Universität Zürich, Philosophische Fakultät

Zentralbibliothek Zürich, MAS Bibliotheks- und Informationswissenschaft

Machine indexing of institutional repositories: indexing Edoc with Annif as proof of concept

 Dr. Maximilian G. Hindermann
Spalentorweg 46, CH-4051 Basel
+41 79 277 0612
maximilian.hindermann@gmail.com

 Silke Bellanger, Referentin

 Dr. Andreas Ledl, Ko-Referent

1. April 2021

Abstract

Lorem ipsum.

Contents

1 Introduction

1.1 Outline

1.2 Method

!! Say something to the effect that all data and code are available on GitHub.

1.3 Tools and configuration

1.3.1 OpenRefine

For data refinement I use OpenRefine version 3.4.1 (https://openrefine.org/). Data manipulation in OpenRefine is tracked: the manipulation history of some data can be exported as JSON file and then reproduced (on the same data on a different machine or on different data) by loading said file. I will supply a corresponding manipulation history whenever appropriate. Note that when using OpenRefine with larger files (and there are many such files in this project), the memory allocation to OpenRefine must be manually increased (see https://docs.openrefine.org/manual/installing/#increasing-memory-allocation for details). ALso note that OpenRefine always counts the number of records by the first column (a fact which caused me many headaches).

2 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.

2.1 Aim

Let us start by stating the aim of the prototype. From a functional perspective, the prototype takes a subset of the data from Edoc as input and provides index terms for each item in this subset as output. In order to fulfill this aim we can distinguish a number of steps that need to be taken:

1. Understand the Edoc data.

2. Select and construct a sample data set.
3. Use Annif to index the items in the sample data set.
4. Construct a gold standard from the keywords.

5. Assess the quality of the output based on the gold standard.

In the sections below, these steps will be discussed in detail.

2.2 Edoc data

2.2.1 Edoc

Edoc is the institutional repository of the University of Basel. It was conceived in 2009 as repository for electronic dissertations and grew in scope when the University of Basel adapted its first open access policy in 2013. As per today Edoc contains roughly 68'000 items.

Edoc runs on the EPrints 3 document management system (https://github.com/eprints/eprints).
!! Add technical description of Edoc.
!! Perhaps add description of how files are added to Edoc, see PDF in `/todo`.

2.2.2 Data extraction

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. The data hence needs to be extracted from Edoc in order to work with it. This could for example be achieved by cloning the database of the server and running it on a local machine. However, this route was not available due to the limited resources of the responsible co-workers. I thus employed a workaround: Edoc has a built-in advanced search tool where the results can be exported. Even though it is not possible to extract all database items by default, only one of the many available search fields has to be filled in. The complete database can hence be extracted by only filling in the `Date` field and using a sufficiently permissive time period such as 1940 to 2020 since the oldest record in the database was published in 1942. On January 21 2021, this query yielded 68'345 results. These results were then exported as a 326 MB JSON file called `1942-2020.json`. `1942-2020.json` has two drawbacks. First, the maximum file size on GitHub is 100 MB. And second, `1942-2020.json` is too big to be handled by standard editors requiring special editors such as Oxygen. For these reasons `1942-2020.json` was split into smaller files that are easier handled and can be uploaded to GitHub. For this `_Utility.split_json` was used yielding 14 files of size 20 MB or less containing 5000 or fewer entries each. These files are called `raw_master_x-y.json` (where x and y indicate the entries as given by `1942-2020.json`) and saved under `/edoc/raw`.

2.2.3 Data description and analysis

!! Explain the Edoc data fields. Say perhaps something about the granularity of the relevant fields. Say something about how the fields are filled in (and by whom).

2.3 Sample data set

In this section, I will explain how the sample data set is selected and constructed. "Selection" means the task of specifying a subset of the Edoc data and "construction" means the task of implementing this selection.

2.3.1 Selection

There are a number of constraints for selecting the sample data set pertaining to an items's text, evaluability and the data set's scope.

The first constraint stems from Annif, the machine indexing framework that will be used. Put simply, machine indexing assigns index terms to a text based on training data. The training data consists of a set of texts, a vocabulary, and function from vocabulary to texts. In this context we mean by "text" a sequence of words in a natural language and by "vocabulary" a set of words. Intuitively, the training data consists of pairs of text and subject terms that meet some standard. The training data and the vocabulary are already supplied by Annif, we only need to provide a text for each item that we want to be assigned index terms. Therefore, we only select items with text.

The second constraint is that we must be able to evaluate the quality of the index terms assigned by Annif to any item in the sample data set.

The third and final constraint takes into account the possibility of having to extend the prototype to the complete Edoc data. On the one hand, the sample data set should be small enough to allow for easy handling and rapid iteration. On the other hand, the sample data set should not be trivial but reflect the quirks and inadequacies of the complete dataset. In other words, we are looking for an abstraction rather than an idealization in the sense of Stokhof and van Lambalgen (2011).

