# 1. Building the prototype

In this chapter, the prototype for a machine indexing of edoc is presented. The focus is primarily on practical implementation although care is taken to spell out design decisions in as much detail as is needed.

## Extracting data from edoc

Even though edoc is a public server, its database does not have a web-ready API. In addition, since edoc is a production server, its underlying database cannot be used directly on pain of disturbing the provided services. To play with the data, it hence needs to be extracted from edoc. This could for example be achieved by cloning the database of the server. However, this route was not available due to the limited resources of the responsible coworkers. I thus employed a workaround: edoc has a built-in search tool where results can be exported in many different formats. It is hence possible to extract the database partitioned by year. The chosen data-format is JSON due to its easy integration with OpenRefine and Annif.

## Description and analysis of the data

[Datengrundlage, siehe die Folien von Viegener!]

## 1.3 Constructing the sample data set

What is the goal of the prototype? From a functional perspective, the envisioned minimal application takes a subset of the data from edoc as input and provides index terms for each item in this subset as output. The success of the prototype depends on comparing the quality and or quantity of output terms with some standard(s) – but more on this below [see X]. The initial task is hence to choose or construct the input data qua subset of data from edoc. For convenience, let us call this subset simply the sample data. The construction of the sample data is important to facilitate development. The sample data need to be small enough to handle yet not be trivial. They hence should contain the quirks of the complete dataset but on a smaller scale. In other words, we are looking for an abstraction rather than an idealization [QUELLE].

Since the extracted data already comes in blocks of years due to the applied extraction process, a natural starting point is a single time slice, say the year 2019 [LINK]. This dataset contains 3486 records. 948 records have both an abstract and keywords.

MeSH-terms as standard: 1202 records with non-blank ID-number field of type PMID. Take this PMID and access full-text by searching for the PMID on <https://pubmed.ncbi.nlm.nih.gov/> (actually the PMID is just added to the main URL, so that should be easy even without API). Then extract MeSH-terms from HTML (ul class keywords list). Open questions: Does Pubmed have an API? To do: Explain MeSH and why it is a gold standard; explain how they to intellectual indexing (see <https://www.nlm.nih.gov/bsd/indexhome.html>).

When using PMID: not all full-texts on Pubmed have MeSH-terms. Also, 519 of these 1202 records also have keywords on edoc. Are these assigned by the researchers uploading the papers? Some might have simply copied keywords provided elsewhere, but not all of them (e.g., <https://edoc.unibas.ch/76461/> has different keywords and MeSH-terms).

We can compare machine indexing based on title, abstract, fulltext and any combinations of these. We can employ different out of the box trained algorithms and compare the results. We could even train Annif for MeSH based on Pubmed! To download, see <https://dtd.nlm.nih.gov/ncbi/pubmed/doc/out/190101/index.html> and look for the fields <MeshHeading><MeshHeadingList> and of course title/abstract/full-text resp. link to it.

How to retrieve mesh data via PMID: <https://dataguide.nlm.nih.gov/eutilities/utilities.html#esummary>, see EFetch

## Gold Standard

Perhaps use Openrefine to reconcile YSO and Mesh or terms from keywords.

## Data-model for indexing

We want to add the indexing result directly to the edoc-items. For this we introduce a category annif with subcategories for each project.

**20201214**

Cleaning edoc keywords

In this section I will be concerned with the subject keywords native to the data-structure of edoc. As explained in above, items are added to edoc must first be added to the research database of the University of Basel. Items are added to the research database by manual process. This process is mandatory and solely the responsibility of the author(s). Entering subject keywords is thus a manual process undertaken by the author. It is hence safe to assume that the resulting subject indexing is of substandard quality. Let me make this assumption explicit:

1. The keywords field is a non-mandatory field. Its importance is hence not ranked very high as compared to other compulsory data fields [above I should have already discussed these data fields] by the operators of the database. This can be seen by two markers, one technical and one procedural.
   1. The keywords field is a free text field. Even though the user manual requires authors to separate keywords by commas (Universität Basel, p 8), this is not checked by the input mask.
   2. Some data fields are manually checked after submission by university employees. For example, bla. However, the keywords field is not validated externally
2. There is no controlled vocabulary from which to choose the keywords. What is more, the user manual does not define what keywords are or how they are to be used.
3. The keywords are entered by the authors. Even though researchers are experts in their domains, there are usually not very good at subject indexing [Quelle]. To wit, most of them are neither information specialists nor librarians.

