OntoPy: a framework to integrate different file types

Pedro Paulo Rezende Silva Domingos[1], José Maria Parente de Oliveira[2]

[1]Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos, SP, Brasil
[2]Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos, SP, Brasil

Abstract

The OntoPy framework is introduced in this paper as a new solution for data integration through ontologies, aiming to provide an accessible approach to integrating various types of data sources. OntoPy responds to challenges in scenarios where organizations are structuring their data and seeking exploratory analyses to derive insights for decision-making. Unlike existing tools such as Ontop, which focus on structured databases and require commercial software and plugins, OntoPy leverages data science-friendly features of Python to simplify the integration of unstructured data in formats like Parquet, CSV, and XLSX. This framework offers a more versatile and efficient method for bridging the semantic heterogeneity of data and ontologies. By emphasizing the relationship between ontology classes and properties and the attributes available in data sources, OntoPy facilitates a seamless and effective data integration process. Initial tests with the framework have shown promising results, with adequate performance in handling larger data volumes compared to other tools such as Morph-KGC. OntoPy has been successfully applied in querying data for diesel-electric locomotive maintenance, handling complex queries across heterogeneous data sources with good performance.

Keywords

Data integration, ontology, mapping, databases, framework, software engineering

1. Introduction

Data integration is considered a recurring problem in data management, and it is observed to be a significant challenge today [1]. It is estimated that 50 to 80% of a data scientist's time is dedicated to manipulating, integrating, and preparing data for effective use [1]. In this context, the use of ontologies emerges as an important tool for semantically modeling concepts and relationships in data domains and integrating different data sources.

An ontology can be defined as a formal representation of a set of concepts within a domain and the relationships between those concepts [2]. The Ontology-Based Data Access (OBDA) approach bridges the semantics of ontologies and data heterogeneity, with mature and widely adopted solutions like Ontop [3] to integrate different data sources.

However, the availability of data in various application scenarios is not always structured, often being in files such as Parquet, CSV, and XLSX formats. In scenarios like these, structuring multiple files to then use a framework like Ontop can result in unnecessary effort in data sources that may not be used later on. It is possible to use Ontop through federated bases, accessing various types of files, but with the need for commercial software and plugins, as well as additional complexity involved in establishing this structure. Ontop is written in Java,

XIV International Confrence on Formal Ontology in Information Systems (FOIS 2024) and X Joint Ontology Workshops (JOWO 2024)

domingos@ita.br (P. P. R. S. Domingos); parente@ita.br (J. M. Parente de Oliveira)

Frameworks	Language	File types	Ontology and mapping
Ontop [4]	Java	RDB	OWL e R2RML
MASTRO [5]	Java	RDB	OWL e R2RML
Morph-KGC [6]	Python	RDB, CSV, TSV, XLSX, Parquet, Feather, JSON, XML, DataFrame Pandas e outros	RML
Morph-CSV [7]	Java	RDB e CSV	RML
Morph-RDB [8]	Scala	RDB e CSV	RML
Morph-xR2RML [9]	Scala	RDB, CSV, XML, TSV, JSON e NoSQL	RML
Owlready2 [10]	Java	-	OWL
RMLMapper [11]	Java	RDB, CSV, XML, TSV, JSON e XLSX e ODS	R2RML
Karma [12]	Java	RDB, CSV, XML, TSV e JSON	K2RML
TripleWave [13]	NodeJS	JSON	R2RML
SPARQL-Anything [14]	Java	RDB, CSV, XML, HTML, JSON e TXT	R2RML
Optique [15]	Java	-	-

Table 1
Features among related frameworks

not making use of data science-focused features as in languages like Python. There is an opportunity to make the OBDA approach and the use of ontologies more accessible, both in terms of application structure and programming language.

Regarding the most widespread mapping standards, it is noted that they require specific knowledge for their development. Since the data scenario in question would be in the process of structuring, it is understood that the use of ontologies for data integration would be at the same level. Thus, it is believed that this process can become more accessible by focusing exclusively on the relationship between classes and properties of the defined ontology and the available attributes in the data sources.

