Building and Serving the Queerlit Thesaurus as Linked Open Data

Arild Matsson¹, Olov Kriström¹

¹The University of Gothenburg, Box 100, 405 30 Gothenburg, Sweden

Abstract
This paper describes the creation of the Queer Literature Indexing Thesaurus (QLIT) as well as the digital infrastructure supporting the workflow for editing and publishing it. The purpose of QLIT is to adequately catalogue Swedish fiction with LGBTQI themes. It is continually edited in plain-text RDF and automatically processed for correctness and storage. Finally, it is published online as Linked Open Data and used with external systems. The technical approach relies on scripts and applications developed ad hoc, rather than existing solutions. Code is available on https://github.com/gu-gridh/queerlit-terms

Keywords
thesaurus, queer, Swedish, Resource Description Framework (RDF), Linked Open Data (LOD)

1. Background

1.1. Queerlit

The Queerlit project aims to identify and index Swedish fiction with LGBTQI themes from the 7th century and onwards. The process of indexing involves assessing a single literary work and choosing suitable subject headings which are then added to the bibliographical record of the work.

Subject headings are organized in a thesaurus, a controlled vocabulary with a hierarchical structure. Subject headings are arranged as broader, more general concepts, and narrower, more specific concepts. In this paper, the term subject heading itself is sometimes also referred to as concept or term.

As a part of the Queerlit project, librarians and literary scholars are together constructing a new thesaurus: Queer Literature Indexing Thesaurus (QLIT). This is because existing applicable thesauri are deemed unsuitable for this bibliography, as they often reflect a majority-population (e.g. heterosexual, cisgender) system of concepts. As such, they fail to express the topics and nuances that scholars within queer (Swedish) literary studies would require to appropriately orient themselves within the bibliography. [1, 2, 3, 4, 5]

As an example, in the major Swedish thesaurus Svenska ämnesord (SAO), there is only a single term relating to intersex people (Intersexualism). The current version of QLIT, on the
other hand, provides 22 different terms relating to intersex people.

The project runs in the years 2021–2023 and is further described on https://www.gu.se/en/research/queerlit-data-base. As of May, 2023, the project has identified and included around 1700 literary works in its bibliography, and indexed around 950 of these using mainly QLIT.

1.2. Libris XL

The National Library of Sweden (Kungliga biblioteket, KB) keeps bibliographical records (as well as physical copies) of literature published in Sweden. As a participant in the Queerlit project, they ensure the indexing work is being performed and recorded in accordance with national standards in the Libris XL cataloging system. Libris XL uses the BIBFRAME data framework [6, 7] which builds on well established global standards for data interoperability, supported also by the Library of Congress. It is implemented with Linked Open Data (LOD). Through Libris XL, other libraries can get access to the QLIT-enriched bibliographical records and implement them in their local catalogs.

As a step in the ongoing development of the cataloging system, KB recommended for the purpose of the Queerlit thesaurus that its content be defined externally to Libris XL. LOD then provides mechanisms to import this structured data into Libris XL. The requirement upon the Queerlit project is then to expose the subject headings as LOD-compatible data, specifically in the Resource Description Format (RDF).

1.3. Homosaurus

QLIT is based on the Homosaurus, a large thesaurus with a similar purpose [8]. The Homosaurus is created in an international or US-American context, with labels in English, so extensive work was put into adapting it to Swedish conditions.

2. Building the thesaurus

The thesaurus has been built in two phases. In the first, it was created based on the Homosaurus, with translation and broad adjustments. The workflow was then reshaped after the development of the data processing scripts, described in Section 3, and the web server and frontend application, described in Section 4.2. All project members could now easily explore the terms and their hierarchy, give feedback and engage in cataloging work. Thus, in the second phase, smaller adjustments have been made continually.

