BioHackrXiv

Exploiting Bioschemas Markup to Populate IDPCentral

Alasdair Gray[1], Petros Papadopoulos[1], Ivan Mičetić[2], and Andras Hatos[2]

1 Heriot-Watt University, Edinburgh, UK **2** University of Padua, Padua, Italy

BioHackathon series:
BioHackathon Europe 2020
Virtual conference 2020

Submitted: 09 Jun 2021

Published by BioHackrXiv.org

Background

One of the goals of the ELIXIR Intrinsically Disordered Protein (IDP) community is create a registry called IDPCentral. The registry will aggregate data contained in the community's specialist data sources such as DisProt (Hatos et al., 2020), MobiDB (Piovesan et al., 2021), and Protein Ensemble (Lazar et al., 2021) so that proteins that are known to be intrinsically disordered can be discovered; with summary details of the protein presented, and the specialist source consulted for more detailed data.

At the ELIXIR BioHackathon-Europe 2020, we aimed to investigate the feasibility of populating IDPCentral harvesting the Bioschemas markup that has been deployed on the IDP community data sources. The benefit of using Bioschemas markup, which is embedded in the HTML web pages for each protein in the data source, is that a standard harvesting approach can be used for all data sources; rather than needing bespoke wrappers for each data source API. We expect to harvest the markup using the Bioschemas Markup Scraper and Extractor (BMUSE) tool that has been developed specifically for this purpose.

The challenge, however, is that the sources contain overlapping information about proteins but use different identifiers for the proteins. After the data has been harvested, it will need to be processed so that information about a particular protein, which will come from multiple sources, is consolidated into a single concept for the protein, with links back to where each piece of data originated.

As well as populating the IDPCentral registry, we plan to consolidate the markup into a knowledge graph that can be queried to gain further insight into the IDPs.

Data and Data Models

Source Data

Before the hackathon, the webpages of the three data sources were harvested for Bioschemas markup using BMUSE. For each page extracted, BMUSE generates an n-quad file containing:

1. The extracted markup stored in an RDF named graph with the IRI being uniquely constructed based on the date of the scrape and the page visited.
2. Provenance data about the scrape. This is stored in the default graph and describes the named graph in which the data is stored. The provenance data provided is:
   - URL of the page visited using `pav:retrievedFrom`
   - Date of extraction using `pav:retrievedOn`
   - The version of BMUSE used to harvest the data using `pav:createdWith`

The `pav:retrievedFrom` property can be used to provide the links back from individual pieces of data to the source from which it came. The other two properties are primarily used for debugging purposes, although the retrieval date can also be used to ensure that the most up to date data is available.

Within the three data sources, we expected to find markup conforming to the following Bioschemas profiles:

- DataCatalog (v0.3-RELEASE)
- Dataset (v0.3-RELEASE)
- DataRecord (v0.2-DRAFT)
- Protein (v0.11-RELEASE)
- SequenceAnnotation (type v0.1-DRAFT)
- SequenceRange (type v0.1-DRAFT)

While all the data conforms to the same data vocabulary, there are differences in the underlying usage. DisProt and MobiDB provide protein centric representations of the data. ProteinEnsemble provides a cluster of proteins on a single page. Additionally, different identifiers have been used to identify a given protein in each source. A consolidation step will need to take place to ensure that these are only presented once in the IDPcentral registry.

IDPcentral Minimal Model

IDPcentral have already designed a minimal JSON model for their data. The model consists of:

```
{
  "idp:name" : "Protein Name",
  "idp:identifier" : "Protein ID",
  "idp:sameAs" : ["Array of URLs"],
  "idp:sequence_range" : [
    {
      "idp:sequence_id" : "Identifier for a sequence",
      "idp:start" : "Start of a sequence range",
      "idp:end" : "End of a sequence range",
      "idp:range_annotation" : "Annotation for the sequence"
    }
  ],
  "idp:resource_name" : "Data source where the data originated",
  "idp:last_update" : "Date of last update"
}
```

Note that we have included `idp:` at the start of each property name to distinguish it from the Schema.org/Bioschemas properties.

We note that this is a subset of the properties available in the Bioschemas model for Proteins and SequenceAnnotations. This confirmed using the Bioschemas markup is a viable approach.

Description of Activities

During the hackathon we focused on converting the scraped data, that has a data source page centric model and consolidating this into a concept centric model, i.e. merging data about the same protein from multiple data sources.

The hackathon actives were carried out in a Jupyter Notebook which has subsequently be cleaned into the two notebooks linked at the end of this report. The original notebook is still available in the GitHub repository.