Thus, this paper presents the OntoPy framework, developed for integrating different types of data sources through ontologies in an accessible manner. It addresses the scenario where an organization is in the process of structuring its data and seeking exploratory analyses to better understand the value data can provide in decision-making processes.

2. Related works

Ontop is the most widespread tool when it comes to OBDA solutions. However, it is a solution that deals with relational databases, either already structured or accessible through federated databases generated by third-party applications. For the proposed framework, a greater versatility is sought, as it relates to a scenario of structuring an organization's data.

Through related works, it is observed that most of them operate on relational databases or files such as CSV, JSON, XML, and RDF (Table 1). Among other solutions, Morph-KGC also stands out, mainly because of its similarity to the method proposed in this present document. It can access various types of different file systems, as well as pandas dataframes, generating results in this format as well. The main difference from the present proposal is that this framework uses only the TTL file as input, focusing on mapping, which may or may not contain additional statements from an ontology. By using only a TTL file, some concepts and practices of using ontologies for data integration are not met, such as reusability, sharing, and portability across multiple platforms, as well as increased maintainability and reliability [16].

In the case of a dedicated file for ontology and another for mapping, such as in Ontop, MASTRO, and the proposed OntoPy framework, the OWL file would be the same for all three solutions, requiring adjustments only in the mapping. This complexity of adjustments becomes

more evident, particularly in extensive ontologies or when dealing with multiple different data sources.

3. OntoPy framework

The research method is primarily based on the development of the OntoPy framework for data integration. In order to develop the OntoPy framework, it must:

- Be developed in Python due to its widespread use for data science applications, its growing community, and ongoing advancements;
- Load an ontology in OWL format, following W3C standards;
- Load a mapping file in JSON format, aiming to make this step more accessible to applications where those involved are in the initial stages of learning how to use ontologies for data integration;
- Materialize data from different types of files based on the loaded ontology;
- Provide the materialized knowledge graph in execution memory for SPARQL queries.

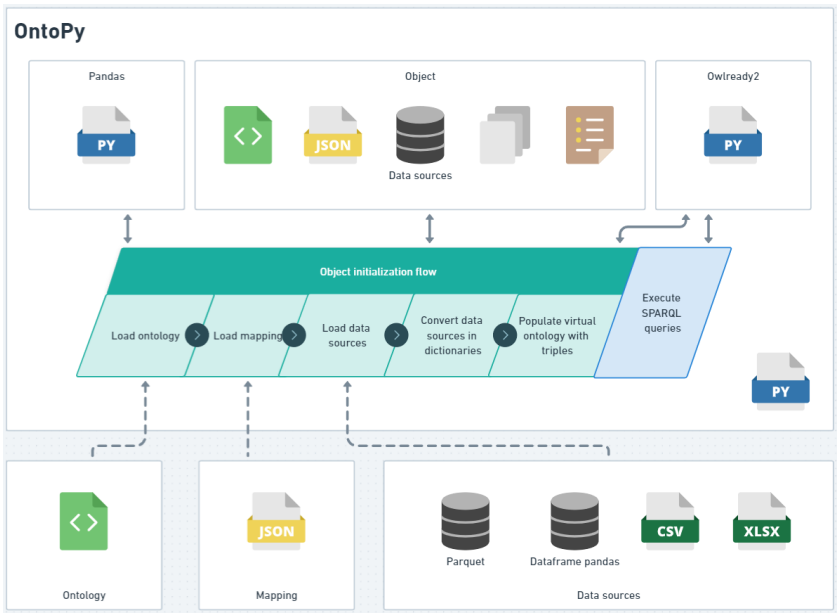

Figure 1: Architecture of the OntoPy framework

The code was developed based on the Owlready2 library to access the ontology and convert the loaded data into triples (Figure 1). Up to the current state of the framework, it is possible to load CSV, XLSX, and Parquet files. Other file types will be considered in the future. However, OntoPy also operates on databases running in pandas DataFrame objects. This capability allows the user to pre-load data sources of other types through Python code in pandas DataFrame

format and use them in Python. The OntoPy framework, its proposal, and implementation case in a freight railway transportation company are available on GitHub[1].