2.1. Thesaurus creation

Choosing the Homosaurus as the basis of constructing QLIT was based on the fact that it is the largest LGBTQ-themed thesaurus, with both institutional users such as The National Archives in the UK and Duke University Library, and community resources like the Australian Queer Archives and the German QueerSearch. In 2019 it went through a major revision, re-launching as a linked data vocabulary with updated terminology making it more explicitly inclusive of queer and trans topics, which suited the aims of Queerlit.
Since the Queerlit project focuses on fiction, and the Homosaurus is designed as a more general thesaurus, applicable for both fiction and non-fiction, a selection was made of the Homosaurus terms used for indexing fiction at two major institutions using the thesaurus – IHLIA LGBTI Heritage and The Transgender Digital Archives. This selection resulted in a subset of 500 terms (out of the total 1647 terms in Homosaurus v2.2). Broader and narrower terms to this selection were included, resulting in a list of 1146 terms which were downloaded as Turtle RDF files [9] and formed the basis for the translation and adaptation.

Initially the work of translating was done manually on a print-out of the list of selected terms, but the Turtle files were soon imported to Visual Studio Code, with an RDF plugin, for text editing. The translations were then made directly in the Turtle files. A benefit of keeping the print-out, however, was that it proved handy as a perspicuous reference for quickly looking up terms.

A benefit of using the Turtle files from the Homosaurus as templates for creating the files for QLIT was that mapping exact and close matches between the two thesauri was easy – a script creating skos:exactMatch values from the Homosaurus identifiers was made, and in a case-by-case review some were revised to skos:closeMatch. A mapping to the Library of Congress Subject Headings (LCSH) thesaurus was also included with the Turtle files from the Homosaurus – these were also subject to case-by-case review, not only in order to adjust exact matches to close matches, but also in order to correct some misleading matches (mainly having to do with homonyms, where for instance the Homosaurus had an exact match between the term Bears, denoting members of a gay subculture, and the LCSH term Bears, denoting members of the mammalian ursidae family). These corrections were shared with the Homosaurus editorial board for them to update.

Mappings were also made to the two main Swedish library thesauri, Svenska ämnesord (SAO) and Barnämnesord (Barn). Based on the experience of faulty homonymic matches from the Homosaurus, this mapping was done manually rather than by an automated process.

Further editorial work was done in deciding what terms needed to be removed, adapted or added based on cultural context. For example terms relating to the anglophone distinction of sex/gender were merged in the broad Swedish concept of Kön, with narrower terms making distinctions of biological, social, legal and subjective aspects, and a term for Sami LGBTQI people was added as a narrower term in relation to Indigenous LGBTQI people.

During the main phase of translating and revising, the identifiers in the Turtle files were handled as plain text, written in camel case (e.g. dcterms:identifier äldreHBTQIPersoner for the preferred term Äldre HBTQI-personer (“Older LGBTQI people”)). This facilitated overview and ready comprehension, but in order to safeguard future term changes, these were substituted with randomized, non-semantic identifiers before launch (e.g. dcterms:identifier ”eg84dq15” for the preferred term Äldre HBTQI-personer).

Files were kept on Sharepoint and shared with other project members. A more easily navigated visual representation (see Section 4.2) was launched on a website for use by the project members in February 2022, making collaborative input easier. This also facilitated workshops with groups of end users, who could be presented with the thesaurus in a preliminary design. In October 2022 the thesaurus section of the webpage was made public, allowing an even broader public the possibility of leaving feedback, both regarding the structure and content of the thesaurus and its visual design.
2.2. Continual editing

When the thesaurus was implemented as linked data in the cataloging platform of Libris XL, a second phase of work was initiated. Based on the experiences of indexing the literature revisions were suggested and ideas for further additions were made by the project members. Uploads of revised versions to Libris XL have been made on a semi-regular basis, depending on the rate of changes made.

At the moment of writing QLIT contains 880 terms.

