Enabling Ontology-Based Data Access to Project Gutenberg

Mattia Egloff[1][0000−0002−1740−519X], Alessandro Adamou[2][0000−0002−9272−908X], and Davide Picca[1][0000−0003−2014−0855]

[1] University of Lausanne, Switzerland {firstname.lastname}@unil.ch
[2] Data Science Institute, The Insight Centre for Data Analytics, NUI Galway, Ireland alessandro.adamou@insight-centre.org

Abstract. Free and open digital libraries have been gaining steady momentum as key resources to support practice in Digital Humanities. Project Gutenberg is one of the oldest repositories of such a kind. The DHTK Python library is able to retrieve content from Gutenberg through querying the RDF metadata that Gutenberg itself publishes regularly, however this process is hampered by said metadata constituting a dataset that lacks a documented ontology, is largely unlinked and significantly bloated with redundant RDF triples. In this paper we detail the processes that were put in place with the aim of improving ontology-based data access to Gutenberg via DHTK, including (a) bottom-up extraction of the Gutenberg Ontology; (b) cleanup, linking and shrinking of the Gutenberg metadata set; (c) refactoring and alignment of said ontology with common vocabularies and (d) incorporation of the enhancements into the DHTK access routines. Early results show that we were able to reduce the size of the Gutenberg metadata set by nearly 29% whilst linking it with Library of Congress datasets, DBpedia and others.

Keywords: Digital Libraries · Ontology-Based Data Access · Gutenberg

1 Introduction and Motivation

Many fields of study in the Humanities, such as literature and history, are increasingly relying on the availability of text corpora that have reached such a status as to be published online and copyright-free [5]. Wikibooks[3] and LiberLiber[4] are examples of digital textbook repositories that enable this practice. The success of the discipline, however, is rarely guided or supported by an appropriate and standardised reorganisation of such resources to facilitate their exploitation. This can be ascribed to a variety of reasons, such the context in which a repository was born, including the technologies available or trending at the time, the capacity in which contributors operate (e.g. volunteering, best-effort, contracted or fully employed) and the content curation policies.

[3] Wikibooks, https://www.wikibooks.org/
[4] LiberLiber, https://www.liberliber.it/

Project Gutenberg[5] is the longest-standing digital library of freely accessible textbooks. Its contribution policy gives volunteers the freedom to choose the encoding formats, layout and distribution packaging of the eBooks they upload [4]. A side-effect of such liberty is that interoperability within and across Gutenberg is limited by the effort of individual contributors, thus hampering the possibility to access an eBook, or part thereof, in a standardised way.

The Digital Humanties ToolKit (DHTK) is a Python software library that was created to ease the access and the exploitation of data and metadata for the Humanities [6]. It has a dedicated module by which it is possible to retrieve any bibliographic resource available in Gutenberg, integrate its metadata with those from semantic datasets like DBpedia, and allow programmers to obtain an enriched and improved version of the same resource. It effectively acts as a proxy API for Gutenberg, which does not provide one of its own. What it does provide, however, is a downloadable export of its metadata in RDF/XML format, which DHTK uses in its backend for implementing its functionalities via SPARQL.

Despite the serialisation format, Gutenberg metadata do not constitute a full-fledged linked dataset, lack an ontology that describes their schema, and present a number of issues that limit their usefulness as a medium for accessing Gutenberg content. Using this dataset, the DHTK Gutenberg module is still able to realise its core functionalities, but has to employ costly and aleatory heuristics at runtime to work around the nonstandard and unlinked nature of the underlying metadata. It would be ideal for DHTK to employ ontology-based data access (OBDA), by using a mediating ontology that describes textbook repositories in a standard way, and then exploiting its alignments with Gutenberg-specific terminologies to parametrise the SPARQL queries that are wrapped into DHTK's Python API. This needs to be backed by an improved Gutenberg metadata set, which not only conforms to said ontology, but that is also as expressive as possible with respect to it, contains linked data and is no larger than necessary.