4. The mapping file

The proposed framework for accessing different file systems also introduces a mapping file in JSON format. The aim of this approach is to explore alternatives or propose a method that simplifies its understanding for creation. The mapping file, in addition to serving as a guide for constructing triples, defines which data should be used in the application. Therefore, even if a data source and the ontology used have many attributes, the mapping file may only include the minimum necessary to form a triple.

Listing 1: Structure of the mapping files for OntoPy in JSON format

```
1  {
2      ``test_database'' :
3      {
4          ``data_source_path'': ``C:\test.csv'',
5          ``separator'': ``;'',
6          ``decimal'': ``,'',
7          ``triples'':
8          [
9              {
10                 ``subject'':
11                 {
12                     ``data_source_attribute_name'': ``Equipment'',
13                     ``ontology_subject_name'': ``Locomotive''
14                 },
15                 ``predicates_and_objects'':
16                 [
17                     {
18                         ``data_source_attribute_name'': ``Model'',
19                         ``ontology_predicate_name'': ``hasModel'',
20                         ``ontology_object_name'': ``Locomotive_model''
21                     },
22                     {
23                         ...
24                     }
25                 ]
26             },
27             ...
28         ]
29     },
30
31     ...,
32 }
```

The mapping file follows the structure seen in Listing 1. Explaining each line index of the Listing 1:

- Index 2: Identifier for the mapped data source;
- Index 4: Path of the data source;
- Index 5: Character defining the column separation in CSV files;

[1]https://github.com/pedropdomingos/OntoPy

- Index 6: Character defining the decimal separation in numerical values in CSV files;
- Index 12: Identification of the attribute in the data source treated as the subject;
- Index 13: Identification of the corresponding ontology class that the subject attribute refers to;
- Index 18: Identification of the attribute in the data source treated as the object;
- Index 19: Identification of the corresponding ontology property responsible for the relationship between the subject and the object;
- Index 20: Identification of the corresponding ontology class that the object attribute refers to;
- Index 23: Next mapping of the predicate for the corresponding subject;
- Index 27: Next sequence of triples with the mapping of a subject and its predicates;
- Index 31: Next data source to be mapped.

The outcome of this application is very significant, as performing joins on tables through any other approach would not be a simple task, especially when considering the semantic heterogeneity of the data and the structure of each source used in the solution.

It was decided to conduct a comparative test regarding performance with Morph-KGC precisely because of the similarities with the proposed OntoPy framework. The idea was to choose a simple and easy-to-use test from Morph-KGC's Github and adapt it to OntoPy to compare the performance of the frameworks. Among the available tests, there is one related to the number of Instagram followers that is very accessible as it uses pandas DataFrames to integrate. To generate the data, a code was developed that generates identifiers from zero to the desired amount, repeating data such as first name, last name, and username on the platform, as well as random numbers for the number of followers. The comparative test was limited to comparing the processing times for inserting data from databases into the ontology or knowledge graph and the times for performing a simple SPARQL query. The query executed was to return the user identifiers and their respective number of followers.

Data	Insert data (OntoPy) (s)	Insert data (Morph) (s)	SPARQL and iterate result (OntoPy) (s)	SPARQL and iterate result (Morph) (s)
10000	2.53	3.71	0.17	0.55
100000	30.31	20.65	7.67	5.13
1000000	364.57	431.43	79.93	297.10

Table 2
Processing times in the integration test of pandas DataFrames proposed by Morph-KGC

The initial tests of OntoPy have shown promissing. In the comparative test with Morph-KGC, OntoPy excelled in performance in scenarios with a large volume of data, which is significant (Table 2). It is under such conditions that the need for performance becomes more apparent. Both presented the same results in the query, also demonstrating the reliability of the proposed approach.