2.3. RDF data model

The RDF data model for QLIT is based on the existing model of the Homosaurus, from which the class skos:Concept and the following properties were transposed:

- dcterm:identifier
- skos:broader
- skos:narrower
- dcterm:issued
- skos:closeMatch
- skos:prefLabel
- dcterm:modified
- skos:exactmatch
- skos:related
- skos:altLabel
- skos:inScheme

After dialogue with KB the property rdfs:comment, which is used in the Homosaurus, was not implemented in QLIT, where we instead integrated the property skos:scopeNote. The reason for this was that the praxis of KB is to treat skos:scopeNote as a general field for both definition and usage guiding, and use of rdfs:comment would have been mapped to this in the Libris XL environment.

Further additions were made with the use of the properties skos:hiddenLabel, in order to allow common misspellings from end users of the search interface, and skos:Collection, as a way to be able to visually organize the thesaurus in a more browsable design, where the (currently) 143 top terms are grouped into 9 thematic categories.

A diagram of the current data model is seen in Figure 1.

3. Data processing

The RDF data created by hand is not guaranteed to be complete or entirely correct. Therefore, the edited Turtle files are synced from the Sharepoint area and input to a script for automatic processing. The script, written in Python, can effortlessly be repeated for each iteration of changes to be published. The steps of the script are described in this section.

The script, named build.py, is available in the queerlit-terms repository on GitHub: https://github.com/gu-gridh/queerlit-terms.

3.1. Input validation

The first step of the script is to read each input file, validate its content and then add it to a compound RDF graph representing the full thesaurus. This validation step primarily makes sure that the Turtle syntax is valid.
Figure 1: The RDF data model used in QLIT.

After all input files are read and added to the same graph, another validation step checks that there are no relationships to missing nodes, and that the skos:identifier value of each term matches its URI.

3.2. Generating randomized identifiers

Each term URI is composed of the thesaurus URI and a term identifier. Early on, identifiers were based on the term label (as mentioned previously, e.g. “äldreHBTQIPersoner” for the term labeled Äldre HBTQI-personer). This has the advantage that the term is easily identifiable by human readers. However, labels may be subject to change, while identifiers should remain constant. Thus, we switched to creating identifiers as randomized strings.

To generate identifiers, the Python library StringGenerator[10] was used with a pattern of four lowercase letters and four numbers, for instance: “mg27td65”. Resulting identifiers have no meaningful connection to the terms to which they are assigned. The pattern allows 4.5 billion different combinations, equivalent to an entropy of approximately 32 bits. For the unlikely event that a newly generated identifier will already exist for another term, an automated check is in place to retry generation upon such collision.
3.3. Inferring relationships and properties

After loading input files into a unified graph, some trivial changes are made automatically. Most importantly, the \texttt{skos:ConceptScheme} node representing the QLIT thesaurus is created, and relationships between it and the terms are added.

Relationships within the thesaurus – \texttt{skos:broader}, \texttt{skos:narrower} and \texttt{skos:related} – are automatically inferred. This means that if \textit{A} has a \texttt{skos:broader} relationship to \textit{B}, the script ensures that there is also a \texttt{skos:narrower} relation from \textit{B} to \textit{A}.

The SKOS standard leaves some freedom as to how the \texttt{skos:topConceptOf} property and its counterpart \texttt{skos:hasTopConcept} are applied \cite{11}. We simply define that a top concept is any term that does not have any \texttt{skos:broader} relationships to another term. The scripts sets these properties accordingly.

3.4. Detecting changes

The result after merging, validating and inferring data is automatically compared to the previous version of the thesaurus. For any added terms, the \texttt{dcterms:issued} property is set to the current time. At release-time, the \texttt{dcterms:modified} property was also set to the same time. From that point on, for any terms with changes, the \texttt{dcterms:modified} property is updated to the current time. This includes both terms in the case where a relationship is added or removed.

```
$ python3 build.py
Parsing 889 files...
Parsed 889 files
Creating new identifiers...
New id cb53to20 for latinesHBTQI
Completing relations...
Checking changes...
Changes for ug69ck69 in narrower
Changes for aa40zb14 in related
Changes for nj44aa03 in narrower
Changes for ho87o142 in narrower
4 changed, 1 new, 1 removed
Writing 889 terms...
Wrote qlit.nt
```

Listing 1: Sample output from the processing script, summarizing changes made since the previous iteration of the data.

