Linguistic Linked Open Data for Humanists

Anas Fahad Khan, Giulia Pedonese, Michele Mallia

Lisbon Summer School in Linguistics, July 1-5, 2024

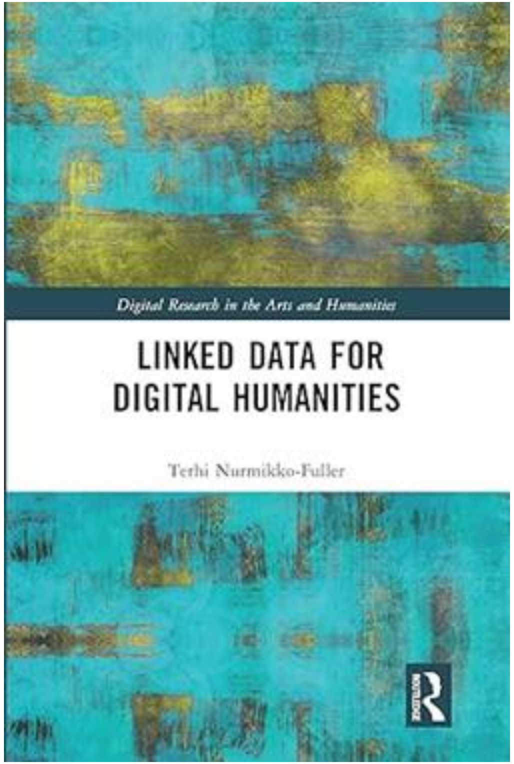
Funded by the European Union NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
Consiglio Nazionale delle Ricerche

But first....

An Introduction to Linked (Open) Data and the Semantic Web

What is Linked Data?

- A means of publishing **structured, interlinked data** by making use of **standard web technologies** such as **HTTP**, **XML** and **URIs**
- Linked data (LD) is intended to be queried **semantically** (we will explain what this means as we go on) and it offers the possibility of querying across **different datasets at the same time remotely**
- It has been around since 2006 and has become a very popular way of **sharing, accessing and querying data** in a number of different disciplines…and has recently begun to make inroads in the humanities too



What is Linked Data?

- Unlike standard HTML web pages, linked data datasets are meant to be **easier to process by machines**
- HTML markup *usually* annotates **the visual appearance of pages** whereas with linked data we're interested in marking up the **meaning/semantics** of our data
- With linked data we are encouraged to **add links from our data to other linked data datasets** (*linked* data, it's in the name :))
- Linked Data relies *heavily* on **shared standards**, including shared vocabularies, in order to ensure the maximum of **interoperability**

But what is the Semantic Web?

- Linked Data is part of the movement towards a **Semantic Web**, the next step on from the existing web, as proposed by **Tim Berners-Lee** the inventor of the **World Wide Web**:

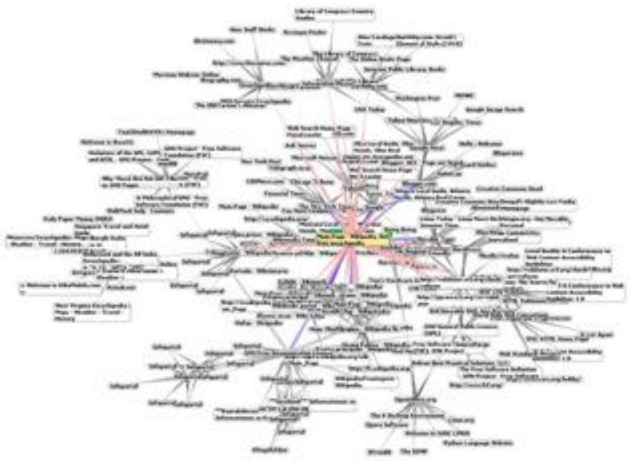
  I have a dream for the Web [in which computers] become capable of analysing all the data on the Web – the content, links, and transactions between people and computers. A "Semantic Web", which makes this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The "intelligent agents" people have touted for ages will finally materialise.

Differences between the Semantic Web and Linked Data?

- **The Semantic Web**: a vision of the World Wide Web in which **web documents** are structured in such a way that **computers can can process them according to their contents** (hence ‘semantic’)
- The Semantic Web is promoted by the **Worldwide Web Consortium (W3C)** and is being implemented through the use of **common, open standards for data and for exchange protocols**.
- **Linked Data** is one of the ways of making the vision of the Semantic Web a **reality**

First Definitions

- **Linked Data** is data published on the **World Wide Web** that obeys the following principles:
  - 1. Use **Uniform Resource Identifiers (URIs)** as names for things.
  - 2. Use **HTTP** URIs so that people can look up those names.
  - 3. When someone looks up a URI, provide **useful information.**
  - 4. Include **links to other URIs**. so that they can discover more things.
- **Linked Open Data**: Linked Data published with an open license



First Definitions

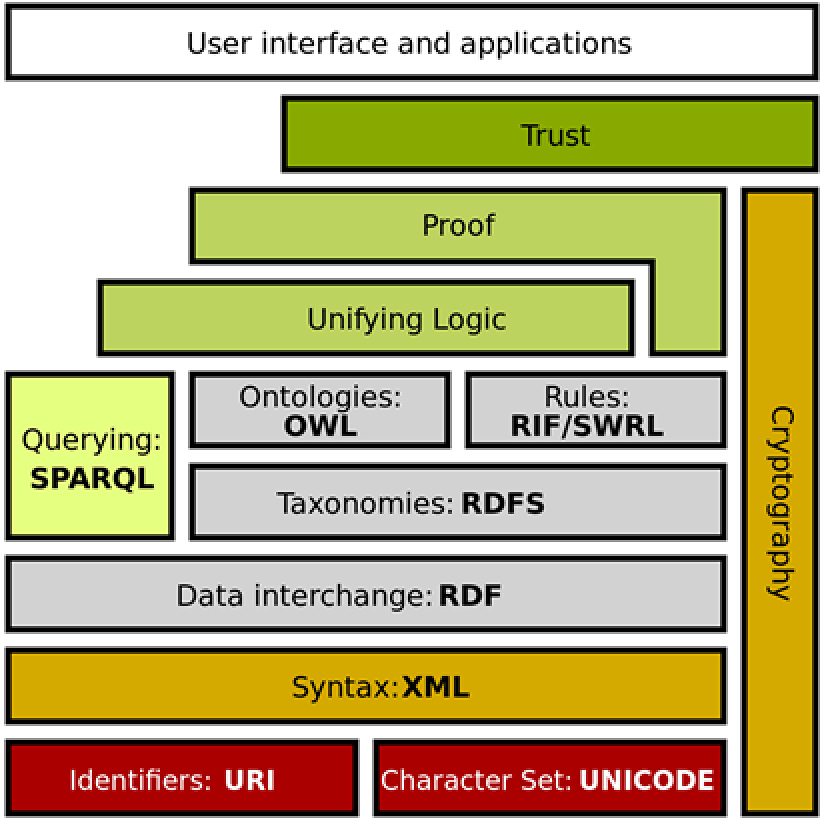
These four principles were simplified and distilled by Tim Berners Lee himself into the following three:

1. *All kinds of **conceptual things**, they have names now that start with **HTTP**.*
2. *If I take one of these HTTP names and I look it up...I will get back some data in a **standard format** which is kind of useful data that somebody might like to know about that **thing**, about that **event**.*
3. *When I get back that information it's not just got somebody's height and weight and when they were born, it's got **relationships**. And when it has relationships, whenever it expresses a relationship then the other thing that it's related to is given one of those names that starts with HTTP.*

Standards for the Semantic Web

- The Semantic Web is based on a stack of open standards, with those higher up on the stack dependent on those below them.
- At the very bottom we have the use of **Uniform Resource Identifiers (URIs)** and the **UNICODE** character set
- Then the use of **XML** as a common serialisation format
- Next the use of **RDF** as a common data interchange framework (specifies organisation of data at an abstract level). Then we come to **RDFS**, another formal language for creating taxonomies.
- Then we come to **OWL** and **SPARQL**

The Semantic Web Stack



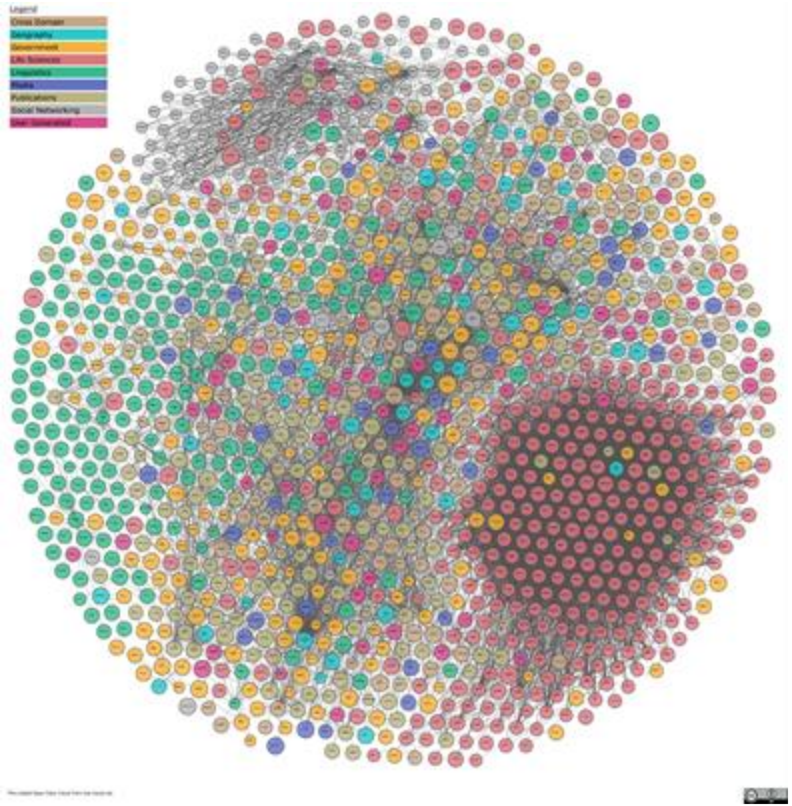
Why Linked Data?

