Ontology Alignment of Two Different Datasets

Table of Contents

Presentation of the project

Goal

The goal is to align two different bus trips datasets to one common ontology, and to generate an RDF version of each, in order to ease the manipulation of their geographical data.

Datasets

Both datasets originate from public transport companies:

The first dataset is the *shape.txt* file from a GTFS-static *.zip* file provided by the Zenbus API[1]. It is in a *.csv* format and represents bus trips in Paris suburbs.

The second dataset is *lyon_bus2.json* from Grand Lyon API[2]. It is in *json* format and represents bus trips in Lyon aggglomeration.

Ontologies

The *eona* ontology to which the datasets are aligned is a simple one and created specifically for this project. It consists of a unique class, *BusTrips*, characterized by three properties: *lat*, *lon*, and *pointSequence*.

Other well-established ontologies are used in this project, such as *wgs84_pos*[3], which is the RDF version of the WGS84 geographical *de facto* standard, used world widely. The full list is:

- ql;
- rdf;
- rml;
- rr;
- wgs84_pos;
- eona.

Problem & Solution

Problem

Both dataset have latitude and longitude data, but they are not reachable in a same manner. Each data source needs their own specific way to extract these data.

1 https://zenbus.net/gtfs/rt/poll.proto?dataset=gpso-rt
2 https://data.grandlyon.com/geoserver/sytral
3 http://www.w3.org/2003/01/geo/wgs84_pos

Solution

By aligning both of them to the Eona-X ontology, a single command is needed to gather all geographical data from both datasets.

Practice

Tools Used

API connection is made with *requests* Python library.

Python data manipulation librairies were used such as *pandas* and *json*.

Most importantly, RML language[4] was used to define the rules to create RDF triples from the *datasets* files. It was executed by the CLI Java program *RMLMappe*[5]*r* in order to generate the RDF triples.

Steps

An *.rml.ttl* was created for each data set:

- *2shapes_csv_to_rdf.rml.ttl* for Zenbus data;
- *lyon_rml.rml.ttl* for Grand Lyon data.

It was then passed to *RMLMapper*, configurated according to the *config1.properties* file, with the following command:

```
java -jar rmlmapper-6.5.1-r371-all.jar -c
~/pro/eonax/oguc_bus_uc/rml_tests/rml_mapping/config2_lyon.properties
```

Results

The resulted RDF triples for each data set are respectively *results_rml.ttl* and *lyon_results_rml.ttl*. We can observe that:

1. they have the same attributes, so they are **reachable in the same fashion**, it is easier to aggregate data. The advantage it provides can be imagined in the case of dozens or hundreds of different data sources;
2. their latitude and longitude values are semantically linked to the WGS84 ontology standard. **This context- knowledge is now understandable to machines**, not only to humans.

4 https://rml.io/specs/rml/
5 https://github.com/RMLio/rmlmapper-java

Conclusion

Two advantages :

- easier manipulation and aggregation of multiple data sources with different formats/schemas/…
- Data are semantically enriched, giving context-awareness to machines and better interoperability.

What’s next?

Show the RDF geographical data on an interactive map.

Semantic computing, deeper data analysis for more actionable knowledge.

Try it on stream data with *RMLStreamer*.

Make alignment with data of different units.

Appendices

2shapes_csv_to_rdf.rml.ttl

```
@prefix : <http://example.org/rules/> .
@prefix csvw: <http://www.w3.org/ns/csvw#> .
@prefix eona: <http://www.eona-x.eu/> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix wgs84_pos: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix zenbus: <http://zenbus.net/> .


:TriplesMap a rr:TriplesMap;
  rml:logicalSource [
    rml:source [
      a csvw:Table;
      csvw:url "/home/maximeb/pro/eonax/oguc_bus_uc/rml_tests/rml_mapping/shapes.csv";
      csvw:dialect [
        a csvw:Dialect;
        csvw:delimiter ","
      ]
    ];
    rml:referenceFormulation ql:CSV
  ].

:TriplesMap rr:subjectMap [
    rr:template "http://zenbus.net/{shape_id}"
    ].

:TriplesMap rr:predicateObjectMap [
  rr:predicate rdf:type;
  rr:objectMap [ rr:constant eona:BusTrip ];
].

:TriplesMap rr:predicateObjectMap [
  rr:predicate wgs84_pos:lat;
  rr:objectMap [ rml:reference "shape_pt_lat" ];
].

:TriplesMap rr:predicateObjectMap [
  rr:predicate wgs84_pos:lon;
  rr:objectMap [ rml:reference "shape_pt_lon" ];
].

:TriplesMap rr:predicateObjectMap [
  rr:predicate eona:pointSequence;
  rr:objectMap [ rml:reference "shape_pt_sequence" ];
].
```

lyon_rml.rml.ttl

```
@prefix : <http://example.org/rules/> .
@prefix eona: <http://www.eona-x.eu/> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
```

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix wgs84_pos: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix lyon: <https://data.grandlyon.com/> .

:TriplesMap a rr:TriplesMap;
  rml:logicalSource [
    rml:source "/home/maximeb/pro/eonax/oguc_bus_uc/lyon_bus2.json";
    rml:referenceFormulation ql:JSONPath;
    rml:iterator "$.features[*]"
  ].

:TriplesMap rr:subjectMap [
  rr:template "https://data.grandlyon.com/{id}"
].

:TriplesMap rr:predicateObjectMap [
  rr:predicate rdf:type;
  rr:objectMap [ rr:constant eona:BusTrip ];
].

:TriplesMap rr:predicateObjectMap [
  rr:predicate wgs84_pos:lat;
  rr:objectMap [ rml:reference "geometry.coordinates[0][*][1]" ];
].

:TriplesMap rr:predicateObjectMap [
  rr:predicate wgs84_pos:lon;
  rr:objectMap [ rml:reference "geometry.coordinates[0][*][0]" ];
].
```

config2_lyon.properties

```
verbose = true
mappingfile =
/home/maximeb/pro/eonax/oguc_bus_uc/rml_tests/rml_mapping/lyon_rml.rml.ttl
outputfile =
/home/maximeb/pro/eonax/oguc_bus_uc/rml_tests/rml_mapping/lyon_results_rml.ttl
serialization = turtle
```