3.5. Version control

At this stage, the RDF graph still exists only transiently in the computer memory. In order to be stored and transferred, it is serialized into the N-Triples format \cite{12}, in which triples are output
in full and separated by newlines. This output is, in turn, sorted alphabetically.

The sorted N-Triples dump is committed to the queerlit-terms git repository. Version control provides long-term storage and facilitates collaboration between developers. Additionally, Git provides a convenient mechanism to diff (i.e. compare) the new output against the previous iteration. We do this routinely as a quick sanity check before committing the output.

There are several serialization formats for RDF, some of which are generally preferred over N-Triples as they are smaller in file size or easier for humans to read. However, they are not quite as easy to sort. Without sorting, the diffs against previous iterations can be less easily interpretable.

3.6. Version number

Versioning of the thesaurus can be considered on two levels. First, the explicit version number is fixed to “v1”, and this is part of the thesaurus URL. This is not actually expected to change, but it does provide room for a large-scale re-instantiation of the thesaurus, should the need appear in the future. Second, a small-scale “version” of the thesaurus is implicit in the dcterms:modified term property.

Furthermore, the publicly available Git commit history shows in greater detail what changes have been made, but this is more of a by-product, not intended as part of the thesaurus per se.

```turtle
@base <https://queerlit.dh.gu.se/qlit/v1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<rf38ex77> a skos:Concept ;
    dcterms:identifier "rf38ex77" ;
    dcterms:issued "2022-05-19T10:12:00"^^xsd:dateTime ;
    dcterms:modified "2022-12-07T08:49:58"^^xsd:dateTime ;
    skos:broader <iu41ao66> ,
        <nf03ub67> ;
    skos:exactMatch ← <http://id.loc.gov/authorities/subjects/sh85061777> ;
    skos:inScheme <https://queerlit.dh.gu.se/qlit/v1> ;
    skos:narrower <pn00dm58> ,
        <y106dh66> ;
    skos:prefLabel "Homosexuella par" ;
    skos:related <vu78ao12> ;
    skos:scopeNote "Används för skildringar av parrelationer mellan
        homosexuella." .
```

Listing 2: The RDF (Turtle) representation of the term *Homosexuella par* ("Homosexual couples") as a result after automatic processing.
Table 1
RDF routes

<table>
<thead>
<tr>
<th>Route name</th>
<th>URL</th>
<th>Content (RDF/Turtle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDF All</td>
<td><a href="https://queerlit.dh.gu.se/qlit/v1/">https://queerlit.dh.gu.se/qlit/v1/</a></td>
<td>The full QLIT thesaurus</td>
</tr>
<tr>
<td>RDF One</td>
<td><a href="https://queerlit.dh.gu.se/qlit/v1/">https://queerlit.dh.gu.se/qlit/v1/</a>&lt;identifier&gt;</td>
<td>A single QLIT subject heading</td>
</tr>
</tbody>
</table>

4. Web server

A lightweight web server application was created to serve the data online for two purposes. Firstly, the full RDF data needs to be made available for importing into Libris XL. Secondly, the Queerlit website (https://queerlit.dh.gu.se/) needs to query and navigate the data. These parts are detailed in each of the following two subsections.

The application loads the version-controlled data dump and serves a few different routes under the URL “https://queerlit.dh.gu.se/qlit/v1/”. It is written in Python, like the data processing script, and it lives in the same code repository. It uses the web server library Flask.

4.1. RDF server

The URLs for the skos:ConceptScheme node representing the QLIT thesaurus, as well as for all the skos:Concept nodes representing QLIT subject headings, are designed to be valid URLs, resolving to their RDF representations. The server application is then configured to serve under the first part of the URL (“https://queerlit.dh.gu.se/qlit/v1/”) and designed to resolve the last part of the URL (the empty string or a subject heading identifier) to the skos:ConceptScheme or a skos:Concept.

