DIS17 – Search Engine Technologies

05 – Queries

Philipp Schaer, Technische Hochschule Köln, Cologne, Germany

Version: WS 2021

Technology
Arts Sciences
TH Köln

Where are we right now?

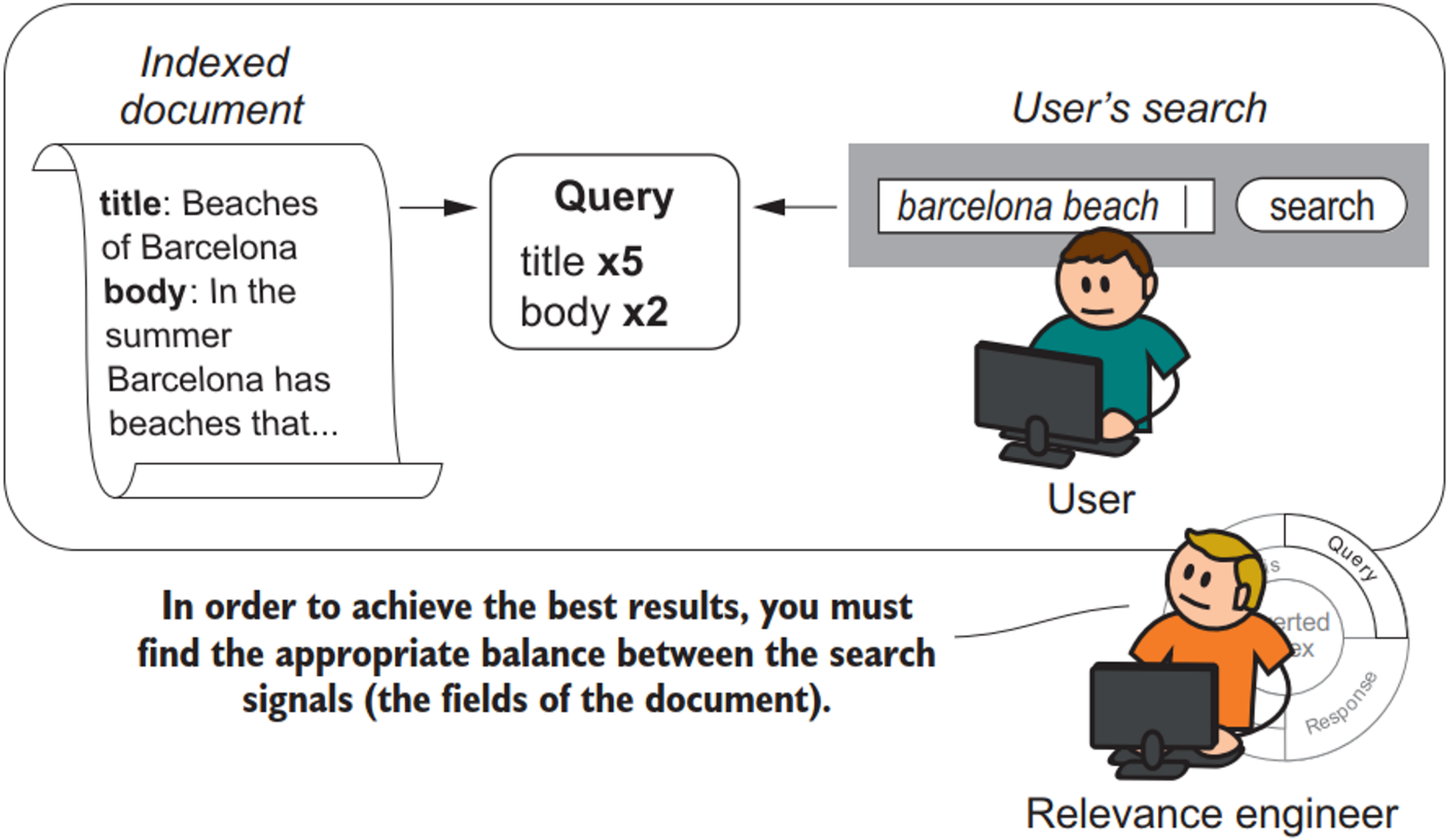
- Solr installed and configured
- CORD data downloaded and indexed into Solr
- We analyzed the data and indexed it along a well-defined indexing-pipeline

- ...right?