2.3.2 Construction

The first constraint is that each item in the sample data set needs to have a text. Intuitively, this text could consist of a title, an abstract, or even a full

text, or any combination of the above. Similar to manual indexing, having more information (that is, longer texts) usually implies a higher indexing quality in machine indexing. However, this assumption needs to be confirmed empirically for the given machine indexing framework and data set. If indeed confirmed, it might be advisable only to index items that have a certain minimal text length. Therefore, in order to construct the sample data set, we require each item to have a non-empty value in the data fields `title`, `abstract`, and `id_number` as proxy to retrieve a full text remotely. Note that even more context could be provided by taking into account other data fields such as `type`, `publication` or `department`. Especially the latter might be valuable when disambiguating homonymous or polysemous words. For example, consider the item `https://edoc.unibas.ch/id/eprint/76510` which is titled "Blacking Out." Without further context, this title could refer (amongst others) to a physiological phenomenon, a sudden loss of electricity, or a measure taken in wartime. Knowing that the item was published by a researcher employed by the Faculty of Business and Economics (and not, say, by the Faculty of Medicine) gives us reason to exclude the first meaning. However, this idea cannot be implemented with the out of the box instance of Annif employed in this chapter (see subsection "Machine indexing").

The second constraint is that the index terms assigned to each item in the sample data set must be assessable. How the assessment is conducted in detail is discussed in section "Assessment". For now, it is sufficient to say that we require a standard against which we can evaluate the assigned index terms. By "standard" I mean that given a vocabulary of index terms and our sample data set, for every item in the sample data set, there is an approved subset of the vocabulary. The production of a standard from scratch is of course is very costly since a person, usually highly skilled in a certain academic domain, must assign and/or approve each item's index terms (!!source). For this reason we try to avoid having to produce a standard. One way of doing so is by requiring items in the sample data set to have non-empty `keywords` data field. The caveats of this approach are discussed in section "Assessment".

Taking into account the third constraint simply means that we do not employ any further restrictions. We can hence construct the sample data set by choosing exactly those items from `/edoc/raw` which have non-empty data fields `title`, `abstract`, `id_number`, and `keywords`. We do so by calling `_Data.select_from_file` iteratively for all files in `/edoc/raw`. The resulting file is saved in `/edoc/selected` as `selected_master.json`.

2.3.3 Analysis

Of the 68'345 items in `/edoc/raw`, all have a title (non-empty `title` data field), a little more than half of the items have an abstract (37'381 items with non-empty `abstract` data field), roughly half of the items have an ID (35'355 items with non-empty `id_number` data field),[1] and less than 10% of the items have keywords (6'660 items with non-empty `keywords` data field). The sample data set as requires all the above data fields to be non-empty; `/edoc/sample/sample_master.json` has 4'111 items and hence constitutes 6% of the raw data.

In order to determine how well the sample data set represents the raw Edoc data, a one-sample χ^2 goodnes of fit test was conducted on each selection data field (following Parke 2013, chap. 1). The results indicate that the sample data proportions of items are significantly different from the raw Edoc data per department (see Figure 1 for more details).

The sample data set is hence not representative of the raw Edoc data. This is not surprising since its construction is strongly biased. This bias has the effect that the sample data set is significantly skewed towards English journal publications in the sciences, medicine and economics from the 21st century (again, this is shown by a one-sample χ^2 goodness of fit test, see Figure 2 for more details). The upshot of this analysis is that the humanities are underrepresented in the sample data set. Therefore, any results with respect to the quality of machine indexing discussed below might not be applicable to the humanities.

2.4 Machine indexing

2.4.1 General overview

!! Give an overview over machine indexing.

2.4.2 Annif

!! Given an intro to Annif. Explain that we are using the out of the box version

---

[1] Note that when using the facet by blank on `id_number` in OpenRefine, there are 57'153 matches; this is due to the fact that many items with an ID such as DOI have secondary or tertiary IDs such as ISI or PMID.
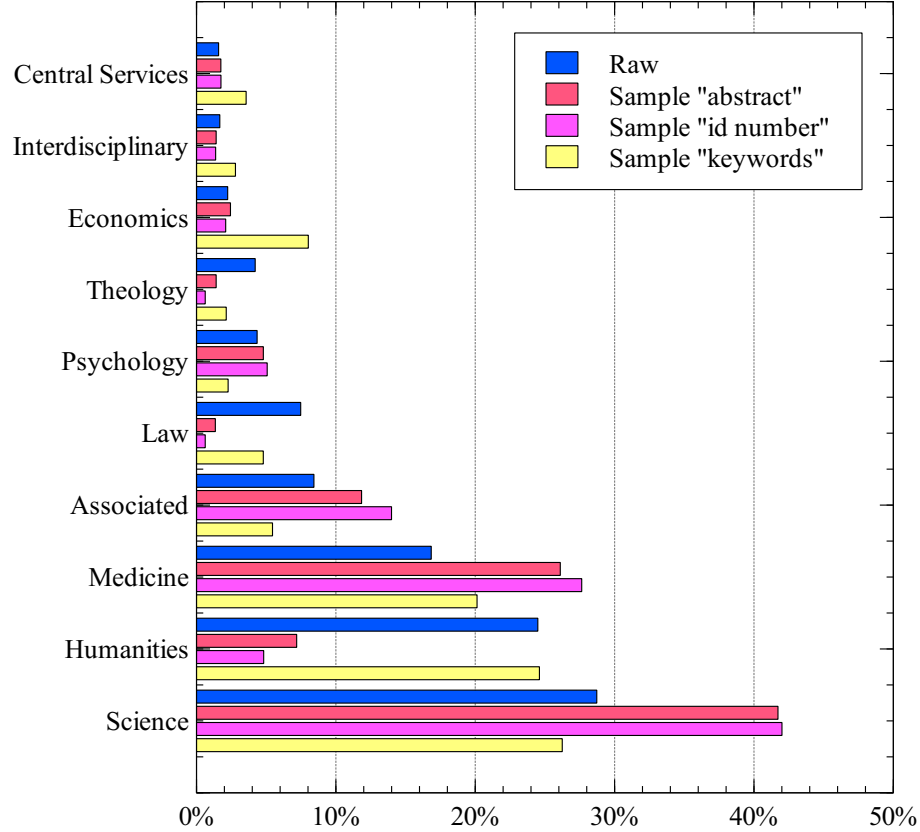
Figure 1: The sample data set ($n = 4$'111) is not representative of the raw Edoc data per department. Data field `abstract`: $\chi^2(df = 9) = 3$'160.556, $p < 0.001$; data field `id_number`: $\chi^2(df = 9) = 4$'209.0285, $p < 0.001$; data field `keywords`: $\chi^2(df = 9) = 2$'314.533, $p < 0.001$. The data foundation is available at `/edoc/analysis/chi_square...` and the χ^2-statistic can be calculated by calling `_Analysis.print_chi_square_fit` on the data foundation files.