We have realised the workflow that makes OBDA possible for Project Gutenberg, with particular focus on DHTK and its functionalities for searching and exploring digital libraries and retrieving eBooks, or part of them, in a more standard, yet enriched, form than originally available. We extracted the Gutenberg metadata schema from the ground up, formalised it into an ontology and used it to query and analyse the metadata set to detect pitfalls. Arising issues were addressed through SPARQL transformations that eliminated blank nodes, materialised external alignments and performed other enhancements. The resulting changes to the schema were backported to the Gutenberg ontology, which was then modularised and reasoned upon. Finally, DHTK was adapted to query Gutenberg through using the general module of the ontology, taking alignments into account. This has produced a refactored, linked version of the Gutenberg metadata set that is reduced by nearly 29% in number of triples without loss of information, with alignments to external sources like Library of Congress and DBpedia, and unstructured data made structured. It also produced a mediating ontology that DHTK can use for querying catalogs using standardised queries.

[5] Project Gutenberg, `http://www.gutenberg.org/`

The rest of the paper is structured as follows: Section 2 describes the Gutenberg metadata and outlines the issues identified. Section 3 is an overview of the methodology for OBDA enhancement: its phases and their results are described in Sections 3.1-3.3. Section 4 provides pointers to implementations and related resources. Section 5 describes related work on Gutenberg and on digital libraries in general, before concluding with an insight into use cases and ongoing work.

2 The Project Gutenberg Metadata Set

Project Gutenberg does not provide a query service API for its metadata, but regularly publishes a downloadable RDF export[6] for loading onto third-party triple stores. The RDF schema mostly covers the following:

– contributors in various capacities (authors, editors etc.), with webpage links to language-specific Wikipedia pages, or occasionally to other custom pages;
– publishing metadata, relating primarily to when the eBook was first published online by Gutenberg;
– media types, formats and links to the corresponding downloadable resources;
– cataloguing and structure metadata, such as eBook languages, the names of the Gutenberg collections – or *bookshelves* – an eBook belongs to, and the table of contents of some of the eBooks;
– subjects and topics, loosely referencing the LCSH[7] and LCC[8] thesauri.

The Gutenberg metadata set is best assessed with four out of five stars.[9] Although available in RDF, the data URIs, when existing, are not externally linked as mandated by the 5-star model. The dataset presents a number of other issues, possibly owing to the age of the repository and the underlying database export not being tailored for data linking. Among them we note:

1. undocumented in-house RDFS classes and properties;
2. many blank nodes representing a small number of entities several times over, resulting in a much larger dataset than necessary;
3. links to the Library of Congress thesauri encoded as literals, not referencing subject headings by identifier, or referencing outdated headings;
4. eBook tables of contents being encoded as a single, nonstandard string literal;
5. non-dereferenceable entity URIs, e.g. for authors and agents in general.

Gutenberg also maintains a curated online wiki,[10] but only part of its content is reflected onto the RDF data: for example, the membership of an eBook in a

[6] Gutenberg catalog feeds, `https://www.gutenberg.org/wiki/Gutenberg:Feeds`
[7] Library of Congress Subject Headings, `http://id.loc.gov/authorities/subjects.html`
[8] Library of Congress Classification, `https://www.loc.gov/catdir/cpso/lcco/`
[9] See the 5-star open data model, `https://5stardata.info/`
[10] Project Gutenberg wiki, `https://www.gutenberg.org/wiki/`

bookshelf is present (and again, only via a blank node pointing to the bookshelf name), but not the hierarchy of bookshelves nor their grouping by language.

In different ways, these issues hinder programmatic access to the actual eBook content, be it ontology-based or not. The DHTK module for the Gutenberg catalog relies upon querying in SPARQL the dataset loaded onto a background RDF store, and then fetching the matching content from Gutenberg. However, it has to process a much larger dataset than it needs to be, using bespoke queries that are hard to adapt across repositories and computationally costly heuristics for retrieving book sections. The goal of this work is to analyse, refactor and enhance the Gutenberg metadata set so that it can lend itself to more streamlined processing by DHTK and other tools. In the following, we address issues 1 to 4.

3 Approach

To work properly, OBDA assumes the existence of (a) an ontology layer, (b) a mapping layer and (c) a data source layer [7]. Our initial environment only includes the data source layer, represented by the Gutenberg metadata set that can be loaded onto a triple store of choice. In order to build the remaining layers and refine the data source one, the following steps were performed (Figure 3).

