SETLr: the Semantic Extract, Transform, and Load-r

Jamie P. McCusker[1], Katherine Chastain[1], Sabbir Rashid[1], Spencer Norris[1], and Deborah L. McGuinness[1]

Rensselaer Polytechnic Institute, Troy, NY 12180, USA,
mccusj@cs.rpi.edu, {chastk,rashis2,norris}@rpi.edu, dlm@cs.rpi.edu,
WWW home page: http://tw.rpi.edu

Abstract. Semantic Extract, Transform, and Load-er (SETLr) is a flexible, scalable tool for providing semantic interpretations to tabular, XML, and JSON-based data from local or web files. It has been used by diverse projects and has shown to be scalable and flexible, allowing for the simplified creation of arbitrary RDF, including ontologies and nanopublications, from many different data formats. Semantic ETL scripts use best practice standards for provenance (PROV-O) and support streaming conversion for RDF transformation using the JSON-LD based templating language, JSLDT. SETLr also supports custom Python functions for remote APIs, entity resolution, external data lookup, or other tasks. We provide case studies for dynamic SETL scripts, ontology generation, and scaling to gigabytes of input and discuss the value and impact of this approach.

Keywords: semantic interpretation, triplification, data curation

1 Introduction

As the interest in Semantic Web applications for advanced visualization and analysis grows, so too does the need for data ingestion methods aimed to support precise generation of RDF that conforms closely to its backing ontologies. Heterogeneous datasets may contain a variety of file formats and sources, including tabular data as well as structured data with increasing semantic encodings (e.g., XML and JSON) that may be gathered from web resources or returned from database endpoints. Data from different sources may require different handling to fit into Semantic Web applications and supporting ontologies; it is equally important to preserve the provenance of such changes, as well as to make the changes themselves. With the aim of facilitating linked data aggregation and translation while maintaining a record of the process, we present SETLr - the Semantic Extract, Transform, and Load-er. SETLr is designed to create RDF from a variety of input file formats. It supports custom programming functionality in the ETL process, thus creating a dynamic tool that can be used in a variety of Semantic Web use cases. Currently we support custom Python inclusion in the ETL process. SETLr utilizes the JSON-LD Template

Language (JSLDT) to create RDF graphs within SETLr scripts, thus enabling a more human-readable way to construct data translation parameters. We also integrate the W3C Provenance Ontology (PROV-O) into the scripts directly in order to preserve the provenance of each step of the Extract, Transform, and Load process.

2 Related Work

RDF Refine[1] is a plug-in for OpenRefine offering RDF capabilities for tabular data, including RDF export and data reconciliation [1]. The tool provides a graphical user interface for construction of an RDF "skeleton" used to map a table to RDF for export. Because of its focus on interactive data curation, OpenRefine does not yet support automated processing of data through those histories without the use of that user interface. CSV2RDF4LOD[2] is a tool for creating RDF from tabular data, specifying the transformation using an RDF vocabulary. It demonstrates how to incorporate attribution, versioning, provenance, and scheduled automation when aggregating data from multiple, disparate parties. However, it is difficult to use to create complex data. The specialized terminology it used was an impoverished shorthand for common structural adjustments that were inadequate for arbitrary RDF output. While the documentation has mostly kept up with the functionality, navigation of it has become more difficult as development has progressed. The result is that, when attempting to use some of the more advanced capabilities together, the generated RDF can become unpredictable, if those capabilities had not been tested together. Other conversion tools based on R2RML[2] assume an underlying relational model to the data, and are not as flexible as these tools or SETLr.

3 Design & Technical Quality

SETLr was designed to integrate and leverage best practice vocabularies and services. SETLr scripts, leveraging the PROV-O vocabulary, drive SETLr. Data artifacts such as the input files and intermediate forms are also described using terms from W3C's Data Catalog Vocabulary (DCAT)[3] and Dublin Core[4], with additional terms added where necessary in a SETLr namespace. The data translation parameters then become the *prov:value* of the transformation, described directly in the script file using the JSON-LD Template language (JSLDT). This script file therefore doubles as *prospective provenance* for the ETL process itself, as well as providing the instructions for each step. The architecture of SETLr, and the process of semantic ETL is shown in Figure 1.