- It helps to resolve the problem of **data silos**: the problem of information being stored in **isolated databases** making it difficult to access and **integrate** or to query across different datasets.
- Linked data enables different datasets to be **interconnected**, breaking down silos and enabling **integrated** access to data from **multiple sources**.

Why Linked Data?

- **Integrating data from disparate sources** can be complex due to the diversity of formats and standards (some of these formats very discipline specific).
    - Linked data helps to make data more **interoperable** by using **common standards** (like RDF) and facilitating easier integration and interoperability across different systems and organisations by **making shared vocabularies and ontologies available** for use across disciplines.
- Data is often reused inefficiently, leading to duplication of effort.
    - Linked data encourages **data reuse and enrichment** and makes it easier for users to **build on existing data rather than starting from scratch.**

The Linked Open Data Cloud



The Linked Open Data Cloud

- A **visualisation** of the accessible linked open data datasets as a massive **RDF knowledge graph** (a ‘cloud’); it can be found at https://lod-cloud.net/
- Periodically **regenerated** (you can track the growth of the cloud by looking at previous versions of the diagram)
- The cloud is partitioned into a number of **different subject areas** (corresponing to node colours)

The Linked Open Data Cloud

- The cloud is partitioned into a number of **different subject areas** (ctd on next slide):
    - **Cross-Domain**: Datasets that span multiple subject areas and can be used in various contexts.
    - **Government**: Datasets related to public administration, policies, and government operations.
    - **Publications**: Information from academic papers, books, journals, and other forms of literature. This includes bibliographic data, citation networks, and publishing metadata.
    - **Life Sciences:** Data pertaining to biology, medicine, and healthcare.
    - **Geography**: Geographic information and spatial data, such as maps, geographic features, locations, and geospatial datasets.

The Linked Open Data Cloud

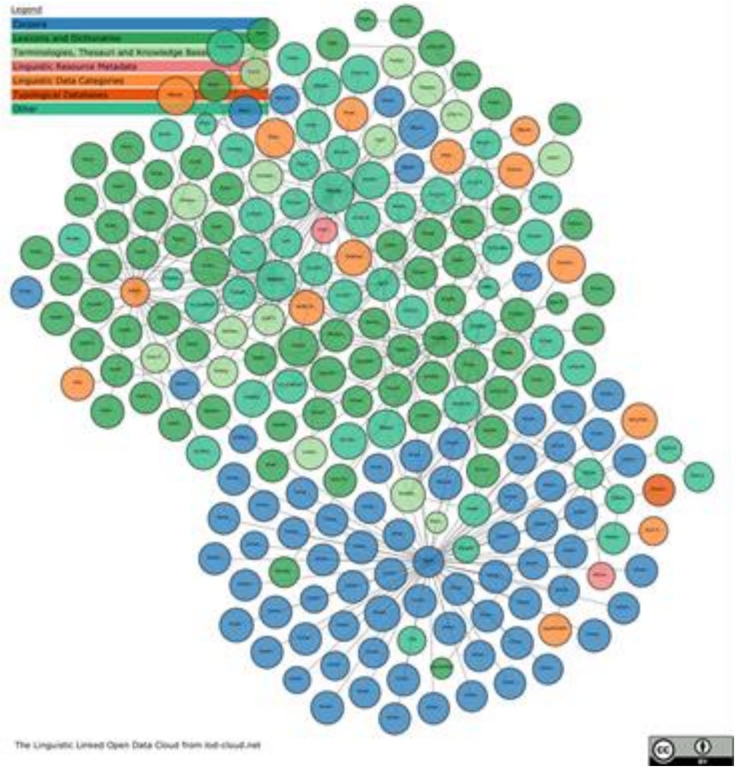
- The cloud is partitioned into a number of **different subject areas** (ctd from previous slide):
  - **Social Media**: Data from social networking sites, blogs, and other social media platforms. This includes user profiles, interactions, and social network graphs.
  - **User-Generated Content:** Information created and shared by users on various platforms. This includes reviews, comments, wikis, and other forms of collaborative content.
  - **Media**: Information related to audio, video, images, and other multimedia content. This includes metadata about media files, usage data, and content descriptions.
  - **Scholarly Data**: Academic and research data, including research projects, experimental results, and academic collaborations.
  - **Linguistics:** Data related to language, such as lexicons, thesauri, language resources, and linguistic annotations.

The Linked Open Data Cloud

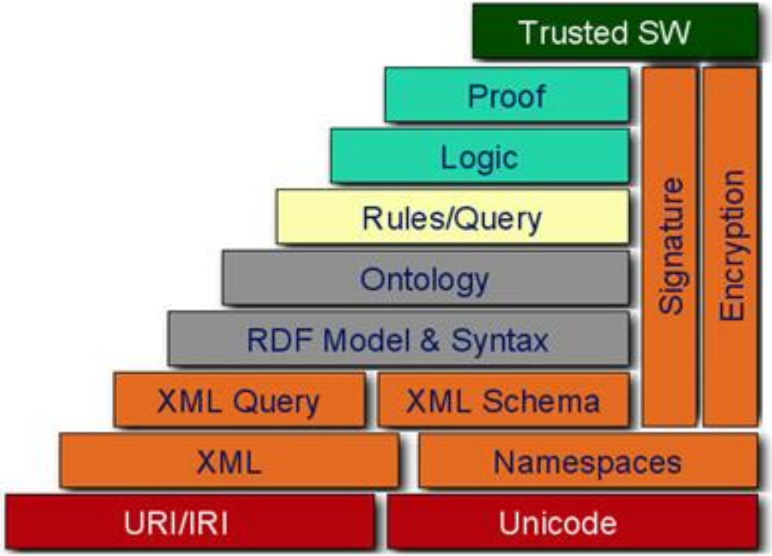
- The cloud is partitioned into a number of **different subject areas** (ctd from previous slide):
    - **Social Media**: Data from social networking sites, blogs, and other social media platforms. This includes user profiles, interactions, and social network graphs.
    - **User-Generated Content:** Information created and shared by users on various platforms. This includes reviews, comments, wikis, and other forms of collaborative content.
    - **Media**: Information related to audio, video, images, and other multimedia content. This includes metadata about media files, usage data, and content descriptions.
    - **Scholarly Data**: Academic and research data, including research projects, experimental results, and academic collaborations.
    - **Linguistics:** Data related to language, such as lexicons, thesauri, language resources, and linguistic annotations.

The Linguistic Linked Open Data (LLOD) Cloud

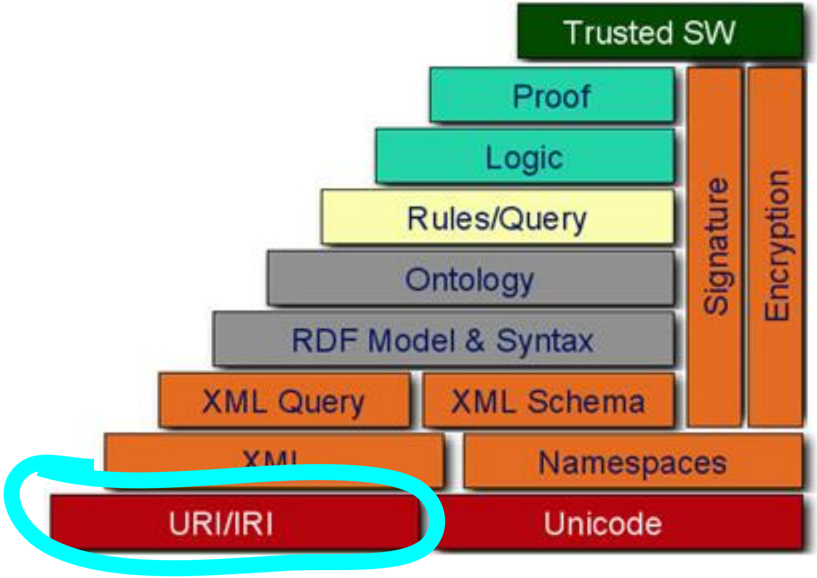
- That part of the LOD cloud dedicated to **language resources**
- The datasets in the LLOD cloud are classified into the following categories:
  - **Corpora (and Linguistic Annotations)**
  - **Lexica and Dictionaries**
  - **Terminologies, Thesauri and Knowledge Bases**
  - **Linguistic Resource Metadata**
  - **Linguistic Data Categories**
  - **Typological Databases**
  - **Other**
- We will describe these categories and the make-up of the cloud in detail **in further lessons**
- Your data too could become part of the LLOD cloud!



