SemTK: A Semantics Toolkit for User-friendly SPARQL Generation and Semantic Data Management

Paul Cuddihy¹, Justin McHugh², Jenny Weisenberg Williams¹, Varish Mulwad¹ and Kareem Aggour¹

> ¹ GE Global Research, Niskayuna NY 12309, USA ² Kobai, Pleasanton, CA 94588, USA cuddihy@ge.com,justin@kobai.io {weisenje,varish.mulwad,aggour}@ge.com

Abstract. We introduce the Semantics Toolkit (SemTK), an integrated platform for user-friendly querying and semantic data management. Through its user interface, SemTK allows users to convert CSV data into RDF triples and ingest them into a triple store. It also allows users to visually explore the ontology and construct SPARQL queries via a drag-and-drop interface. SemTK also provides novel SQL stored procedure-like support for saving and executing semantic queries with run-time constraints. Additionally, SemTK provides REST APIs for its functionality--including allowing data ingestion and queries to be executed programmatically--dramatically simplifying the deployment of knowledge-driven applications. SemTK is open-sourced under the Apache License, Version 2.0 and is available at https://github.com/ge-semtk/semtk

Keywords: Visual SPARQL querying, SPARQL generation, data triplification, data ingestion, semantic data management.

1 Introduction

The Semantic Web technology stack provides a powerful foundation to construct, query and maintain knowledge graphs, but in practice can be challenging to adopt by both non-semantic subject matter experts in scientific and industrial communities and by application developers building knowledge-driven applications. Tools such as Protégé [1] and SADL¹ [2] have made rapid strides in reducing these barriers for ontology design and creation. However, there exist few tools with the same level of maturity to explore, query and manage semantic data in knowledge graphs.

The Semantics Toolkit (SemTK) is designed to lower the barrier to using semantic technologies by making semantics accessible in a user-friendly manner to both subject matter experts and application developers. With an "ontology first" approach, SemTK has been designed in the context of the needs of a large industrial business, and in the context of the migration of large disparate data sources into Linked Data. Through its SPARQLgraph interface, SemTK allows both domain experts and application developers to upload ontologies, browse existing ontologies, map tabular data to a nodegroup

¹ http://sadl.sourceforge.net/

graph pattern, and upload data. Finally, a graphical drag-and-drop tool is used for SPARQL query generation. Further, SemTK allows application developers to save both queries and data triplification mappings and access them programmatically via REST APIs, dramatically reducing knowledge-driven application development time. To the best of our knowledge, SemTK is the first open source system that provides a seamlessly integrated experience to perform all of these tasks together.

2 The Semantics Toolkit Demonstration

The features of SemTK are best demonstrated using the SPARQLgraph web interface, using the demo ontology and data shown at: https://github.com/ge-semtk/semtk/wiki/demo. This demo includes a SADL ontology and data in a CSV file that can be loaded into a knowledge graph and queried against.



Fig. 1. SPARQLgraph screen capture illustrates three basic building blocks SemTK: (i) the connection, (ii) the ontology information cache, and (iii) nodegroup. This example ontology represents a collection of equipment test data, with results organized into layers of measurements.

Connections and Ontology Information. Use of SPARQLgraph starts with defining a new "connection" to a knowledge graph. The connection consists of the URLs and graph names of multiple SPARQL endpoints each for the ontology and the data. The "model domain" is a regex used to select the URI's which make up the ontology. After loading, the class hierarchy is displayed in the left-hand pane, and keyword searchable.

Nodegroup Construction. A nodegroup is a graphical representation crucial to almost every SemTK function. It represents a subgraph of interest, and is used for generating

queries such as SELECT and DELETE, and for ingesting data. The main pane provides a drag-and-drop-interface for constructing nodegroups and complex SPARQL queries.

Path-finding Algorithm. Automated path-finding uses a modified A* algorithm to simplify the task of building nodegroups . When new classes are dropped on the canvas, possible connection paths are automatically generated and suggested.

VALUES Clause Generation. SemTK's automated VALUES generation capability assists further by suggesting all valid values for a property based on the current nodegroup and the contents of the data endpoints, and automatically build VALUES clauses.

Runtime Constraints. Properties in a nodegroup can be flagged as runtime constrainable, in which case their values must be provided when the query is invoked, mimicking stored procedure functionality common in the relational database world.