Given this assumption, we can expect the data to be very heterogeneous.

At each step, describe and analyze the data.

1. Extract keywords
2. Clean keywords
3. Reconcile keywords with Wikidata
   1. Run reconciliation with set parameters
   2. Correct reconciliation (at least for the most important terms)
4. Clean keywords again (remove duplicates from reconciliation)
5. HHistogram, see also <https://bolt.mph.ufl.edu/6050-6052/unit-1/one-quantitative-variable-introduction/describing-distributions/>

**20201215**

In order to elevate quality of reconciliation btwn edoc keywords and wikidata:

1. Manual checking of top 500 most used terms.
2. Exclude bad matches via additional properties:
   1. On the reconciled column, “add column from reconciled values”.
   2. Choose “instance of” as property to add.
   3. On this new column, add facet by text.
   4. Manually check all choices that are eo ipso false such as academic journal or article (this step could also be automated)
   5. Note that this works only for reconciled cells that are matched, so auto-match first.
      1. To do this we want to first set a threshold for how bad automatically matched results can be. There are different measures available (word similarity, Levenshtein distance) or simply the score supplied by the wikidata reconciliation service. We can display this score via a GREL expression (see <https://github.com/OpenRefine/OpenRefine/wiki/Variables>) with “add column based on this column” an then use cell.recon.best.score
      2. We can then use the score as facet to use as threshold. The chosen number is somewhat arbitrary. When choosing 50, we include 13827 items or more than 86% of the items.
      3. With the facet filter employed, we then choose Reconcile>Action>Match each cell to its best candidate.
3. Wikidata can then be used to get GND, MeSH and YSO data!!

To do: Regenerate wikidate-IDs, steps 2. und 3. Oben

**20201217**

Remove trailing s (plural s) from items with wrong scholarly article class (see above for how to filter them). More generally, transform plurals into singulars. We do this as follows:

1. On master column: Add column based on this column called master-singular.
2. Choose Python and add:

if value is None:

return None

if "studies" in value:

return value.replace("studies", "study")

else:

return value.rstrip("s")

1. Reconcile on master-singular with wikidata to create the column wikidata-singular

**20210112**

What is really important: The keywords in the edoc file must be transformed (cleaned etc) for each article similar to the transformation elsewhere in order to allow for a comparison with the newly indexed Annif-keywords! So we basically need a map for each edoc keyword to a Wikidata or whatever keyword. The histogram alone is not sufficient. => So the function clean\_keyword has to be applied to a edoc keyword; the output should then match one of the keywords in the clean (histogram) keywords list. All good.

**20210113**

Instance of to exclude resp. manually check reconciliation (Reconcile/Actions/Clear reconciliation data): scholarly article, clinical trial, scientific journal, human, academic journal, album, open access journal, film, musical group, business, literary work, television series, organization, family name, written work, video game, thesis, doctoral thesis, natural number, single, television series episode, painting, commune of France, city of the United States, magazine, studio album, year, nonprofit organization, border town, international organization, political party, software, song, website, article comic strip, collection, commune of Italy, fictional human, film, government agency, village, academic journal article, female given name, poem… But most of them have only 1 occurrence. Perhaps focus only on the x+? Define x.., say 20

Some entries have “.” at the end, needs to be cleaned.

**20210115**

Fore the entries with none judgement, we apply singe-strategy from above.

Here is the draft-outline for the edoc data section:

1. Edoc
2. Edoc data structure
3. How and why to clean data

**20210118**

We work on enriching the keywords in OpenRefine with MeSH. To do this: on Wikidata-column, edit column > add column from reconciled values > add MeSH descriptor ID. Similar for YSOI-ID.

Now how to retrieve MeSH for article based on article ID? As indicated above see <https://dataguide.nlm.nih.gov/eutilities/utilities.html#efetch>

Example: <https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=pubmed&id=3030142&retmode=xml>

Then look under <MeshHeadingList>

Pyhton module for entrez <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6821292/> This module is complete shit, none of the examples work.

With our custom method we now have 1653 of 4111 items that have been enriched with MeSH terms from PubMed

**20210121**

Run script to check each entry in keywords\_clean\_histogram that lacks yso id with

https://api.finto.fi/rest/v1/yso/search?query=drug

wobei query für den term substitiert.

**20210129**

References

Universität Basel, *Research Database of the University of Basel User Manual*.