



Prominent search engines (frameworks)



Good reads on Solr and Elasticsearch

First Chapter of “Solr in Action”

- great overview of the basics of search
- great overview of the and functionality of Solr
- <https://www.manning.com/books/solr-in-action>



Third Chapter of “Relevant Search”

- detailed description on how to use Elasticsearch
- default ranking and its problem
- how to make it better
- <https://www.manning.com/books/relevant-search>



Solr



- is a **search engine** (no database!)
- based on **Lucene** (written in Java)
- provides **out-of-the-box** indexing functionalities
- works over **HTTP** and according to the REST principle
→ use CURL, Python API etc.
- has also a **GUI**, if you prefer clicking over coding
- is **open source** (Apache license)
- focuses on **text data**
- and much more. ...

In Solr it's all about text!

Solr is a search engine that **focuses on text**, which means it's

- **Text-centric** (handles unstructured text well, as opposed to a database, which tends to focus on structured data)
- **Read-centric** (more content is read out of a search engine than written into it)
- **Document-centric** (the indexed items are flat units of information, mostly documents - no multimedia, no network data, etc.)
- **Flexible schema** (the data to be indexed does not have to follow the same format and structure - there may also be different distributions of content)

Where Solr struggles...

There are things for which Solr is not well suited, e.g.

- when **more than the usual 10-100** documents are expected as a result, e.g. 1 million result documents;
- when **large subsets** of the index are to be analyzed;
- when **relationships** between documents are important;
- when **access rights** and **security** are important;

Again, in all clarity:

- Solr is not a web search engine like Google or Bing!
- Solr has nothing to do with Search Engine Optimization (SEO)!

Solr – an awesome community

- Solr provides a lot of information about itself, and the Solr community is very active in helping each other with issues.
- There is also a very extensive and extraordinary well-explained tutorial about various aspects of Solr:
 - <https://solr.apache.org/guide/solr/latest/getting-started/solr-tutorial.html>
 - very illustrative from scratch on
 - all you need to start with Solr

This will be your first assignment!
Work on the tutorial, to get a feeling for the software...

Attention! Don't fall into the Cloud trap of Solr!

- The tutorial uses the cloud features of Solr – a lot!
- But in our cases, these are not really needed. A single core is enough!
- Tips for later use (after you have completed the tutorials)
 - Just start Solr with **bin/solr start** (no **-c** or **-e cloud**), although it is mentioned in the tutorial a lot
 - When stuck with false data from old examples, read the section about “**Cleanup**” in the tutorial.

ElasticSearch



- is a full-text, distributed **NoSQL database**
- also based on **Lucene**
- also **Open Source**
- also uses **REST** for communication
- can index many different types of content, not only on text
 - Geodata, Business data
- is a store, a search and an **analytics engine**

Where ElasticSearch **struggles**:

- Has a slight latency in indexing.
- Does not support permission management (security)

Solr vs. Elasticsearch

Querying

- Solr: Uses simple URL parameters
- Elasticsearch: Uses JSON

Advantage Solr

- working with static data (e-commerce), as it uses an uninverted reader for faceting and sorting

Advantage Elasticsearch

- working with time series, like log analysis, business analytics, etc.

There are a lot of differences, too many to cover, but if you're interested, here is a good review: <https://sematext.com/blog/solr-vs-elasticsearch-differences/>

NETFLIX

Kids

Kategorien ▾

Q disney



KIDS

Kids-Bereich verlassen

Titel auf Grundlage von Disney-Filme und -Serien | Disney | Disney Channel | Monsters, Inc. | Disney-Filme für Teenager





Computer and information sciences



Showing 1 - 13 of 20.529

View by:

