Enhancing JSON to RDF Data Conversion with Entity Type Recognition

Fellipe Freire, Crishane Freire and Damires Souza
Academic Unit of Informatics, Federal Institute of Education, Science and Technology of Paraiba, João Pessoa, Brazil

Keywords: RDF Data Conversion, Entity Type Recognition, Semi-Structured Data, JSON Documents, Semantics Usage.

Abstract: Nowadays, many Web data sources and APIs make their data available on the Web in semi-structured formats such as JSON. However, JSON data cannot be directly used in the Web of data, where principles such as URIs and semantically named links are essential. Thus it is necessary to convert JSON data into RDF data. To this end, we have to consider semantics in order to provide data reference according to domain vocabularies. To help matters, we present an approach which identifies JSON metadata, aligns them with domain vocabulary terms and converts data into RDF. In addition, along with the data conversion process, we provide the identification of the semantically most appropriate entity types to the JSON objects. We present the definitions underlying our approach and results obtained with the evaluation.

1 INTRODUCTION

The Web has evolved into an interactive information network, allowing users and applications to share data on a massive scale. To help matters, the Linked Data principles define a set of practices for publishing structured data on the Web aiming to provide an interoperable Web of Data (Heath and Bizer, 2011). These principles are based on technologies such as HTTP, URI and RDF (Heath and Bizer, 2011). By using the RDF model, data or resources are published on the Web in the form of triples (composed by a subject, a predicate and an object), where each resource is individually identified by means of URIs. To achieve this, it is essential to provide the conversion of data available in different formats (e.g., XML, JSON) to RDF data.

Publishing structured data on the Web faces some challenges related mainly to the large number of data sources, their autonomous nature, and the heterogeneity of their data (Alexe et al., 2013; Fanizzi et al., 2012). It is usually hard to convert data without considering the knowledge domain (e.g., “Healthy”, “Music”, “Books”) in which the data exist. Delimitating a specific knowledge domain at conversion time may help coping with the amount of heterogeneous data (Fanizzi et al., 2012; Silva et al., 2013). With respect to this, one of the principles of this work is using semantics provided by the knowledge domain of the data to enhance their conversion to RDF data. To this end, we use recommended open vocabularies which are composed by a set of terms, i.e., classes and properties useful to describe specific types of things (LOV Documentation, 2016).

Regarding data already on the web, the deployment of applications and services generating semi-structured data has increased every day. For instance, most of the information published on the Web as semi-structured data are particularly in JSON or XML formats. These are indeed flexible formats, where missing values or extra fields are allowed, enabling easier data exchange between sources (Klettke et al., 2015). This means that in semi-structured data, elements of the same “type” might have different number of properties.

Particularly, nowadays, it is usual to deal with JSON data on the web. For instance, data acquired from social networks are commonly obtained in such format. JSON stands for JavaScript Object Notation. It is considered as a lightweight data-interchange format (JSON, 2016). In a JSON document, the instances are called objects. Objects are an unordered enumeration of properties, consisting of name (key) and value pairs (JSON, 2016). In this work, JSON object keys are considered as structural metadata, i.e., high-level information about the

Freire, F., Freire, C. and Souza, D.
Enhancing JSON to RDF Data Conversion with Entity Type Recognition.
DOI: 10.5220/0006302900970106
In *Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST 2017)*, pages 97-106
ISBN: 978-989-758-246-2

schema and internal structure of a dataset. JSON data are defined as semi-structured ones, and are mainly characterized by a lack of a fixed schema.

Since a large amount of data published on the Web is structured as JSON format, which thus cannot be directly used in the Web of data, it is necessary to convert them into RDF data. According to their nature, JSON documents are constructed in a flexible way and may have a metadata schema minimally defined if we consider their key-value pairs. Nevertheless, the entity types (or classes) associated with the existing instances (objects) are not commonly identified. This can be defined as an *entity type recognition problem*, i.e., the process of recognizing entities and their types (Sleeman and Finin, 2015). Thus, for the generation of RDF data, it is rather important to identify the metadata present in a JSON document (properties) and, particularly, to recognize semantically adequate entity types associated to their instances. In this scenario, this work presents a proposal for JSON data conversion to RDF, including in this task, the identification of the most appropriate entity types for the instances (objects). Also, it includes the usage of open vocabularies in order to reference the data with suitable domain terms. The goal is to generate richer RDF documents and to facilitate query formulation and consequently more accurate results.

Our contributions are summarized as follows:

(i) We present a domain-based approach to convert JSON documents into RDF data by using open domain vocabularies;

(ii) We propose an entity type recognition approach along with the data conversion process;

(iii) We present a conversion tool that implements the proposed approach; and

(v) We describe some experiments regarding the effectiveness of the data conversion and entity type recognition processes.