[1] http://refine.deri.ie/
[2] https://www.w3.org/TR/r2rml/
[3] https://www.w3.org/TR/vocab-dcat/
[4] http://dublincore.org/documents/dcmi-terms/

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.26476v1 | CC BY 4.0 Open Access | rec: 31 Jan 2018, publ: 2 Feb 2018

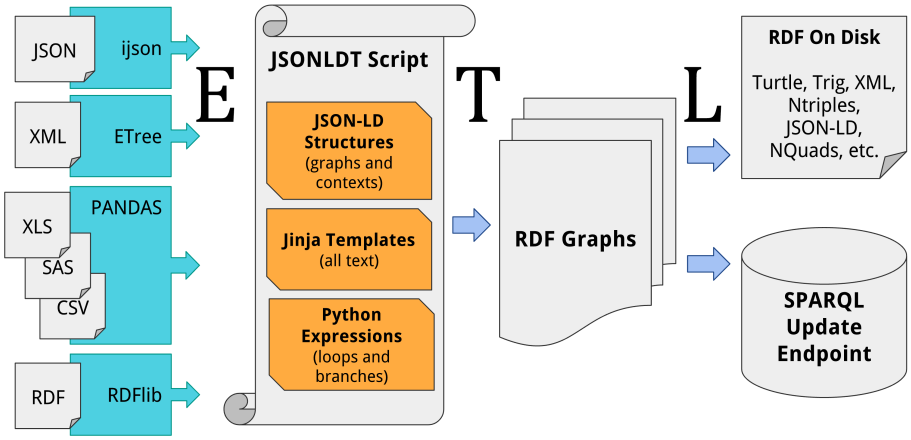

Fig. 1. SETLr processes are broken into three kinds of activities: Extract, Transform, and Load. Extract can process tabular (CSV, Excel, SAS, HDF, etc.), HTML, XML, and JSON files, including streaming processing for XML and JSON using XPath-like constructs. Transforms use JSON-LD based templating for RDF generation (JSLDT). The resulting RDF can be stored on disk or uploaded to a SPARQL Update endpoint.

3.1 Extract

SETLr handles a variety of source data formats, including but not limited to tabular files. The Pandas Python library[5] enables tabular formats including SAS XPort, XLS, and CSV/TSV. SETLr also ingests JSON data using ijson,[6] HTML using BeautifulSoup,[7] XML using ElementTree,[8] and RDF using RDFlib.[9]

3.2 Transform

SETLr uses JSLDT for dictating the transform section of the ETL process. These templates are structurally based on JSON-LD, and leverage capabilities such nested named graphs and contexts to facilitate the transformation. JSLDT makes use of Python for flow control, including loops and conditionals. The Jinja template engine[10] adds the ability to add flexibly data values. The full specification for JSLDT is available in the SETLr documentation.[11]

[5] http://pandas.pydata.org/
[6] https://pypi.python.org/pypi/ijson
[7] https://www.crummy.com/software/BeautifulSoup/
[8] https://docs.python.org/2/library/xml.etree.elementtree.html
[9] https://github.com/RDFLib/rdflib
[10] http://jinja.pocoo.org/
[11] https://github.com/tetherless-world/setlr/wiki/JSLDT-Template-Language

3.3 Load

Load operations consist of either serializations to files on disk using an RDFlib-supported RDF file format from one of the generated graphs, or SPARQL UPDATE operations to remote SPARQL endpoints. These operations are the end-stage of SETL scripts, and are not usually used in other operations.