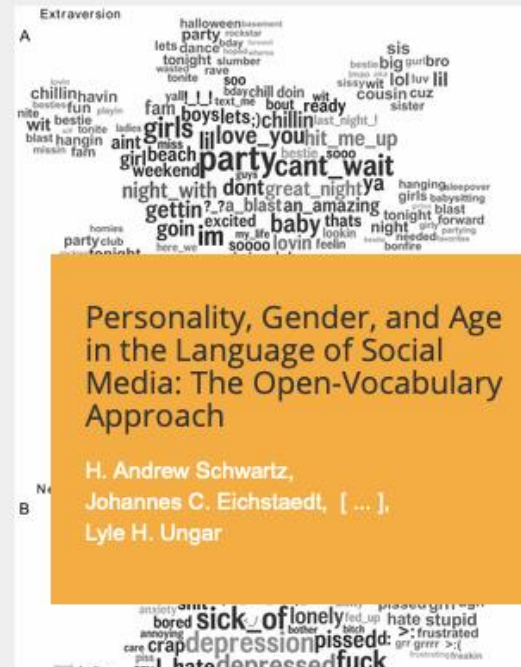
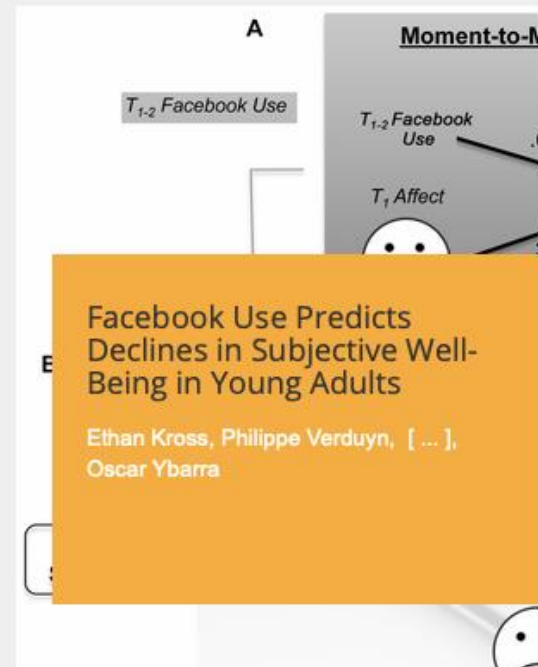
[cover page](#)

[list articles](#)

Sort by:

[recent](#)

[popular](#)

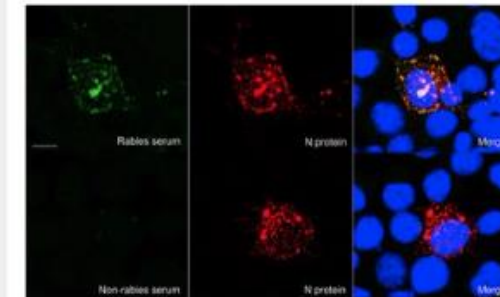


Tweets by @PLOSONE



PLOS ONE
@PLOSONE

#Rabies N protein #ELISA as an additional option for the detection of virus-specific #antibodies in antemortem human cerebrospinal fluid or serum specimens. #Virology #MedHighlight #PubHealthHighlight