We initially loaded the data into Jena Fuseki triplestore hosted on the same machine that is running the notebook. The triplestore was queried using the Python SPARQLWrapper library. However, to increase reusability of the notebook, and the scripts contained within them, we decided to directly use the querying capabilities of RDFLib (at the time of the Hackathon version 5.0.0 of the library was used and the readthedocs documentation extensively consulted).

We will now discuss the different aspects of the work. Throughout the hackathon we followed an iterative process, gradually adding more data and functionality. We started by focusing on three files scraped from the DisProt site (files) corresponding to disprot:DP00003, disprot:DP00004, and disprot:DP00005. Once we were able to reliably extract and transform the data, we added in two pages from MobiDb (files) corresponding to mobidb:Q12959 and mobidb:P03265. The page mobidb:P03265 was specifically selected as it represents the same protein as the disprot:DP00003 page. Before adding in the final source, we implemented the protein disambiguation process. Finally, three Protein Ensemble pages (files) were added corresponding to the pages ped:PED00001, ped:PED00148, and ped:PED00174.

Data Extraction by Profile

In this section we describe the SPARQL queries developed to extract the data content and convert it into the target data model.

The queries were designed around the Bioschemas profiles, with one query written for each profile against which we are extracting data. The query extracts each property listed in the corresponding profile. All of the queries developed can be found in this GitHub directory.

A Bioschemas profile consists of three levels of marginality constraints for properties: *minimal* are properties that must always be present, *recommended* are properties that should be present but may not be relevant to all sources, and *optional* are other properties that may be of interest. However, after developing the queries over the data sources, we found that we could not rely on any of the properties being present, except for the declaration of the type being a `schema:Protein`. As such, the queries were written to use the `OPTIONAL` clause for every property.

The listing below shows the query developed for extracting Protein information. The properties have been grouped by their Bioschemas marginality level to make editing the query easier when consulting the Protein profile. Note that due to a bug in BMUSE, the provenance information is currently contained in the named graph rather than the default graph. Lines 35 and 36 of the query should be moved outside of the `GRAPH` clause once this is rectified.

```
# Query to retrieve protein data
# Defensive query: assumes that data does not conform to Protein profile
# Retrieve all properties that are mentioned in the Protein Profile

PREFIX pav: <http://purl.org/pav/>
PREFIX schema: <https://schema.org/>

SELECT *
WHERE {
    GRAPH ?g {
        # Bioschemas Minimal Properties
        ?s a schema:Protein .
        OPTIONAL {?s schema:identifier ?identifier}
        OPTIONAL {?s schema:name ?name}
        # Bioschemas Recommended properties
        OPTIONAL {?s schema:associatedDisease ?associatedDisease}
        OPTIONAL {?s schema:description ?description}
        OPTIONAL {?s schema:hasSequenceAnnotation ?annotation}
        OPTIONAL {?s schema:isEncodedByBioChemEntity ?encodedBy}
        OPTIONAL {?s schema:taxonomicRange ?taxonomicRange}
        OPTIONAL {?s schema:url ?url}
        # Bioschemas Optional properties
        OPTIONAL {?s schema:alternateName ?alternateName}
        OPTIONAL {?s schema:bioChemInteraction ?bioChemInteraction}
```

```
        OPTIONAL {?s schema:bioChemSimilarity ?bioChemSimilarity}
        OPTIONAL {?s schema:hasBioChemEntityPart ?bioChemEntity}
        OPTIONAL {?s schema:hasBioPloymerSequence ?sequence}
        OPTIONAL {?s schema:hasMolecularFunction ?molFunction}
        OPTIONAL {?s schema:hasRepresentation ?representation}
        OPTIONAL {?s schema:image ?image}
        OPTIONAL {?s schema:isInvolvedInBiologicalProcess ?process}
        OPTIONAL {?s schema:isLocatedInSubcellularLocation ?cellularLocation}
        OPTIONAL {?s schema:isPartOfBioChemEntity ?parentEntity}
        OPTIONAL {?s schema:sameAs ?sameAs}
        ?g pav:retrievedFrom ?source ;
            pav:retrievedOn ?date .
    }
}
```

Protein Merging

As already stated, each of the data sources uses their own identifier scheme to identify concepts in their data. Within the IDPcentral registry, we need to aggregate the data from the multiple sources about a given protein into a single consolidated entry, which will need its own identifier. The IDPcentral team have decided they will use UniProt accessions (The UniProt Consortium et al., 2021) as this central spine for identifying proteins. This means that for each webpage about a protein, where the protein is identified by the data sources IRI, e.g. `https://disprot.org/DP00003`, the conversion process needs to align and merge this to a UniProt accession number.

