Data Search and Discovery in RDF Sources

Zoé Chevallier[1,2], Zoubida Kedad[1], Béatrice Finance[1], and Frédéric Chaillan[2]

[1] David Lab. University of Versailles Paris-Sclay, Versailles, France
firstname.lastname@uvsq.fr
[2] Grand Paris Sud, Evry-Courcouronnes, France
{z.chevallier, f.chaillan}@grandparissud.fr

Abstract. The RDF data sources published on the Web represent an unprecedented amount of knowledge. However, querying these sources to extract the relevant information for some specific needs represented by a target schema is a complex task, as the alignment between the target and the source schemas might not be provided or may be incomplete. This paper presents a system that aims to automatically populate the classes of a target schema from RDF data sources by identifying candidate instance patterns. This identification process relies on a semi-supervised learning algorithm and the system automatically generates the SPARQL queries that populate the target schema.

Keywords: RDF data sources · Target Schema Instantiation · SPARQL Query Generation · Semi-supervised learning.

1 Introduction

The Web represents a huge space of available data from which various applications can extract meaningful knowledge. However, finding relevant data for some specific need is not obvious, especially for irregular data sources. This problem has been addressed by dataset discovery approaches [1, 3], which aim at discovering the relevant datasets that could complement a given target dataset. These approaches are designed for structured datasets.

Considering that the specific needs of an application are described by a target schema, our problem is the identification and the extraction of relevant data to populate this target schema. A similar problem is the one of mapping generation which has been the subject of several works [2, 4], targeting relational or XML data. Sacramento et al. [5] have addressed the problem of expressing an RDF data source in the terms defined by an ontology, which consists in generating a mapping between a source and this ontology. This requires the alignment between the source schema and the target schema, which is not always provided.

In this paper, we present a system that identifies candidate instances from RDF data sources to populate a given target schema. It relies on a semi-supervised learning algorithm to extract candidate instance patterns from an RDF data source and automatically generates the queries that extract these instances. This paper is organized as follows. Section 2 presents the architecture of our system. Section 3 details a use-case scenario, and section 4 presents some future works.

2 System Architecture

Fig. 1 depicts the architecture of our system. We consider that the target schema is described in RDFS[3] /OWL[4] . The alignment between the source schemas and the target schema can be exploited if they are available, but they are not required.

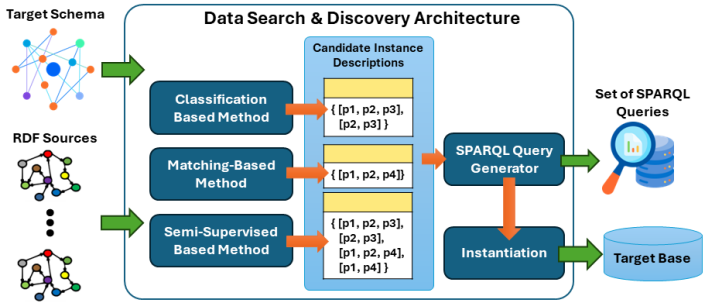

Fig. 1: The Data Search and Discovery System Architecture

Our system identifies in RDF sources candidate instance patterns in order to populate the target schema. Each pattern is a property set which describes some candidate instances in the data source. Given a candidate pattern, a SPARQL Query is generated to retrieve all the candidate instances and automatically populate the target schema.

We assume that each class C of the target schema is described by a set of properties, $Prop(C)$, such that $Prop(C) = \{p \mid < p, rdfs{:}domain, C > \in T\}$. An entity e in an RDF data source S is a resource that is neither a class, a property, a literal, or a blank node; it is characterized by a set of properties, $Prop(e)$, such that: $Prop(e) = \{p \mid < e, p, * > \in S\}$.

A straightforward way to identify candidate instance patterns is to compute the similarity between source entities and a target class C. If the similarity is higher than a threshold, then $Prop(e)$ is a candidate pattern for C. We refer to this process as ***classification-based***. If the output of some schema matching tool is provided, and if a target class C_T is equivalent to a source class C_S, then all the instances of C_S are candidate instances for C_T. We refer to this process as ***matching-based*** instantiation.