The remainder of this paper is organized as follows: Section 2 introduces a motivating example with some background concepts; Section 3 proposes our approach; Section 4 presents the developed tool; Section 5 describes some accomplished experiments to evaluate our proposal; Related works are discussed in Section 6. Finally, Section 7 draws our conclusions and points out some future work.

2 MOTIVATING EXAMPLE AND SOME CONCEPTS

Suppose a web application which presents data regarding *software projects*. Together with the data visualization option, it provides a web service where data can be acquired in JSON format. For instance, consider a software project description as showed in Figure 1.

In Figure 1, we have a set of keys (properties) with their respective values, which describes a given project. Nevertheless, in the presented JSON document, the described object does not have its entity type explicitly defined, i.e., it is not stated that such instance description refers to a “software project”. Such information is usually optional.

```
{
"homepage":"www.arboviz.com.br",
"repository":"github.com/arboviz",
"title":"arboViz",
"developer":"Marcio Alves",
"tester":"Carina Monte",
"helper":"Jonas Brim",
"group":"SIDE Group",
"programmingLanguage":"Java",
"status":"Doing"
}
```

Figure 1: JSON dataset example.

The “Resource Description Framework" or RDF (RDF Documentation, 2016) is a data model that was proposed by the World Wide Web Consortium (W3C) as a standard for publishing datasets on the Web. Its schema-free model makes RDF an interesting mechanism for describing objects or resources in such a way that diverse data publishers can add information about the same object/instance (resource), or create named links between different or similar ones (Heath and Bizer, 2011).

The RDF syntax is based on triples in the form of subject-predicate-object expressions. An RDF graph is a set of such triples with nodes being subjects or objects, and labelled edges being predicates. In addition, an RDF dataset is a set of RDF graphs (RDF Documentation, 2016).

According to the JSON data provided in Figure 1, a possible correspondent RDF distribution could be as the one depicted in Figure 2. In this case, the data converted to RDF has been serialized in Turtle syntax.

In this example, we have used the *doap* vocabulary in order to semantically refer the data (DOAP, 2016). As an illustration, we have also pointed out a possible named link with the

DBPEDIA concept "Project" (DBPEDIA, 2016). On the other hand, an RDF triple indicating the entity type of that object is indeed missing. This is the problem we address in this paper. We are interested in discovering the entity types and attributes that can be used to generate richer RDF data along with the data conversion itself. This example will be used throughout the paper to illustrate our approach and experiments.

```
   @prefix rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-
ns#>.
   @prefix rdfs:
<http://www.w3.org/2000/01/rdf-schema#>.
   @prefix xsd:
<http://www.w3.org/2001/XMLSchema#>.
   @prefix doap:
<http://usefulinc.com/ns/doap#>.
   @prefix dbp:
<http://dbpedia.org/ontology/>
   <http://www.ifpb.edu.br/side/projects/pro
ject10#id>
    doap:homepage
           "www.arboviz.com.br";
    doap:repository
           "github.com/arboviz";
    doap:title "arboViz",
    doap:developer "Marcio Alves",
    doap:tester "Carina Monte",
    doap:helper "Jonas Brim",
    doap:group "SIDE Group",
    doap:programmingLanguage "Java",
    doap:status "Doing"
      dbp:primaryTopicOf
  http://dbpedia.org/ontology/Project.
```

Figure 2: Example Data in RDF/Turtle.

3 OUR APPROACH

In this section, we introduce the SenseRDF approach which has been extended in this current work. Then we present some definitions underlying our work along with the proposed extension.

3.1 The SenseRDF Principles

This work extends the RDF data conversion approach introduced in Silva et al., (2013). Named as the SenseRDF approach, it allowed the conversion of existing datasets in XML format into RDF data.

In the SenseRDF approach, the data conversion effort is incrementally accomplished if the datasets to be converted belong to the same knowledge domain (e.g., "Bibliographic Data", "Music"). To this end, it is essential to have assistance of a *Domain Expert* (DE). The DE is a person that has an understanding of the content to be converted and the knowledge domain underlying the data. The DE is responsible for identifying the knowledge domain (hereafter called as domain) at hand. Then s/he verifies the datasets to be converted and also points out the recommended vocabularies to be used. Thus, at conversion time, all datasets and vocabularies must belong to the same domain. For instance, suppose that the defined domain regards *Bibliographic data*, i.e., data (e.g., publications, conferences, journals) compiled upon some common principle, as subject, or place of publication. In this case, the DE may indicate the SWPO reference ontology (SWPO, 2016) as a recommended vocabulary to be used at datasets conversion time.

