BioHackrXiv

Knowledge graphs and wikidata subsetting

Jose Emilio Labra Gayo[1], Ammar Ammar[2], Dan Brickley[3], Daniel Fernández Álvarez[1], Alejandro González Hevia[1], Alasdair J G Gray[4], Eric Prud'hommeaux[5], Denise Slenter[2], Harold Solbrig[6], Seyed Amir Hosseini Beghaeiraveri[4], Benno Fünkfstük[8], Andra Waagmeester[7], Egon Willighagen[2], Liza Ovchinnikova[3], Guillermo Benjaminsen[3], Roberto García[9], and Leyla Jael Castro[10]

1 WESO research group, University of Oviedo, Spain **2** Maastricht University **3** Google, London, UK **4** Heriot-Watt University, UK **5** Janeiro Digital, W3C/MIT **6** Johns Hopkins University **7** Micelio/Gene Wiki **8** TU Dresden **9** Universitat de Lleida, Spain **10** ZB MED Information Centre for Life Sciences, Germany

BioHackathon series:
BioHackathon Europe 2020
Virtual conference 2020
Wikidata subsetting

Submitted: 29 Mar 2021

Published by BioHackrXiv.org

Abstract

Knowledge graphs have successfully been adopted by academia, governement and industry to represent large scale knowledge bases. Open and collaborative knowledge graphs such as Wikidata capture knowledge from different domains and harmonize them under a common format, making it easier for researchers to access the data while also supporting Open Science.

Wikidata keeps getting bigger and better, which subsumes integration use cases. Having a large amount of data such as the one presented in a scopeless Wikidata offers some advantages, e.g., unique access point and common format, but also poses some challenges, e.g., performance. Regular wikidata users are not unfamiliar with running into frequent timeouts of submitted queries. Due to its popularity, limits have been imposed to allow for fair access to many. However this suppreses many interesting and complex queries that require more computational power and resources. Replicating Wikidata on one's own infrastructure can be a solution which also offers a snapshot of the contents of wikidata at some given point in time.

There is no need to replicate Wikidata in full, it is possible to work with subsets targeting, for instance, a particular domain. Creating those subsets has emerged as an alternative to reduce the amount and spectrum of data offered by Wikidata. Less data makes more complex queries possible while still keeping the compatibility with the whole Wikidata as the model is kept.

In this paper we report the tasks done as part of a Wikidata subsetting project during the Virtual BioHackathon Europe 2020 and SWAT4(HC)LS 2021, which had already started at NBDC/DBCLS BioHackathon 2019 in Japan, SWAT4(HC)LS hackathon 2019, and Virtual COVID-19 BioHackathon 2019. We describe some of approaches we identified to create subsets and some susbsets from the Life Sciences domain as well as other use cases we also discussed.

Introduction

There are several benefits and use cases derived from extracting knowledge graphs subsets. For example, it is possible to extract datasets for a specific research task from a given knowledge graph with multiple data. It also allows users to store these subsets locally and reduce database scaling and costs. The extracted subsets can be a useful resource for researchers which can analyze the evolution of the data and develop on-the-fly transformations and subsets for their specific needs.

During the last years there were multiple BioHackathon efforts to develop mechanisms which extract subsets from linked data. One example would be the G2G language created in a 2019 BioHackathon Europe project, which used a mixture of SPARQL expressions and Cypher patterns to extract property graphs.

An initial effort started during BioHackathon 2019 and SWAT4(HC)LS 2019 to define the topical use cases and main methods which were collected as informal notes that were later added to a Wikidata project. Later, at the virtual Covid-19 BioHackathon in April, an initial prototype was started to use ShEx schemas to define the subsets to be extracted from a wikibase instance.

This report collects the main advances developed during the virtual BioHackathon 2020, project 35 which were complemented with the SWAT4HCLS virtual hackathon in January 2021.

Description of activities

In this section we report the different activities that were done during the BioHackathon Europe 2020 and the SWAT4HCLS hackathons at the beginning of 2021.

General overview

As a running example we departed from the GeneWiki (**???**) project following (Waagmeester et al., 2020). That paper includes a figure with a UML-like data model representing the main concepts related with life sciences in the GeneWiki project. That figure was taken as the initial point with the goal of obtaining a WikiData subset that followed the data model represented in that figure.

It depends on the license –>

We classify the activities in 4 parts: - Describing the subsets: how do we describe what subset of wikidata we are interested in? - Extraction techniques: which approach can we follow to extract the subset from wikidata? - Publishing and using the subset: once we have the subset, how do we publish it so it can be used? - Use cases: what use cases did we identify?