Exploring the Semantic Web Stack



Exploring the Semantic Web Stack



Uniform Resource Identifiers

- To start off we need a way of identifying people, things, locations, etc, that can be re-used across datasets
- **Uniform Resource Identifiers (URIs)** answer the need for uniquely identifying resources across the whole Semantic Web (global IDs)
    - NB. Resources here aren't just documents on the web but **anything that we want to describe in a Semantic Web datasets**, i.e., Cristiano Ronaldo, the Portuguese language, Portugal, the number 23,
    - They can also be relationships: **located in**, **bigger than**, **has name**
- URIs are strings that are structured according to a standard schema
    - They are the basis on which the Semantic Web is constructed and give us the most basic building blocks of our linked data datasets
- **Uniform Resource Locators (URLs)** are a kind of URI of identifying internet domain resources:
    - https://clunl.fcsh.unl.pt/en/

Uniform Resource Identifiers

- All URI's follow a set of syntactic rules represented by the following schema:
  - `URI = scheme:[//authority]path[?query][#fragment]`
    - where `scheme` can be for instance **http** or **urn**
    - `authority` can include hostnames, domain names and top level domains such as in www.cnr.it
- URIs *should* be **dereferenceable** whenever we enter them in a browser or make a GET request they should give us back some relevant data :
    - e.g., whenever we enter/click on such a URI in a browser you should get back a relevant HTML document or automatically download an RDF file
- Ideally URIs should **be stable** (i,e,, not re-used for different things) and **persistent** (be around for the long term)
- We should re-use pre-existing URIs to refer to things and not just randomly invent new ones

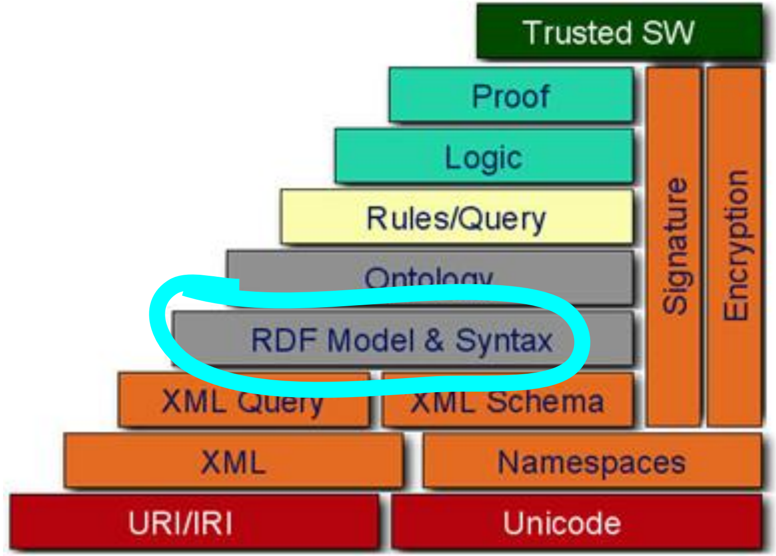
Example URIs

- The City of Lisbon:
  - https://dbpedia.org/resource/Lisbon
  - https://www.wikidata.org/wiki/Q597
  - htp://viaf.org/viaf/124321959
  - http://yago-knowledge.org/resource/Lisbon
- The property/relationship of country (‘The country where the thing is located’)
  - https://dbpedia.org/ontology/country
  - http://www.wikidata.org/entity/P17

URIs vs IRIs vs URNs

- **IRIs** (Internationalized Resource Identifiers):
  - An IRI is an **extended form of a URI** that allows a wider range of characters than are allowed with URIs.
  - Facilitates the **use of non-ASCII characters**, enabling resource identification in various languages and scripts.
  - Examples:
    - http://my.dbpedia.org/resource/လစ်စဘွန်းမြို့
    - http://pa.dbpedia.org/resource/ਲਿਸਬਨ
- **URNs** (Uniform Resource Names):
  - **A subset of URIs** that provides a unique and persistent identifier for a resource, **independent of its location**.
  - Focuses on the **resource's identity** rather than its location.
  - urn:cts:greekLit:tlg0012.tlg001.perseus-grc1

Exploring the Semantic Web Stack



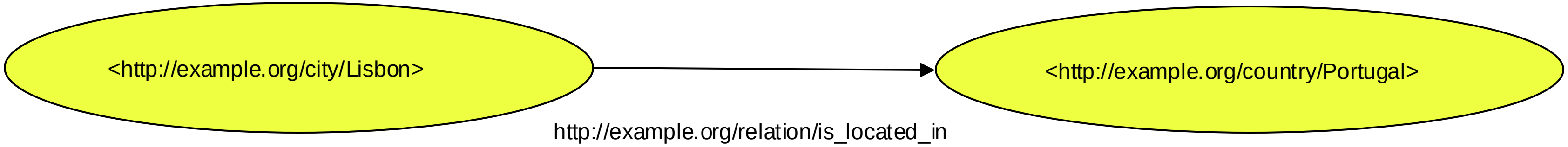
Resource Description Framework

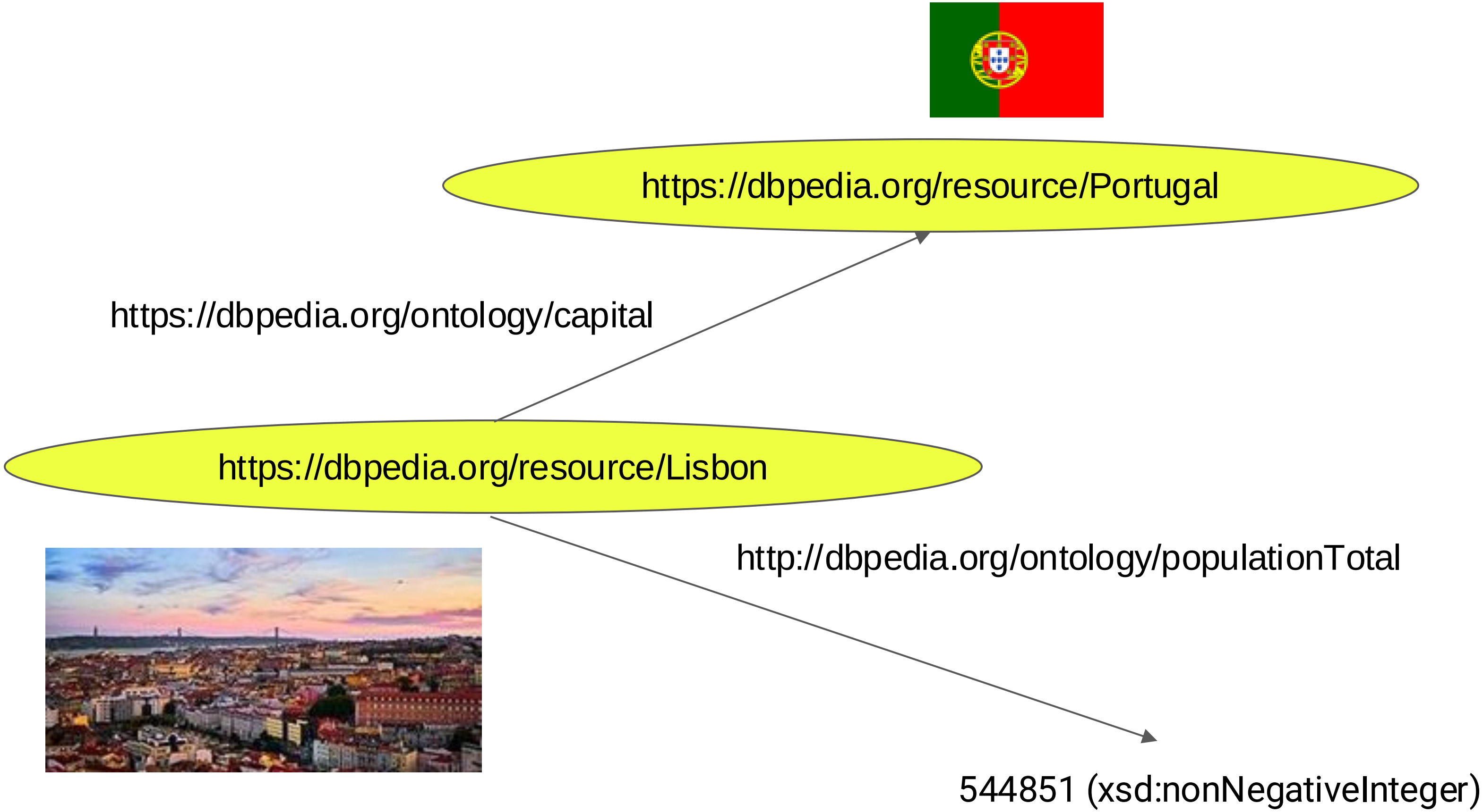
- We have URIs as universal IDs for identifying resources in the Semantic Web (which can be things in the world, abstract things including words, events)
- How do use these to describe the world and in a way that ensures interoperability at all levels of description?
- The answer is the **Resource Description Framework (RDF)**!
- RDF is a **standard metamodel** for **data interchange** that utilises the **linking structure** of the World Wide Web via URIs
- All linked data resources adhere to this standard metamodel

Resource Description Framework

- RDF stipulates that data be structured in the form of
  - **(subject, predicate, object)** triples,