An initial inspection of the scraped markup showed that the sources all included a `schema:sameAs` declaration to the UniProt accession. However, different sources used different forms of the UniProt namespace for this identifier, e.g. the DisProt data included declarations such as

```
disprot:DP00003 schema:sameAs <https://www.uniprot.org/uniprot/P03265> .
```

while MobiDB used

```
mobidb:P03265 schema:sameAs <http://purl.uniprot.org/uniprot/P03265> .
```

The following SPARQL query was used to extract a list of source protein IRIs with their corresponding UniProt IRI; note that the sources contain `schema:sameAs` declarations to other sources as well. The `FILTER` clause matches the different UniProt IRI patterns found in the data.

```
PREFIX schema: <https://schema.org/>
SELECT ?proteinIRI ?uniprot
WHERE {
    GRAPH ?g {
        ?proteinIRI a schema:Protein ;
            schema:sameAs ?uniprot .
        FILTER regex(str(?uniprot),
            "^(https://www|http://purl).uniprot.org/uniprot/")
    }
}
```

The results of this query were used to populate a templated form of the Protein information query that was extended with a `CONSTRUCT` clause. The UniProt accession was extracted from the IRI and used to mint a new IRI for the IDPcentral representation of the concept. For convenience we used the Bioschemas namespace (`https://bioschemas.org/entity/`) for our identifiers but this can easily be changed. The `CONSTRUCT` clause also added in a `schema:sameAs` property back to the original data source page, i.e.

```
bs:P03265 schema:sameAs disprot:DP00003 .
```

Knowledge Graph Construction

While constructing the IDPcentral knowledge graph, it was assumed that the data sources would contain declarations of the same property of information. There are two cases to consider.

The first is that each source contains different values but these compliment each other, e.g. a list of synonyms where no source will necessarily have a complete set but by merging the data from the sources the IDPcentral knowledge graph would have a more complete set.

The second case is where two sources have differing values for a property which should have a single specific value. Rather than decide that a specific source's value should be used, we have decided to include all values available in the sources together with the provenance. Users of the data can then decide on the correct value, and feedback issues to the source with the erroneous value.

To support providing statement level provenance, we tried to replicate the data model of Wikidata (Vrandečić & Krötzsch, 2014), where each statement has a reference node identified by a unique universal identifier (UUID) (Leach, Mealling, & Salz, 2005) that contains the provenance information. This approach would provide a simple property graph that can be queried directly in a single graph, with additional properties used to carry the provenance data. While SPARQL has the ability to generate UUIDs in a query (SPARQL1.1#UUID), we could not do this to ensure reuse of UUIDs when some reference nodes required reuse across queries. As such, we abandoned this approach in favour of using named graphs (RDF1.1) to separate the data statements by where they have come from, and provenance data declared about the named graph and stored in the default graph. The named graph approach will increase the complexity of querying the data since multiple graphs must be considered, but simplifies the generation of the knowledge graph.

Analysis Queries

The final strand of activity within the Hackathon was to develop queries over the resulting IDP Knowledge Graph that would allow further insights to be made about the data, i.e. to demonstrate the benefit of aggregating the source data into a single knowledge graph using Bioschemas.

We first began by using SPARQL queries to extract statistical data about the generated knowledge graph according to the Health Care and Life Sciences Dataset Description Profile (Gray et al., 2015, @dumontier_health_2016). This provides valuable insights into the size of the knowledge graph, the internal structure, and the types of information that it contains and the properties used to relate these types.

We then started developing queries that would be of benefit to bioinformaticians, again with a focus on demonstrating the benefit of the integrated knowledge graph. The following questions were deemed to be of interest:

- Find proteins with data from multiple sources
- Find proteins with annotations in multiple data sources
- Find proteins with annotations in only one source
- Find proteins with annotations of type X
- Find annotations with identical ranges
- Find annotations with overlapping ranges

During the hackathon we only had time to develop SPARQL queries for the first two questions.

Discussion

During the hackathon we explored the use of Bioschemas markup extracted from a number of real-world data sources. The Bioschemas Markup Scraper and Extractor tool (BMUSE) was used to extract the data from three sources, although only a sample of the markup was used during the hackathon. A Jupyter Notebook was used to develop a transformation pipeline to process the harvested data into a consolidated knowledge graph which included detailed metadata, and to query the resulting knowledge graph.