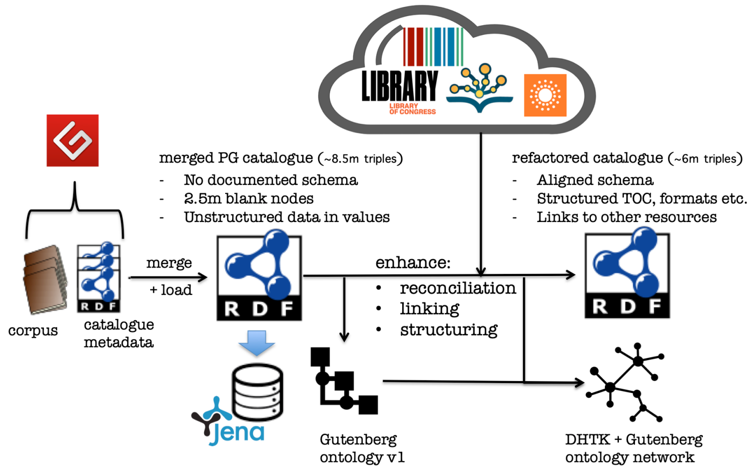

Fig. 1. Workflow for enabling OBDA to Project Gutenberg.

1. The Gutenberg metadata schema was extracted bottom-up with exploratory SPARQL queries and formalised as the Gutenberg ontology, resulting in the first iteration of the ontology layer.
2. The Gutenberg ontology was aligned with well-known vocabularies such as the DBpedia ontology (first iteration of the alignment layer).

3. The above was reused to sample the Gutenberg metadata, analyse the values in the dataset and identify pitfalls and obstacles to standardised queries.
4. Transformations to the metadata were applied, using a combination of scripting and SPARQL UPDATEs, to address such issues as the reconciliation and elimination of blank nodes, materialisation of data alignments and transformations of literals into RDF resources. To that end, Library of Congress datasets were also loaded onto the triple store. This generated changes to the Gutenberg data schema, resulting in an updated data source layer.
5. The changes to the data schema were backported to the Gutenberg ontology. In-house terms were modularised into a separate, aligned ontology. OWL reasoning was applied to both, and the materialised inferences were loaded into a separate graph (second iteration of ontology and mapping layers).
6. Finally, DHTK queries were refactored to only use terms from the remaining, general ontology, taking inferences into account as well.

These steps, and the issues they address, are described in greater detail in Section 3.1 (steps 1-2), Section 3.2 (steps 3-4) and Section 3.3 (steps 5-6).

3.1 Extraction of the Gutenberg Ontology

One advantage of OBDA is that it can be used to either (a) issue general SPARQL queries that take inferences into account and can be satisfied by many datasets, or (b) easily parametrise them so that they can be adapted to several datasets that use different, but aligned, terms for representing their data. To enable OBDA to a resource, the ontology that represents its data schema is needed, especially any alignments from nonstandard ontological terms in the target dataset to standard ones. The first step was therefore to build from the ground up the ontology that represents Gutenberg in its original state. This process was also useful for the early detection of pitfalls in the data model.

Lacking a formal description of the in-house Gutenberg terms, the ontology was constructed out of exploratory SPARQL queries on the data themselves, having loaded the Gutenberg metadata onto a triple store with a SPARQL endpoint. Bottom-up schema extraction is made possible by Linked Data not enforcing database schemas. The queries used were adapted from those recommended by VoID for building dataset descriptions,[11][12] plus other queries to detect (subclasses of) the domains and ranges of properties as utilised in the dataset.

The ontology relies extensively upon terms from the Dublin Core nomenclature (`dcterms`) and uses classes generated in-house in the Gutenberg namespace (shortened to `pgterms`), with the addition of Library of Congress terms to indicate specific contributor roles like editor or translator. Also, `pgterms:ebook` is the only type of entities in the data with no incoming links, which allows the whole ontology to be extracted out of navigating the description of eBooks alone.