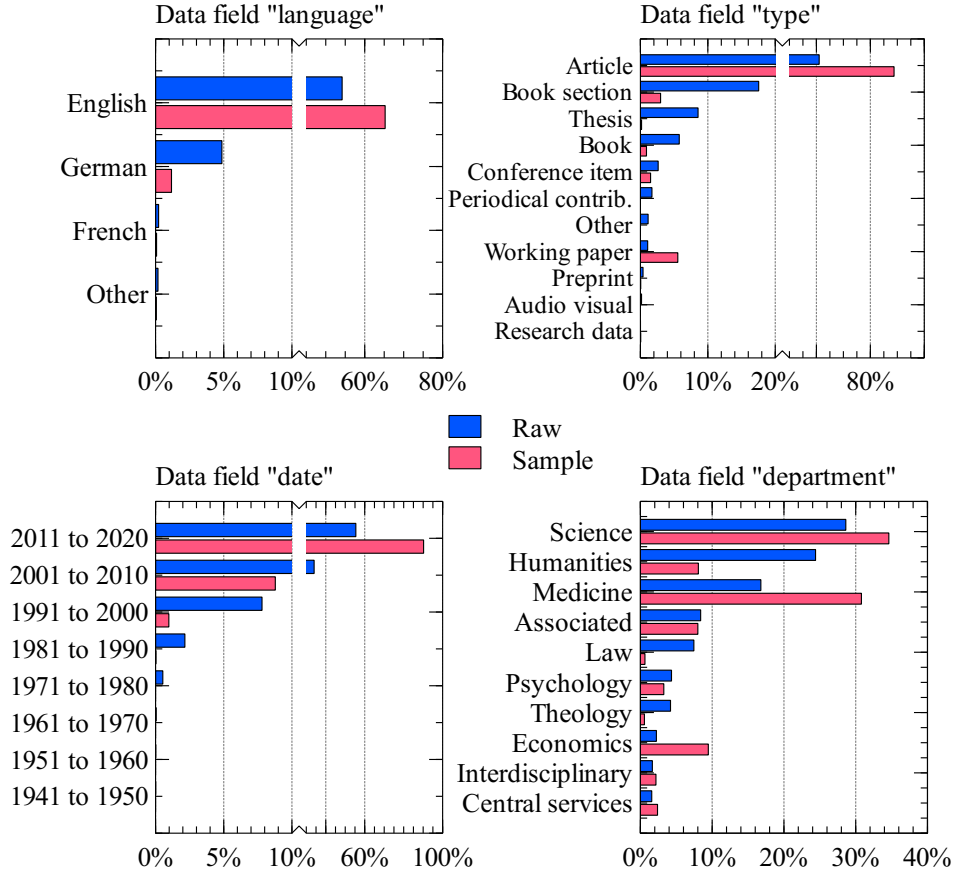Figure 2: The sample data set ($n = 4$'111) is significantly skewed towards English (data field `language`: $\chi^2(df = 3) = 1$'441.414, $p < 0.001$) journal publications (data field `type`: $\chi^2(df = 10) = 52$'519.743, $p < 0.001$) in the sciences, medicine and economics (data field `department`: $\chi^2(df = 9) = 14$'447.276, $p < 0.001$) from the 21st century (data field `date`: $\chi^2(df = 7) = 35$'878.493, $p < 0.001$) as compared to the raw Edoc data. The data foundation is available at `/edoc/analysis/chi_square...` and the χ^2-statistic can be calculated by calling `_Analysis.print_chi_square_fit` on the data foundation files.

2.4.3 Implementation

As explained above, for the prototype we will use the out

!! Explain how to implementation works.