To allow the data conversion, the SenseRDF generates correspondences between the converting dataset metadata and the domain terms, which belong to the chosen vocabularies. The resulting alignment is persisted and will be reused for each other new dataset conversion that belongs to the same domain. When converting other document which belongs to the same knowledge domain, if there is no correspondence between a given metadata and a vocabulary term, the SenseRDF proceeds with another correspondence identification. As a result, it adds this new correspondence to the existing domain alignment. Still considering the bibliographic data domain, examples of correspondences between metadata and domain terms are:

```
creator ≡ dc:creator  and
publication ≡ swpo:publication
```

where

creator and *publication* are metadata from an input dataset; and

the prefixes *dc* and *swpo* are related to the DC vocabulary (DC, 2016) and to the SWPO ontology, respectively.

In this work, the SenseRDF approach has been extended in order to allow the conversion of JSON data into RDF. Furthermore, it is now able to identify entity types along with the data conversion process itself both to XML and JSON data formats. These features are presented in the next sections.

3.2 Definitions

We provide some definitions regarding the concepts underlying our approach. They refer to semi-

structured to RDF data conversion topics that we take into account.

Let *D* be a data domain (ex: "Music", "Education") which is identified by a *Domain Expert* (DE). According to *D*, source datasets are converted to RDF ones. These datasets and their instances are defined as follows.

Definition 1 (Source Dataset and Instance). Let $DS=\{ds_1,...,ds_i\}$ be a set of source semi-structured datasets which belong to *D*. Each ds_i is composed by a set of instances (or objects) $I=\{Ids_{i1},...,Ids_{ik}\}$. Each instance Ids_{ik} is defined by a set of properties (keys) as $Ids_{ik} = \{p_1, p_2, \ldots, p_n\}$.

Definition 2 (Target RDF Dataset). Let dsr_j be an RDF dataset which represents a source description ds_i after a semantic-based data conversion process. dsr_j is composed by RDF triples and instances $Idsr_m$ in such a way that $Idsr_m \in dsr_j$.

A target RDF dataset dsr_j is composed by concepts and properties, which are semantically formalized by means of domain vocabularies.

A Domain Vocabulary *dv* is a domain ontology or an open vocabulary which contains terms (classes and properties) belonging to a particular data domain *D*. A *dv* should be a reliable domain reference available on the web. Since we can have some associated vocabularies (one or more) to a given *D*, we are able to establish the set of domain vocabularies we will use to provide semantics to the data conversion process. The set of domain vocabularies to be used is defined as follows.

Definition 3 (Set of Domain Vocabularies). We state a Set of Domain Vocabularies *SDV* as $SDV=\{dv_1, dv_2,..., dv_l\}$ which have been chosen as reliable domain vocabularies to be used as background knowledge in a data conversion process.

In the light of the motivating example (Section 2), ds_1 is a JSON dataset which will be converted to an RDF one. In this example dataset, there is one instance Ids_{11} composed by some properties, as follows.

```
Ids11 = {homepage, repository, title,
developer, tester, helper, group,
programmingLanguage, status}
```

In this example, the DE considers *D* as "*Software Projects*". S/he chooses two domain vocabularies to be used to assist the data conversion process, namely `SDV={doap, bibo}`. Domain Vocabularies may be found, for instance, in the Linked Open Vocabularies (LOV) repository (LOV Documentation, 2016). `doap` is an RDF vocabulary to describe software projects, and, in particular, open source projects (DOAP, 2016). The Bibliographic Ontology Specification (BIBO) provides concepts and properties for describing documents on the Semantic Web (BIBO, 2016).

The goal of our approach is twofold: (i) to convert semi-structured data to RDF and (ii) meanwhile to establish the most appropriate entity type(s) for each instance Ids_{ik}. Since we do not know the meaning of $\{p_1, p_2, \ldots p_n\}$ for each Ids_{ik}, we firstly map them to *SDV* terms (properties). As a result, we find out a set of correspondences between $\{p_1, p_2, \ldots p_n\}$ and the vocabularies' properties.

Definition 4 (Property Correspondence). A property correspondence is defined as a triple $\langle p_a, v_b, n\rangle$, where p_a and v_b are matching properties (with $p_a \in ds_i$ and $v_b \in dv_j$) and *n* expresses the level of confidence underlying such correspondence.

In this work, we only consider *equivalence* relationship holding between p_a and v_b. Also, we define a threshold to indicate when a matching may be considered as an equivalence one. Thus, *n* must be higher than a given threshold to assess equivalence.

The output of a matching process between ds_i and dv_j is called an alignment A_{pv}. It contains a set of equivalence correspondences indicating which properties of the two datasets (ds_i and dv_j) correspond to each other.

Properties belong to Entity Types (ET) in domain vocabularies. With this in mind we need to identify the properties which belong to entity types that are candidates to be the most appropriate one to a given Ids_i.

Definition 5 (SDV Types). Given a SDV, we state that SDV Types $SDVT = \{ET_1, ET_2, \ldots ET_o\}$ is the set of entity types (or classes) which compose the vocabularies belonging to a particular SDV. Each element $SDVT_o$ has associated properties $\{v_1, v_2, ..., v_n\}$. Thus, $SDVT_o=\{v_1, v_2, ..., v_n\}$.