4 Reusability

As shown below, SETLr has been applied to a number of tasks in several projects, and is ready for use by the Semantic Web community. A Google Groups mailing list[12] has been created for support and discussion. As of publication, the list has 13 subscribers from 4 organizations. Prior, SETLr was supported through a slack.com channel, which has had 148 messages posted. A tutorial for creating SETL scripts is on the SETLr GitHub page.[13] Further documentation is linked off of the tutorial. A talk was also published on YouTube that discusses the motivations of SETLr, how it was designed, and how to use it.[14]

4.1 CHEAR

The NIH-funded Child Health Exposure Analysis Resource (CHEAR) [3] project uses the Laboratory Information Management System (LIMS), Labkey[15], to store and manage ontology concepts and annotations for the CHEAR Ontology [4]. LabKey has allowed us to break the ontology up into specialized lists for domain experts. For example, CHEAR has lists for Agents, Instruments, Studies and Attributes. While the use of specific groups is great for management of terms, the conversion into a single coherent ontology is nontrivial. We use SETLr to build the majority of the ontology from Labkey. Not every part of the ontology can be expressed in a table, however. For the CHEAR Ontology, we create *fragments* for several complex topics. For example, a fragment was created to define levels of severity in the disease Eosinophilic Esophagitis. Another was created to represent different races and ethnicities. These fragments are dynamically added to the SETL script using fabric[16] for inclusion in the ontology. The fabric script and SETL file are available on GitHub,[17] however generating the ontology needs credentials to access LabKey.

4.2 Nanomine

The NanoMine project aims to create a nanomaterials knowledge graph that relates processing methods, structure, and properties of the nanomaterials to

[12] https://groups.google.com/forum/#!forum/setlr
[13] https://github.com/tetherless-world/setlr/wiki/SETLr-Tutorial
[14] https://www.youtube.com/watch?v=ocbyTjyBD5w
[15] http://www.labkey.com
[16] http://www.fabfile.org/
[17] https://github.com/tetherless-world/chear-ontology

help design new materials and predict their properties. The initial version of NanoMine was based on NIST's MDCS prototype and used an XML schema as a data standard for curating findings from papers. Because of this, the search capability was limited. We have been developing an ontology for NanoMine using SETLr by adding classes, definitions, and class relationships to a Google Sheets spreadsheet. A SETLr script is used to generate the NanoMine ontology from the spreadsheet. To construct the knowledge graph, we use fabric with a SETL script to iterate over XML files, creating nanopublications [5] for each one. The resulting knowledge graph and ontology are loaded into a SPARQL endpoint, and is much easier to query than the original XML database. We use the generated ontology to provide enriched visualizations and user interfaces.

4.3 Global Childhood Health

One SETLr script can be used to process multiple data files, provided the script has instructions for how to transform any column header it may encounter across the files. One of our childhood health projects aims to compare numerous datasets from studies that have already been performed, by way of a semantic web application for browsing and visualizing data. This project has a data dictionary, which we expand into a Semantic Data Dictionary (SDD), by including the URIs for the entities in the table as well as for the relationships between them. With URI definitions for every column, we can generate a SETLr script for it using an additional script. This script contains the target structure information for the linked data conversion - in our use case, based on the Semanticscience Integrated Ontology (SIO) [6]. SETLr is also used to construct ontologies collaboratively. Users contribute to a spreadsheet including fields such as *rdfs:label*, *rdfs:subClassOf*, and *skos:definition* to build up a list of concepts for a domain ontology and describe their hierarchy. SETLr extracts the table from Google Sheets, so any changes that may be necessary can be made in the sheet. Re-running the SETLr script updates the ontology accordingly.