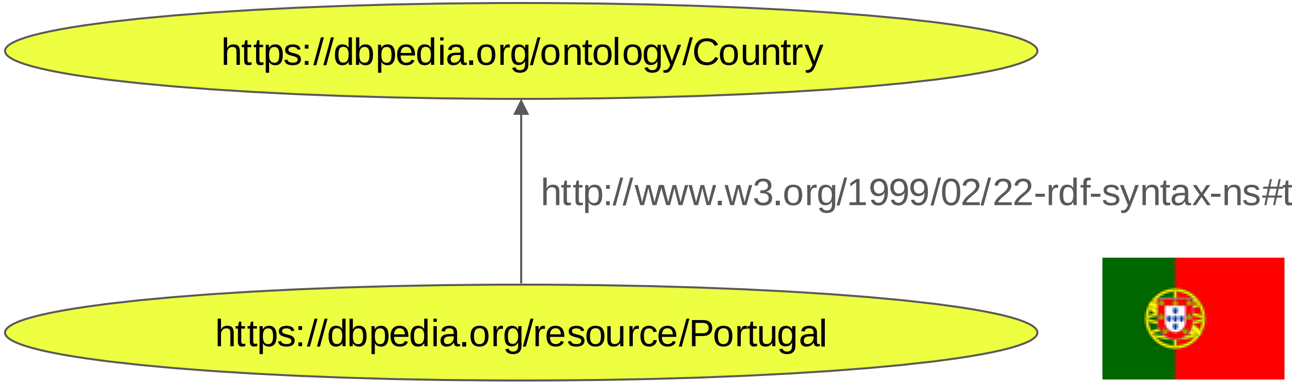
  where the subject and predicate are **URI resources** identified and the object can be either a **URI resource or a data value (literal).**
- RDF literals can be expressed using the ^^ syntax followed by the XSD datatype.
- A set of RDF triples in an RDF dataset describes a **directed, labelled graph** where the **predicate** relates together the **subject** and **object**
- Resources relating together other resources as predicates are called **properties** (convention to represent them as labelled arrows)




https://dbpedia.org/resource/Portugal
https://dbpedia.org/ontology/capital
https://dbpedia.org/resource/Lisbon
http://dbpedia.org/ontology/populationTotal
544851 (xsd:nonNegativeInteger)

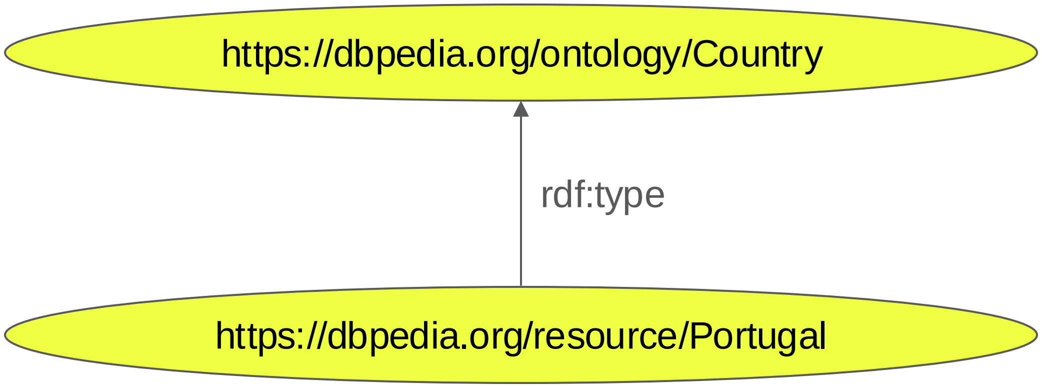
RDF

- RDF is also a vocabulary which gives us a number of built in classes and properties built-in **type** property, introduced in the RDF namespace



RDF

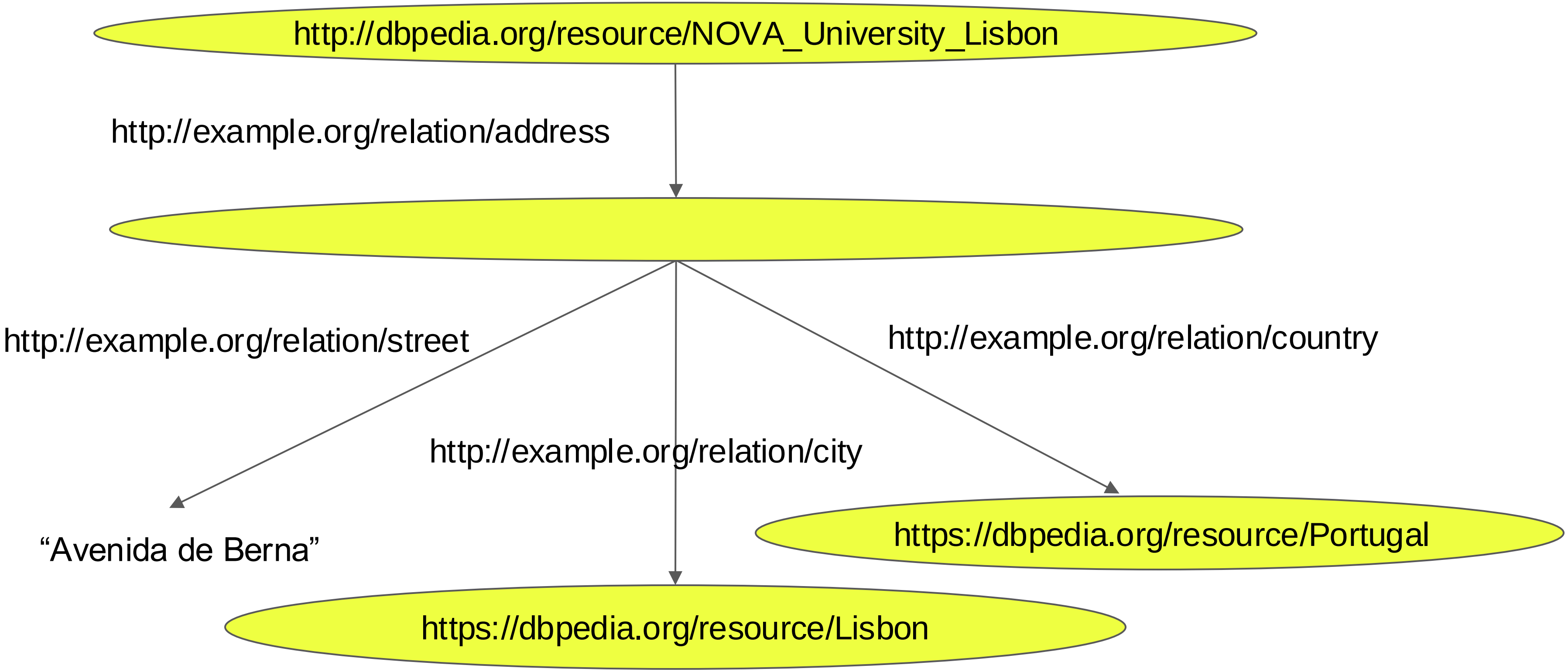
- RDF is also a vocabulary which gives us a number of built in classes and properties built-in **type** property, introduced in the RDF namespace



Blank Nodes

- We *don't* always need to give resources **a global identifier (URI)**.
- **Blank nodes** are **anonymous resources** in RDF and are only identifiable within a specific RDF graph.
- They can be useful for **representing complex data** without cluttering the RDF graph with numerous URIs.
- Blank nodes often act as intermediate nodes connecting various parts of an RDF structure.

Blank Nodes



http://dbpedia.org/resource/NOVA_University_Lisbon
http://example.org/relation/address
http://example.org/relation/street
http://example.org/relation/country
http://example.org/relation/city
“Avenida de Berna”
https://dbpedia.org/resource/Portugal
https://dbpedia.org/resource/Lisbon

Blank Nodes

- Pros:
  - They **simplify RDF** models by avoiding unnecessary URIs.
  - Can improve **readability and manageability** of RDF data (Can be useful in presenting examples, especially in a classroom setting :))
- Cons:
  - Blank nodes **cannot be referenced outside their graph**.
  - Identification Issues: Difficult to **merge RDF graphs** containing blank nodes.
- Best Practices:
  - Use blank nodes when the **identity of the resource is not important**.
  - Avoid **excessive use** to maintain graph clarity and interoperability.

XSD Datatypes

- **XSD (XML Schema Definition) datatypes** provide a way to define **the data types of RDF literals.**
  - Here xsd stands for http://www.w3.org/2001/XMLSchema
- `xsd:string`: Represents a **sequence of character**s.
  - Example: "Hello World"^^xsd:string
- `xsd:integer`: Represents an **integer** (can also have nonnegativeinteger).
  - Example: "42"^^xsd:integer
- `xsd:decimal`: Represents a **decimal number**.
  - Example: "3.14"^^xsd:decimal
- `xsd:boolean`: Represents a **boolean value**.
  - Example: "true"^^xsd:boolean
- `xsd:float`: Represents a **floating point number**.
  - Example: "3.14"^^xsd:float
- `xsd:double`: Represents a **double precision floating point number**.
  - Example: "2.71828"^^xsd:double