Describing the subsets

A first step to obtain a Knowledge Graph subset is to describe what we expect to extract. Given that we are talking about really big data, this process must be done in a machine-processable way. Also, given that those subsets are intended to be used by people, they should be easily described by some domain experts.

A common use case is to define the boundaries of the subset by some topic identifying the type of entities or properties that are of interest for a given use case. Next, it is necessary to describe the bounderies of that subset in its larger context of all of Wikidata.

We identified several approaches to describe subsets: - SPARQL Construct queries - Filtering by rule patterns - Shape Expressions and entity schemas - Defining a domain specific language

SPARQL construct queries

One approach to extract data from any SPARQL endpoint is to use SPARQL construct queries. As an example, the [following query] (https://github.com/ingmrb/WikidataSubsetting/blob/main/Public%20queries%20method/SPARQL%20queries/Wiki%20types/anatomical_structure.sparql) can be used to retrieve anatomical structures from Wikidata.

```
CONSTRUCT {
  ?anatomical_structure wdt:P31 wd:Q4936952.
  ?anatomical_structure wdt:P361 ?part_of.
  ?anatomical_structure wdt:P527 ?has_part.
} WHERE {
  ?anatomical_structure wdt:P31/wdt:P279* wd:Q4936952
} UNION {
  ?anatomical_structure wdt:P31/wdt:P279* wd:Q4936952.
  ?anatomical_structure wdt:P361 ?part_of.
} UNION {
  ?anatomical_structure wdt:P31/wdt:P279* wd:Q4936952.
  ?anatomical_structure wdt:P527 ?has_part.
}
```

SPARQL construct queries can also transform the data retrieved on-the-fly. For example, the previous query could be expressed to transform it into using terms from the schema.org vocabulary (original query).

```
CONSTRUCT {
  ?anatomical_structure a schema:AnatomicalStructure.
  ?anatomical_structure schema:partOfSystem ?part_of.
  ?anatomical_structure schema:subStructure ?has_part.
} WHERE {
  ?anatomical_structure wdt:P31/wdt:P279* wd:Q4936952
} UNION {
  ?anatomical_structure wdt:P31/wdt:P279* wd:Q4936952.
  ?anatomical_structure wdt:P361 ?part_of.
} UNION {
  ?anatomical_structure wdt:P31/wdt:P279* wd:Q4936952.
  ?anatomical_structure wdt:P527 ?has_part.
}
```

Filtering by rule patterns

WDumper is a tool created by Benno Fünkfstük that generates Wikidata RDF dumps on demand. The tool is based on Wikidata Toolkit and allows the user to select the desired entities and properties according to rule patterns, as well as other settings like labels, descriptions, aliases, sitelinks, etc. Upon request the service creates the RDF dumps which can later be downloaded.

Internally, the rules are represented by a JSON configuration file. The rules are patterns that identify either an intended entity or property and retrieve all content that is related with them. An example of a rule that obtains all entities that are instances of `wd:Q4936952` (anatomical structure) could be:

```
{
    "version": 1,
    "__name": "anatomical_structure",
    "entities": [
      {
        "id": 3,
        "type": "item",
        "properties": [
          {
            "id": 4,
            "type": "entityid",
```

```
            "rank": "all",
            "value": "Q4936952",
            "property": ""
          }
        ]
      }
    ],
    "meta": true,
    "aliases": true,
    "sitelinks": true,
    "descriptions": true,
    "labels": true,
    "statements": [
      {
        "id": 5,
        "qualifiers": false,
        "simple": true,
        "rank": "all",
        "full": false,
        "references": false
      }
    ]
  },
  . . .
```


The JSON configuration file contains several properties related with the wikidata data model so it is possible to declare if we also want to retrieve qualifiers, ranks, etc.

ShEx and entity schemas

ShEx was created in 2014 as a human-readable and concise language for RDF validation and description (Prud'hommeaux, Labra Gayo, & Solbrig, 2014). In 2019, ShEx was adopted by Wikidata to define entity schemas (Thornton et al., 2019) and there is already a directory of entity schemas which have been collaboratively defined by the community.

During the BioHackathon Europe 2020 we defined a ShEx schema based on the GeneWiki data model. The full ShEx schema is here. Using RDFShape, it is also possible to have UML-like visualizations of the ShEx schemas. As an example, the shape of `anatomical_structure` is:

```
:anatomical_structure EXTRA wdt:P31 {
  wdt:P31  [ wd:Q4936952 ] ;
  wdt:P361 @:anatomical_structure * ;
  wdt:P527 @:anatomical_structure *
}
```

ShEx validators can use a technique called "slurp", which consists of keeping track of the triples that they have visited during validation. With this technique it is possible to obtain Wikidata subsets directly from the ShEx schemas.