[11] See VoID legacy wiki at Google Code, `https://code.google.com/archive/p/void-impl/wikis/SPARQLQueriesForStatistics.wiki`

[12] See also Mark Wallace, Exploratory RDF SPARQL queries. SemApps (2015), `http://semapps.blogspot.com/2015/12/exploratory-rdf-sparql-queries.html`

The alignment process and result We leverage the adoption of Dublin Core by both Gutenberg and the DBpedia ontology to build the first alignment layer. This involved two main linking processes:

1. between Gutenberg and Dublin Core terms;
2. between the terms obtained from the previous step and the DBpedia ones.

Figure 3.1 shows the result as the first versions of the ontology layer and alignment layer combined. The alignment was performed first at the class level using subsumption or equivalence. We identified the most appropriate terms in the Gutenberg catalog and aligned `Agent` from Gutenberg to `Agent` in Dublin core. The same holds true between `Bookshelf` and `Collection`. Data properties and object properties are shown in the figure as blue arrows. So for example the property `dcterms:hasFormat` always has `pgterms:ebook` as subject class (subset of domain) and `pgterms:File` as object class (subset of range).

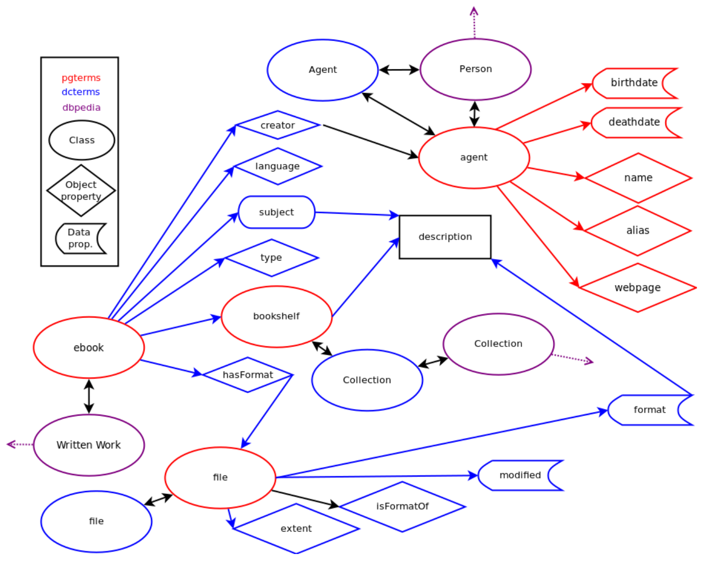

Fig. 2. The ontology and alignment layers. Namespaces are abbreviated to *pgterms* (red) for Gutenberg, *dcterms* (blue) for Dublin Core and *dbpedia* (purple) for DBpedia.

To aid comprehensibility, the Dublin Core term ontology was incorporated in the resulting OWL resource. This is freely available for research purposes.[13]

[13] This version of the ontology was first presented at EADH 2018, Galway, Ireland (unpublished). `https://drive.switch.ch/index.php/s/jLV7LcC5m8bGwEP`

3.2 Enhancements to Gutenberg Metadata

We used the class-property relationships as extracted for the Gutenberg ontology to perform random sampling of the dataset using SPARQL. We sampled property usage (as object or datatype properties), named and blank nodes, URI patterns and literal values. We tackled the issues emerging from the analysis as follows.

Blank Node Reconciliation and Entity Linking The Gutenberg metadata have a large number of object property predicates whose objects are blank nodes. On top of that, most of these blank nodes are used for re-describing the same entities over and over. Blank node statistics can be extracted on a per-property basis by issuing the following exploratory query:

Listing 1.1. Exploratory SPARQL query for grouping predicates with blank nodes.

```
SELECT (COUNT(DISTINCT ?bnode) as ?c) ?pin ?pout
WHERE {
  ?x ?pin ?bnode . ?bnode ?pout []
      FILTER (isBlank(?bnode))
} GROUP BY ?pin ?pout ORDER BY ?pin
```

The results of this query tell us not only how many distinct blank nodes there are, but also the RDF predicates `pout` used for describing each. By multiplying `c` by the number of tuples where the incoming predicate `pin` is the same, we obtain the number of blank-node triples for each `pin` in Table 1 (second column).