4.4 SemNext

SETLr has also been used to generate RDF from tabular data describing gene-disease interactions in support of the SemNExT project [7]. SETLr allowed the output RDF to be formatted according to the properties of the entities represented in the tables. This was done by directly embedding Python conditional statements in the SETLr workflow description. This greatly simplified or eliminated the need for data cleaning when pulling in results from Excel spreadsheets generated by collaborators since any unexpected values could be handled using simple checks. For example, *owl:sameAs* assertions between a SemNExT gene URI and its Bio2RDF [8] counterpart were created given that the ingested value was an integer, as opposed to a string representing an HGNC name. The end result was a small, well-formed .trig file which was used to completely replace an existing subgraph in the SemNExT knowledge graph.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.26476v1 | CC BY 4.0 Open Access | rec: 31 Jan 2018, publ: 2 Feb 2018

4.5 Education Ontology

In response to many states having adopted the Common Core State Standards[18] an ontology based on these standards was created. The ontology was designed with the use case of an intelligent tutoring system in mind. In the management and compilation of the Education Ontology, SETLr proved to be an invaluable tool. An approach for managing the ontology was to create shared tables using Google Sheets, with each containing various aspects of the ontology. One table may contain information regarding education levels, grades, and subjects, while another contains more granular topics for each of the subjects, as well as their applicable grades. A different table include references to Common Core standards that are applicable to a specific course. A final table represents questions and the possible answers, as well as related content, such as figures. SETLr was used to convert these various tables into a single coherent ontology. The use in this project is an example of how SETLr can process multiple inputs and return a single RDF/RDFS output.

4.6 DBLP

As part of an undergraduate research project for creation of scientific knowledge graphs, we attempted to create a semantic interpretation for the data from the computer science bibliography project, DBLP. Initial efforts have resulted in a SETL script that uses streaming XML processing to create records that are aligned with the standards used by the DOI Linked Data service, but are expanded to include non-DOI papers as well as citation graphs. The SETL file is available on GitHub.[19]

5 Discussion

SETLr focuses on building capabilities from existing best-of-breed software and standards. These capabilities are combined in ways that allow for maximum flexibility in the creation of RDF, and attempt to minimize any limitations on the use of any one tool. For instance, JSON-LD is fully supported in the JSLDT language. That means that contexts, nested resources, and named graphs are all available to easily create complex data. Further, every string that is embedded in JSLDT, except for Python expressions, is evaluated as a template. As a result, contexts can be generated dynamically, and property names can be generated or even returned from complex Python functions (or even remote API calls). This design also makes maintenance much simpler. The use of ETree, ijson, BeautifulSoup, and Pandas libraries means that the capabilities of those libraries are available by the user, including complex summarization and data transformation operations available in Python. We found that the different projects using SETLr required slightly different approaches to things like ontology generation and data

[18] http://www.corestandards.org/standards-in-your-state/
[19] https://github.com/jerrylei98/graphene/blob/master/curation/dblp/dblp.setl.ttl

curation. One project (CHEAR) uses a JSON-based API to access data, another (Nanomine) provides a dump of XML files. A third (DBLP) uses a single large XML file, which required inclusion of stream processing capabilities in XML. As we applied SETLr to these projects, we were thoughtful about extending its capabilities. Instead of focusing on incorporating special-purpose capabilities, like entity resolution or identifier lookups, we provide hooks for those capabilities by allowing secondary tables and custom Python functions that can call out to remote APIs. Because of this approach, we have found SETLr to be quick to learn for simple tasks, powerful to use for complex tasks, and easy to maintain.

6 Availability

SETLr is licensed under Apache License 2.0 and is available at GitHub.[20] It is also available through the Python Package Index (PyPI) as the package *setlr*. SETLr builds on existing open standards, especially the W3C PROV-O [9]. SETLr also uses the W3C draft CSV on the Web (CSVW) vocabulary[21] to describe the formatting of CSV documents. It also uses properties and classes from Dublin Core Terms (DC-Terms) [10], Provenance Vocabulary[22], W3C SPARQL Service Description,[23], Vocabulary of Interlinked Datasets (VoID),[24], and the Linked Data API Vocabulary.[25]

7 Future Work

We plan to expand work on streaming-based data processing. For instance, tabular data is processed row-by-row, and if the data is not reused in the workflow, only the current row needs to be stored in memory for processing, allowing for very large tabular datasets. We will generalize the custom Python code process to make SETLr fully modular- all activities would be defined in terms of imported SETL scripts, with the core Python code only serving as an execution engine. As part of this process, we plan to provide activity templates that are direct REST API calls by templating the URL (and parameters) as well as the POST data using the Jinja2 template language. These methods would be called as functions like current python functions. In prior prototypes of SETLr, we developed methods for constructing RDF using SPARQL UPDATE and CONSTRUCT queries. Those methods my be useful in some cases, especially since they allow the incorporation of SPARQL federated queries to external sources. We are also considering adding support for ODBC data sources. Finally, we are planning to create a Javascript-based user interface for editing SETL scripts.