What could possibly go wrong…?

- Java hickups
- No correct index field (cord_uid != id)
  - ids have to be unique!
- ATM we have no clue, what to do with the results of our simple single topic searches
  - (except to make a screenshot and send it to Prof. Schaer)
- Running Timo Breuer's `simple-run.py` didn't work out.
- Where can I find and how can I use `trec_eval`?

Using a query against the index



Picture from: Turnball at al., Relevant Search - With applications in Solr and ElasticSearch, Manning, 2016

Keyword search - Lucene Query Syntax

Search for „janeway“ in **the field** description_txt_de

-

Search for the **phrase** „captain janeway“

-

Search for the phrase „captain janeway“ in description_txt_de AND for the phrase „star trek“ in rating_txt

-

Search for „janeway" but for **not** “captain" in description_txt_de

-

Keyword search - Lucene Query Syntax

Search for „janeway“ in **the field** description_txt_de

- `description_txt_de:janeway`

Search for the **phrase** „captain janeway“

- `description_txt_de:"captain janeway"`

Search for the phrase „captain janeway“ in description_txt_de AND for the phrase „star trek“ in rating_txt

- `description_txt_de:"captain janeway" AND rating_txt:"star trek"`

Search for „janeway" but for **not** “captain" in description_txt_de

- `description_txt_de:janeway -description_txt_de:captain`

Brackets and default search

The following statements do the same

- `title_t:(captain janeway)`
- `title_t:captain title_t:janeway`
- `title_t:captain OR title_t:janeway`

Warning: When no field name is given we search in ALL FIELDS (as long a we have specified the catch-all field)

- `title_t:captain janeway`
- `title_t:captain ...` same result

Search for the phrase „captain janeway" in title_t AND the phrase „star trek“ in plot_t or for „Picard“ in title_t

- `(title_t:"captain janeway" AND plot_t:"star trek") OR title_t:Picard`

Proximity search (Sloppy Phrase Search)

Search for two words, that are in a max. distance of 10 words

- `"captain janeway"~10`

Distance = 1

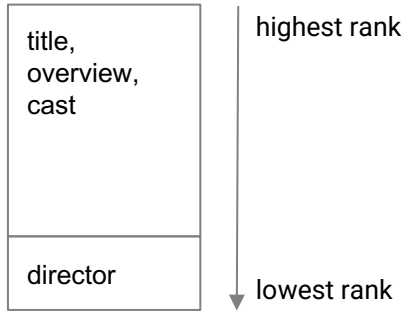
- `"captain janeway"~1` matches the following
- `"captain janeway"`
- `"captain foo janeway"`

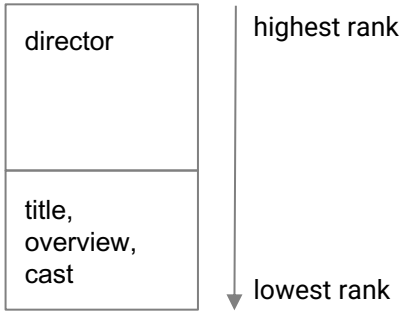
Boost

The operator ^ can alter the weigth of a search term or a phrase.

- If the name „Patrick Stewart“ appears in the director field, give it a higher weight:

- **`director:“Patrick Steward“^2 OR cast:“Patrick Steward“`**





Range Queries

Search for values between an upper and lower threshold:

- Years, IMDB ratings...
- but also text-based fields

Examples

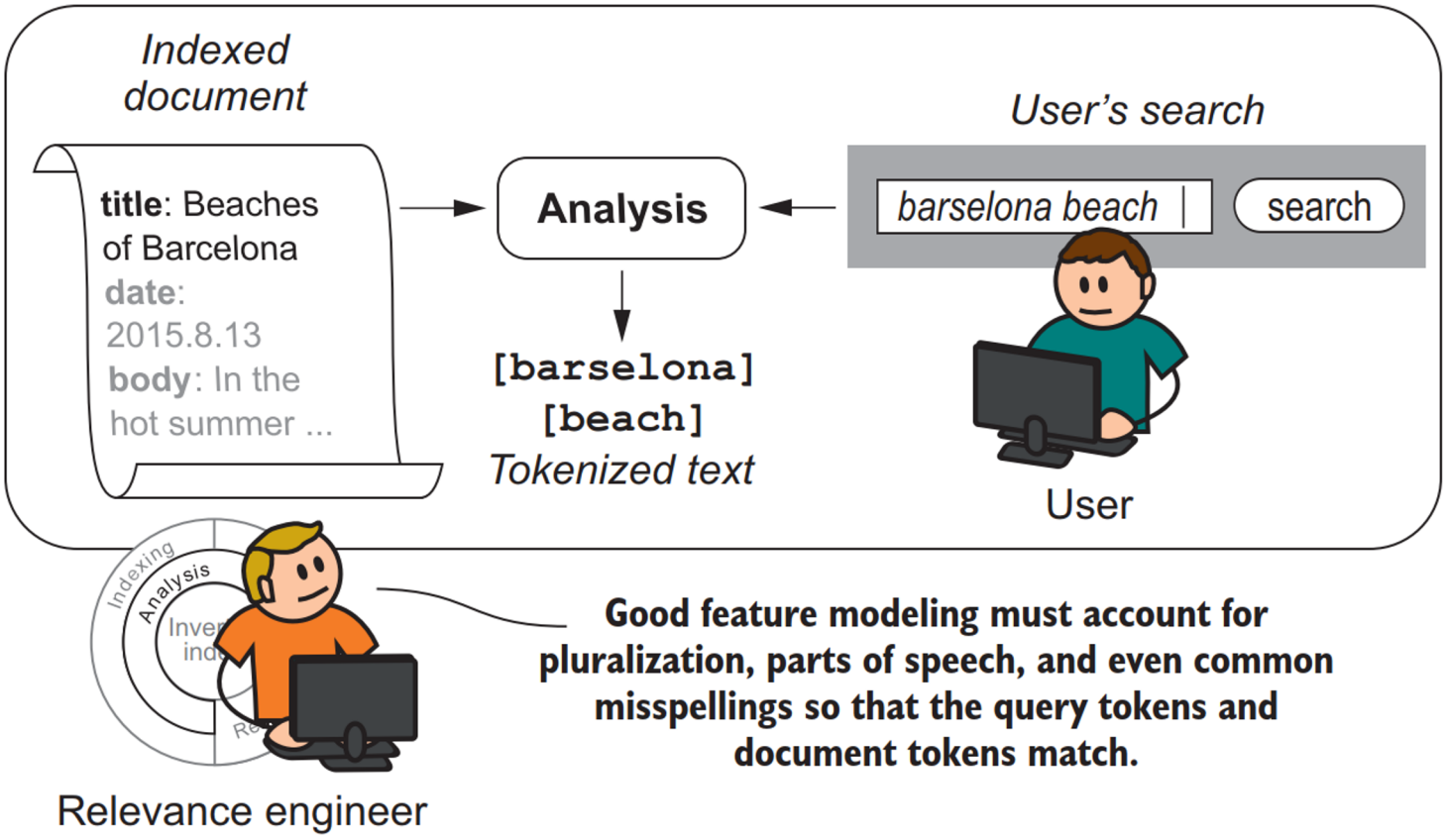
- `imdbRating_f:[8 TO 10]` `// between 8 and 10`
- `imdbRating_f:[8 TO *]` `// „open end“`
- `imdbRating_f:[* TO *]` `// „everything“`

- **Nice detail**: You can invert a range query (NOT or -) and get all documents that do NOT contain any rating:
- `-imdbRating_f:[* TO *]`

