Linked Data Creation with ExcelRDF

Karl Hammar[0000−0001−8767−4136]

Jönköping AI Lab
Jönköping University, Sweden
karl.hammar@jail.ai

1 Introduction

Constructing an RDF-based knowledge graph requires designing a data model (typically an OWL ontology) and transforming one's data into an RDF representation that is compliant with said model. There are a multitude of tools built to support these two tasks, and of the ones adressing the first task, several that are specifically intended to enable less experienced users to construct, maintain, or analyse ontologies easily and with confidence: these include WebProtégé [8], VOWL-based visualizations [5], CoModIDE [7], etc.

The second task, transforming existing data into RDF representation, can either be carried out in a batch manner (e.g., using OpenRefine, or an R2RML-based [2] transformation tool like DB2Triples), or at query time (e.g., using databases that provide RDF views over relational data, again typically employing R2RML mappings). Neither is easy for a linked data beginner. In the former case they must typically learn a non-trivial mapping tool and its vocabulary; in the latter case, a server daemon needs to be setup (and possibly licensed), a mapping definition needs to be defined, etc. In neither case is the user guided on how to create RDF data in accordance with a specific ontology.

By contrast, Microsoft Excel is a well-established and well-understood software for data wrangling in industry. It is installed on a large number of desktop machines already, and office workers tend to navigate and use its basic functionalities with minimal, if any, training. Integrating user-friendly ontology-based RDF creation functionalities in Excel enables this group of users to easily contribute to knowledge graph construction; that is the intuition behind the ExcelRDF[1] tool. ExcelRDF was created in the Building Knowledge project, where it is used by real estate owners to populate knowledge graphs using the RealEstateCore [3] smart buildings ontology. Its key design criteria are that it should:

- Be easy to install, update, and start; no IT support should be required.
- Employ a transparent syntax for mapping cells to RDF constructs; nothing should be "hidden" in the underlying Excel file format.
- Support users in creating said mappings from a source ontology.
- Generate Excel files that can be shared across an organisation, even by users who do not have ExcelRDF installed, without the RDF mappings being lost.
- Provide simple and direct data export from spradsheet to RDF graph; any data transformation can be done in Excel itself.

[1] https://dev.realestatecore.io/ExcelRDF/

2 Related Work

There are several tools that enable spreadsheet-to-RDF translation, but to my knowledge, none that emphasize the ExcelRDF design criteria described above.

XLWrap [4] and Spread2RDF[2] operate on spreadsheets and translate these (batch-based or at query time) using custom mapping languages. DB2Triples[3] and D2RQ [1] do the same, but employ mapping languages that have been standardised (the R2RML and D2RQ languages, respectively). These tools are geared toward users who are already quite familar with linked data and who are comfortable with writing their own mapping rules.

OpenRefine[4] is a well-established tool for data transformation and its RDF plugin supports GUI-based mapping of tabular data (e.g., from Excel) to an RDF graph structure. Users can modify their data using both GUI approaches (e.g., merging or splitting columns, filtering values, etc) and for more fine-grained data manipulation on cell-by-cell level, through the GREL language. However, OpenRefine does not allow for easy sharing of work, as each participant needs to import the shared project into their own on-machine OpenRefine install; and installing it, and the RDF extension, is non-trivial.

TabLinker[5] uses spreadsheet styling to indicate the mapping of cells, rows, and columns to values, types, properties, etc. A spreadsheet that has been annotated using TabLinker styles can be shared and edited by multiple users before being run through the command line script that exports RDF. Compared with ExcelRDF, TabLinker however lacks an ontology import feature, so users need the develop style-based mappings by hand.

Other approaches to bring ontology structures into spreadsheets include RightField [9] (for Excel) and OntoMaton [6] (for Google Spreadsheets). These tools allow for the annotation of spreadsheet data by terms in an ontology; but they do not include RDF export functionality.