We observed that each blank node is described by at most two predicates with no outgoing links: one is always `rdf:value` over a string literal; the other, when present, is a `dcam:memberOf` predicate[14] that indicates the provenance of that value. The objects of this predicate are themselves DC vocabulary schemes. This hinted that the actual combinations of RDF values and provenance vocabularies may be less than the blank nodes used. Indeed, by counting the unique values for each blank property object, the figures of the third column of Table 1 emerged.

Table 1. Blank-node triple count per predicate in Gutenberg (as of February 2020).

Property	#Triples	Distinct values
`dct:format`	1,930,260	55
`dct:language`	61,563	67
`dct:subject`	404,724	34,309
`dct:type`	122,734	7
`pgterms:bookshelf`	39,644	337

From the statistics in the table, the figure for `dct:format` is the most striking: nearly 2 million triples are used for describing as little as 55 different formats.

[14] DCAM is a Dublin Core subset of terms for vocabulary descriptions.

From the table as a whole, this means that over 2.5m triples are used for describing entities that could be described, potentially even more effectively, with a largely lower number of triples, were those entities named with URIs. This is not an obstacle to OBDA per se, but still a serious problem of scale and a hurdle to data reconciliation that should be addressed.

The values of `dct:format` were all HTTP content types, i.e. combinations of MIME types and character sets: they were parsed, decomposed, refactored as instances of `ContentType` and modelled using the UWA[15] and NIE[16] vocabularies, which are able to model content types. For bookshelves we minted new URIs and linked them to automatically-generated corresponding Gutenberg Wiki URIs.

For the remaining blank nodes, their provenance predicate values suggested that they could be aligned with external datasets, which is discussed next.

External Data Alignments Many provenance triples of blank nodes point to external authorities, such as IMT (media types), the Library of Congress or Dublin Core, which hinted that external links should be created when possible.

The seven distinct `dct:type` values matched Dublin Core types - as also indicated by their provenance value `dcterms:DCMIType` - such as `Text`, `StillImage` or `MovingImage`, and were easily transformed into the corresponding URIs. Likewise, language values were all ISO 639-1 and 639-2 codes and were converted to the corresponding Linked Data URIs as published by the Library of Congress.[17]

The values for `dct:subject` were less trivial to handle: according to their provenance triples, all the string values were taken from the Library of Congress Classification (LCC) or Subject Headings (LCSH). However in the case of LCSH ones the values were the heading titles, not their codes. In addition, they are largely outdated as many subject headings can no longer be found in the current LCSH thesaurus and older ones are no longer published by the LOC. We loaded the latest LCSH RDF dataset onto the same triple store as Gutenberg and performed string manipulation and matching over the heading titles, with a recall of 43%, whereupon we generated `owl:sameAs` links. Matching the obsolete headings will be part of our future work. Subjects from the LCC were converted directly using the LOC Linked Data URI scheme, however we note that the LOC is still working on turning their classification scheme into Linked Data, therefore only part of the generated URIs can be dereferenced at the time of writing.[18]

Alignments with DBpedia were carried out: (i) for authors, by refactoring and cross-checking existing Wikipedia page links in Gutenberg; (ii) for eBooks, using cascaded DHTK heuristics that match book titles and the aforementioned author links. We refer to our previous work [3] for details on such heuristics.

[15] Ubiquitous Web Applications, `http://www.w3.org/2007/uwa/context/common.owl`

[16] NEPOMUK Information Elements ontologies, `http://www.semanticdesktop.org/ontologies/2007/01/19/nie#`

[17] See e.g. `http://id.loc.gov/vocabulary/iso639-1` for ISO 639-1.

[18] See `http://id.loc.gov/authorities/classification.html` for the list of LCC classes available as Linked Data.

Table-of-Contents Refactoring Approximately 10% of the eBooks on Gutenberg come with a table of contents (TOC), represented by `dct:tableOfContents`, whose values encode the entire table in a single string. This limits the exploitability of the metadata for accessing specific chapters of an eBook. Our goal is to replace these unstructured values with named RDF resources of type `fabio:TableOfContents` that would encode the sequence of headings, preserving their order and each heading with its individual URI and title, so that the dataset can satisfy this triple pattern:

Listing 1.2. SPARQL triple pattern for listing book headings (after refactoring).

```
?book a pgterms:ebook
  ; dct:tableOfContents/rdf:member/dct:title ?heading
```

Gutenberg maintains a single-level TOC for its eBooks; we therefore extracted the headings by mining the TOC values for recurring string patterns. This task was carried out programmatically, yet availing ourselves of string manipulation functions available in the Jena SPARQL engine. The task succeeded for 96% of the eBooks with a TOC, for which multiple headings were generated. This makes it possible for DHTK to retrieve sections of a book without resorting to its heuristics for text part detection, which are available but computationally expensive due to acting upon the texts themselves rather than the metadata.