In our example, according to the chosen SDV, we may identify SDVT as the union of *doap* and *bibo* entity types. For the sake of space, we show an excerpt from *SDVT*={doap ET U bibo ET}, as follows.

```
SDVT = {doap:Project,
 doap:Repository, doap:Version,
 foaf:Person, foaf:Organization,
bibo:Collection, bibo:Document,
bibo:Event, foaf:Agent,
bibo:Thesisdegree}
```

For instance, some of these entity types and their associated properties are shown in the following:

```
doap:Project.properties = {homepage,
```

```
old-homepage, release,mailing-list,
category, repository, download-
page,download-mirror, wiki,bug-
database, screenshots,maintainer,
developer,documenter, translator,
tester,helper, programming-language,
 os, implements, service-
endpoint,language,vendor,platform,
audience,blog}

doap:Repository.properties = {anon-
root, browse, module, location}

doap:Version.properties = {revision,
 file-release, os, platform}

bibo:Document.properties = {citedBy,
distributor,listOfContributors,
owner,producer, status, edition,
identifier,locator,numberOfPages,
shortTitle,shortDescription}

bibo:Event.properties = {organizer}
```

Therefore, for each instance $Ids_{ik} \in ds_i$, we wish to associate a set of candidate ET which belongs to *SDVT*.

Definition 6 (Candidate Entity Types). Given an $Idsi_k \in ds_i$, a *SDVT* and an A_{pv}, we set the candidate entity types *CET*= {<ET_1, o_1>, <ET_2, o_2>, ..., <ET_f, o_h>} as the ones which have corresponding properties in A_{pv}. Each CET_f has a value o_h which indicates the number of property occurrences for a given ET_f.

With the current example in mind, we present A_{pv} between ds_1 and *SDV* as follows.

```
Apv(ds1 x sdv) =
{homepage ≡ doap.homepage
 repository ≡ doap.repository
 developer ≡ doap.developer
 tester ≡ doap.tester
 helper ≡ doap.helper
 programmingLanguage ≡
     doap.programming-language
 title ≡ bibo.shorttitle
 status ≡ bibo.status
}
```

There is no direct correspondence to the source property `group`. It would be included in an own vocabulary, which is created when no defined vocabulary may be used (this will be explained in Section 4). Although the correspondence `title ≡ bibo.shorttitle` has n <> 1, it is also included because n > threshold (which is defined as 0.8).

Therefore, according to A_{pv}, we would have the following CET.

```
CET = {<bibo.Document, 2>,
<doap.Project, 7>}
```

Our algorithm then ranks the top *m* ETs for each instance Ids_{ik} according to the number of existing properties o_h. It thus indicates the top one as a suggestion of the most semantically appropriate entity type for Ids_{ik}.

Definition 7 (Most Semantically Appropriate ET). Given *CET*, we state the most semantically appropriate entity type *MET* for Ids_{ik} as the one(s) which have the max number of property occurrences in CET.

Thus, each RDF target instance $Idsr_m \in dsr_j$ is labelled with an *ET* belonging to *SDVT*. If there is more than one ET with the same max number of property occurrences, all of them will be added as ETs of $Idsr_m$.

In our current example, for Ids_{11}, the identified and suggested *MET* is *doap:Project*. Thereby, metadata (properties) for the target RDF $Idsr_{11}$ along with the definition of the entity type (predicate rdf:type) will be derived as follows.

```
Idsr11 = {rdf:type doap:Project,
doap:homepage, doap:repository,
bibo:shorttitle, doap:developer,
doap:tester, doap:helper, own:group,
doap:programmingLanguage,
bibo:status}
```

Where
Project represents the ET of $Idsr_{11}$;
own represents an own ontology created to supply missing terms;
doap and *bibo* are prefixes for the doap and bibo vocabularies, respectively.

3.3 The MET Identification Algorithm

A high-level algorithm regarding the identification of candidate entity types and the definition of the most appropriate one is shown in *Algorithm_MET*.

Algorithm_MET shows how to identify the candidate entity types for a given JSON object. It also provides a suggestion of the most semantically appropriate one. To this end, at first, the set of properties which belongs to a JSON instance Ids_{ik} is read and saved as an OWL file (line 1). Then the vocabulary terms from SDV are selected (line 2) and mapped to an OWL file as well (line 3), thus enabling the use of the AlignmentAPI (David *et al.* 2011). The reason is due to the fact that this API is able to read OWL data. After mapping the input properties and vocabulary terms to OWL, the

algorithm generates a candidate alignment (line 4) and, for each identified correspondence (line 5), it verifies its confidence measure *n* (line 6). The correspondences with *n* above the defined threshold are saved in the domain alignment A_{pv} (line 7).