3 System Design and Features

ExcelRDF is implemented as a .NET-based Microsoft Office VSTO Add-In. The .NET underpinnings allows ExcelRDF to reuse the DotNetRDF[6] library, saving significant development effort. The VSTO plugin infrastructure also provides a dead-simple deployment mechanism, "ClickOnce", which generates a user-friendly installer, and provides automated Internet-based updates.

Using ExcelRDF consists of three distinct steps. First, the user loads an ontology, and through a friendly GUI selects which classes and properties from that ontology that they intend to use (Figure 1) – based on their selection, the tool creates corresponding works sheets and column headers in an otherwise

[2] `https://github.com/marcelotto/spread2rdf`
[3] `https://github.com/antidot/db2triples`
[4] `http://openrefine.org/`
[5] `https://github.com/Data2Semantics/TabLinker`
[6] `https://www.dotnetrdf.org/`

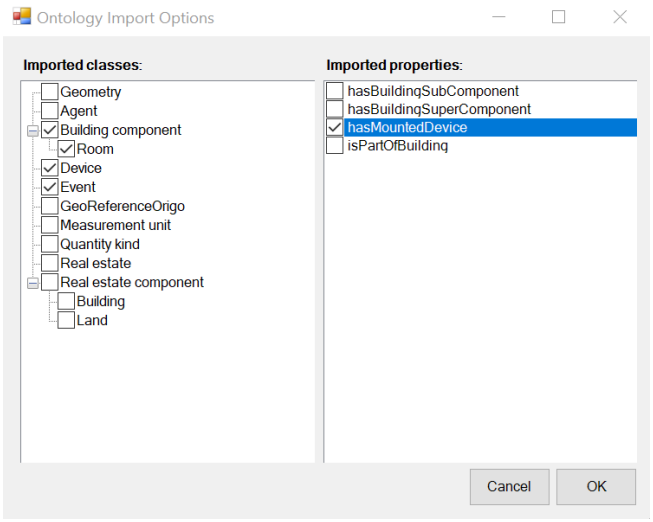

Fig. 1: Ontology import dialog

empty Excel skeleton file. Second, the user fills out this skeleton file with their data, using standard Excel tools and existing workflows. Third, once the data is complete the user exports it into RDF that is compliant with the initially loaded ontology – a simple GUI is provided to configure data namespaces and prefixes (Figure 3). Each of these steps is described in detail below.

OWL Import The ExcelRDF ontology import GUI (Figure 1) is launched from the "Data" ribbon menu. The user is asked to select an on-disk OWL ontology file[7]. The named classes in this file are parsed and added to the class selection GUI; the properties for each such class (i.e., that have the class asserted as `rdfs:domain`) are added to the property selection GUI for that class. The user selects the classes and properties that they wish to use for their data, and the tool then constructs one work sheet (i.e., Excel tab) per selected class, and for each such work sheet adds columns corresponding to the selected properties. Additionally, a special identifier column is inserted and used for IRI minting. For examples of the complete structure, see Figures 4a and 4b.

The header row cells on each generated work sheet are marked up with Excel notes that describe the properties that underlie each column; these notes (see Figure 2a for an example) act as instructions for the RDF exporter. Optionally, the user may when importing an ontology select to embed anonymous individuals on a work sheet, spanning over several columns; when doing so, the cells of these columns will correspond with nested objects through an intermediate anonymous node. In the latter case, the RDF exporter instructions become a little more complex: see Figure 2b for an example.

[7] Supported serializations: `XML/RDF`, `Turtle`, `JSON-LD`, `NTriples`, `NQuads`, and `TriG`

<https://w3id.org/rec/core/associatedWithDevice>
<http://www.w3.org/2002/07/owl#ObjectProperty>
<https://w3id.org/rec/core/Device>

(a) Column representing the `associatedWithDevice` property.

<https://w3id.org/rec/core/hasRealEstateComponent>
<http://www.w3.org/2002/07/owl#ObjectProperty>
<https://w3id.org/rec/core/RealEstateComponent>
<http://www.w3.org/2000/01/rdf-schema#label>
<http://www.w3.org/2002/07/owl#AnnotationProperty>
<http://www.w3.org/2001/XMLSchema#string>

(b) Column representing the label of a nested individual of the type `RealEstateComponent`.