By the end of the entire enhancement process, the refactored Gutenberg metadata set had been reduced by 2,439,332 triples, amounting to nearly 29% of its grand total, while at the same time adding links to external sources and structured versions of previously unstructured data, like content types and TOCs.

3.3 Backporting the Enhancements

The ontology alignment and the enhancements applied to the Gutenberg metadata set entail changes to the schema used by DHTK for querying its metadata, however some of these changes (such as the usage of UWA and NIE for representing content types) are of interest for digital libraries in general, whereas others – such as `pgterms`-specific axioms – make sense for Gutenberg only. We therefore refactored the ontology and alignment layers as follows:

– The new **ontology layer** becomes a general-purpose ontology module that describes access to digital libraries in general (e.g. without Gutenberg terms).
– The new **alignment layer** includes the mappings with `pgterms` classes and properties, *plus* the materialised inferences obtained by running the HermiT 1.3.8 reasoner over the general ontology and Gutenberg mappings combined.

This means that DHTK, as well as any other client, can now employ general queries that can be satisfied by multiple digital library metadata sets, without having to hard-wire all the queries in bespoke modules. So for example, the Gutenberg-specific query that DHTK uses to list the available books by Molière:

Listing 1.3. Retrieval of an author's publications in Gutenberg (before treatment).

```
SELECT DISTINCT ?book ?title WHERE {
  ?book rdf:type pgterms:ebook
      ; dct:creator pgagent:791
      ; dct:title ?title
}
```

becomes, considering ontology and data alignments, this cross-dataset query:

Listing 1.4. Retrieval of an author's publications across libraries (after treatment).

```
SELECT DISTINCT ?book ?title WHERE {
 ?book rdf:type/rdfs:subClassOf? dbo:WrittenWork
     ; dct:creator/^owl:sameAs? pgagent:791
     ; dct:title ?title
}
```

where `pgagent` is prefix for `<http://www.gutenberg.org/2009/agents/>`.

Note that, in order to capture all subclasses of `dbo:WrittenWork`, one should use the star operator `*` on `rdfs:subClassOf` instead of `?`. This would add a significant computational overhead on the evaluation of the query: however, having loaded the materialised inferences (including subclass ones) allows us to use the zero-or-one operator instead, which has a negligible impact on query efficiency.

4 Resources

The outputs of the work described in this paper are publicly available. An open source project on GitHub groups the scripts and queries used for refactoring the Gutenberg dataset,[19] as well as the ontology and alignment layers in the `ont` directory. The enhanced Gutenberg dataset – sans the DBpedia alignments, which are separate work – can be freely downloaded and used under the same conditions as the original.[20]

DHTK itself will be released as open source during 2020: its homepage[21] will be updated accordingly as this happens. In the meantime, a Web app named WeDH,[22] which allows the construction of custom corpora, has been made available to demonstrate the functionalities of DHTK [2].

5 Related Work

Access to Gutenberg's resources is a topic that arouses community interest, due to its wealth of cultural resources. Other works are worth mentioning in this regard, which aim to exploit the Gutenberg.org repository in a variety of ways.

[19] gutenberg-ld, `https://github.com/alexdma/gutenberg-ld`
[20] Gutenberg metadata (cleaned), temporary location: `https://tinyurl.com/quvrx66`
[21] Digital Humanities ToolKit, `https://dhtk.unil.ch/`
[22] WeDH, `https://dhtk.unil.ch/WeDH/`. The following temporary credentials can be used: USER:`demo@dhtk.unil.ch`, PWD:`DemoWeDH`