Then, for each property *p* belonging to the domain alignment A_{pv} (line 8), a SPARQL query is performed in order to acquire to which class that term is associated with (line 11). The idea is querying on each property and verifying which classes are then returned. Those returned classes become candidate entity types for the converting JSON object. Thus, classes with associated properties (from the JSON object) become candidate entity types (CETs).

```
---------------------------------------------------------------------------
Algorithm_MET: IdentifyMET()
---------------------------------------------------------------------------
Input:
 Idsik: Set of JSON instance properties
 SDV: set of domain vocabularies
Output:
 MET: entity type for Idsik
Begin
1: owlp = generateProperty(Idsik);
2: vocTerms = selectVocTerms(SDV);
3: owlv = generateVocTerms(vocTerms);
4: Apvc = AlignmentAPI(owlp, owlv);
5: For Each Corresp ϵ Apvc do
6:   if (Corresp.n>Threshold)then
7:      Add Corresp to Apv;
8:   end if;
9: End For Each;
10:For each property p ϵ Apv do
11:  CETf = Execute
        (SELECT ?entitytype
         WHERE {"+p+" rdfs:domain
         ?entitytype .});
12: End For each;
13: For each c ϵ CET do
14:   CETf = Add(CountPOcurrences(c,
            oh));
15: End For each;
16: RankedCET = RankCET(oh);
17: MET = Top(RankedCET);
18: Return (MET);
EndIdentifyMET;
---------------------------------------------------------------------------
```

The algorithm counts the number of property occurrences o_h for each CET_f according to a given property present in A_{pv} (line 14). Such measure is used to provide a rank of the top ETs in concordance with o_h (line 16).

After ranking ETs, the algorithm determines the entity type that has the highest number of property occurrences o_h (line 17). Finally, the most semantically appropriate entity type (MET) for a JSON instance (object) is returned (line 18).

A_{pv} is also used to provide how the target RDF dataset is generated, i.e., how the properties are semantically referred by the corresponding domain vocabulary terms.

For example, suppose that a JSON instance Ids_{i2} has the properties "name", "first_name", "age", and "developer". As SDV, consider SDV={*foaf*, *doap*}. At the end of the alignment (A_{pv}) generation, there are correspondences between Ids_{i2} properties and the terms "name", "firstName" and "age" of the FOAF vocabulary; the term "developer" is matched with *doap:developer*. From that, queries are executed to identify to which classes these four terms are associated with. In this example, the *doap:Project* and *foaf:Person* classes are returned as candidate ETs. Given this scenario, according to the number of obtained occurrences for each class w.r.t. the properties, the most appropriate entity type (MET) for the properties "name", "firstName", "age" and "developer" is *foaf:Person*.

4 IMPLEMENTATION

We have implemented the approach within a tool developed in Java which is an extension of the SenseRDF one. In the current version, the tool is able to convert PDF (only its metadata), XML and JSON data to the RDF model. In this work we show the conversion process along with the identification of the entity type for a given JSON object. Figure 3 shows a screenshot of the tool's main window that is split into four parts: (i) data file identification; (ii) area with the file's metadata or its content; (iii) the dataset type at hand (PDF, XML or JSON), and (iv) the generated RDF dataset.

In Figure 3, we present an example regarding a JSON dataset conversion. This dataset belongs to the "*Software Projects*" domain. Two domain vocabularies have been defined by the DE, namely: *doap* and *bibo*.

As mentioned in Section 3, the tool works with a domain alignment, where correspondences are set between the JSON object properties and the domain vocabulary terms. This alignment is produced by a linguistic matcher (David *et al.,* 2011). Each correspondence is defined with a confidence measure (between 0 and 1). Accomplishing tests, we have defined a threshold of 0.8 to identify correspondences to be set as equivalences.