In addition to the already used algorithms we should also try https://ai.finto.fi/?locale=en

!! Somewhere here we talk about indexing based on title and/or abstract and/or fulltext. Fulltext is not yet implemented. Some observations to do so: - the link to the fulltext is constructed from the data fields `offical url` and `documebts - main`, e.g., `https  ://edoc.unibas.ch/79633/` + `1/` + `2020_18_Informed by wet feet_How do floods affect property prices.pdf` to get `https://edoc.unibas.ch/79633/1/2020_18_Informed by wet feet_How do floods affect property prices.pdf`

2.5 Gold standard

In this chapter I will construct two distinct gold standards in order to assess the quality of machine indexing the sample data set with Annif.
!! Say something about assessments of Annif that have already been carried out.

2.5.1 Definition

The most common approach for assessing the output of machine indexing is by systematically comparing it with a gold standard (sometimes also referred to as model or reference). Golub et al. (2016, 10) define a gold standard as "a collection in which each document is assigned a set of [subject terms] that is assumed to be complete and correct" where "*complete* means that all subjects that should be assigned to a document are in fact assigned, and *correct* means that there are no subjects assigned that are irrelevant to the content." Put inversely, if an item in the gold standard lacks subject terms describing its content, the assignment is not complete; and if an item in the gold standard has been assigned subject terms that are not relevant to its content, the assignment is not correct.

A gold standard is usually the product of manual indexing by "information experts, subject experts, or potential or real end users" (Golub et al. 2016, 10). This entails its own host of epistemic problems relating to objectivity and consistency of the assigned subject terms. Most importantly, however, the construction of a gold standard from scratch is very expensive. It is

therefore not an option for the project at hand. Rather, I will construct what could be called derivative gold standards by reusing indexing data that is already available. Here we can distinguish two distinct kinds of derivative gold standards: first, I will construct a derivative gold standard based on the author keywords available in the sample data set; call this the "native" gold standard. Second, I will construct a derivative gold standard based on indexing metadata available in repositories distinct from Edoc; call this the "foreign" gold standard.

There are other methods for assessing machine indexing quality besides comparison with a gold standard that are worth mentioning. These include an evaluation in the context of indexing workflows (Golub et al. 2016, 13ff.), the assessment of retrieval performance (Golub et al. 2016, 15–23.), and so-called model free assessments (Louis and Nenkova 2013).

2.5.2 Native gold standard

The construction of the native gold standard takes x steps:

1. Extract the keywords from the sample data set.

2. Clean the extracted keywords.
3. Reconcile the extracted keywords with. . .

The intended output of this transformation process is. . .

These steps are explained in more detail in what follows.

2.5.2.1 Extract keywords In a first step, the keywords must be extracted from the sample data set `/sample/sample_master.json`. Recall that we mandated a non-empty `keywords` data field for an item to be selected from the raw Edoc data (see subsection "Selection"). We can thus simply copy the information in the `keywords` data field on a per item basis. To do this, we call `_Keywords.extract_keywords` with the sample data set as argument and save the output as `keywords/keywords_extracted.json` like so:

```
keywords = _Keywords.extract_keywords(DIR + "/sample/sample_master.json")
_Utility.save_json(keywords, DIR + "/keywords/keywords_extracted.json")
```

2.5.2.2 Clean keywords In second step, a list of all single keywords must be created. In order to do so, let us consider now in more de-

tail the exact information extracted from the `keywords` data fields as per
`/keywords/keywords_extracted.json`. In Edoc, the `keywords` data field
of an item is a non-mandatory free text field that is filled in by the user
(usually one of the authors) who undertakes the data entry of an item to
Edoc. Even though the Edoc user manual specifies that keywords must be
separated by commas (Universität Basel, n.d., 8), this requirement is neither
validated by the input mask nor by an administrator of Edoc. Furthermore,
neither the manual nor the input mask provide a definition of the term
"keyword." A vocabulary or a list of vocabularies from which to choose the
keywords is also lacking. Taken together, these observations are indicative
of very heterogeneous data in the `keywords` data field. To wit, the items of
`/keywords/keywords_extracted.json` are strings where single keywords
are individuated by any symbols the user saw fit. So, for each item in
`/keywords/keywords_extracted.json`, the user input must be parsed into
single keywords.

Next the so parsed single keywords must be cleaned or normalized: we
want the keywords to follow a uniform format thereby joining mor-
phological duplicates such as "Gene," "gene," "gene_," "gene/," and
so on. Also, some keywords are in fact keyword chains, for example
"Dendrites/metabolism/ultrastructure." Keyword chains must be broken
into their component keywords and then parsed again. The reason for this is
that Annif only assigns flat keywords and not keyword chains.

