Report Web DataMining

# Modeling the Ontology

For this project, we have chosen the vélib' dataset. In this dataset we can observe all velib’ stations, with latitude and longitude, how many bikes are available, how many places are available and some other information.

We define differents classes in our protege project:

The class station with the information on each one

The class user with the name, the age and his favorite station

The class Trajet who links two vélib’ station with a average time of travel

Finally we add a subclass named “big Station” if a station has a capacity of more than 40.

# Populating the ontology

To populate this, we use the data on the website :

<https://opendata.paris.fr/explore/dataset/velib-disponibilite-en-temps-reel/table/?disjunctive.name&disjunctive.is_installed&disjunctive.is_renting&disjunctive.is_returning&disjunctive.nom_arrondissement_communes>

But at the beginning, we created the individuals one by one in Protege so we can start coding quickly.

# Querying the ontology

We put the queries we made in the [Appendix](#_fpf2ysl6341p)

# Manipulating the ontology

The final goal of the project is to determine a route for someone who wants to move from a POI to another in paris. To achieve that goal, we started by showing someone all the POI, and if there are bikes and borns available, then he can choose from where to where he wants to go. This has been coded in Java, using Eclipse.

# Appendix

Here we can see all the queries we have done with Jena :

Name of all stations :

SELECT ?Station\_names

WHERE {

?stat a ns:Velib\_Station .

?stat ns:name ?Station\_names

}

All stations and their city :

SELECT ?Station\_names ?city

WHERE {

?stat a ns:Velib\_Station .

?stat ns:name ?Station\_names .

?stat ns:city ?city

}

People who are more than 45 years :

SELECT ?name ?age

WHERE {

?people a ns:Users .

?people ns:user\_name ?name .

?people ns:user\_age ?age .

FILTER(?age>=45) .

}

Routes with "Harpes" as departure station:

SELECT ?route ?nb\_bike\_free

WHERE {

?departure ns:departure ?route .

?route a ns:Velib\_Station .

?route ns:name "Harpes" .

}

Number of free borns at "Harpes" :

SELECT ?nb\_free\_born

WHERE {

?stat a ns:Velib\_Station .

?stat ns:name "Harpes" .

?stat ns:nb\_born\_free ?nb\_free\_born .

}

### 

### Part I: Modeling the ontology

1. We aim to create an ontology, using the Protégé editor, which models geolocated POI, travelers, trips resources
2. Define or reuse a vocabulary for describing the resources (Schema.org, FoaF, etc.). You can make it more specific (for instance, for train stations, for parking lots, for bicycle sharing stations, etc.) but it should have generic concepts that can be reused for many things.
   1. Define and/or reuse classes
   2. Define and/or reuse datatype and object properties
   3. Define and/or reuse restrictions
   4. Define and/or reuse defined classes

### Part II: Populating the ontology

1. Create some individuals to the different classes: geolocated POI, travelers, trips resources, etc
2. Setup a triplestore. The simplest is to use files, and the best use stores such as: [Apache Jena Fuseki](https://jena.apache.org/download/index.cgi), but you may also install a OpenLink's [Virtuoso server](https://virtuoso.openlinksw.com/) (triplestore used by DBpedia in its backend) or [Blazegraph](https://blazegraph.com/) (triplestore used by Wikidata) or [Stardog](https://www.stardog.com/) (another commercial triplestore that has a free version) or [GraphDB](https://www.ontotext.com/) (yet another commercial triplestore with a free 60-day licence). A [list of triplestores](https://en.wikipedia.org/wiki/Comparison_of_triplestores) is available on Wikipedia.
   1. Have a look on this [lab](https://devinci-online.brightspace.com/d2l/le/lessons/65692/topics/239084) to explore RDF data management and processing ;
3. Convert static data (from open data sources, see bellow) into RDF, and load the resulting data to the triplestore. You can simply generate an RDF file that you load manually to the triplestore, or (better) add the RDF programmatically using SPARQL Update queries. You need for instance to add Json Context to the different available Json files from open data sources.

### Part III: Querying the ontology

1. Write SPARQL queries to response to the following:
   1. List the instances of the geolocated POI
   2. List the name of all train station. For each one, display its city.
   3. List the name of trips that have Paris (or any other chosen city) as destination.
   4. List the name of travellers older than 51 years.
2. Propose 5 SPARQL queries:
   1. A query that contains at least 2 Optional Graph Patterns
   2. A query that contains at least 2 alternatives and conjunctions
   3. A query that contains a CONSTRUCT query form
   4. A query that contains an ASK query form
   5. A query that contains a DESCRIBE query for
3. Define some SWRL rules

### Part IV: Manipulating the ontology using Jena

1. Make an application (Ideally a website, or a GUI application, or a terminal application) that will allow one to select places such as a city (in a list or on a map) and get the associated data. Ideally, provide links to nearby resources or associated resources (travelers, trips), or allow the selection of resources by types (e.g., hospitals, train stations, travelers, trips, etc.).
2. You may also make the data available in RDF (Turtle, RDF/XML, or JSON-LD)
3. The resulting lists and entities are recommender to be available in HTML with RDFa or JSON-LD. While the real time data may be generated on the fly, static data should be extracted from the triplestore using a SPARQL query.