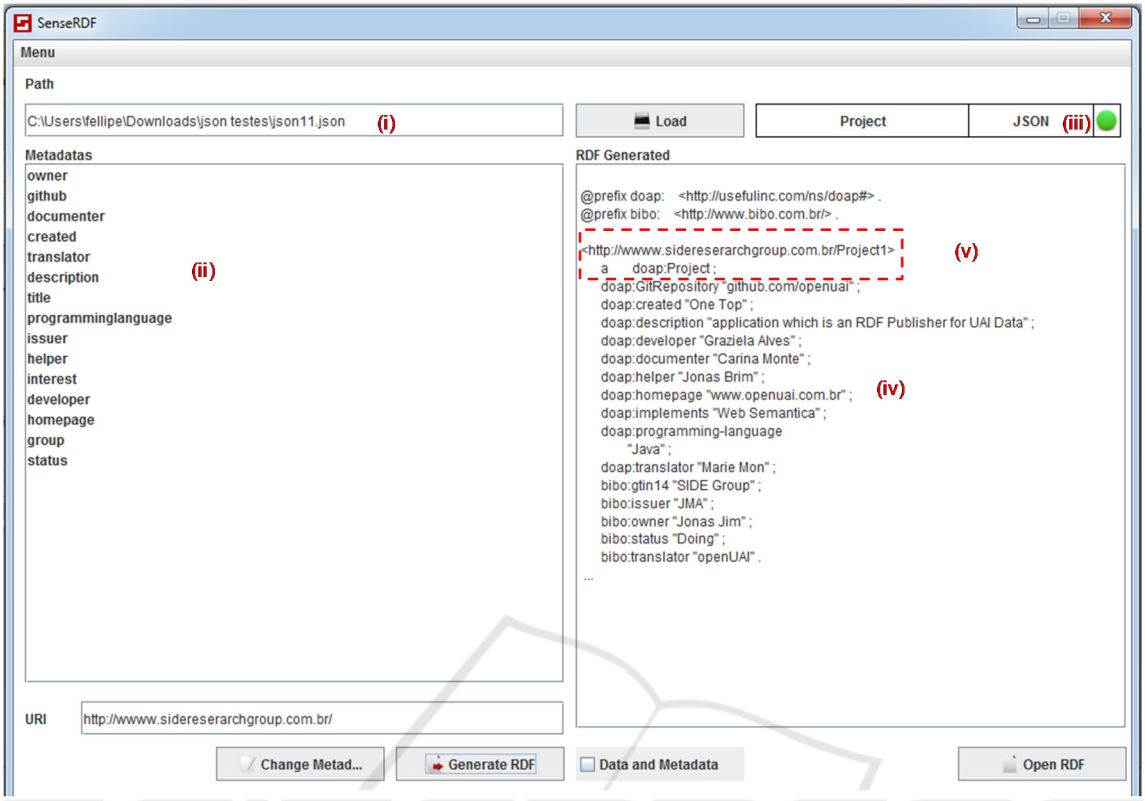

Figure 3: Main Interface.

The idea is to use correspondences set in the alignment to identify domain terms and refer the data at RDF generation time. Nevertheless, if it was not possible to identify a corresponding term in the chosen vocabularies, the tool incrementally "learns" the new terms, produces a specific ontology with them and defines the complementing correspondences.

The tool allows the generation of the RDF dataset in XML or turtle syntaxes. Considering the data shown in Figure 3 (part ii), it identifies the referring terms for the resources, thus producing the resulting RDF dataset shown in Figure 3 (part iv). Furthermore, it identifies the MET and includes such information in the form of an RDF triple in the target dataset (Figure 3 – part v).

5 EXPERIMENTS

We have conducted some experiments to verify the effectiveness of our approach. The goal was twofold: (i) to assess the ability to identify the most appropriate entity type for a given JSON object and (ii) to measure the data completeness w.r.t. the RDF data conversion process. The former aims to identify the degree of precision, recall and f-measure regarding the statement of the MET for a given JSON Object. The latter intends to measure the degree of RDF triples that represent JSON object properties which are present on the generated RDF dataset. In both goals, we intend to verify in which degree the choice of the used vocabularies imply in the data conversion process itself as well as in the MET definition. To this end, we have performed both experiments *considering* and *not considering* adequate domain vocabularies provided by the DE and comparing the obtained results. In this particular evaluation, we have used ten datasets regarding "Software Projects" and also "Books" domain. As recommended vocabularies, we have used the *DOAP*, *BIBO* and *CBO* Ontologies (CBO, 2016). As a not suitable vocabulary, in order to provide tests, we have used the *FOAF* one.

Regarding the first goal, we consider precision measure as the ratio of correctly found entity types (true positives) over the total number of returned entity types (true positives and false positives) (Rijsbergen, 1979). This is supposed to measure the correctness of MET statement provided by our approach. On the other hand, recall measures the

ratio of correctly found entity types (true positives) over the total number of expected entity types (true positives and true negatives) (Rijsbergen, 1979). To achieve the expected number of entity types, we have produced gold standards regarding the MET identification for the ten source datasets. These gold standards have been manually produced by domain users of our group. Also, we have used the traditional F-measure as the harmonic mean of precision and recall (Rijsbergen, 1979). The used formulas are shown in the following:

$$\text{Recall}(\text{CorrectET}, \text{ExpectedETs}) = \frac{\#\text{CorrectET}}{\#\text{ExpectedETs}} \quad (1)$$

$$\text{Precision}(\text{CorrectET}, \text{ReturnedETs}) = \frac{\#\text{CorrectET}}{\#\text{ReturnedETs}} \quad (2)$$

$$\text{F-Measure} = 2 * \frac{(\text{Re}\,call * \text{Pr}\,ecision)}{(\text{Recall} + \text{Precision})} \quad (3)$$

Where

#CorrectET is the number of correct suggested ETs produced by our approach;

#ExpectedETs is the total number of all possible ETs that could be produced by considering the defined gold standards; and

#ReturnedETs is the total number of all retrieved ETs produced by the approach (i.e., correct or incorrect ones).