Correct queries and boosting helps, but

- Sometimes it is not sufficient to use the query terms at hand.
- What if …
  - the user makes **spelling mistakes** in her query?
  - the user is searching for **couch**, but our index contains only **sofas**?
- To deal with this problem, we need... **Query Expansion**

We cannot expect the user to know our index and make no mistakes



Picture from: Turnball at al., Relevant Search - With applications in Solr and ElasticSearch, Manning, 2016

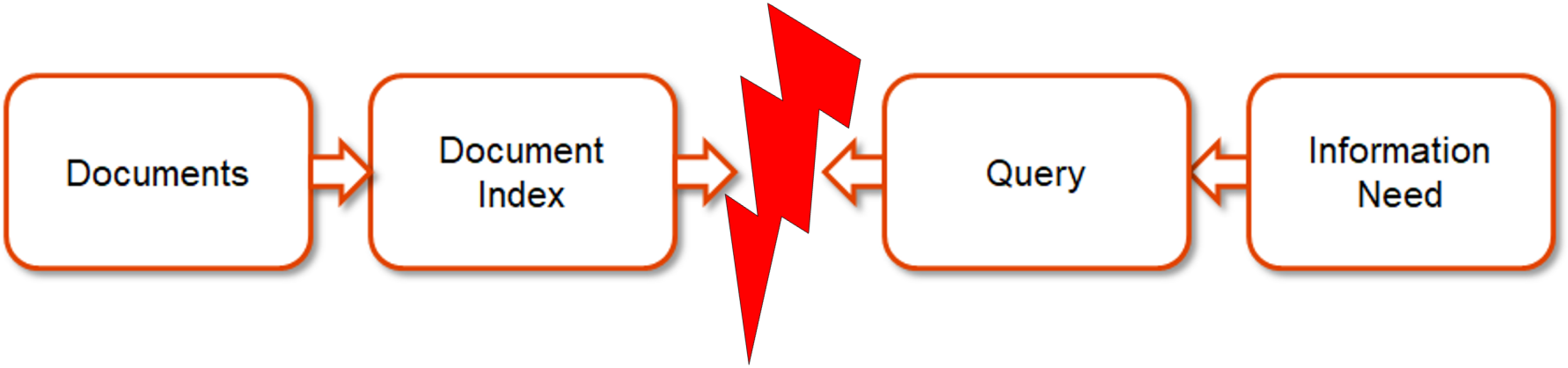
Dealing with Typos

- Small typos can be approached by using **Levenshtein Distance**
  - measures the difference between two strings
  - needs a dictionary to work to match a query token to a dictionary token
- NLP libraries (e.g. SpaCy) can help correcting typos
- Heavier typos are typically approached with Language Models
  - very sophisticated approaches which usually need a lot of training
  - pre-trained models (such as BERT) can be used
- But typos are rather the smaller problem. The bigger one is...

Which one to take

Remember from IR lecture

Why is this a problem?



- At least two different vocabularies, usually more:
  - the vocabulary of the authors of the indexed documents,
  - the vocabulary of the indexer (e.g. the relevance engineer) used to store a representation of the documents in the database,
  - the vocabulary of the user describing her information need,
  - the vocabulary the user actually chooses to issue a query

How to mitigate this problem?

Bring the index and the query to a common denominator, if possible

- One possibility is to use **stemming**, but that is not always helpful, as stemmers are based on heuristics and are error-prone. Therefore, we can try and use synonyms for terms that are used by different groups.

Why synonyms?

- When users type a query “Coronavirus”, the search engine searches also for
  - SARS-CoV-2
  - COVID-19
- When users type a query “large”, the search engines searches also for enormous, huge, great, big, colossal, massive, etc.

Working with Synonyms

To work with synonyms,

- use a **thesaurus** or other dictionaries
  - **Wordnet** (https://wordnet.princeton.edu/) for more general purposes.
  - Some **specific thesaurus** for the medical/biological domain
- use classifications, topic models, embeddings, language models…
- use taxonomies, ontologies, domain-specific knowledge graphs…
- perform **co-occurrence** analysis and/or calculate Jaccard-Index
- analyze log-data
  - which search terms were used together?
  - what did others search for?
  - etc.