Date and Time XSD Datatypes

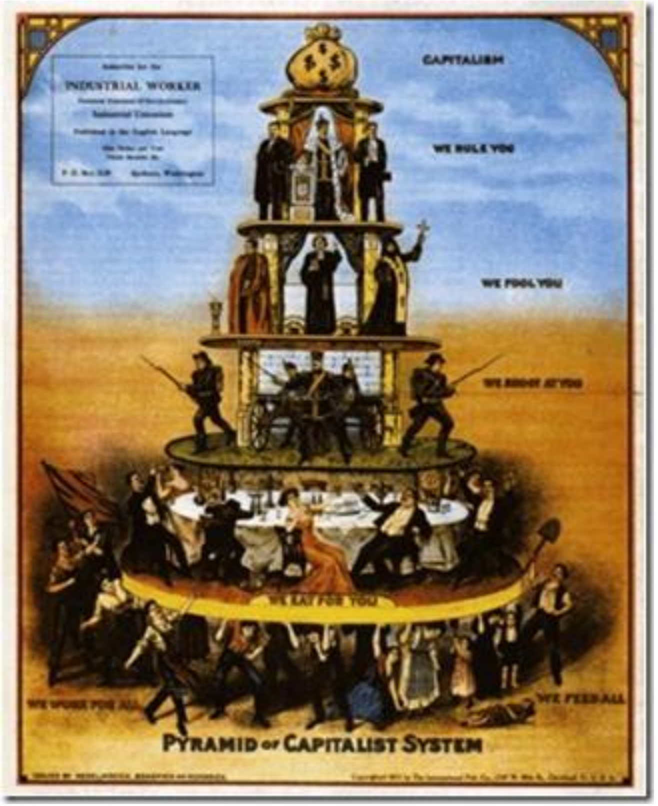
- `xsd:date`: Represents a **date** (YYYY-MM-DD).
  - Example: "2024-06-21"^^xsd:date
- `xsd:time`: Represents a **time** (HH:MM).
  - Example: "14:30:00"^^xsd:time
- `xsd:dateTime`: Represents a **date and time** (YYYY-MM-DDTHH:MM).
  - Example: "2024-06-21T14:30:00"^^xsd:dateTime

Language Tags in RDF Literals

- Language tags are **used to specify the language of a string literal in RDF**.
- Syntax: Literal followed by @ and the language code (e.g., from ISO 639).
- Example:
  - "Olá"@pt (Portuguese)
  - "Hello"@en
  - "Hola"@es
  - "Hallo"@de
  - "你好"@zh
- NB: RDF literals **cannot** have both an XSD datatype and a language tag simultaneously.

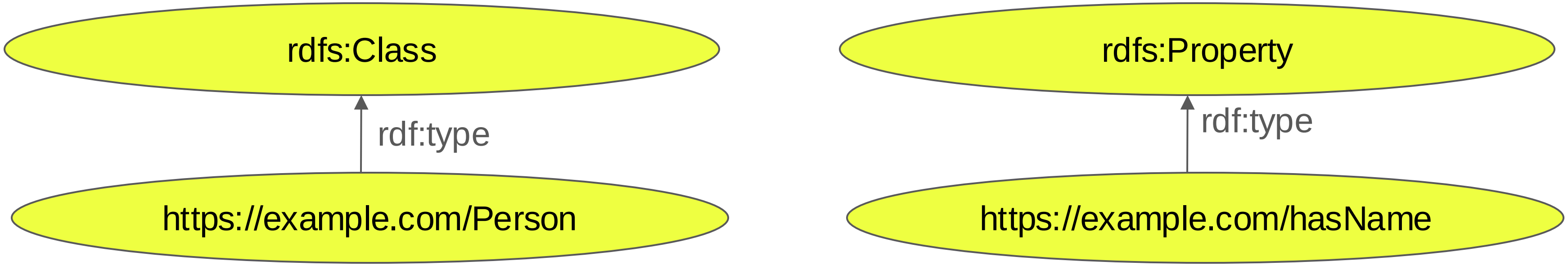
RDFS

- **Resource Description Framework Schema (RDFS)** is a semantic extension of RDF that builds on top of the former and allows us to begin adding semantics to our data:
  - *[RDFS] provides mechanisms for describing groups of related resources and the relationships between these resources. … These resources are used to determine characteristics of other resources, such as the domains and ranges of properties* (source)
- With RDFS we can begin to describe salient things about **classes (sets of things) and properties (relationships between things)** and how they relate to each other classes and properties; in particular RDFS allows us to specify hierarchies.



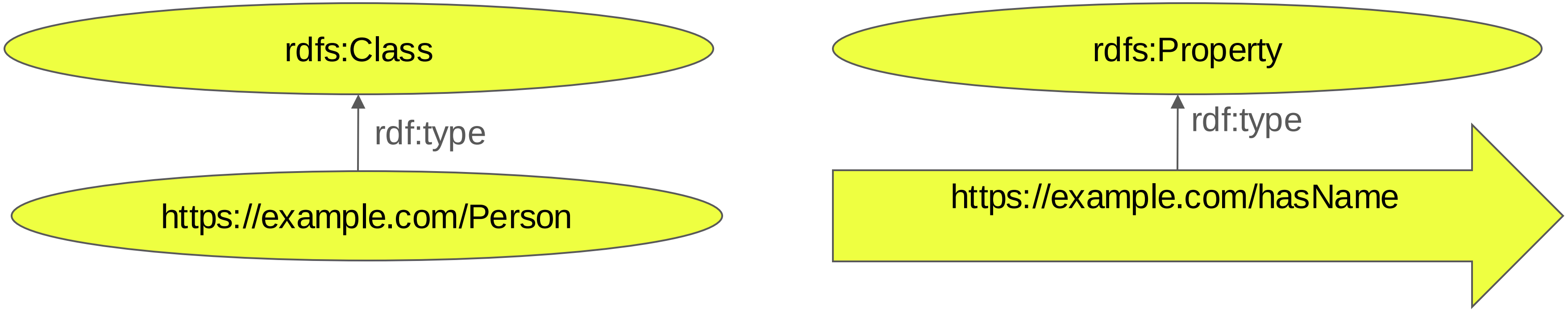
Defining Classes and Properties

- For instance we can use the rdfs **`Class`** resource (http://www.w3.org/2000/01/rdf-schema#Class or rdfs:Class for short) to define a class and we can also use the rdfs resource **`Property`** (http://www.w3.org/2000/01/rdf-schema#Property or rdfs:Property) to define a property (a relationship)
- We can additionally use the properties **`rdfs:subClassOf`** and **`rdfs:subPropertyOf`** to define subclass and subproperty hierarchies

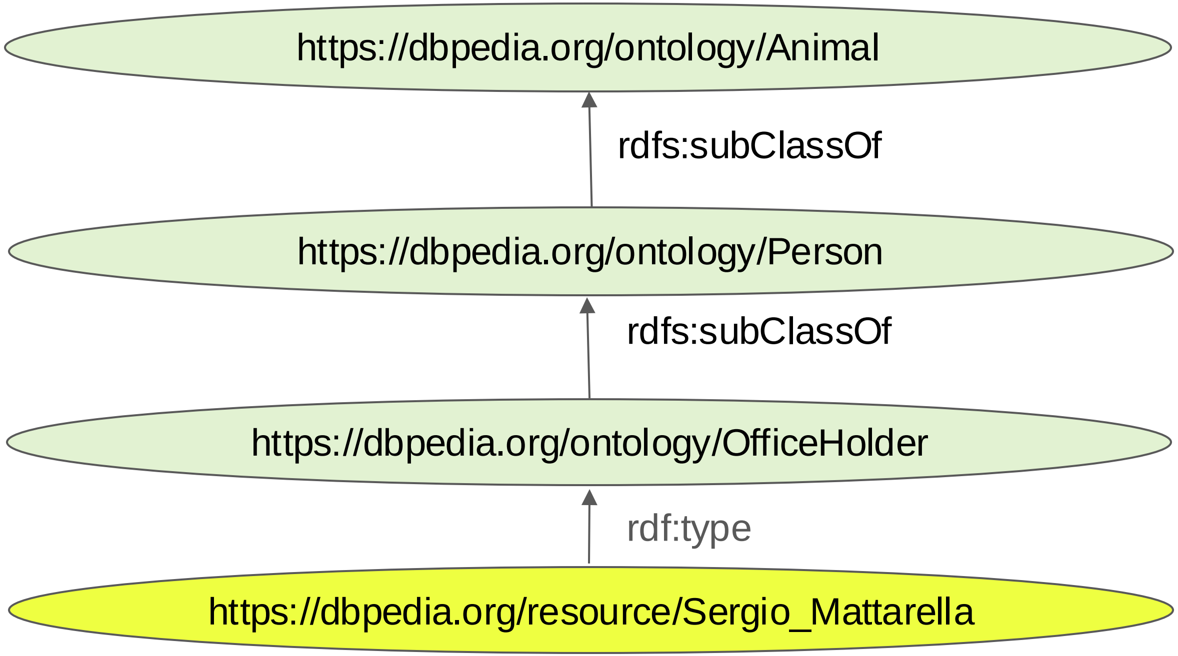


Defining Classes and Properties

- For instance we can use the rdfs **`Class`** resource (http://www.w3.org/2000/01/rdf-schema#Class or rdfs:Class for short) to define a class and we can also use the rdfs resource **`Property`** (http://www.w3.org/2000/01/rdf-schema#Property or rdfs:Property) to define a property (a relationship)
- We can additionally use the properties **`rdfs:subClassOf`** and **`rdfs:subPropertyOf`** to define subclass and subproperty hierarchies