While N-Triples was used as output format for the version-controlled data dump, Turtle is chosen in this case for its readability. The RDF All route is in principle just a re-serialization of the data dump into Turtle.

The RDF content of these routes is intended as the canonical, “source of truth” variant of the QLIT thesaurus to the outside world.

4.2. JSON server for frontend application

Another goal of the Queerlit project is creating a website for searching the Queerlit bibliography and browsing the QLIT thesaurus. It is a frontend-only application which runs in the user’s web browser and fetches data primarily from the Libris XL web API.

Thesaurus data could also be fetched from Libris XL, but is instead fetched from the QLIT server application for two reasons. First, the QLIT data in Libris XL is in theory only a cached version of the canonical data served by us. Second, we can enrich the data with labels from matching Homosaurus terms. Third, the output format is designed to be easy to use in the frontend application code.

Each route corresponds to a method of traversing the RDF graph and identifying a set of terms. Each term is translated to a Python object with properties corresponding roughly but not exactly to RDF triples connected to the term.
Table 2
JSON routes

<table>
<thead>
<tr>
<th>Route name</th>
<th>Content (JSON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>List of all terms in no specific order</td>
</tr>
<tr>
<td>One</td>
<td>One term</td>
</tr>
<tr>
<td>Autocomplete</td>
<td>List of terms matching a given search string</td>
</tr>
<tr>
<td>Roots</td>
<td>List of all terms having no skos:broader relation</td>
</tr>
<tr>
<td>Children</td>
<td>List of all terms having a skos:broader relation to a given term</td>
</tr>
<tr>
<td>Parents</td>
<td>List of all terms having a skos:narrower relation to a given term</td>
</tr>
<tr>
<td>Related</td>
<td>List of all terms having skos:related relation to a given term</td>
</tr>
</tbody>
</table>

Figure 2: A screenshot from the thesaurus section of the Queerlit website (https://queerlit.dh.gu.se/subjects).

The Homosaurus is loaded as a separate RDF graph. For QLIT terms with an skos:exactMatch relation to a Homosaurus term, the term is looked up and the skos:prefLabel and skos:altLabel values are added to the Python object.

The list of objects (or single object, in the case of the One route) are serialized to the JSON format, chosen because of its easy and widespread use in frontend applications.

The API is openly accessible in practice, but not intended for this purpose. It is not publicly documented, and changes are not announced.
5. Discussion

5.1. Custom implementation

As detailed in this paper, the process has involved custom implementation for data validation, processing and web publication. Writing our own code, rather than employing existing available software, has enabled solving specific problems without adding irrelevant cruft. However, as the project evolves, there is an increased probability that problems get more complex and alike those that existing software is solving.

Some solutions which are already openly available, and which could perhaps have been used instead of custom code, are:

- Editor applications specialized for RDF (e.g. Protégé [13]). We started working with a non-specialized text editor because it was easier to get started. Investing time in learning a new application might have saved us time in the long run, but we did not consider it until later in the process. Any mistakes that would have been avoided with a specialized editor (e.g. relations to non-existing nodes) were detected and corrected in the input validation process.

- Applications for publishing RDF online (e.g. Skosmos [14]). We considered existing applications, but writing a custom application was not a very time-consuming alternative. It also gave us the possibility to write endpoints suitable for use with the search website.

- Applications for visualization of the thesaurus. We wanted to integrate this visually with the search website, which is also designed and implemented as part of the research project. We made the judgment that employing existing visualization tools with their own design would break the user experience, and overriding their design would be time-consuming.

References

[7] Vad är Libris och XL?, ????.
URL: https://libris.kb.se/katalogisering/about.

[8] About, ????.
URL: https://homosaurus.org/about.

URL: https://www.w3.org/TR/turtle/.

URL: https://github.com/paul-wolf/strgen.

URL: https://www.w3.org/TR/skos-reference/#schemes.

URL: https://www.w3.org/TR/n-triples/.

doi:10.1016/S1071-5819(02)00127-1, place: LONDON Publisher: Elsevier Ltd.