In some cases, manually creating a ShEx data model may detract the users which just want to obtain a subset which is similar to some example data. sheXer is a tool that automatically extracts ShEx schemas from instance data which can be used in this case.

Wikidata subsetting language

During the BioHackathon, we considered that it was possible that ShEx was too expressive and it contained constructs that were not necessary for just creating Wikidata subsets. As an example the shape:

```
:anatomical_structure EXTRA wdt:P31 {
  wdt:P31  [ wd:Q4936952 ] ;
  wdt:P361 @:anatomical_structure * ;
  wdt:P527 @:anatomical_structure *
}
```

As a possible alternative we considered a configuration like:

```
 { "anatomical_structure":
     { "on-type": "wd:Q4936952",
      "wdt:P361": "anatomical_structure",
      "wdt:P527": "anatomical_structure" },
       "wikidata-specific-magic-extras": "..."
    {  // etc.
    }
}
```

Given a JSON file like the above, a processor could generate SPARQL construct queries or WDumper JSON configuration files.

The team considered that further work could be done in this direction in the future.

Extraction of subsets from Wikidata

Using Shape Expressions and Slurp

During validation, ShEx processors can keep track of the triples used for the validation process and create an RDF dump with them, which will follow the ShEx schema. As an example, given the ShEx schema of anatomical structure. We may start the validation by using the following ShapeMap:

```
{FOCUS wdt:P31 wd:wd:Q4936952}@:anatomical_structure
```

which means that we will validate all nodes that have property `wdt:P31` with value `wd:Q4936952` (anatomical_structure). The validator would find candidate nodes to validate like:

```
wd:Q1074 (skin)
wd:Q7891 (respiratory system)
...
wd:Q168291 (cornea)
wd:Q169342 (retina)

. . .
```

Notice that the shape indicates that anatomical structures can be part of other anatomical structures using property `wdt:P361`. When the ShEx validator validates the cornea node, it will find the triple `wd:Q168291 wdt:P361 wd:Q7364`, collect it, and continue validating the `wd:Q7364` node. This graph traversal process allows to collect an RDF dump which is based on the nodes that are really linked by the graph.

The following example uses a Wikidata ShEx definition to construct a minimal conforming graph from Wikidata using PyShEx slurper. It has been deployed as a Jupyter notebook.

During the BioHackathon we detected 2 issues:

- The Wikidata's SPARQL endpoint uses blank nodes.
- The slurp process can generate too many requests to Wikidata's endpoint.

  Talk about issue with blank nodes and endpoints... Phabricator ticket: https://phabricator.wikimedia.org/T267782 2 appearances of blank nodes: - to represent ontological constructs like `owl:complementOf` which may be justified. - to express unknown and no values. This use could be replaced TODO: Extend this explanation?

WDumper

WDumper can extract information from a complete Wikidata dump at once. This dump must be in the Wikidata JSON format. Wikidata JSON dumps are published weekly under this Wikimedia dumps page. The extractor part of WDumper uses the Wikidata Toolkit Java library to scroll the JSON dump and extract entities based on the filters specified by the specification file.

The JSON Wikidata dump file is actually a single array of Wikidata entities. For each entity, the related information like labels, descriptions, and claims are stored in different sub-arrays (see the Wikibase JSON data model for more information). In its navigation, WDumper selects any entity that matched in the filters, and for each selected entity, it extracts labels, descriptions, claims, etc. based on the specification file. Finally, WDumper builds an RDF structure of the extracted data using the Wikidata namespace and returns the final data into a compressed N-Triple format. It takes approximately 10 hours for WDumper to build a subset from the current Wikidata full dump.

- WDumper + SPARQL Construct Queries to extract data from wikidata, which was done in https://github.com/ingmrb/WikidataSubsetting
- Scrapping and linking data from web data portals which contain bioschemas metadata like mobidb and disprot. This step was done in https://github.com/elizusha/scraper.
- Creating a Wikibase instance from RDF dumps with a script and WikidataIntegrator
-

Publishing and using the subsets

During the BioHackathon 2020 the main focus of the team was to create a Wikidata subsetting. After the BioHackathon, a team lead by Dan Brickley created a docker image of a Wikidata subset based on GeneWiki which at the same time contained triples that have been scrapped from mobidb and disprot. In this way, during SWAT4(HC)LS hackathon the team focused on how those wikidata subsets could be used and published.

Documenting the subset

We have employed ShEx schemas to document the extracted subset as well as sheXer to automatically extract the shapes.