`_Keywords.clean_keywords` is the implementation the parser while the re-
cursive cleaner is implemented by `_Keywords.clean_keyword`. To create the
desired list of clean keywords, saved as `/keywords/keywords_clean.json`,
we call `_Keywords.clean_keywords` with the extracted keywords as argu-
ment:

```
keywords_extracted = _Utility.load_json(DIR + "/keywords/keywords_extracted.json")
keywords_clean = _Keywords.clean_keywords(keywords_extracted)
_Utility.save_json(keywords_clean, DIR + "keywords/keywords_clean.json")
```

2.5.2.3 Analysis `/keywords/keywords_clean.json` has 36'901 entries
many of which are duplicates. We hence deduplicate and count the
number of occurrences of each keyword. This is achieved by calling
`_Keywords.make_histogram` on `/keywords/keywords_clean.json` and
saving the output as `/keywords/keywords_clean_histogram.json`:

14

```
keywords_clean = _Utility.load_json(DIR + "/keywords/keywords_clean.json")
keywords_histogram = _Keywords.make_histogram(keywords_clean)
_Utility.save_json(keywords_histogram, DIR + "keywords/keywords_clean_histogram.json"
```

An analysis of `/keywords/keywords_clean_histogram.json` shows that the lion's share of keywords has only one occurrence but that the total occurrences are predominantly made up of keywords with more than one occurrence (see Figure 3 for details).

To analyse the spread of keywords in the sample data set, the keywords per item are counted. To do so, we call `_Keywords.make_count` on `/indexed/indexed_master_mesh_enriched.json` and save the output as `/analysis/keywords_counted.json`:

```
sample_data_set = _Utility.load_json(DIR + "/indexed/indexed_master_mesh_enriched.jso
keywords_counted = _Keywords.make_count(sample_data_set)
_Utility.save_json(keywords_counted, DIR + "/analysis/keywords_counted.json")
```

The median number of keywords for an item in the sample data set is 6 with an IQR of 4 (min = 0, Q1 = 4, M = 6, Q3 = 8, max = 87). Of course, items in the sample data set with a number of keywords below the first and above the third quartile are highly suspect from a qualitative point of view: when it comes to subject indexing, some terms are required, but more is usually worse [!! source]. Items with too few or too many keywords will be discussed in more detail in section section Assessment

2.5.2.4 Reconciliation As explained in section X, Annif assigns index terms from a controlled vocabulary. If we want to assess the quality of the indexing via a gold standard, we must therefore ensure that the gold standard makes use of the vocabulary used by Annif. As seen in section Y, the relevant (English) vocabularies are Wikidata YSO. The next step in constructing the native gold standard is hence to match the extracted and cleaned keywords with keywords from YSO and Wikidata. This process is called "reconciliation" (see https://docs.openrefine.org/manual/reconciling) and the tool of choice for this task is OpenRefine (see section OpenRefine).

In this section, I will describe how the cleaned keywords were reconciled with Wikidata and YSO, and which additional steps for refinement were undertaken. In total, 2'104 data transformation operations were performed; the complete operation history is available as `/keywords/operation_history.json`.

Figure 3: In `keywords/keywords_clean_histogram.json`, the distribution of keywords is strongly skewed right (min = Q1 = M = Q3 = 1, max = 910). However, even though keywords with only one occurrence constitute over 75% of the total keywords, their occurrences constitute less than 35% of the total occurrences. The most common keywords with 50 or more occurrences are extreme outliers but make up almost 20% of the total occurrences.

The data from `/keywords/keywords_clean_histogram.json` was imported into OpenRefine. The `keyword` column was then duplicated and reconciled with Wikidata. Here the parameters were chosen as follows: reconcile against no particular type, and auto-match candidates with high confidence.

The reconciliation service API returns automatic matches (call them "suggestions"). A suggestion is correct if and only if the meaning of the keyword from `/keywords/keywords_clean_histogram.json` corresponds to the meaning of the suggested concept from Wikidata. Note that for homonymous or polysemous keywords, it is impossible to confirm a correct match without further context; those keywords therefore cannot be reconciled (but see section "Construction" for a possible solution). Unfortunately, random sampling showed that the overall quality of the reconciliation was not satisfactory, that is, there were too many incorrect suggestions. A two-pronged strategy was adopted to ameliorate the quality of the reconciliation.

First, the suggestions to the top 500 keywords were manually verified. These keywords account for 14'996 occurrences or 40.638% of the total occurrences and thus constitute an effective lever.

Second, systematic biases were identified and removed. The most prevalent bias was an due to a suggestion's type. As stated above, the reconciliation service API was not constrained by type but had access to the complete Wikidata database. Since Wikidata is an ontology that encompasses everything, it also features types whose concepts cannot qualify as subject terms (at least in the present context). The most prominent example is the type Q13442814 "scholarly article." Wikidata contains the metadata of many scholarly articles. Now, for some of our keywords, there is a scholarly article with a title that exactly matches the keyword; and since there is no restriction concerning the type, the scholarly article is suggested with high confidence (see https://github.com/OpenRefine/OpenRefine/wiki/Reconciliation-Service-API for details). For example, [keyword/article]. To generalize, suggestions with types whose concepts are proper names are usually incorrect. Based on this observation, suggestions with the following types were rejected: "scholarly article," "clinical trial," "scientific journal," "academic journal," "open access journal," "thesis," "doctoral thesis," "natural number." Suggestions with the following types were manually verified (i.e., checked for correctness): "human," "album," "film," "musical group," "business," "literary work," "television series," "organization," "family name," "written work," "video game," "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."