Fig. 2: RDF generator instructions embedded in Excel skeleton

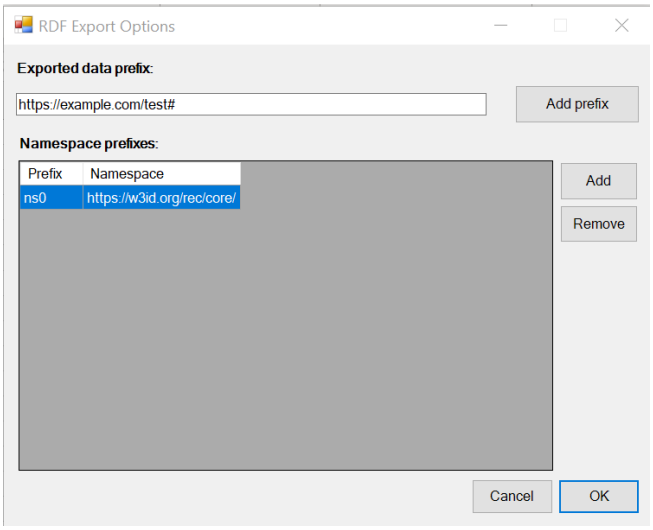

Fig. 3: Ontology import dialog

RDF Export Once the user has populated the spreadsheet with data, they launch the RDF export GUI (Figure 3). ExcelRDF extracts URI namespaces from the classes and properties mentioned in the note objects, and suggests that these be added to the namespace prefix mapping in the same GUI; additionally, the user is asked for a data namespace, that will be prepended to the identifiers that the user has given in the identifier column.

ExcelRDF generates an RDF graph[8] using the aforementioned notes objects it finds in the work sheet headers. Every cell on the sheet will generate an RDF statement where the `subject` is the row identifier, the `predicate` is the column header, and the `object` is the literal value held in the cell, or in the case of an object property, is a URI with that value as local name (Figure 4c); unless if the embedded anonymous individuals feature has been used, in which case a more complicated structure such as the one in Figure 4d is generated instead.

[8] Supported serializations: `RDF/XML`, `Turtle`, and `NTriples`

	A	B	C	D	E	F
1	**Identifier**	**associated with device**	**event measurement unit**	**event quantity kind**	**has start time**	**has stop time**
2	Event1	Thermometer1	C	Temperature	20200305T15:58:	20200305T15:58:30Z

(a) Example data for Event class/work sheet

	A	B
1	**Identifier**	**rdfs:label (through hasRealEstateComponent)**
2	JU-Campus	JU Building E (School of Engineering)

(b) Example data for RealEstate class/work sheet

```
example:Event1 a rec:Event;
               rec:associatedWithDevice example:Thermometer1;
               rec:eventMeasurementUnit example:C;
               rec:eventQuantityKind example:Temperature;
               rec:hasStartTime "20200305T15:58:30Z"^^xsd:dateTime;
               rec:hasStopTime "20200305T15:58:30Z"^^xsd:dateTime.
```

(c) RDF generated from data in Figure 4a

```
example:JU-Campus a rec:RealEstate;
                  rec:hasRealEstateComponent [
                                 a rec:RealEstateComponent ;
                                 rdfs:label "JU Building E (School of Engineering)"^^xsd:string].
```

(d) RDF generated from data in Figure 4b

Fig. 4: Excel and generated RDF data

4 Discussion and Future Work

The beauty of ExcelRDF lies in its simplicity. The tool does not purport to enable complicated schema or data transformation scenarios; it simply provides a round-trip translation from ontology to spreadsheet and back to RDF graph. This enables data owners to maintain and user their existing Excel-based tools or workflows. Since the RDF exporter instructions are embedded in the generated Excel file itself, these files can be shared through the organisation and data collated from multiple sources by users who may not have ExcelRDF installed. And, since the RDF generation instructions are stored in a transparent manner using Excel notes, modifying them is easy.