RDF HDT

All data generated by the WDumper from the Blazegraph Docker image (gcr.io/wikidata-collab-1/full_graph) was extracted ans is available at https://rhizomik.net/html/~roberto/swat4hcls

as separate Turtle files and an HDT file that combines all them.

We also published the RDF HDT file using Fuseki through a SPARQL endpoint, which is available as a Docker image, fuseki-hdt-docker.

1. Download HDT file:

```
curl -L -o combined.hdt \
https://rhizomik.net/html/~roberto/swat4hcls/combined.hdt
```

2. Deploy fuseki-hdt-docker to serve the HDT file using SPARQL

```
docker run -d -p 3030:3030 \
-v $(pwd)/life-combined.hdt:/opt/fuseki/dataset.hdt:ro \
--name fuseki-hdt rogargon/fuseki-hdt-docker
```

3. Consume the data through Fuseki's SPARQL endpoint at `http://localhost:3030/dataset/sparql`. You could use a ny SPARQL client or send queries directly using `curl`:

```
curl -X POST localhost:3030/dataset/sparql \
     -d "query=SELECT DISTINCT ?class (COUNT(?i) AS ?count) WHERE { ?i a ?class }
```

Interactive visualizations/explorations

Rhizomer is a web application for interactive exploration of semantic and linked data available from SPARQL endpoints. With Rhizomer, lay users can explore datasets performing the 3 data analysis tasks:

- **Overview**: get the full picture of the data set at hand
  - Rhizomer generates a wordcloud to provide an overview of the kinds of things in the dataset. Future work is to generate other overview components like site map, site index or treemap.
- **Zoom & Filter**: zoom in on items of interest and filter out uninteresting items
  - Once a class is selected, Rhizomer generates a faceted view. It zooms in and allows filtering resources of the chosen type based on their properties. Future work is to also allow pivoting among classes through facets.
- **Details**: after zooming and filtering the user arrives at the resources of interest
  - All properties and values are shown for every selected resource. It is also possible to browse linked resources. Future works it to include visualisations like maps or timelines.

Rhizomer is available as two Docker containers; one for the frontend and another for the backend. More details about how to deploy Rhizomer are available from https://github.com/rhizomik/rhizomerEye

SPARQL queries and demos

We also developed several SPARQL queries that demoed the resulting dataset with some information provided in this repo.

Use cases

We identified several use cases where knowledge graphs subsetting could be interesting like the Gene Wiki, Scholia, Chemistry and fact-checking.

Other activities

Given that the BioHackathon Europe 2020 and SWAT4(HC)LS 2021 were virtual events, the team missed the opportunity of more personal interaction but gained the possibility of inviting external contributors to collaborate in our activities. In this way we did the following activites: - Meeting with ShExCG. Given that the [W3C ShEx Community group] meets regularly every 2 weeks on wednesday and that the event overlapped with the BioHackathon, on 11th november 2020, the ShExCG was invited to participate in the BioHackathon and we presented our work to the ShExCG. - Joint meeting with Alasdair Gray, Ivan Micetic and Petro Papadopoulos from the Bioschemas project to talk about the interaction with the creation of wikidata subsettings and the mappings to external data annotated using bioschemas vocabulary. - Meeting with Benno Funkstük, creator of WDumper to talk about the possibilities and limits of the tool. - Meeting with Lydia Pintscher, from Wikidata, to talk about the creation of Wikidata dumps using RDF HDT. - Meeting with Javier Fernández and Wouter Beek to talk about RDF HDT

As Dan Brickley pointed out during the event, "The nice thing about virtual events over real events is that it allows instantly reaching out to expertise needed, and for experts on the matter join. We had several guests that would be almost impossible at a face to face event.". See https://twitter.com/andrawaag/status/1350063285287215106.

References

Prud'hommeaux, E., Labra Gayo, J. E., & Solbrig, H. (2014). Shape expressions: An RDF validation and transformation language. In *Proceedings of the 10th international conference on semantic systems, SEMANTICS 2014* (pp. 32–40). ACM.

Thornton, K., Solbrig, H., Stupp, G. S., Labra Gayo, J. E., Mietchen, D., Prud'hommeaux, E., & Waagmeester, A. (2019). Using shape expressions (ShEx) to share RDF data models and to guide curation with rigorous validation. In *The semantic web* (pp. C1–C1). Springer International Publishing. doi:10.1007/978-3-030-21348-0_40

Waagmeester, A., Stupp, G., Burgstaller-Muehlbacher, S., Good, B. M., Griffith, M., Griffith, O. L., Hanspers, K., et al. (2020). Wikidata as a knowledge graph for the life sciences. *eLife*, *9*. doi:10.7554/elife.52614