With these improvements in place, each keyword with a suggestion was assigned a QID via the "Add cloumns from reconciled values"-function (and similar for YSO and MeSH identifiers). The data was then exported and saved as `/keywords/keywords_reference.json`. Keywords with a QID now constitute 69% of all keywords and 78% of their total occurrences in the sample data set, but these numbers are significantly lower for the MeSH identifier (53.6% of all keywords with only 28.8% of all occurrences) and especially low for the YSO identifier (26.9% of all keywords with 11% of all occurrences). In both cases, the problem is due to the fact that Wikidata's mapping of MeSH respectively YSO is only partial. There can be two reasons for this state of affairs for a given entry: either there is no match between Wikidata and YSO or MeSH (after all, Wikidata is much larger than MeSH and YSO taken together), or there is a match but it has not yet been added to the mapping. In the latter case, at least with respect to YSO, there is a solution: Finto provides a REST-sytle API to access the YSO vocabulary directly (see https://api.finto.fi/). For each keyword in the list of reference keywords that lacks a YSO identifier, the Finto API is queried; if a term turns up, it is added to the keyword. The effect of this second reconcilement is detailed in Figure 4. It is achieved by calling `_Keywords.enrich_with_yso` on `/keywords/keywords_reference.json` and saving the output as `/keywords/keywords_reference_master.json`

```
keywords_reference = _Utility.load_json(DIR + "/keywords/keywords_reference.json")
keywords_reference_new = _Keywords.enrich_with_yso(keywords_reference)
_Utility.save_json(keywords_histogram, DIR + "keywords/keywords_reference_master.json
```

Finally, consider the distribution of the cleaned and reconciled keywords per item in the Edoc sample data set. The corresponding data is generated with `_Keywords.make_count` as described in subsection Analysis above and available as `/analysis/keywords_counted.json`. Figure 5 shows that the median number of keywords from MeSH or YSO might be too low to be adequate. This problem can be amended by imposing further constraints on the sample data set and such a solution is discussed in section [!! section].

2.5.2.5 Discussion Let us now turn to the evaluation of the reconciliation: how many of the suggestions were correct? This question
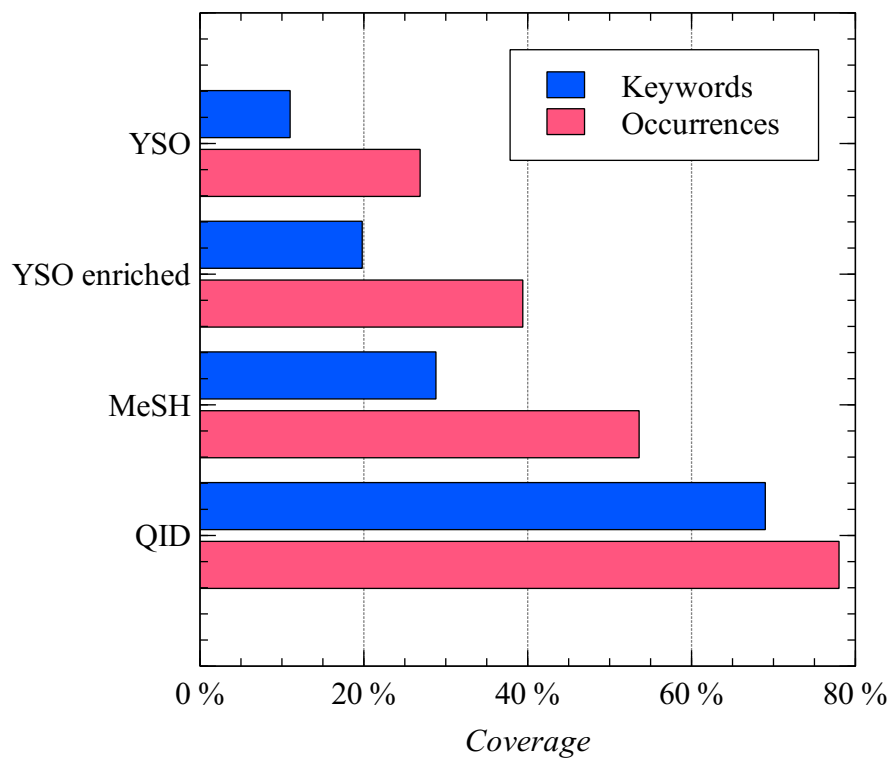
Figure 4: The coverage of `/keywords/keywords_reference_master.json` by the controlled vocabularies of Wikidata (QID), Medical Subject Headings (MeSH), and YSO (General Finnish Ontology). Both MeSH and YSO are dependent on QID but independent of each other. YSO enriched is the superset of YSO created by reconciling keywords that lack a YSO identifier directly with the YSO database provided by the Finto API; it is hence independent of Wikidata.
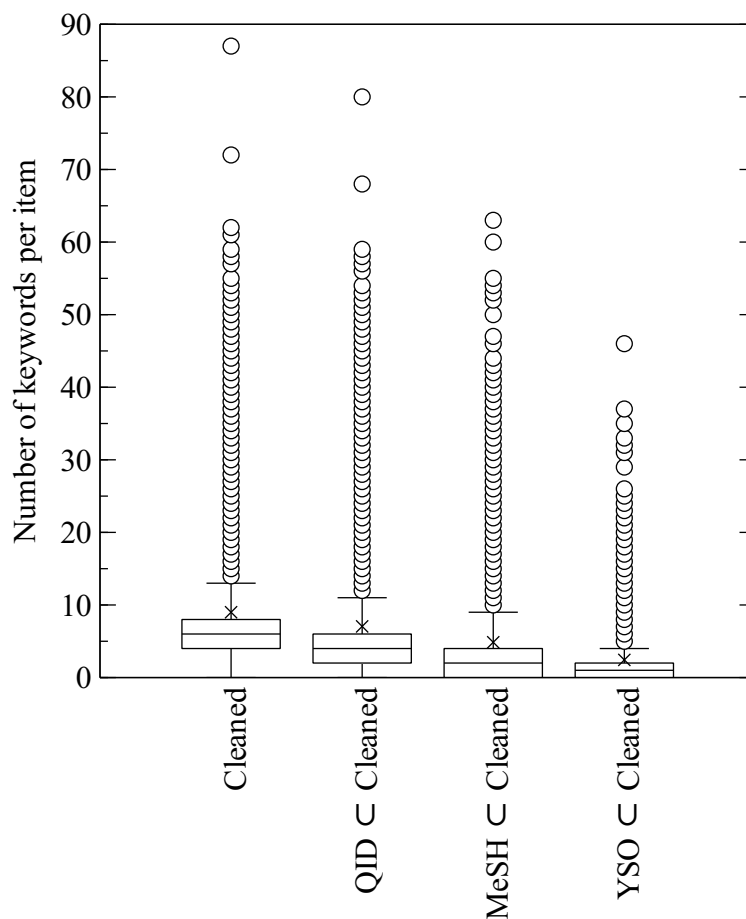
Figure 5: The distribution of keywords per item in the Edoc sample data set. The leftmost boxplot shows the distribution of cleaned keywords (min = 0, Q1 = 4, M = 6, Q3 = 8, max = 87); the other boxplots show the distribution of cleaned keywords with reconciled QID (min = 0, Q1 = 2, M = 4, Q3 = 6, max = 80), MeSH ID (min = Q1 = 0, M = 2, Q3 = 4, max = 63), and YSO ID respectively (min = Q1 = 0, M = 1, Q3 = 2, max = 46). All four distributions are similarly consistent, but there is a linear shrinkage of the center leaving MeSH and YSO with potentially too few descriptors to represent an adequate indexing.