Synonyms must be taken with care

Remember:

- When users type a query “large”, the search engines looks also for → enormous, huge, great, big, colossal, massive, etc.
- If using synonyms for every word in the query, it is likely to lower our precision drastically → see “Volltextfalle” when looking for synonyms in full text field

Therefore, we need to examine very carefully how we send the query terms through the “get synonyms gate” of our Solr

- Can be achieved by choosing a domain specific Thesaurus
- Can be achieved by boosting specific fields, e.g., subject or keyword fields

Synonyms in Solr

We can use synonyms in a two-fold way:

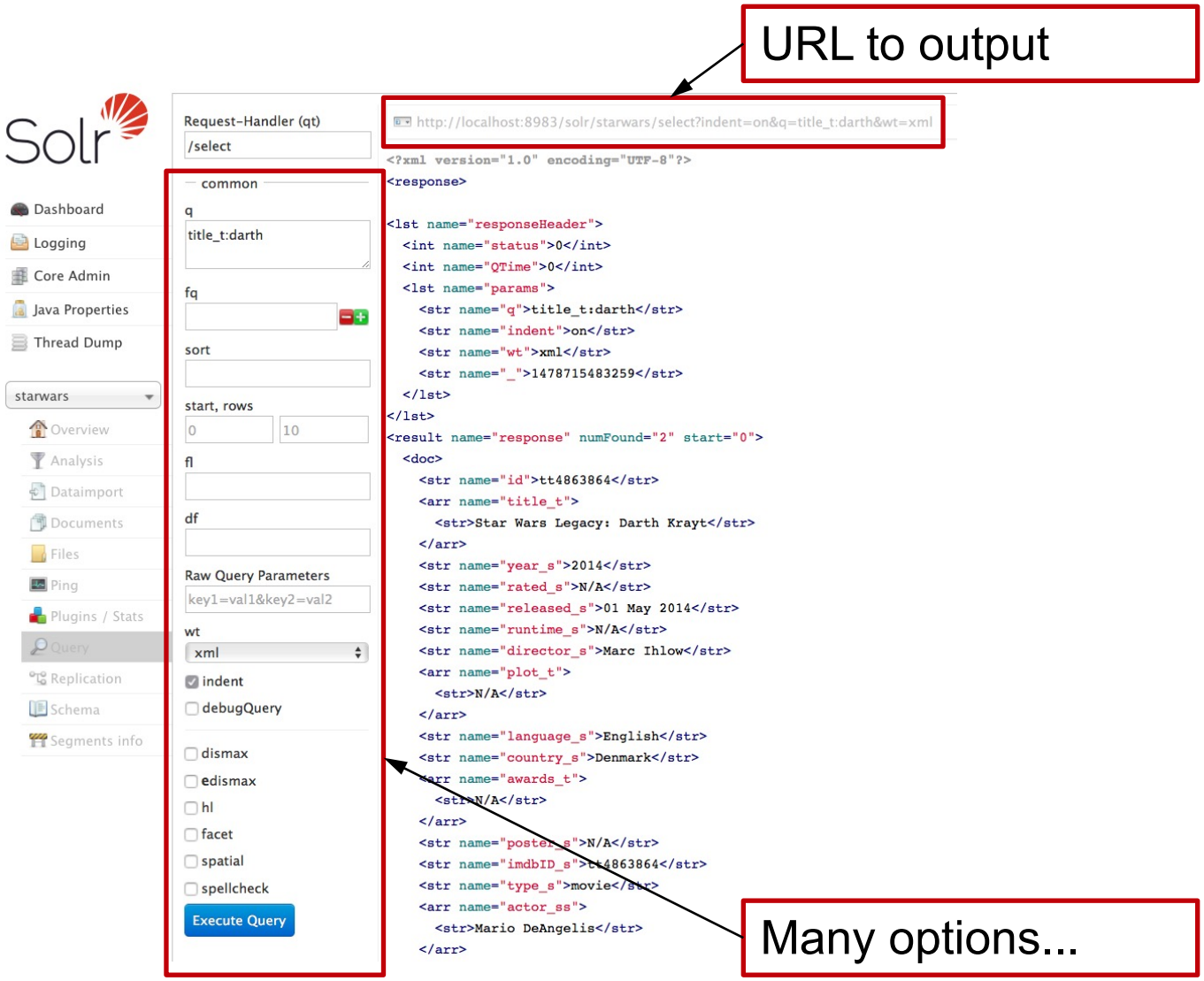
- Use synonyms at **indexing time**
  - Synonyms are written into the index!
  - synonyms might also go through the entire indexing pipeline including stemmers, etc.
  - blows up the index, but can give us further possibility, e.g. related searches, embedding similarity, etc.
- Use synonyms at **query time**
  - Synonyms will be added to query tokens
  - synonyms pass the query processing pipeline, which can also include stemmers, filters, etc.,
  - index remains small, but we cannot perform query suggestions or related searches