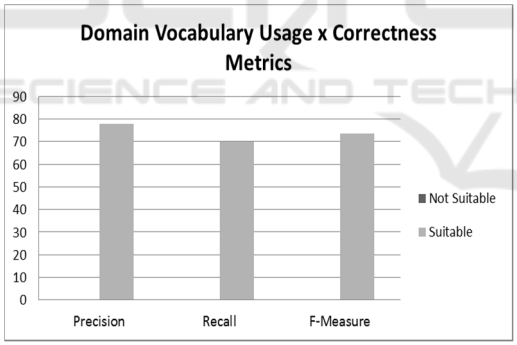

Figure 4: Measures w.r.t. the Correctly Suggested METs.

The experimental results regarding precision, recall and f-measure for ten datasets of both data domains are shown in Figure 4. In order to evaluate them, we have applied two experimental variables: (i) MET statement *without* suitable domain vocabularies and (ii) MET statement *with* suitable domain vocabularies. Regarding the first variable, the precision and recall measures were equal to zero. This means that it was not possible to identify the METs if an inappropriate domain vocabulary is used to provide the needed underlying semantics. On the other hand, by considering suitable domain vocabularies, we achieve a precision value of almost 78% (Formula 2), a recall of 70% (Formula 1) and f-measure of almost 74% (Formula 3) (Figure 4).

With respect to the second experiment goal, we have measured the data completeness when obtaining an RDF dataset from a source JSON document. Particularly, this experiment is concerned with evaluating each $Idsr_m \in dsr_j$ with respect to $Ids_{ik} \in ds_i$. In order to measure the data completeness, we use the following formula.

$$DC_{ds_i} = \frac{\#Propdsrm}{\#\text{Propdsi}} \quad (4)$$

Where #Propdsrm is the number of properties obtained in the generated RDF dataset dsr_m that represents original properties;
#Propdsi refers to the number of original properties in *DSi*.

A summary of the results regarding the data completeness along with the data conversion process for ten datasets is shown in Figure 5. In Figure 5, **NS** means a **N**ot **S**uitable domain vocabulary, i.e., it does not contain appropriate terms for that dataset domain; and **S** means a **S**uitable domain vocabulary, which has been carefully chosen for that dataset domain. Thereby, we are able to observe that the usage of a suitable domain vocabulary makes all the difference. If it is well chosen by the DE, the results are complete w.r.t. the original dataset.

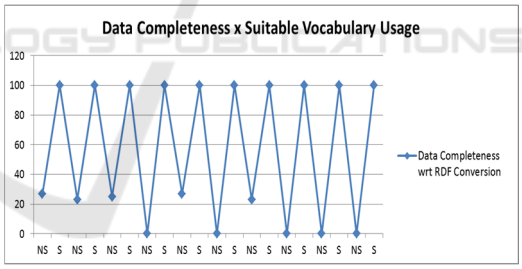

Figure 5: Data Completeness w.r.t. the data conversion process.

In conclusion, we verify that our approach for RDF dataset conversion along with the identification of the MET of a JSON object is promising. However it is dependent on the semantics provided by the chosen domain vocabularies.

6 RELATED WORK

Sleeman et al. (2015) address the problem of identifying coreferences in heterogeneous semantic graphs where the underlying ontologies may not be informative or even known. For such cases, they use

supervised machine learning to map entity attributes via dictionaries to predict instance entity types. ´Particularly, this method uses a way to map attributes from different domains to a common set regarding people, location and organization types. They show how to reduce work when using entity type information as a pre-filter for instance matching evaluation.

Bernardo et al. (2012) propose an approach to integrate data from hundreds of spreadsheets available on the Web. For that, they make a semantic mapping from data instances of spreadsheets to RDF/OWL datasets. They use a process that identifies the spreadsheet domain and associates the data instances to their respective class according to a domain vocabulary.

Taheriyan et al. (2014) use a supervised machine learning technique based on Conditional Random Fields with features extracted from the attribute names as part of a process to construct semantic models. The goal is to map the attributes to the concepts of a domain ontology and generate a set of candidate semantic types for each source attribute, each one with a confidence value. Next, an algorithm selects the top k semantic types for each attribute as an input to the next step of the process.

Tonon et al. (2013) propose a method to find the most relevant entity type given an entity (instance) and its context. This method is based on collecting statistics and on the graph structure interconnecting instances and types. This approach is useful for searching entity types in the light of search engines.