RDFS

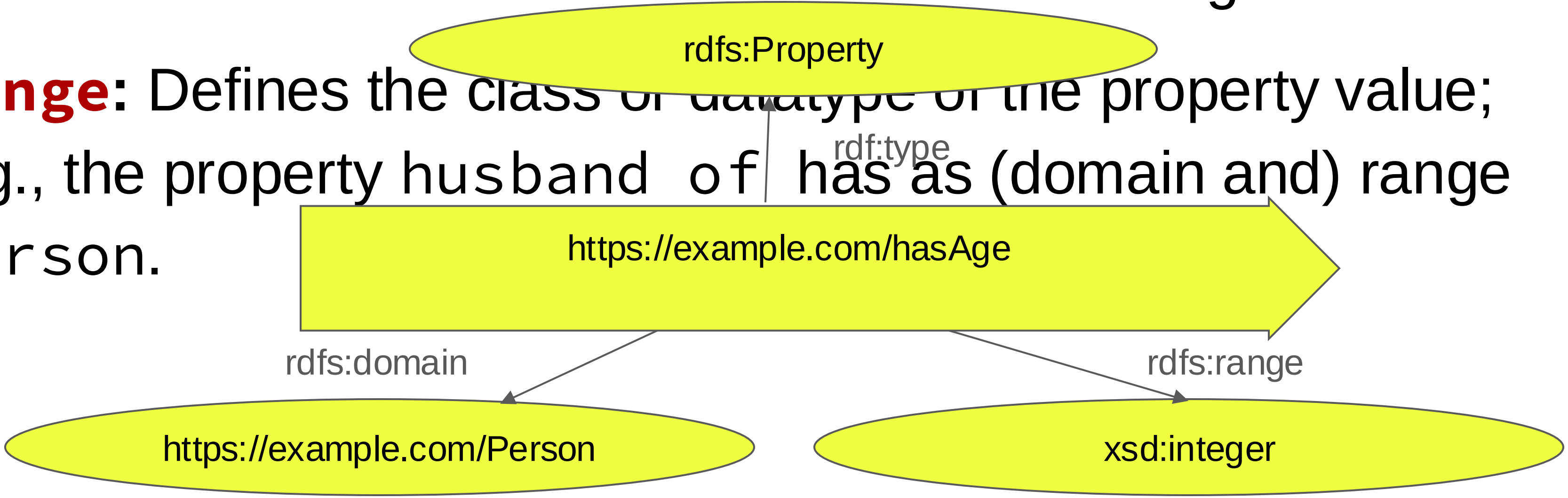


Attribuzione: Quirinale.it

RDFS: Domain and Range

Domain: Defines the class that a property applies to; e.g., the property `name of` has as domain the class `Person`. It does this via the use of the resources domain and range

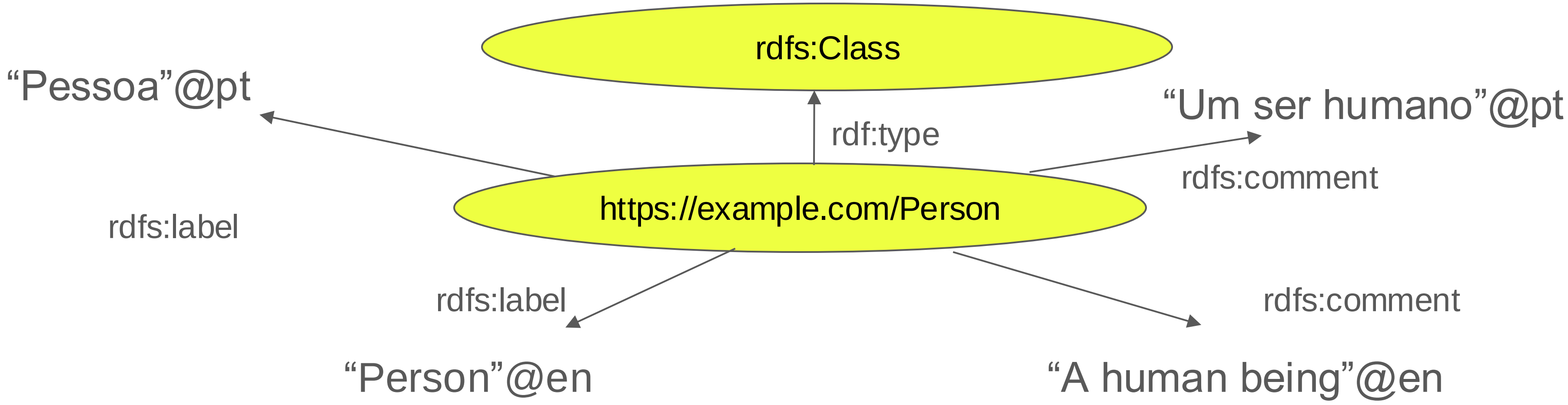
Range: Defines the class or datatype of the property value; e.g., the property `husband of` has as (domain and) range `Person`.



RDFS: Comment and Label

rdfs:label: Provides a human-readable name for a resource.

rdfs:comment: Provides a description of a resource.

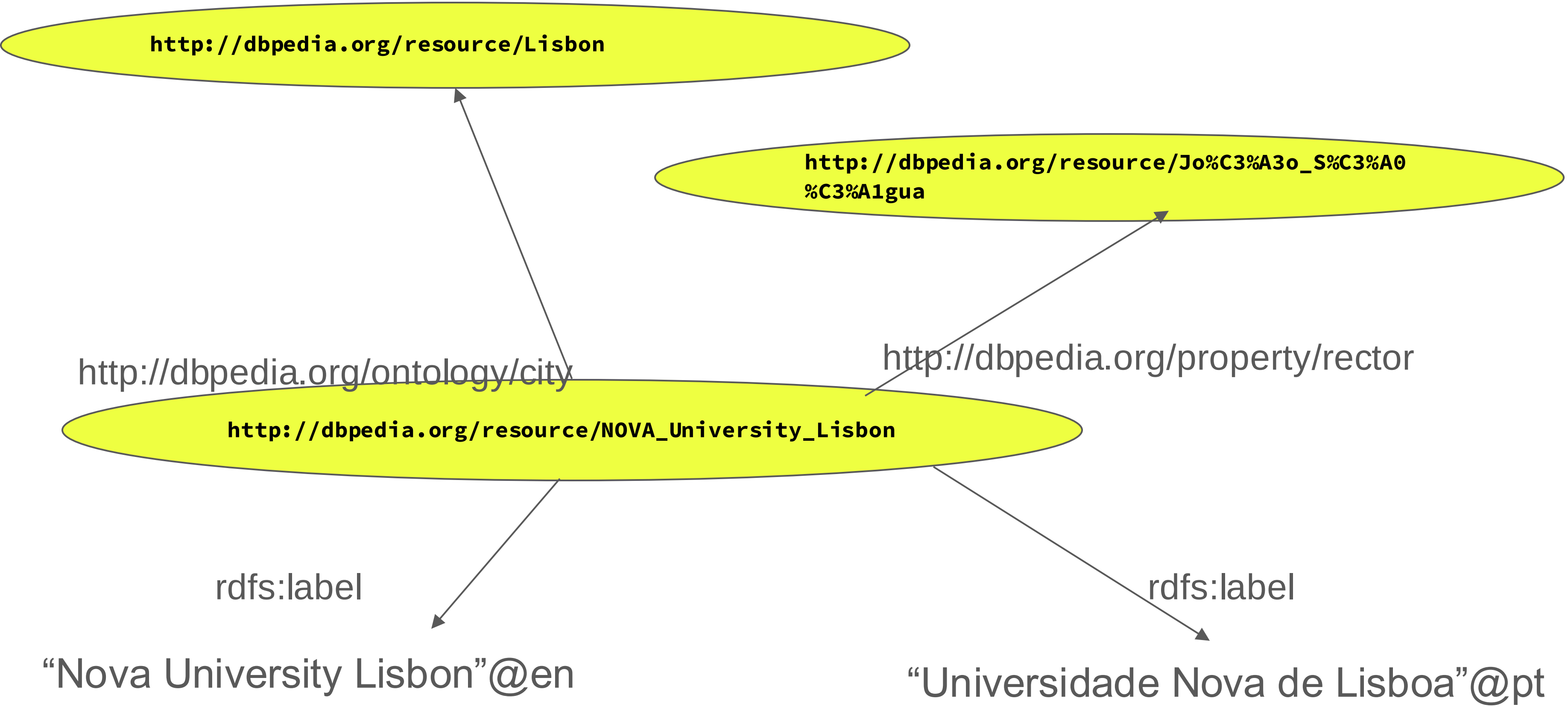


Introduction to Serialisations

- So far we have represented our RDF examples as **diagrams** of graphs
- When it comes to actually producing RDF files in a format that we can **store** or browse there are a number of different so called **serialisations** (or serialization in American English) we can choose.
- We can choose these on the basis of various criteria including compatibility with various technologies as well as readability/efficiency of processing.

Introduction to Serialisations

- We will look at the following common serialization formats:
  - **RDF/XML**
  - **N-TRIPLES**
  - **TURTLE**
  - **JSON-LD**
  - **RDF-A**
- You can convert between formats in the following site:
  - https://www.easyrdf.org/converter


http://dbpedia.org/resource/Lisbon
http://dbpedia.org/resource/Jo%C3%A3o_S%C3%A0
%C3%A1gua
http://dbpedia.org/ontology/city
http://dbpedia.org/property/rector
http://dbpedia.org/resource/NOVA_University_Lisbon
rdfs:label
rdfs:label
“Nova University Lisbon”@en
“Universidade Nova de Lisboa”@pt

RDF/XML

- Uses **XML** to represent RDF data.
- Pros:
  - **Widely supported.**
  - Good for **integration** with XML-based tools and systems.
- Cons:
  - **Verbose** and less readable for humans.