was answered via random sampling (following Roth and Heidenreich 1993, 204–25). The random sample ($n = 500$) was created by calling `_Analysis.make_random_sample` on `/keywords/reference_keywords_master.json`; it is available at `/analysis/random_keywords.json`. The sample was then imported into OpenRefine and the judgement (1 for correct, 0 for incorrect) of the manual verification was recorded in the column `verification`. The data was then exported and is available at `/analysis/random_keywords_verified.csv`.

An analysis of this data shows that 53% of the suggestions in the random sample were correct with a 95% confidence interval of 48.8% to 57.3%. We can therefore conclude that 8'507 ± 692.1 of the 16'050 keywords in `/keywords/reference_keywords.json` have correct suggestions. Note that of the 235 incorrect suggestions in the random sample, 188 were incorrect by default because they were missing a QID; the share of non-empty yet incorrect suggestions is only 9.4% in the random sample meaning that the quality of the reconciliation is not as disappointing as it might seem at first glance.

2.5.3 Foreign gold standard

- !! Explain MeSH
- !! Explain how to get MeSH for subset of sample data set

2.6 Assessment

In this chapter I will assess the quality of the sample data set's indexing with Annif based on the native and foreign gold standards.

2.6.1 Precision, recall, F1-score

The metrics used for the assessment are precision, recall and F1 score. Precision and recall are standard metrics for indexing quality (e.g., Gantert 2016, 197) whereby the F1 score plays are more prominent role in the assessment of machine indexing (e.g., Suominen 2018, 11–14; Toepfer and Kempf 2016, 93f.). Of course, there is a host of alternative metrics (such as indexing consistency, transparency, reproducability) that are neglected here.

Let us briefly look at the definitions and motivations of the chosen metrics. Remember that a suggestion of a subject term is correct if and only if the subject term is in the native gold standard. The possible outcomes are summarized in Table 1.

	Suggested by Annif?	
	No	Yes
No	True negative	False positive
Yes	False negative	True positive

In gold standard?

Table 1: Annif confusion matrix.

2.6.1.1   Precision   "Precision" is the fraction of the correctly suggested subject terms; a suggestion is correct if and only if it is in the native gold standard:

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}}$$

Or put as question: what fraction of the subject terms suggested by Annif are also in the native gold standard?

2.6.1.2   Recall   "Recall" is the fraction of correct subject terms out of all correct subject terms:

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}}$$

Put as question: what fraction of the subject terms in the gold standard were suggested by Annif?

2.6.1.3   F1-score   The F1-score is the harmonic mean between precision and recall:

$$\text{F1} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.6.1.4   Implementation   The above scoring metrics were implemented using the scikit-learn machine learning library (Pedregosa et al. 2011). It is important to note that each metric comes in three different flavors dubbed "macro," "micro," and "weighted" respectively. In the macro flavor, the metric represents simply the mean per class (i.e., correct or incorrect suggestion). The weighted metric is the macro metric but each class is weighted by its true positives. By contrast, a metric with the micro flavor is computed globally over all true positives and false positives respectively negatives.