SPARQL Generation. Once a nodegroup has been constructed, SELECT, COUNT, and DELETE queries can be executed. Runtime constraints can be added (for saved queries) and executed with the VALUES generation capability used to suggest constraint values.

Data Ingestion. Using the Map Input screen, a CSV file can be aligned with the nodegroup for data ingestion by dragging-and-dropping columns to corresponding ontology elements. Ingestion values can be transformed and combined with text and with each other. URI lookup can be used to connect new data to existing data using a powerful dynamic lookup strategy. During ingestion, the data is type-checked, then one copy of the nodegroup is generated for each line of input data, and a SPARQL-generation engine builds and executes an INSERT query.

REST API. All features demonstrated via SPARQLgraph are accessible via REST APIs. Nodegroups can be saved by name in the service layer or as JSON, and invoked to perform select, delete, count and ingestion tasks. This provides a simple yet powerful integration of semantics in applications via a standard REST interface.

The default demo can be accessed at http://semtk.research.ge.com/sparqlGraph/index.html. For ease of use, this demo scenario comes pre-loaded with an existing ontology. Interested readers can find more technical details in [3].

3 Related Work

Previous research efforts into simplifying the use of Semantic Web technologies have focused both in visual SPARQL querying and data triplification. Tools exist that enable users to visually generate SPARQL queries to interact with triple store data including OptiqueVQS [4] and QueryVOWL [5] which come closest to SemTK in terms of func-

tionality. SemTK expands and improves upon these tools by providing advanced features geared toward subject matter experts who are not well-versed in OWL or SPARQL. Through its oInfo panel, SemTK allows non-expert users to explore domain ontologies. With SemTK's pathfinding, users are not burdened with the task of finding a path to connect two classes, which is a fairly common step in constructing SPARQL queries. Finally, SemTK makes it very simple to add constraints on both object and data properties to filter query results. A detailed comparison between SemTK and other tools can be found in [3]. As with visual SPARQL querying, there has been a significant amount of prior work focused on mapping CSV, spreadsheets and relational data to RDF². SemTK's data triplification and ingestion is similar to RDF Refine³, SemTK simplifies this process by providing a simple, intuitive, graphical drag-and-drop approach to define mappings. However, unlike RDF Refine, which allows users to define arbitrary classes, properties and links between them, SemTK uses classes and properties from an existing domain ontology, thus conforming the data to existing schemas. In addition, SemTK's data triplification is tightly coupled with its data ingestion process. Once a mapping is defined based on the user-provided CSV, the data within the CSV is triplified and ingested. This tight integration between triplification and ingestion makes SemTK considerably more practical and useable. These features make it dramatically simpler for non-semantic experts to construct knowledge-driven applications.

4 Future Work

SemTK remains an ongoing development effort. Future enhancements include generating more complex SPARQL features such as UNION clauses. While SemTK is designed to work with any SPARQL1.1-compliant triple store, work remains to ensure smooth operation with a wide range of semantic stores.

References

- 1. M. A. Musen, "The Protégé Project: A Look Back and A Look forward," AI Matters, 2015.
- A. Crapo and A. Moitra, "Toward a Unified English-like Representation of Semantic Models, Data, and Graph Patterns for Subject Matter Experts.," International Journal of Semantic Computing, vol. 7, no. 3, pp. 215-236, 2013.
- P. Cuddihy, J. McHugh, J. W. Williams, V. Mulwad and K. S. Aggour, "SemTK: An Ontology-first, Open Source Semantic Toolkit for Managing and Querying Knowledge Graphs," arXiv preprint arXiv:1710.11531., 2017.
- 4. A. Soylu, M. Giese, E. Jiménez-Ruiz, G. Vega-Gorgojo and I. Horrocks, "Experiencing OptiqueVQS: a multi-paradigm and ontology-based visual query system for end users," Universal Access in the Information Society, vol. 15, no. 1, pp. 129-152, 2016.
- 5. F. Haag, S. Lohmann, S. Siek and T. Ertl, "QueryVOWL: Visual Composition of SPARQL Queries," in ESWC Satellite Events, 2015.

² https://github.com/timrdf/csv2rdf4lod-automation/wiki/Alternative-Tabular-to-RDFconverters

³ http://openrefine.org/