RDF/XML

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:ns0="http://dbpedia.org/ontology/"
         xmlns:ns1="http://dbpedia.org/property/">
  <rdf:Description rdf:about="http://dbpedia.org/resource/NOVA_University_Lisbon">
    <rdfs:label xml:lang="en">NOVA University Lisbon</rdfs:label>
    <rdfs:label xml:lang="pt">Universidade Nova de Lisboa</rdfs:label>
    <ns0:city rdf:resource="http://dbpedia.org/resource/Lisbon"/>
    <ns1:rector rdf:resource="http://dbpedia.org/resource/Jo%C3%A3o_S%C3%A0%C3%A1gua"/>
  </rdf:Description>
</rdf:RDF>
```

N-Triples

- Line-based, Plain Text: Each RDF triple on a **separate** line.
- Pros:
  - Simple and **easy to parse.**
  - Suitable for **large datasets** and streaming.
- Cons:
  - **Less readable** due to lack of abbreviations and prefixes.

N-Triples

```
<http://dbpedia.org/resource/NOVA_University_Lisbon> <http://www.w3.org/2000/01/rdf-schema#label> "NOVA University Lisbon"@en .

<http://dbpedia.org/resource/NOVA_University_Lisbon> <http://www.w3.org/2000/01/rdf-schema#label> "Universidade Nova de Lisboa"@pt .

<http://dbpedia.org/resource/NOVA_University_Lisbon>
<http://dbpedia.org/ontology/city> <http://dbpedia.org/resource/Lisbon> .

<http://dbpedia.org/resource/NOVA_University_Lisbon>
<http://dbpedia.org/property/rector>
<http://dbpedia.org/resource/Jo%C3%A3o_S%C3%A0%C3%A1gua> .
```

JSON-LD (JSON for Linking Data)

- **JSON**-based Syntax: Integrates seamlessly with JSON-based systems.
- Pros:
  - Easy to use with **web APIs**.
  - **Embeds** RDF in web documents.
- Cons:
  - JSON format might be **unfamiliar** to some RDF users.

JSON-LD

```
[
    {"@id":"http://dbpedia.org/resource/Jo%C3%A3o_S%C3%A0%C3%A1gua"},
    {"@id":"http://dbpedia.org/resource/Lisbon"},
    {"@id":"http://dbpedia.org/resource/NOVA_University_Lisbon",
        "http://www.w3.org/2000/01/rdf-schema#label":
            [
                {"@value":"NOVA University Lisbon","@language":"en"},
                {"@value":"Universidade Nova de Lisboa","@language":"pt"}
            ],
        "http://dbpedia.org/ontology/city":
            [{"@id":"http://dbpedia.org/resource/Lisbon"}],
        "http://dbpedia.org/property/rector":
            [{"@id":"http://dbpedia.org/resource/Jo%C3%A3o_S%C3%A0%C3%A1gua"}]
            }
]
```

RDFa (RDF in Attributes)

- **XML**-based Syntax: Embeds RDF directly into XML based formats, usually in HTML.
- Pros:
  - **Enhances** web documents with semantic data.
  - Search engines can parse RDFa for better indexing.
  - Can be used to insert RDF triples inside **TEI-XML documents**
- Cons:
  - Can make HTML more **complex**.

TURTLE

- The **Terse RDF Triple Language (TURTLE)** is the most **human readable** of the common serialisation formats adopted for RDF
- It is more readable for several reasons including the possibility of shortening URIs by defining prefixes at the beginning of turtle documents
- Turtle allows for the same subject, predicate to have more than one object (separated by a comma)
  - i.e., `s p o . s p o’. -> s p o,o’ .`
- It also allows for the same subject to have more than one predicate object pairs (separated by a semi-colon)
  - i.e., `s p o . s p o’. s p’ o’’ .-> s p o, o’; p’ o’’.`

Namespaces

- **Namespaces** are a method to group related terms and avoid naming conflicts.
- They are essential for defining URIs in **a concise and readable** manner.
- A namespace is defined by a URI and a prefix. Prefix declarations are usually made at the beginning of a file.
  - `@prefix ex: <http://example.org/> .`
- This defines the prefix `ex` for the namespace http://example.org/

Common Namespaces

- RDF: http://www.w3.org/1999/02/22-rdf-syntax-ns#
  - Prefix: rdf:
- RDFS: http://www.w3.org/2000/01/rdf-schema#
  - Prefix: rdfs:
- OWL: http://www.w3.org/2002/07/owl#
  - Prefix: owl:
- XSD: http://www.w3.org/2001/XMLSchema#
  - Prefix: xsd:

<http://dbpedia.org/resource/NOVA_University_Lisbon> <http://www.w3.org/2000/01/rdf-schema#label> "NOVA University Lisbon"@en .

<http://dbpedia.org/resource/NOVA_University_Lisbon> <http://www.w3.org/2000/01/rdf-schema#label> "Universidade Nova de Lisboa"@pt .

<http://dbpedia.org/resource/NOVA_University_Lisbon><http://dbpedia.org/ontology/city> <http://dbpedia.org/resource/Lisbon> .

<http://dbpedia.org/resource/NOVA_University_Lisbon> <http://dbpedia.org/property/rector> <http://dbpedia.org/resource/Jo%C3%A3o_S%C3%A0%C3%A1gua> .

@prefix db: <http://dbpedia.org/resource/> .

@prefix dbo: <http://dbpedia.org/ontology/> .

@prefix dbp: <http://dbpedia.org/property/> .

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

dp:NOVA_University_Lisbon rdfs:label "NOVA University Lisbon"@en .

dp:NOVA_University_Lisbon rdfs:label "Universidade Nova de Lisboa"@pt .

dp:NOVA_University_Lisbon dbo:city db:Lisbon .

dp:NOVA_University_Lisbon dbp:rector db:Jo%C3%A3o_S%C3%A0%C3%A1gua .

TURTLE Example

```
<http://dbpedia.org/resource/NOVA_University_Lisbon> <http://www.w3.org/2000/01/rdf-schema#label> "NOVA University Lisbon"@en .

<http://dbpedia.org/resource/NOVA_University_Lisbon> <http://www.w3.org/2000/01/rdf-schema#label> "Universidade Nova de Lisboa"@pt .

<http://dbpedia.org/resource/NOVA_University_Lisbon><http://dbpedia.org/ontology/city> <http://dbpedia.org/resource/Lisbon> .

<http://dbpedia.org/resource/NOVA_University_Lisbon> <http://dbpedia.org/property/rector> <http://dbpedia.org/resource/Jo%C3%A3o_S%C3%A0%C3%A1gua>
.
```

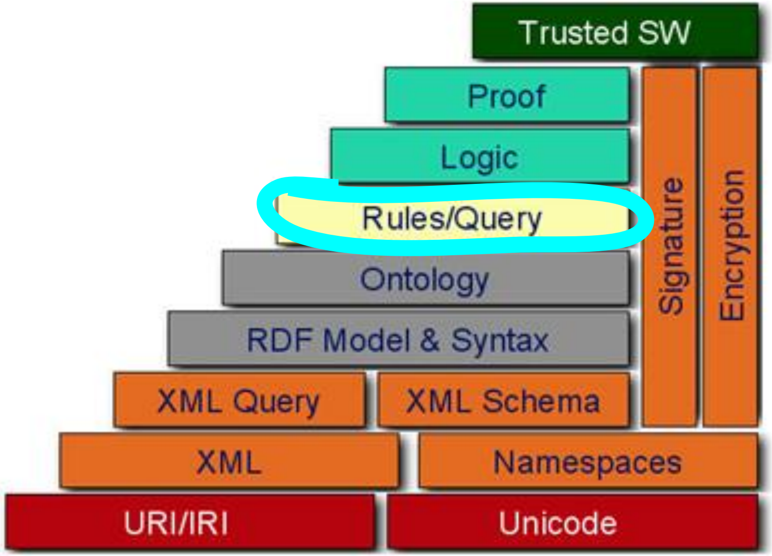
```
db:NOVA_University_Lisbon rdfs:label  "NOVA University Lisbon"@en, "Universidade Nova de Lisboa"@pt;

                          dbo:city db:Lisbon;

                          dbp:rector db:Jo%C3%A3o_S%C3%A0%C3%A1gua .
```

What Serialisation Should I Use?

- Each of these different serialisations has its own advantages and disadvantages
- **Human Readability**: Turtle,
- **Machine Efficiency**: N-Triples, RDF/XML.
- **Integration with Web**: JSON-LD, RDFa.
- Best Practice: Use the serialization that best fits your **specific needs** and environment .
- We will use **Turtle** in the rest of the course because it is the most readable of the common serialisations

Exploring the Semantic Web Stack

A first tutorial on SPARQL



What is SPARQL?

- Stands for the recursive acroyntm **SPARQL Protocol and RDF Query Language**
- As the name suggests it is used as a query language and protocol used to query RDF data (we can also use it to update RDF datasets)
- W3C standard since 2008.
- Uses: Retrieve and manipulate data stored in RDF format
- The syntax is based on the database query language **SQL** but also uses RDF-like syntax:

Why SPARQL? SPARQL Endpoints

- Querying Linked Data: SPARQL is designed specifically for RDF data.
- **Flexibility**: Can extract information in a flexible way from diverse data sources.
- **Integration**: Works well with other semantic web technologies (e.g., RDF, RDFS, OWL).
- **SPARQL endpoint**: Web services accepting remote SPARQL queries and returning results
- One of the most well known SPARQL endpoints is that made available by the DBPedia node. We will look at this endpoint in this short tutorial

Basic Structure of a SPARQL Query

SELECT: Specifies the variables which we would like to appear in the query results.

WHERE: Contains triple patterns to be matched against the RDF data .

```
SELECT ?subject ?predicate ?object
WHERE {
  ?subject ?predicate ?object.
}
```

Basic Structure of a SPARQL Query

SELECT: Specifies the variables which we would like to appear in the query results.

WHERE: Contains triple patterns to be matched against the RDF data .