Example in Solr (managed-schema.xml)

```
<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100"   multiValued="true">
      <analyzer type="index">
            <tokenizer class="solr.StandardTokenizerFactory"/>
            <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
            <filter class="solr.SynonymFilterFactory" synonyms="index_synonyms.txt" ignoreCase="true"/>
                  <filter class="solr.LowerCaseFilterFactory"/>
      </analyzer>
            <analyzer type="query">
            <tokenizer class="solr.StandardTokenizerFactory"/>
            <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
            <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true"/>
            <filter class="solr.LowerCaseFilterFactory"/>
      </analyzer>
</fieldType>
```

More on this in the next lecture…

The query interface



Many, many options (some examples)

Option	Meaning
q	The query string itself
fq	An additional FilterQuery (narrows down the result set but doesn‘t contribute to the ranking itself)
sort	Sorts results (and overwrites the relevance ranking)
rows	Number of results in the output (default=10)
fl	What fields should be part of the output?
indent	true | false – pretty print the output
wt	xml | json | csv – What is the output format
facet	Activate the facets (needs additional parameters)
...	

Check the **Sorl Reference Guide:**
https://solr.apache.org/guide/8_10/common-query-parameters.html

But what are good queries…?

topics-rnd5.xml — trec-covid

simple-run.py | topics-rnd5.xml

```
<topics task="COVIDSearch 2020" batch="5">
  <topic number="1">
    <query>coronavirus origin</query>
    <question>what is the origin of COVID-19</question>
    <narrative>seeking range of information about the SARS-CoV-2 virus's origin, including its evolution, animal source, and first transmission into humans</narrative>
  </topic>
  <topic number="2">
    <query>coronavirus response to weather changes</query>
    <question>how does the coronavirus respond to changes in the weather</question>
    <narrative>seeking range of information about the SARS-CoV-2 virus viability in different weather/climate conditions as well as information related to transmission of the virus in different climate conditions</narrative>
  </topic>
  <topic number="3">
    <query>coronavirus immunity</query>
    <question>will SARS-CoV2 infected people develop immunity? Is cross protection possible?</question>
    <narrative>seeking studies of immunity developed due to infection with SARS-CoV2 or cross protection gained due to infection with other coronavirus types</narrative>
  </topic>
  <topic number="4">
    <query>how do people die from the coronavirus</query>
    <question>what causes death from Covid-19?</question>
    <narrative>Studies looking at mechanisms of death from Covid-19.</narrative>
  </topic>
```