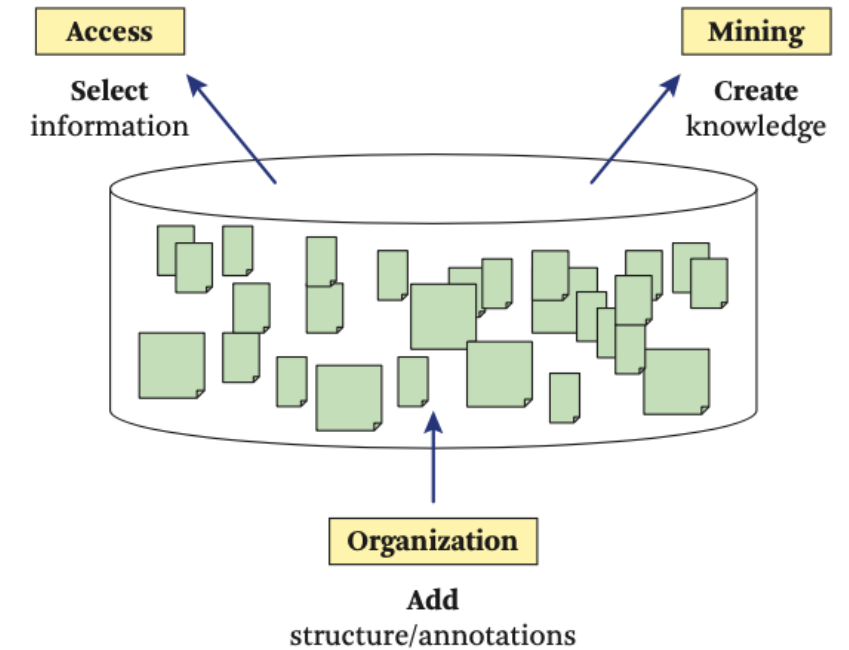
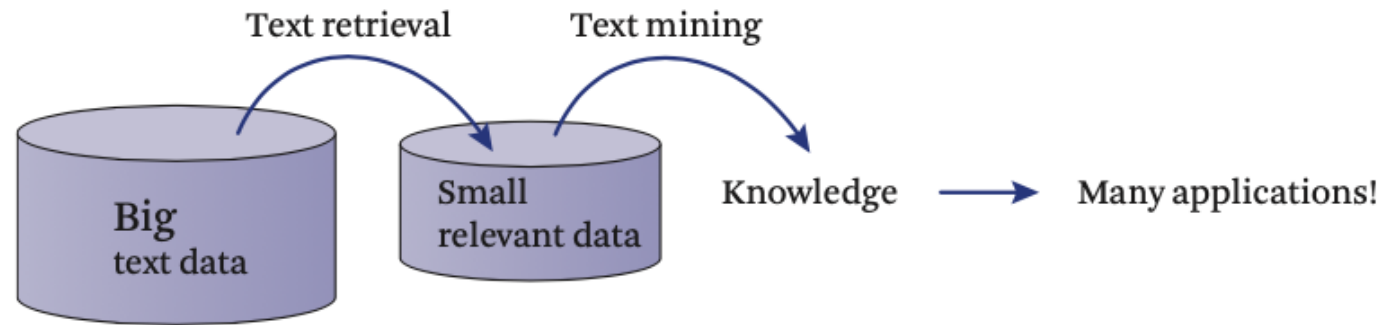
<http://api.plos.org/search?q=title:DNA>