New features being considered for the future roadmap include:

1. Support for `owl:Imports` – At present, the tool only operates on an ontology file loaded from disk. Adding imports resolution (possibly over the Internet) adds significant complexity, and arguably, an ontologist could anyway integrate imports in a pre-processing step, e.g., using Protégé. That said, as an optional feature, imports support may be very useful.
2. Pre-loading A-box from ontology – The tool ignores any A-box entities (i.e., `owl:NamedIndividual`) in the imported ontology. In some use cases it is useful to have a base set of individuals already in the ontology; I am considering how they should be represented in the generated Excel file.

3. Type checking of values – The tool does not validate that the types of values provided in cells are correct with regard to the `rdfs:range` of the column's underlying property. Such checking should raise an error if, for instance, the user has entered a string in a cell that should generate an XSD integer object.

Additionally, while ExcelRDF has been used successfully in the Building Knowledge project, it has not been rigorously evaluated in a more formal setting; this remains to be done in the near future.

Finally, it should be noted that since ExcelRDF is based on the VSTO architecture, it will run only on Excel for Windows. Microsoft provides an alternate add-in-architecture that is platform-agnostic, based on web technologies; but since ExcelRDF depends on .NET-based libraries this architecture has until recently not been available to use. However, with the uptake of WebAssembly, it may in the not so distant future be possible to compile those .NET libraries into WASM that can be executed in a web environment, in which case ExcelRDF could certainly be re-engineered to also become entirely platform-agnostic, running anywhere Excel runs (including in the browser, on macOS, iOS, etc).

References

1. Bizer, C., Seaborne, A.: D2RQ – Treating Non-RDF Databases as Virtual RDF graphs. Poster at the 3rd International Semantic Web Conference, Hiroshima, Japan (November 2004)
2. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF Mapping Language. `https://www.w3.org/TR/r2rml/` (September 2012)
3. Hammar, K., Wallin, E.O., Karlberg, P., Hälleberg, D.: The RealEstateCore Ontology. In: The Semantic Web – ISWC 2019. pp. 130–145. Springer (October 2019), `http://dx.doi.org/10.1007/978-3-030-30796-7_9`
4. Langegger, A., Wöß, W.: XLWrap – Querying and Integrating Arbitrary Spreadsheets with SPARQL. In: International Semantic Web Conference. pp. 359–374. Springer (October 2009), `https://doi.org/10.1007/978-3-642-04930-9_23`
5. Lohmann, S., Negru, S., Haag, F., Ertl, T.: Visualizing Ontologies with VOWL. Semantic Web **7**(4), 399–419 (2016), `http://dx.doi.org/10.3233/SW-150200`
6. Maguire, E., González-Beltrán, A., Whetzel, P.L., Sansone, S.A., Rocca-Serra, P.: OntoMaton: a Bioportal powered ontology widget for Google Spreadsheets. Bioinformatics **29**(4), 525–527 (2013), `https://doi.org/10.1093/bioinformatics/bts718`
7. Shimizu, C., Hammar, K.: CoModIDE – The Comprehensive Modular Ontology Engineering IDE. In: ISWC 2019 Satellites. CEUR Workshop Proceedings, vol. 2456 (October 2019), `http://urn.kb.se/resolve?urn=urn:nbn:se:hj:diva-46397`
8. Tudorache, T., Nyulas, C., Noy, N.F., Musen, M.A.: WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the Web. Semantic web **4**(1), 89–99 (2013), `http://dx.doi.org/10.3233/SW-2012-0057`
9. Wolstencroft, K., Owen, S., Horridge, M., Krebs, O., Mueller, W., Snoep, J.L., du Preez, F., Goble, C.: RightField: embedding ontology annotation in spreadsheets. Bioinformatics **27**(14), 2021–2022 (2011), `https://doi.org/10.1093/bioinformatics/btr312`