Line: 1 | XML | Tab Size: 4

Reminder: TREC_EVAL

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

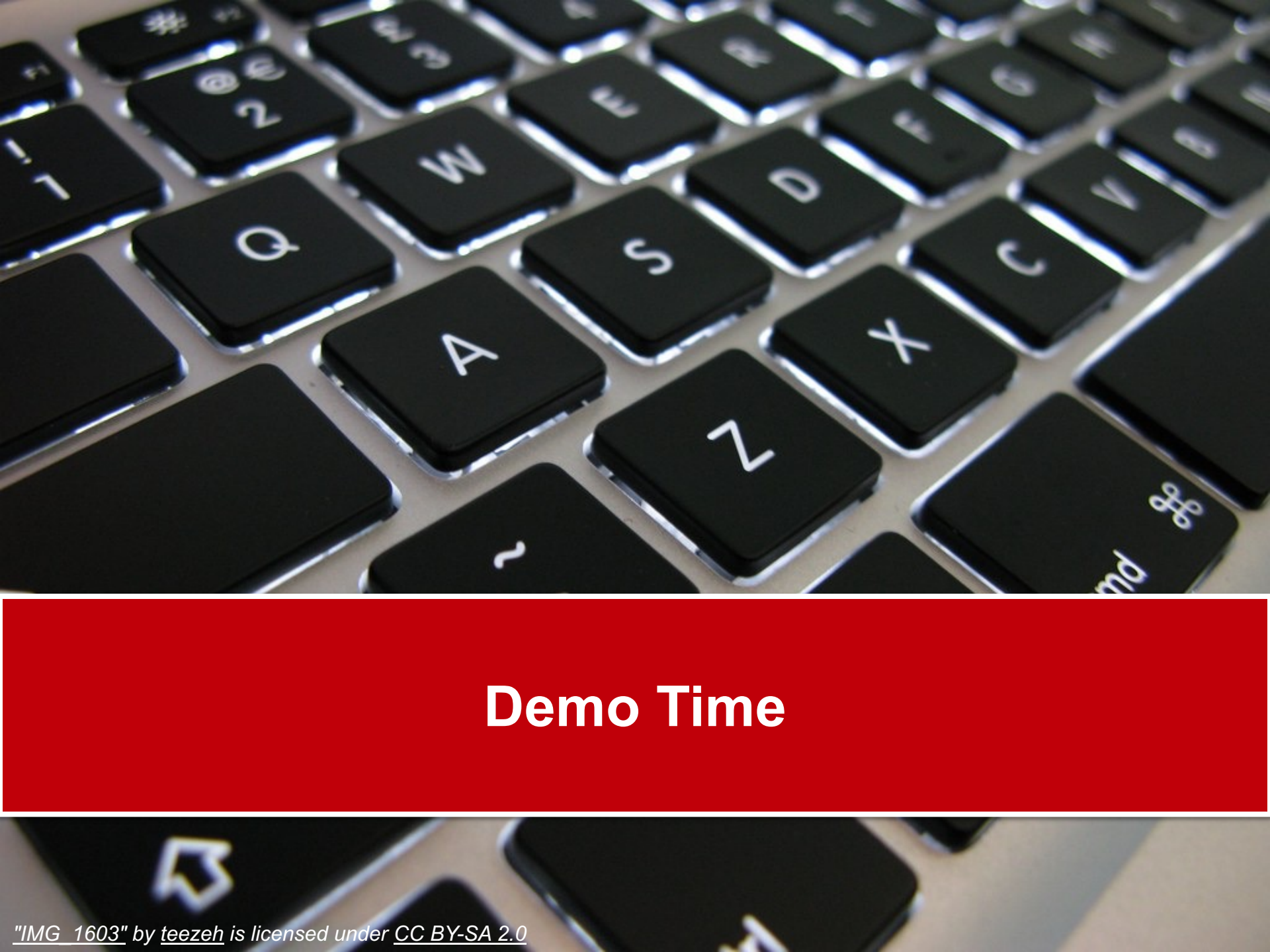
- Remember, in TREC, retrieval results are called "**runs**"
- allows a common foundation for evaluating runs
- is open source and therefore completely reusable and open for verification
- allows the extension by own evaluation methods
- is actually used :-) (both by science and industry)
- is used via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

Get it, while it's fresh: https://github.com/usnistgov/trec_eval

Compiling your own trec_eval

- Download the source: https://github.com/usnistgov/trec_eval
- Unzip the archive
- On the Mac shell:
  ```
  $> xcode-select –install
  ```
- For Linux and Mac: simply compile the source code using the Makefile:
  ```
  $> make
  ```

- Or visit our GitHub repo to get some binaries for Windows. ☺



Demo Time