However, these correspondences are not always provided. Besides, the schema in RDF sources is only descriptive, and an instance can be characterized by a property set that is different from the one defined in the schema for its class. Therefore, new candidate instance patterns should be identified based on the class description in the target schema, but also based on the candidate instance patterns already identified for a class. To do so, we propose an approach based on a semi-supervised learning algorithm [6] to identify candidate patterns.

[3] `https://www.w3.org/TR/rdf-schema/`
[4] `https://www.w3.org/TR/owl-features/`

An entity which is similar to a candidate pattern already identified for a class C could also be a candidate instance for C. We propose an approach that iteratively computes the similarity between each entity and the candidate patterns already identified. If the similarity between e and the candidate instance patterns of C is above a predefined threshold, then $Prop(e)$ is considered as a candidate instance pattern for C. We introduce the notion of candidate instance description that represent the patterns of candidate instances for a target class C. It is such that:

$$\mathcal{D}(C) = \{\mathcal{D}_i \,|\, \mathcal{D}_i = Prop(e), \text{e candidate instance of C}\} \tag{1}$$

If $\mathcal{D}(C) = \{\ \mathcal{D}_1, ..., \mathcal{D}_n\ \}$ is a set of candidate instance descriptions, the similarity based on the instances is denoted $Sim_I(e, \mathcal{D}(C))$, and is defined as follows:

$$Sim_I(e, \mathcal{D}(C)) = MAX_{1 \leq i \leq n} \left(\frac{|Prop(e) \cap \mathcal{D}_i|}{|Prop(e) \cup \mathcal{D}_i|} \right) \tag{2}$$

The semi-supervised algorithm first computes candidate instance descriptions using both the matching-based and classification-based approaches. For a target class C, the similarity between each entity e and C, Sim_I is computed. If the similarity is higher than a given threshold, then $Prop(e)$ is a candidate instance description of C. If new candidate instance descriptions are identified, a new iteration is started. This identification process iterates until no more new candidate instance descriptions are found.

Consider $\mathcal{D}_i$ a candidate instance description of C such that $\mathcal{D}_i = \{\ p_1, ..., p_n\}$. A SPARQL query is generated for $\mathcal{D}_i$ in order to extract from the considered dataset S all the entities characterized by a property set identical to $\mathcal{D}_i$, i.e., the set in which each entity e is such that $Prop(e) = \mathcal{D}_i$.

The entities and their associated triples are extracted, and a type declaration is defined for each of them stating that their corresponding type is C. This is represented by adding the triple pattern <?e, rdf:type, C> in the basic graph pattern.

In addition, if p is a property in $\mathcal{D}_i$, then the triple pattern <?e, p ?o_p> is added to the basic graph pattern.

We implement an exclusion mechanism to filter out the entities which are characterised by a superset of $\mathcal{D}_i$. We used a *minus* operator in the query that exclude the entities that are described by properties which are not contained in the description, i.e. <?e, ?prop, ?o>, where ?$prop \notin \mathcal{D}_i$.

Fig. 2 depicts the SPARQL query generated for the description:
$\mathcal{D}(t{:}Person) = \{$ firstname, lastname $\}$.

3 Demonstration Scenario

Our system is implemented in Java, using Apache Jena. An online presentation of the demonstration is available[5]. The demonstration will proceed in four phases:

[5] `https://mega.nz/folder/QK0lRSha#dSW_77ur0QBPNp7ff4_qXg`

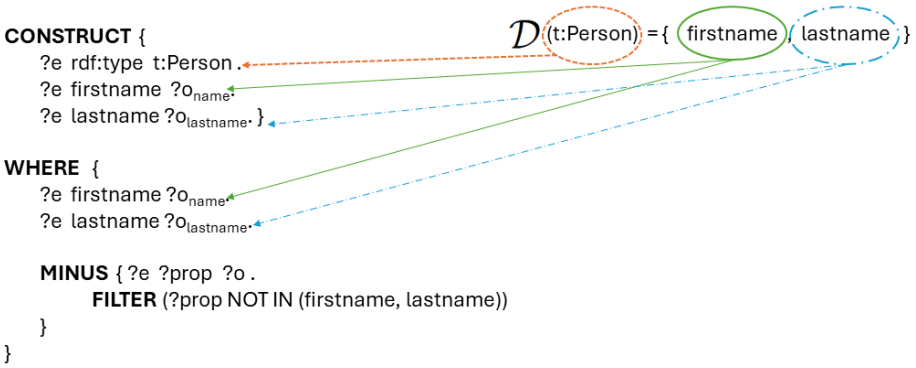

Fig. 2: A Candidate Instance Description and the Corresponding SPARQL Query

1. **Configuration.** The process starts by selecting a project composed of a RDFS/OWL target schema, RDF sources, and optionally a set of correspondences between the target schema and the RDF sources. A graphical representation of both the target schema and the data sources is presented.
2. **Identification of Candidate Instance Descriptions.** In this phase, we will show the generation of candidate instance descriptions using the different approaches. The candidate instance descriptions identified for each target class will be presented as well as their provenance, i.e. the data source from which they have been extracted. Moreover, a visualization tool allow the user to highlight the candidate instances identified in the source.
3. **Generation of SPARQL Queries.** We will then demonstrate the query generation process, and show the queries generated for each candidate instance description and the execution result, such as the one depicted in Fig.3.
4. **Comparison of the Candidate Instance Sets.** In order to highlight the quality of the semi-supervised approach, we will compare its resulting candidate instance sets with the ones extracted using a baseline method which consists in extracting the candidate instance descriptions obtained using both the matching-based approach and the classification-based approach.

4 Future Works

In future works, we will improve the query generation process in our data discovery system by taking into account some constraints defined on the target schema so as to extract only the source instances for which the constraints are verified. These constraints will be expressed in SHACL[6], and exploited during query generation. We will also explore the possible ways of minimizing the number of generated queries by grouping the candidate instance descriptions that share some properties before generating the queries.

[6] `https://www.w3.org/TR/shacl/`

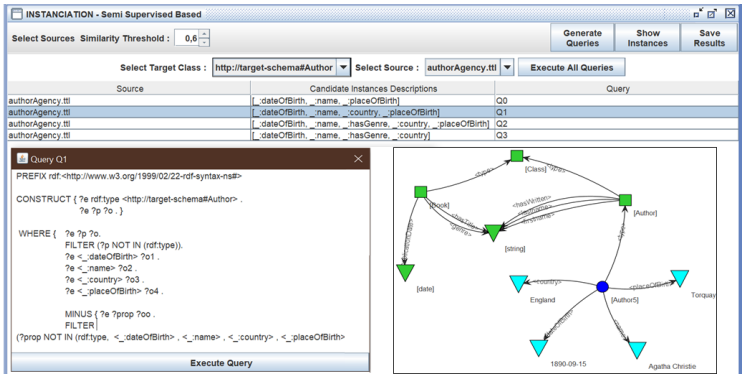

Fig. 3: A Screenshot of the Query Generated from a Candidate Description

References

1. Castro Fernandez, R., Abedjan, Z., Koko, F., Yuan, G., Madden, S., Stonebraker, M.: Aurum: A Data Discovery System. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE). pp. 1001–1012. IEEE, Paris (Apr 2018)
2. Fagin, R., Haas, L.M., Hernandez, M., Miller, R.J., Popa, L., Velegrakis, Y.: Clio: Schema Mapping Creation and Data Exchange | SpringerLink. In: Conceptual Modeling: Foundations and Applications, Lecture Notes in Computer Science, vol. 5600, pp. 198–236 (2009)
3. Koutras, C., Siachamis, G., Ionescu, A., Psarakis, K., Brons, J., Fragkoulis, M., Lofi, C., Bonifati, A., Katsifodimos, A.: Valentine: Evaluating Matching Techniques for Dataset Discovery. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE). pp. 468–479 (Apr 2021)
4. Mazilu, L., Paton, N.W., Fernandes, A.A., Koehler, M.: Schema mapping generation in the wild. Information Systems **104**, 101904 (Feb 2022)
5. Sacramento, E.R., Vidal, V.M.P., de Macêdo, J.A.F., Lóscio, B.F., Lopes, F.L.R., Casanova, M.A.: Towards automatic generation of application ontologies. J. Inf. Data Manag. **1**(3), 535–550 (2010)
6. Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. In: Proceedings of the 33rd annual meeting. Association for Computational Linguistics, Cambridge, Massachusetts (1995)