Aiteam
NATIONAL & KAPODISTRIAN
UNIVERSITY OF ATHENS

From Earth Observation Data to the Triple Store

Kostas Plas
Sergios-Anestis Kefalidis
Manolis Koubarakis

Part 1: From Unstructured EO Data to RDF with Python

- In this part of the tutorial, we want to transform EO Data into RDF data using our team's python library ToposKG

- The EO Data we will be working with are retrieved from OSM (The city of Hamburg, POIs and rivers) and sentinel repositories available from DA4DTE project

- The EO Data is available in GeoJSON format. Our goal in this part of the tutorial is to generate triples (s,p,o) that cover all information (both thematic and spatial) in these GeoJSON files.

What is GeoJSON?

- GeoJSON is an extension of JSON made to better represent spatial information

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [
          9.979228498387096,
          53.55638368037634
        ]
      },
      "properties": {
        "type": "poi",
        "hasName": "Planten un Blomen",
        "wikilink": "Q494440"
      }
    }
```

- The key “features” is an array of spatial entities
- Each spatial entity has a key named “geometry” that represents the type of spatial object the entity is (Line,Point, Polygon etc) and its coordinates
- Each spatial entity also has a key named “properties” which is a simple JSON object that holds thematic information about the entity e.g A location’s name, A city’s population etc

Goal of Part 1

- Create an RDF graph from .geojson files!

- An geospatial RDF graph looks something like this:

```
ex:entinty_1 rdf:type ex:poi

ex:entity_1 geo:hasGeometry ex:geometry_entity_1

ex:geometry_entity_1 geo:asWKT “POINT (9.9792284 53.55638368)”
```

In RDF graphs spatial information is often stored in WKT format.

Well-Known Text (WKT)

A WKT (Well-Known Text) is a text-based format used to represent geometric shapes such as points, lines, and polygons.

- Point: `POINT (30 10)`
- LineString: `LINESTRING (30 10, 10 30, 40 40)`
- Polygon: `POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))`
- MultiPoint: `MULTIPOINT ((10 40), (40 30), (20 20), (30 10))`
- MultiLineString: `MULTILINESTRING ((10 10, 20 20), (15 15, 30 15))`
- MultiPolygon: `MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)))`

Part 2: Storing and Querying RDF Data with GraphDB

- After transforming our EO data to spatial RDF data, we need to store them in a triple store to query them
- A triple store or RDF store is the equivalent of RDBS for RDF data.
- In this tutorial the RDF store we will use is GraphDB

Querying RDF Data

- Like RDMS use SQL, triple stores use their own querying language similar to SQL called SPARQL:

```
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix geo: <http://www.opengis.net/ont/geosparql#>
prefix geof: <http://www.opengis.net/def/function/geosparql/>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix schema: <http://schema.org/>
prefix onto: <https://ai-team-uoa.github.io/LTEOI-BIDS-2025/ontology/>
prefix resource: <https://ai-team-uoa.github.io/LTEOI-BIDS-2025/resrouce/>

select distinct ?s2 where {
    ?s2 rdf:type onto:sentinel2 .
    ?s2 geo:hasGeometry ?g2 .
    ?g2 geo:asWKT ?wkt2 .

    ?poi rdf:type onto:poi .
    ?poi geo:hasGeometry ?g_poi .
    ?g_poi geo:asWKT ?wkt_poi .

    FILTER(geof:sfWithin(?wkt_poi, ?wkt2))
} limit 100
```

Thank you!

Find us in our web page: https://ai.di.uoa.gr/
Or follow us on LinkedIN @ai-team-uoa

Aiteam
NATIONAL & KAPODISTRIAN UNIVERSITY OF ATHENS