Comparing these works with ours, in our work we are interested in identifying the entity types to give more semantics to RDF generated datasets. Also, we use a semantic matcher to identify the vocabulary terms which are associated with the structural metadata from the converting dataset. Differently from the presented related works, the entity type is defined as the one which has the max number of property occurrences. This is recognized according to the semantics provided by domain vocabularies which have been chosen by a DE. Although there is such dependency, our work may be used in any data domain.

7 CONCLUSIONS

We presented a data domain-driven approach to converting semi-structured datasets, particularly in JSON formats, to RDF. By using the semantics underlying the domain of the data, it makes the conversion process less demanding. It attempts to automate as much of the conversion process by maintaining a domain alignment composed by correspondences between the converting metadata (properties) and the domain terms, and reusing it in each new conversion process. Also, in order to enrich the target generated RDF dataset, the object's entity types are identified and included in the code.

Accomplished experiments show that our approach is promising. By using the domain vocabularies, it is able to produce complete RDF datasets w.r.t. the original source data. Furthermore, it identifies in almost 77% the most appropriate entity type for a given object.

As future work, we intended to extend the approach and tool to deal with CSV files. Furthermore, we intend to use the MET recognition process to assist a coreference resolution task when integrating some datasets at conversion time.

REFERENCES

Alexe, B., Burdick, D. Hernandez, M., Koutrika, G., Krishnamurthy, R., Popa, L., Stanoi, I., and Wisnesky, R., 2013. High-Level Rules for Integration and Analysis of Data: New Challenges. *In Search of Elegance in the Theory and Practice of Computation: Essays Dedicated to Peter Buneman*. Springer Berlin Heidelberg. Pp 36-55.

Bernardo, I. R., Mota, M. S., Santanchè, A., 2012. Extraindo e Integrando Semanticamente Dados de Múltiplas Planilhas Eletrônicas a Partir do Reconhecimento de Sua Natureza. *In Proceedings of Brazilian Symposium on Databases (SBBD 2012)*: 256-263

BIBO, 2016. Available at http://lov.okfn.org/dataset/lov/vocabs/bibo. Last access on December, 2016.

CBO, 2016. Available at http://comicmeta.org/cbo/. Last access on December, 2016.

David, J., Euzenat, J., Scharffe, F., and Trojahn dos Santos, C., 2011. The alignment api 4.0. *In Semantic web journal 2 (1)*: 3–10, 2011.

DBPEDIA, 2016. Available on http://wiki.dbpedia.org/. Last access on December, 2016.

DC, 2016. Available at http://dublincore.org/documents/2008/01/14/dcmi-type-vocabulary/. Last access on December, 2016.

DOAP, 2016. Available on http://lov.okfn.org/dataset/lov/vocabs/doap. Last access on December, 2016.

Fanizzi, N., dAmato, C., and Esposito, F. 2012. Mining linked open data through semi-supervised learning methods based on self-training. *In Proceedings of the IEEE Sixth International Conference on Semantic Computing (ICSC), 2012. IEEE*, Palermo, Italy, pp. 277–284, 2012.

Heath, T. and Bizer, C. 2011. Linked data: Evolving the web into a global data space. *In Synthesis lectures on the semantic web: theory and technology*. (1): 1–136, 2011.

JSON Documentation. Available at http://json.org/. Last access on December, 2016.

Klettke, M., Störl, U., Scherzinger ,S., 2015. In: OTH Regensburg - BTW, 2015

LOV Documentation, 2016. Available on https://lov.okfn.org/dataset/lov/. Last access on December, 2016.

RDF Documentation, 2016. Available on www.w3.org/RDF/. Last access on December, 2016.

Rijsbergen, C. J. 1979. Information Retrieval, 2nd Ed. Stoneham, MA: Butterworths, 1979.

Silva, A.; Chaves, L. C.; Souza, D. 2013. A Domain-based Approach to Publish Data on the Web. *In Proceedings of the 15th International Conference on Information Integration and Web-based Applications & Services (iiWAS2013)* 2-6 December, 2013, Vienna, Austria.

Sleeman, J., Finin, T., & Joshi, A., 2015. Entity type recognition for heterogeneous semantic graphs. *In: AI Magazine*, 36(1), 75-87.

SWPO, 2016. Available at http://lov.okfn.org/dataset/lov/vocabs/swpo. Last access on December, 2016.

Taheriyan, M., Knoblock, C. A., Szekely, P., & Ambite, J. L., 2014. A scalable approach to learn semantic models of structured sources. *In Semantic Computing (ICSC), 2014 IEEE International Conference on (pp. 183-190). IEEE.*

Tonon, A., Catasta, M., Demartini, G., Cudré-Mauroux, P., & Aberer, K., 2013. Trank: Ranking entity types using the web of data. *In International Semantic Web Conference (pp. 640-656). Springer Berlin Heidelberg.*