2.6.2 Annif versus native gold standard

In this section, I will assess the perfomance of Annif versus the native gold standard.

2.6.2.1 Creating the data foundation I will now describe how the data foundation for the assessment was created. There are three parameters that need to be distinguished (see section "Annif"):

1. The Annif project (algorithm plus vocabulary) responsible for the indexing of the sample data set. As per the Annif REST API, these are `yso-en`, `yso-maui-en`, `yso-bonsai-en`, `yso-fasttext-en`, and `wikidata-en`.
2. The text base per item, namely title versus title and abstract.
3. The maximum number of suggestions per item. Since we required 10 suggestions per item, we can choose between 1-10 suggestions.

Combining these parameters, we really have 100 Annif configurations whose perfomance we want to compare and assess: Annif project $*$ text base $*$ maximum number of suggestions $= 5 * 2 * 10 = 100$.

For each configuration, the F1-score (macro, micro and weighted) was computed. To do so, we call `_Analysis.super_make_metrics` on `/indexed/indexed_master_mesh_enriched`. The 100 output files are saved as `/metrics/metrics_{marker}` where `marker` specifies the Annif configuration; a single file for analysis is saved as `/analysis/metrics.json`:

`_Analysis.super_make_metrics(DIR + "/indexed/indexed_master_mesh_enriched.json")`

Note that if an item had an empty native gold standard, no score was computed; this is the case exactly if none of the cleaned keywords had been matched to Wikidata or YSO respectively. Configurations with a Wikidata vocabulary had a scoring coverage of 94.38% of the items in the sample data set as compared to only 62.84% for configurations with a YSO vocabulary.

2.6.2.2 General results we can then discuss whether the best perfomance is actually useful; perhaps do this by comparing absolute scores with other assessments. if results are much lower it is to be suspected that the native gold standard is not very good (i.e., that the measure we use to judge annif performance is inadequte => motivation for foreign standard)

construct all metrics: __Analysis.super_make_metrics(file_path=DIR + "/indexed/indexed_master_mesh_enriched.json")

construcht all stats for metrics: __Analysis.super__make__stats()

2.6.2.3 YSO configurations

2.6.2.4 Wikidata configurations

2.6.3 Annif versus foreign gold standard

!! but to do this we would first have to map the annif suggested terms to mesh...! yes.

2.6.4 Digression: truth-conditions for indexing

(Correspondence theory of indexing: subject term T fits text X iff text X is about T) ### Targets !! Find out what I meant here.

2.6.5 Analysis

2.7 Discussion

3 Outlook

3.1 Refinement

3.2 Implementation strategy

3.3 Other use cases

4 Conclusion

!! Clean up: rename `/indexed/indexed_master_mesh_enriched.json` to `/indexed/master_final.json` or something short.

5 Appendix

6 Bibliography

Gantert, Klaus. 2016. *Bibliothekarisches Grundwissen.* 9., vollständig neu bearbeitete und erweiterte Auflage. Berlin: De Gruyter Saur. https://doi.org/10.1515/9783110321500.

Golub, Koraljka, Dagobert Soergel, George Buchanan, Douglas Tudhope, Marianne Lykke, and Debra Hiom. 2016. "A Framework for Evaluating Automatic Indexing or Classification in the Context of Retrieval." *Journal of the Association for Information Science and Technology* 67 (1): 3–16. https://doi.org/10.1002/asi.23600.

Louis, Annie, and Ani Nenkova. 2013. "Automatically Assessing Machine Summary Content Without a Gold Standard." *Computational Linguistics* 39 (2): 267–300. https://doi.org/10.1162/COLI%7B/textunderscore%2 0%7Da%7B/textunderscore%20%7D00123.

Parke, Carol. 2013. *Essential First Steps to Data Analysis: Scenario-Based Examples Using SPSS.* Thousand Oaks: SAGE Publications. https://doi.org/10.4135/9781506335148.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research*, no. 12: 2825–30. https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?source=post_page-------------------------.

Roth, Erwin, and Klaus Heidenreich. 1993. *Sozialwissenschaftliche Methoden: Lehr- Und Handbuch für Forschung Und Praxis.* 3., völlig überarbeitete und erweiterte Auflage. München: R. Oldenbourg Verlag.

Stokhof, Martin, and Michiel van Lambalgen. 2011. "Abstractions and Idealisations: The Construction of Modern Linguistics." *Theoretical Linguistics* 37 (1-2): 1–26. https://doi.org/10.1515/thli.2011.001.

Suominen, Osma. 2018. "Annif: DIY Automated Subject Indexing Using Multiple Algorithms." *LIBER Quarterly* 29 (1): 1–25. https://doi.org/10.18352/lq.10285.

Toepfer, Martin, and Andreas Oskar Kempf. 2016. "Automatische Indexierung Auf Basis von Titeln Und Autoren-Keywords – Ein Werkstattbericht." *027.7 Zeitschrift für Bibliothekskultur / Journal for Library Culture* 4 (2): 84–97. https://0277.ch/ojs/index.php/cdrs_0277/article/view/156/354.

Universität Basel. n.d. "Research Database of the University of Basel User Manual." Edited by Universität Basel.