<http://api.plos.org/solr/search-fields/>

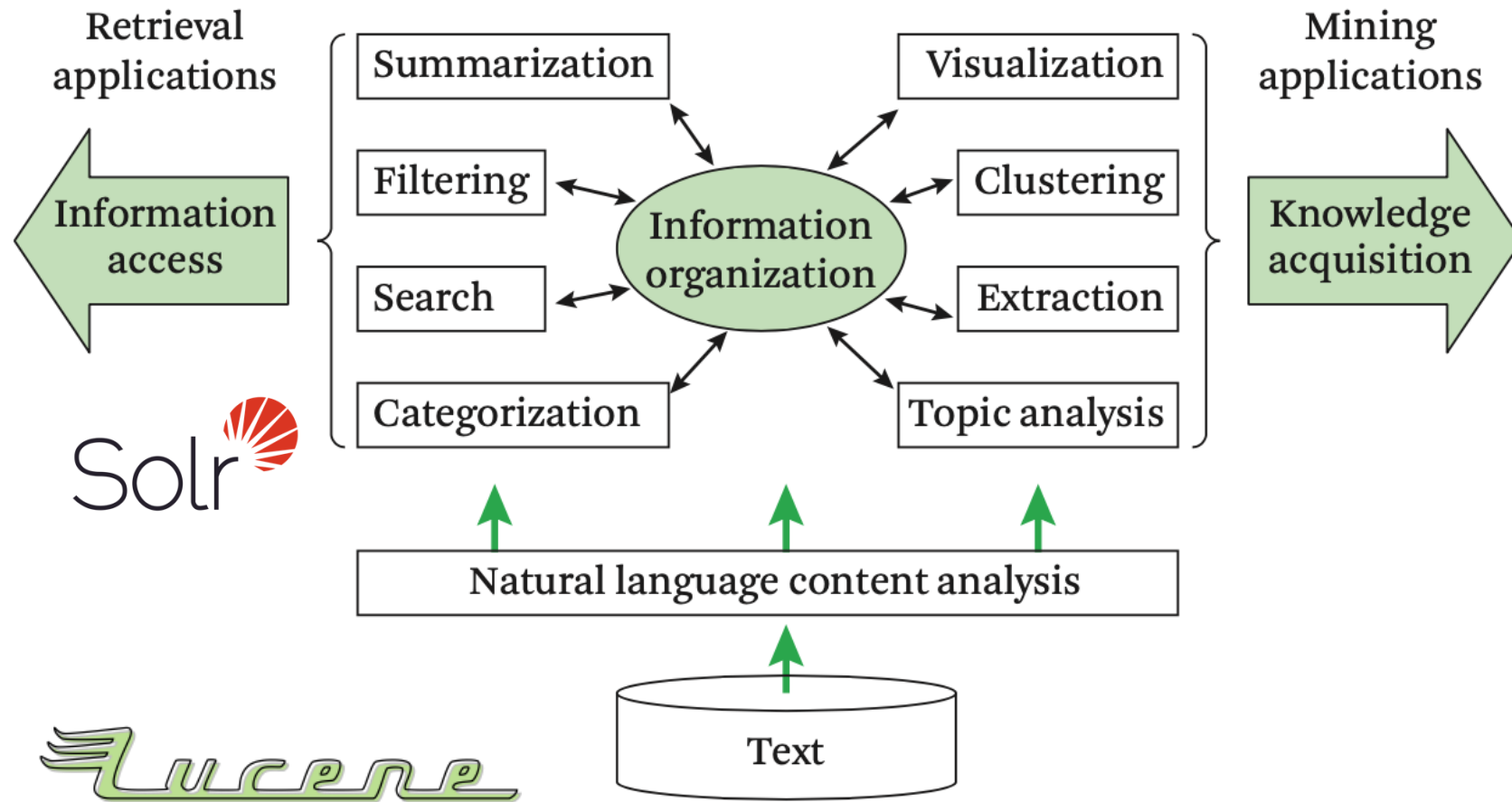
Text Information Processing (TIP)

Two main techniques for analyzing big text data:

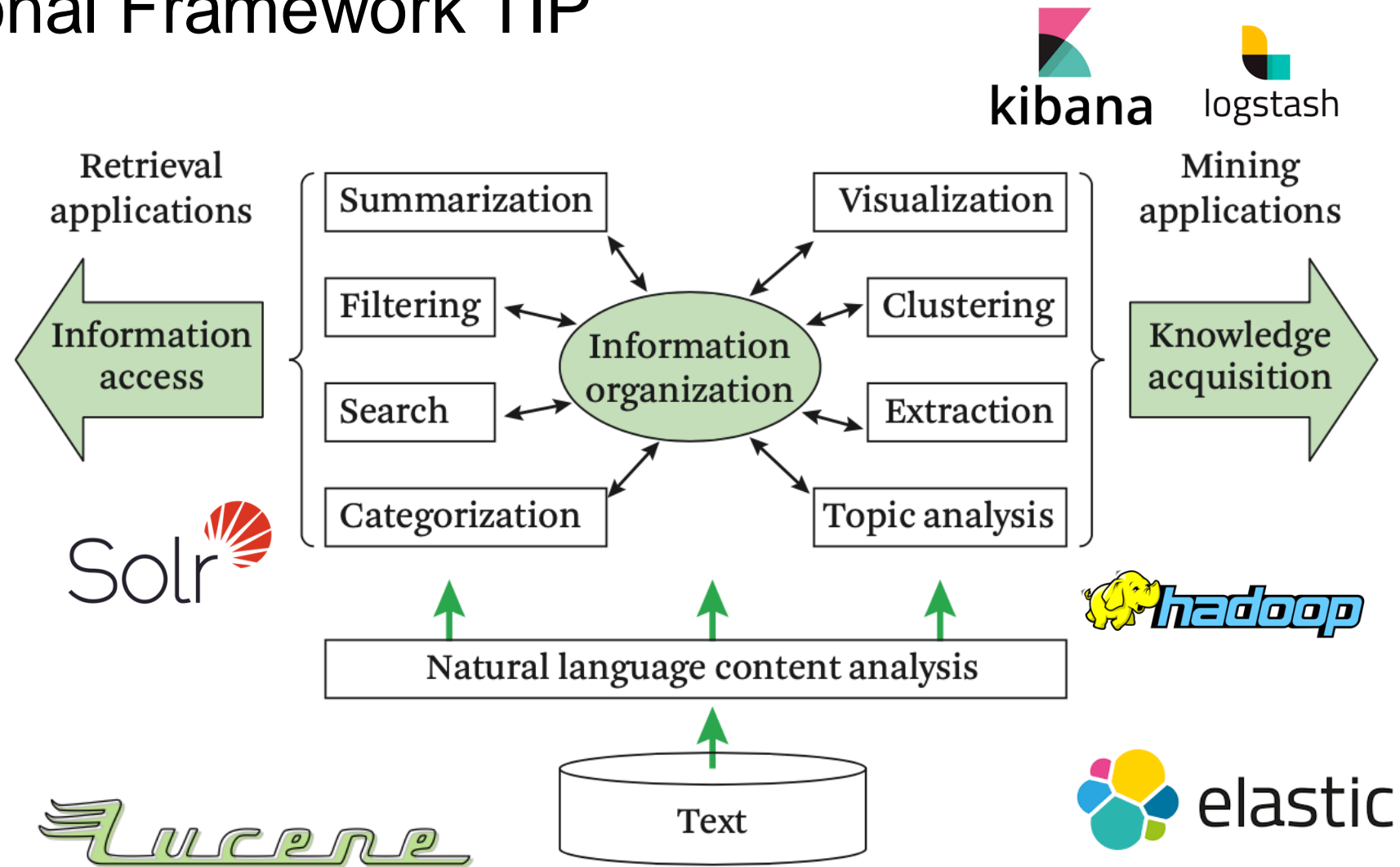
- Text retrieval and
- text mining.