[20] https://github.com/tetherless-world/setlr
[21] https://www.w3.org/ns/csvw
[22] http://purl.org/net/provenance/ns
[23] http://www.w3.org/ns/sparql-service-description
[24] http://rdfs.org/ns/void
[25] http://lov.okfn.org/dataset/lov/vocabs/api

8 Conclusions

SETLr is a powerful, new tool for creating RDF from existing data sources, including tabular data, XML, JSON, and HTML. It uses best of breed standards and libraries, like PROV-O, JSON-LD, RDFlib, Pandas, and ETree, to enable powerful transformations of data into RDF. It has been used in a number of projects to build ontologies from online spreadsheets and to convert data from many different formats and scales into RDF, including nanopublication generation, which requires intensive use of named graphs. It has active use through a number of projects, and has seen interest from many external collaborators. SETLr is in active development and is available on GitHub and through PyPI as the *setlr* package.

References

1. Maali, F., Cyganiak, R., Peristeras, V.: Re-using cool uris: Entity reconciliation against lod hubs. LDOW **813** (2011)
2. Lebo, T., Erickson, J.S., Ding, L., Graves, A., Williams, G.T., DiFranzo, D., Li, X., Michaelis, J., Zheng, J.G., Flores, J., Shangguan, Z., McGuinness, D.L., Hendler, J.: Producing and using linked open government data in the twc logd portal (to appear). In Wood, D., ed.: Linking Government Data. Springer, New York, NY (2011)
3. Stingone, J.A., Mervish, N., Kovatch, P., McGuinness, D.L., Gennings, C., Teitelbaum, S.L.: Big and disparate data: considerations for pediatric consortia. Current opinion in pediatrics **29**(2) (2017) 231–239
4. McCusker, J.P., Rashid, S.M., Liang, Z., Liu, Y., Chastain, K., Pinheiro, P., Stingone, J.A., McGuinness, D.L.: Broad, interdisciplinary science in tela: An exposure and child health ontology. (2017)
5. Groth, P., Gibson, A., Velterop, J.: The anatomy of a nanopublication. Information Services and Use **30**(1) (2010) 51–56
6. Dumontier, M., Baker, C.J., Baran, J., Callahan, A., Chepelev, L., Cruz-Toledo, J., Del Rio, N.R., Duck, G., Furlong, L.I., Keath, N., Klassen, D., McCusker, J.P., Queralt-Rosinach, N., Samwald, M., Villanueva-Rosales, N., Wilkinson, M.D., Hoehndorf, R.: The semanticscience integrated ontology (sio) for biomedical research and knowledge discovery. Journal of Biomedical Semantics **5**(1) (2014) 14
7. Patton, E.W., Brown, E., Poegel, M., De Los Santos, H., Fasano, C., Bennett, K.P., McGuinness, D.L.: Semnext: A framework for semantically integrating and exploring numeric analyses. In: SemStats@ ISWC. (2015)
8. Callahan, A., Cruz-Toledo, J., Ansell, P., Dumontier, M.: Bio2rdf release 2: improved coverage, interoperability and provenance of life science linked data. In: Extended Semantic Web Conference, Springer (2013) 200–212
9. Lebo, T., Sahoo, S., McGuinness, D.: PROV-O: The PROV Ontology. http://www.w3.org/TR/prov-o/ (2013)
10. Dublin Core Metadata Initiative: DCMI metadata terms. http://purl.org/dc/terms/ (2012)