```
SELECT *
WHERE {
  ?subject ?predicate ?object.
}
```

DBpedia -- the Linked Data Version of Wikipedia

- A project aimed at extracting structured information from Wikipedia and making this information available on the Semantic Web.
- Transforms the unstructured data of Wikipedia into structured linked data in RDF.
- Data is extracted from Wikipedia infoboxes, categories, and other metadata.
- Offers a SPARQL endpoint for querying the dataset
  - https://www.dbpedia.org/resources/sparql/
- Used for data integration, semantic web applications, and linked data projects.Supports a variety of applications including search, data mining, and AI
- Pages available in human readable form too (thanks to content negotiation)

DBpedia First Query

We will look at the first 10 triples in the dataset using the LIMIT keyword:

```
select * WHERE {?s ?p ?o} LIMIT 10
```

DBpedia Second Query

Explanation: The basic building block of SPARQL queries, similar to RDF triples (subject-predicate-object).

```
select *

{?p <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://dbpedia.org/ontology/City>; <http://dbpedia.org/ontology/country>
<http://dbpedia.org/resource/Portugal>}
```

DBpedia Second Query

Explanation: The basic building block of SPARQL queries, similar to RDF triples (subject-predicate-object).

```
PREFIX  dpo: <http://dbpedia.org/ontology/>
PREFIX dpr: <http://dbpedia.org/resource/>


select *
{?p a dpo:City;
dpo:country dpr:Portugal}
```

USEFUL SPARQL KEYWORDS

FILTER: Restricts the results based on a condition.

OPTIONAL: Includes data if it exists but does not require it for the result.

BIND: Assigns a value to a variable within a query.

UNION: Combines results from multiple graph patterns.

GROUP BY: Groups results by one or more variables.

ORDER BY: Orders the results based on a variable.

LIMIT: Restricts the number of results returned.

OFFSET: Skips a number of results before returning the rest.

DESCRIBE: Returns an RDF graph that describes resources.

ASK: Returns a boolean indicating if a query pattern matches.

CONSTRUCT: Returns an RDF graph constructed from the query results.

ORDER BY

ORDER BY: Orders the results based on a variable.

List all the cities in Portugal and order them on the basis of their population

```
PREFIX  dpo: <http://dbpedia.org/ontology/>
PREFIX dpr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>

select ?c ?p
{?c a dpo:City;
dpo:country dbr:Portugal;
dbp:populationTotal ?p
}
ORDER BY DESC(?p)
```

YOUR TURN

Find all the rivers in Portugal ordered by their length

A Potential Solution

What about this?

```
PREFIX  dpo: <http://dbpedia.org/ontology/>
PREFIX dpr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>

select ?r
{?r a dpo:River;
dpo:country dbr:Portugal
}
```

A Potential Solution

What about this?

```
PREFIX  dpo: <http://dbpedia.org/ontology/>
PREFIX dpr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>

select ?r
{?r a dpo:River;
dpo:country dbr:Portugal
}
```

A Better One!

```
select ?r

{?r a
<http://dbpedia.org/class/yago/WikicatRiversOfPortugal>

}
```

Moral: DBpedia can be a bit strange

OPTIONAL

OPTIONAL: Includes data if it exists but does not require it for the result.
A list of Portuguese cities and their mottoes (if they have one)

```
PREFIX  dpo: <http://dbpedia.org/ontology/>
PREFIX dpr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
select ?c ?m
{?c a dpo:City;
dpo:country dbr:Portugal.
OPTIONAL {?c dbo:motto ?m}
}
```

COUNT

COUNT: Aggregates the number of results matching a pattern.

How many cities are there in Portugal?

```
PREFIX  dpo: <http://dbpedia.org/ontology/>
PREFIX dpr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>

select COUNT(?c) AS ?ct
{?c a dpo:City;
dpo:country dbr:Portugal
}
```

```
PREFIX yago: <http://dbpedia.org/class/yago/>

select COUNT(?c) AS ?ct
{?c a yago:WikicatCitiesInPortugal
}
```

AVG

AVG: Calculates the average value of numeric data.

Find the average population of the cities in Portugal:

```
PREFIX  dpo: <http://dbpedia.org/ontology/>
PREFIX dpr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>

select AVG(?pcp) AS ?apc
{?pc a dpo:City;
dpo:country dbr:Portugal;
dbp:populationTotal ?pcp
}
```

UNION

UNION: Combines results from multiple graph patterns.

All the rivers in either Spain or Portugal

```
SELECT  (AVG(?r) as ?a) WHERE {
{
     ?r a <http://dbpedia.org/class/yago/WikicatRiversOfPortugal>;


    }
  UNION
  {
      ?r a <http://dbpedia.org/class/yago/WikicatRiversOfSpain>
  }
}
```

YOUR TURN

Find the average length of all rivers in Spain

FILTER

FILTER: Restricts the results based on a condition.

HOW MANY PORTUGUESE CITIES HAVE A POPULATION MORE THAN 10,000?

```
PREFIX  dpo: <http://dbpedia.org/ontology/>
PREFIX dpr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>

select COUNT(?c) AS ?cc
{?c a dpo:City;
dpo:country dbr:Portugal;
dbp:populationTotal ?p
FILTER(?p > 10000)
}
```

FILTER

FILTER: Restricts the results based on a condition.

Give me the names of the Portuguese cities in Arabic

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?city ?arName
WHERE {
  ?city a dbo:City ;
        dbo:country dbr:Portugal ;
        rdfs:label ?arName .
  FILTER (lang(?arName) = "ar" )
}
GROUP BY ?arName
```

FILTER

FILTER: Restricts the results based on a condition.

Give me the names of Portuguese places starting with ‘Al’

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?place
WHERE {
  ?place dbo:country dbr:Portugal ;
         rdfs:label ?name .
  FILTER (strstarts(?name, 'Al'))
}
```

DISTINCT used to ensure that the results returned by a query are unique, eliminating any duplicate results.

YOUR TURN

Give a list of all Portuguese locations beginning with ‘P’ with their name in English and their population

HOMEWORK OPTIONAL

THE POWER OF REGEX

All of the DBPedia URIs of names of cities in Portugal with ‘, Portugal’ in the title

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?city
WHERE {
  ?city a dbo:City ;
        dbo:country dbr:Portugal .
  FILTER regex(str(?city), ",_Portugal" )
}
```

MORE FILTER

What does the following do?

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
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#>

SELECT ?p ?name (YEAR(?birthDate) AS ?birthYear) WHERE {
  ?p rdf:type <http://dbpedia.org/class/yago/WikicatPortugueseMonarchs> ;
 dbp:birthDate ?birthDate;
rdfs:label ?name .
FILTER (datatype(?birthDate) = xsd:date)
FILTER (lang(?name) = "pt" )
}
ORDER BY ?birthYear
```

BIND

BIND: Assigns a value to a variable within a query

Order the list of Portuguese monarchs by their year of birth and classify them on the basis of whether they were also Spanish and/or Brazilian monarchs

```
SELECT ?name (YEAR(?birthDate) AS ?birthYear) (BOUND(?spanish) AS ?spanishMonarch)
(BOUND(?brazil) AS ?brazilMonarch)  WHERE {
  ?p rdf:type <http://dbpedia.org/class/yago/WikicatPortugueseMonarchs> ;
 dbp:birthDate ?birthDate;
rdfs:label ?name .
FILTER (datatype(?birthDate) = xsd:date)
FILTER (lang(?name) = "pt" )
OPTIONAL { ?p rdf:type <http://dbpedia.org/class/yago/WikicatSpanishMonarchs> .
BIND(true AS ?spanish) }
OPTIONAL { ?p rdf:type <http://dbpedia.org/class/yago/WikicatBrazilianMonarchs> .
BIND(true AS ?brazil) }
}
ORDER BY ?birthYear
```

BIND

Even better

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
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#>

SELECT ?name (YEAR(?birthDate) AS ?birthYear) (IF(BOUND(?spanish), "Yes", "No") AS
?spanishMonarch)  (IF(BOUND(?brazil), "Yes", "No") AS ?brazilMonarch)  WHERE {
  ?p rdf:type <http://dbpedia.org/class/yago/WikicatPortugueseMonarchs> ;
 dbp:birthDate ?birthDate;
rdfs:label ?name .
FILTER (datatype(?birthDate) = xsd:date)
FILTER (lang(?name) = "pt" )
OPTIONAL { ?p rdf:type <http://dbpedia.org/class/yago/WikicatSpanishMonarchs> . BIND(true
AS ?spanish) }
OPTIONAL { ?p rdf:type <http://dbpedia.org/class/yago/WikicatBrazilianMonarchs> . BIND(true
AS ?brazil) }
}
ORDER BY ?birthYear
```

Using Prompting to Write SPARQL Queries

- **Confession**: I used **ChatGPT** to help me write some of these queries
- SPARQL is **a hard language** to maintain fluency in if you don’t use it every day
- The important thing is to have a basic grasp of the way that the **keywords** work
- You can then use prompts to your favourite LLM to help your write your queries
- **BONUS**: open up ChatGPT and use prompting to help write SPARQL queries of your own/use it to help you write some of the queries in the previous slides/exercises

SPARQL Query Forms

- **SELECT**: Retrieve specific variables.
- **CONSTRUCT**: Create RDF graph.
- **ASK**: Boolean query (true/false).
- **DESCRIBE**: Retrieve RDF graph about resources.