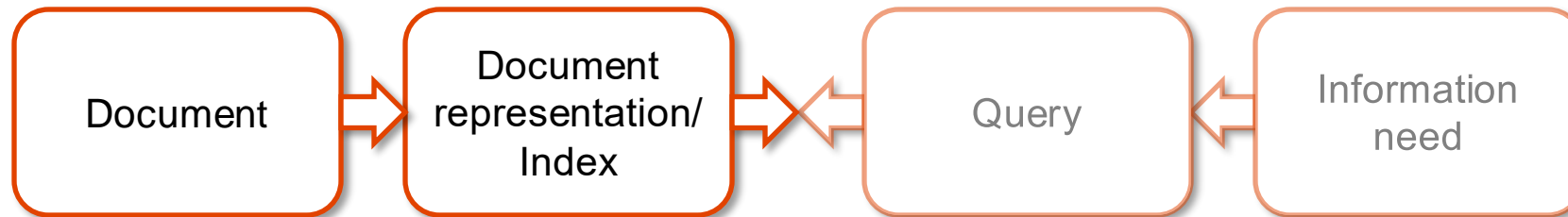
Conceptional Framework TIP



Conceptional Framework TIP



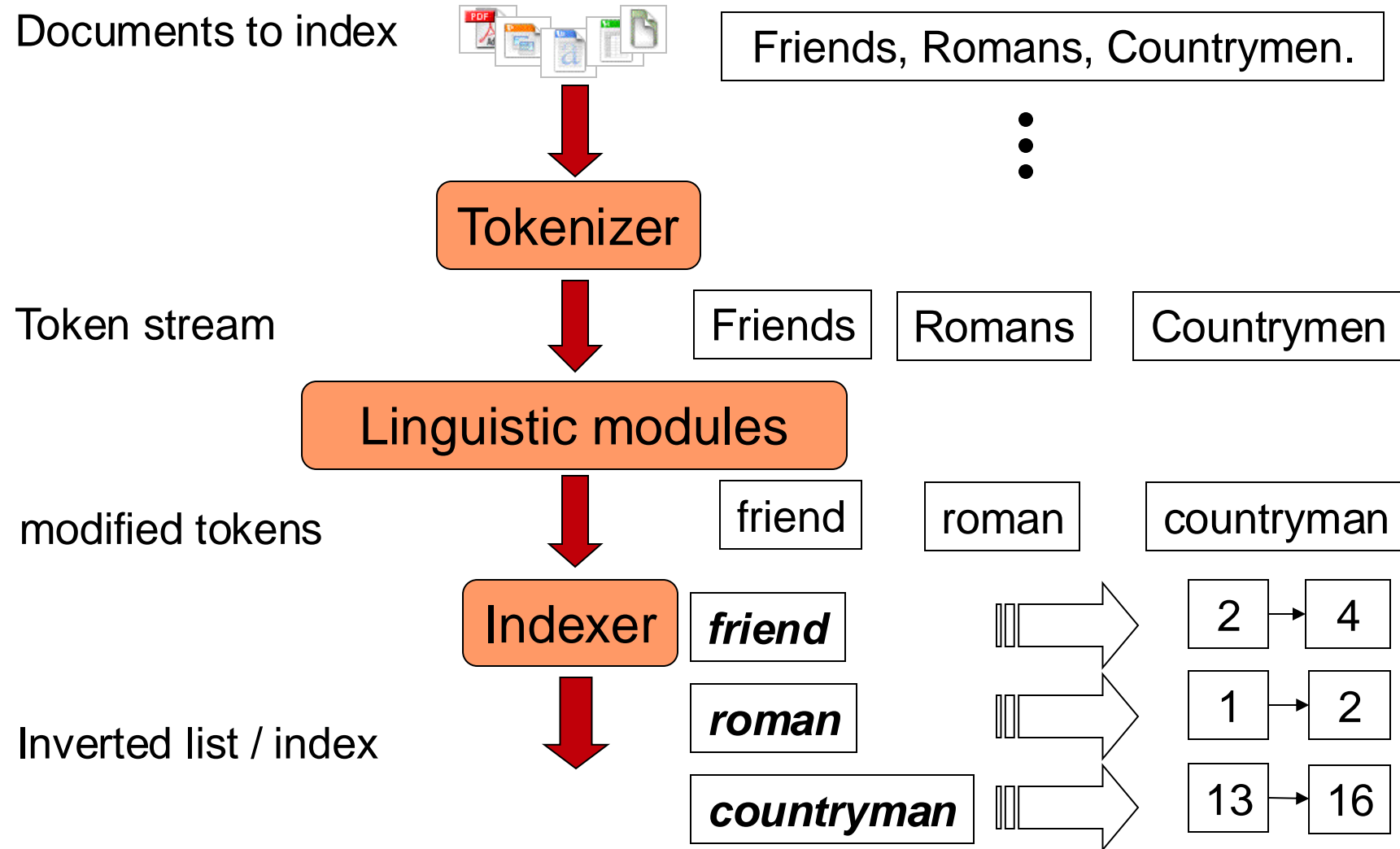
From theory to practice: The Index

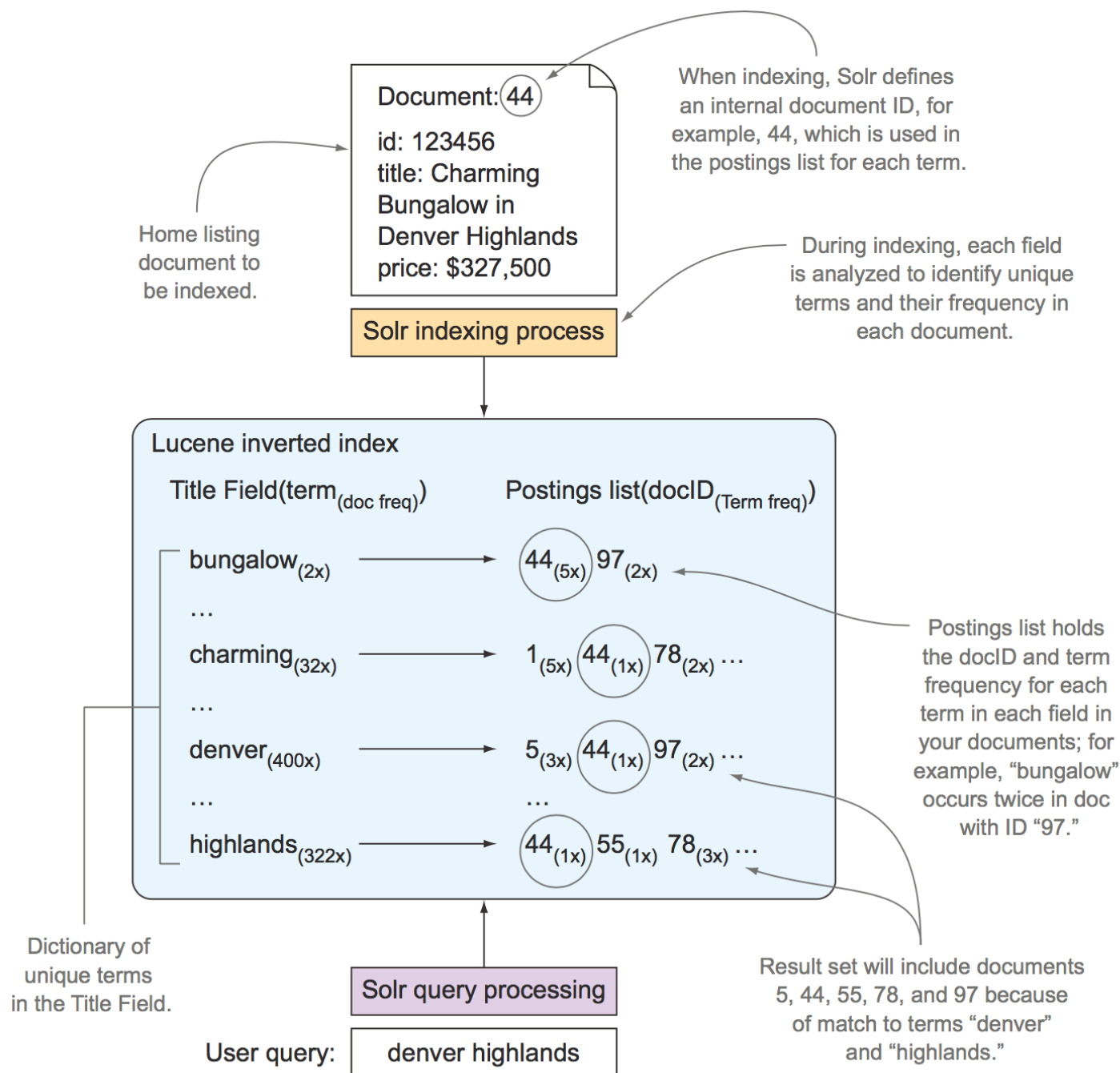


Basic task to solve before you begin to index:

- What preprocessing steps are necessary to build up the index and a good **vocabulary**?
- **Which terms** to include in the index?
- **How** and **in which form** to include the terms in the index?
- In Solr we define these in the so-called **schema** ...

Basic indexing pipeline





Well, Solr and Elastic do everything. We're done, right?

Remember the **diversity of search**?

- web search, e-commerce, expert search, location search, etc.

Solr or Elasticsearch **can do a lot**, but

- don't work well for your problem out of the box
 - if it did, there is nothing unique about your product!
- Solr and ElasticSearch are “**search programming frameworks**”
 - it lets you program **your understanding** of what's relevant
 - you can focus on the art and science of delivering relevant results
 -and of course meet the business goals!! ;-)



Demo Solr