GutenTag is an NLP-driven tool for digital humanities research that exploits the Gutenberg corpus. The authors state that the tool exploits the Gutenberg repository to automatically create corpora by using text part recognition heuristics [1]. Unlike GutenTag, *Gitenberg*[23] does not offer the possibility to build corpora automatically, but is meant to support a collaborative, open source community for the curation and publishing of highly usable eBooks in the public domain. Whilst valuable contributions in their own right, neither of the aforementioned solutions offers actual enhancements of Gutenberg resources, nor do they harness encyclopedic knowledge to enrich texts with structured metadata. The only work that, to our knowledge, exploits resources derived from encyclopedias is *Corpus-DB*[24], a textual corpus database for Digital Humanities. This project aggregates public domain texts, enhances their metadata from sources such as Wikipedia, and makes those texts available according to those metadata [8]. As no development toolkit is provided, the project lacks programmatic access to its content. Currently, the downloadable content is limited to novels.

DHTK and the work presented here is a response to these shortcomings in one solution: to offer programmatic access to Gutenberg resources through a Python SDK, as well as an enrichment of these resources through the reconciliation and integration of data from the Linked Open Data Cloud.

6 Conclusion and Future Work

In this paper, we described our processes of enabling ontology-based data access to Project Gutenberg. Our use case was for the benefit of our DHTK library, whose access routines were improved thanks to this work, however the outcomes are reusable for other use cases and can be applied in a similar fashion to other bigliographical resources. The retrofitting of these enhancements to DHTK calls for particular attention, as one of the further improvements we have planned is the construction and automatic integration of a TEI[25] export of the repository.

Another use case of interest to us is the automatic costruction of character profiles in fiction and drama, whose early outcomes are described in [3]. It is expected that the introduction of an OBDA system able to dissolve the inconsistencies in the Gutenberg repository, along with TEI support, will allow us to target more ambitious use cases relying, for example, on NLP techniques. These range from the detection of character roles to the creation of ad-hoc datasets related to specific literary domains of fictional narrative.

References

1. Brooke, J., Hammond, A., Hirst, G.: GutenTag: an NLP-driven tool for digital humanities research in the Project Gutenberg corpus. In: Feldman, A., Kazantseva,

[23] Gitenberg, `https://www.gitenberg.org/`
[24] Corpus-DB, `http://corpus-db.org/`
[25] Text Encoding Initiative, `http://www.tei-c.org/`

A., Szpakowicz, S., Koolen, C. (eds.) Proceedings of the Fourth Workshop on Computational Linguistics for Literature, CLfL@NAACL-HLT 2015, June 4, 2015, Denver, Colorado, USA. pp. 42–47. The Association for Computer Linguistics (2015). https://doi.org/10.3115/v1/w15-0705

2. Egloff, M., Picca, D.: WeDH: a friendly tool for building literary corpora enriched with encyclopedic metadata. In: LREC 2020 (2020), to appear
3. Egloff, M., Picca, D., Adamou, A.: Extraction of character profiles from the Gutenberg archive. In: Garoufallou, E., Fallucchi, F., De Luca, E.W. (eds.) Metadata and Semantic Research - 13th International Conference, MTSR 2019, Rome, Italy, October 28-31, 2019, Revised Selected Papers. Communications in Computer and Information Science, vol. 1057, pp. 367–372. Springer (2019). https://doi.org/10.1007/978-3-030-36599-8_32
4. Hart, M.: Project Gutenberg mission statement, `https://www.gutenberg.org`
5. Kitchin, R.: Big Data, new epistemologies and paradigm shifts. Big Data & Society **1**(1) (2014). https://doi.org/10.1177/2053951714528481
6. Picca, D., Egloff, M.: DHTK: The digital humanities toolkit. In: Adamou, A., Daga, E., Isaksen, L. (eds.) Proceedings of the Second Workshop on Humanities in the Semantic Web (WHiSe II) co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 22, 2017. CEUR Workshop Proceedings, vol. 2014, pp. 81–86. CEUR-WS.org (2017), `http://ceur-ws.org/Vol-2014/paper-09.pdf`
7. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semantics **10**, 133–173 (2008). https://doi.org/10.1007/978-3-540-77688-8_5
8. Reeve, J.: Corpus-DB: a scriptable textual corpus database for cultural analytics. Proceedings of Digital Humanities 2020, Ottawa (2020), to appear