5. Conclusion and results

OntoPy has been used for querying data in the domain of diesel-electric locomotive maintenance. It has been possible to perform queries that are challenging due to the heterogeneity of the

data sources, both in terms of attributes and structures, and it has shown good performance. It is understood that there is room for improvement, particularly in the stage of populating the materialized ontology with instances, but the solution has already yielded consistent results.

There are possibilities for improving the efficiency of the framework by using threads and multiprocessing resources. Additionally, the use of the Owlready2 library should be re-evaluated in terms of performance.

References

[1] G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati, Using ontologies for semantic data integration, A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years (2018) 187–202.

[2] B. Dorneanu, S. Zhang, H. Ruan, M. Heshmat, R. Chen, V. S. Vassiliadis, H. Arellano-Garcia, Big data and machine learning: A roadmap towards smart plants, Frontiers of Engineering Management 9 (2022) 623–639.

[3] D. Calvanese, D. Lanti, T. M. De Farias, A. Mosca, G. Xiao, Accessing scientific data through knowledge graphs with ontop, Patterns 2 (2021) 100346.

[4] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, G. Xiao, Ontop: Answering sparql queries over relational databases, Semantic Web 8 (2017) 471–487.

[5] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, D. F. Savo, The mastro system for ontology-based data access, Semantic Web 2 (2011) 43–53.

[6] J. Arenas-Guerrero, D. Chaves-Fraga, J. Toledo, M. S. Pérez, O. Corcho, Morph-KGC: Scalable knowledge graph materialization with mapping partitions, Semantic Web 15 (2024) 1–20. doi:10.3233/SW-223135.

[7] D. Chaves-Fraga, L. Pozo-Gilo, J. Toledo, E. Ruckhaus, O. Corcho, Morph-csv: Virtual knowledge graph access for tabular data., in: ISWC (Demos/Industry), 2020, pp. 11–16.

[8] F. Priyatna, O. Corcho, J. Sequeda, Formalisation and experiences of r2rml-based sparql to sql query translation using morph, in: Proceedings of the 23rd international conference on World wide web, 2014, pp. 479–490.

[9] F. Michel, L. Djimenou, C. F. Zucker, J. Montagnat, Translation of relational and non-relational databases into rdf with xr2rml, in: 11th International Confenrence on Web Information Systems and Technologies (WEBIST'15), 2015, pp. 443–454.

[10] L. Jean-Baptiste, Ontologies with Python: Programming OWL 2.0 Ontologies with Python and Owlready2, Springer, 2021.

[11] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van de Walle, Rml: A generic language for integrated rdf mappings of heterogeneous data., Ldow 1184 (2014).

[12] J. Slepicka, C. Yin, P. A. Szekely, C. A. Knoblock, Kr2rml: An alternative interpretation of r2rml for heterogenous sources., in: Cold, 2015.

[13] A. Mauri, J.-P. Calbimonte, D. Dell'Aglio, M. Balduini, M. Brambilla, E. Della Valle, K. Aberer, Triplewave: Spreading rdf streams on the web, in: The Semantic Web–ISWC 2016: 15th

International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part II 15, Springer, 2016, pp. 140–149.
[14] E. Daga, L. Asprino, P. Mulholland, A. Gangemi, et al., Facade-x: an opinionated approach to sparql anything, Studies on the Semantic Web 53 (2021) 58–73.
[15] S. Kamm, N. Jazdi, M. Weyrich, Knowledge discovery in heterogeneous and unstructured data of industry 4.0 systems: challenges and approaches, Procedia CIRP 104 (2021) 975–980.
[16] H. Li, Ontology-Driven Data Access and Data Integration with an Application in the Materials Design Domain, Ph.D. thesis, Linköping University Electronic Press, 2022.