We encountered numerous issues with both the deployed markup and the BMUSE tool during the hackathon. Several fixes or workarounds were put in place during the hackathon and subsequently. These fixes included editing the sample data to meet the expected form for the data to enable experimentation with the transformation pipeline. BMUSE and the deployed markup are being updated to remove the need for this editing.

One substantial issue related to the use of the Bioschemas DataRecord type. Within the deployed markup we found issues with identifying the record and the protein itself. There were also issues with different resources in the data reusing the same IRI with undesirable consequences. This led to discussions about the benefits of the proposed Bioschemas DataRecord type versus the complexity of developing correct markup and ultimately consuming that markup. The outcome of this was that the Bioschemas Community have decided to deprecate the use and development of the DataRecord type (Bioschemas#475).

The notebooks included in this report have been extracted from the single notebook used during the hackathon, and the subsequent development work. This has been done to aid understandability of the work, and therefore simplify reuse of the approach. The original (unfinished) notebook is also available in the GitHub repository.

Future work

The framework developed within this hackathon has only been used on a small sample of data from each of the data sources. Once the deployments of the markup of these sources has been updated, we intend to use the transformation framework over the entire scraped data.

At the time of the Hackathon, IDPcentral were developing a MongoDb backend for their repository. While the transformation process was able to generate the intended data model, it would be desirable to have a single collection of data for IDPcentral. IDPcentral are considering changing their backend data store to a SPARQL endpoint to enable them to use the IDP knowledge graph directly. This would enable the more detailed data in the knowledge graph to be exploited through queries whilst still supporting the surfacing of basic data about a given protein by the web site.

Jupyter notebooks, GitHub repositories and data repositories

- GitHub repository: https://github.com/elixir-europe/BioHackathon-projects-2020/tree/master/projects/24/IDPCentral
- Jupyter Notebooks:
  - Knowledge Graph Construction: https://github.com/elixir-europe/BioHackathon-projects-202 blob/master/projects/24/IDPCentral/notebooks/ETLProcess.ipynb
  - Analysis Queries: https://github.com/elixir-europe/BioHackathon-projects-2020/blob/master/projects/24/IDPCentral/notebooks/AnalysisQueries.ipynb

Acknowledgements

This work was done during the BioHackathon Europe 2020 organized by ELIXIR in November 2020. We thank the organizers and fellow participants.

References

Dumontier, M., Gray, A. J., Marshall, M. S., Alexiev, V., Ansell, P., Bader, G., Baran, J., et al. (2016). The health care and life sciences community profile for dataset descriptions. *PeerJ*, *4*, e2331.

Gray, A. J., Baran, J., Marshall, M. S., Dumontier, M., Alexiev, V., Ansell, P., Bader, G., et al. (2015). *Dataset descriptions: HCLS community profile*.

Hatos, A., Hajdu-Soltész, B., Monzon, A. M., Palopoli, N., Álvarez, L., Aykac-Fas, B., Bassot, C., et al. (2020). DisProt: Intrinsic protein disorder annotation in 2020. *Nucleic Acids Research*, *48*(D1), D269–D276. doi:10.1093/nar/gkz975

Lazar, T., Martínez-Pérez, E., Quaglia, F., Hatos, A., Chemes, L. B., Iserte, J. A., Méndez, N. A., et al. (2021). PED in 2021: A major update of the protein ensemble database for intrinsically disordered proteins. *Nucleic Acids Research*, *49*(D1), D404–D411. doi:10.1093/nar/gkaa1021

Leach, P., Mealling, M., & Salz, R. (2005). *Hjp: Doc: RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace*. Retrieved from https://www.hjp.at/doc/rfc/rfc4122.html

Piovesan, D., Necci, M., Escobedo, N., Monzon, A. M., Hatos, A., Mičetić, I., Quaglia, F., et al. (2021). MobiDB: Intrinsically disordered proteins in 2021. *Nucleic Acids Research*, *49*(D1), D361–D367. doi:10.1093/nar/gkaa1058

The UniProt Consortium, Bateman, A., Martin, M.-J., Orchard, S., Magrane, M., Agivetova, R., Ahmad, S., et al. (2021). UniProt: The universal protein knowledgebase in 2021. *Nucleic Acids Research*, *49*(D1), D480–D489. doi:10.1093/nar/gkaa1100

Vrandečić, D., & Krötzsch, M. (2014). Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, *57*(10), 78–85. doi:10.1145/2629489