KNOWLEDGE AND DATA ENGINEERING

GROUP PROJECT

TEAM MEMBERS:

CHAO CHEN (19310133)

CHAVVI NIHAL CHANDANI (19316970)

GEORGE JOHN CHAVADY (19305272)

PARITOSH CHAUHAN (19320006)

SARVANI CHAKRABARTY (19305730)

Contents

1. Introduction

Galway is the third largest city in Ireland with a population of almost 80,000. More than a million tourists visit Galway every year. With the tourist visits and population increase graphs steadily increasing in Galway, it is necessary to have useful data in hand which provides information about basic amenities. Galway hosts around 122 events annually and easy to get information on amenities such as parking sites in a particular area of the city which happens to be near the events becomes useful for visitors. Galway has around 30 parks and playgrounds which have different facilities and opening hours. Having smooth and easy access to these information saves time and enhances experience for visitors as well as the local people.

Keeping this objective and the main problem statement of the project in mind, an interactive ontology driven application is built which will provide users access to information about various parks, playgrounds and sports facilities available in Galway.

2. Approach to Ontology Modelling

2.1. Datasets

Datasets	Links
Parks in Galway City	https://data.gov.ie/dataset/parks-in-galway-city/resource/a3a3a88c-11e9-4439-b28e-bede607b5f55
Playgrounds in Galway City	https://data.gov.ie/dataset/galway-city-playground-locations/resource/ed089579-ab97-4ddc-af01-42f248b51160
Public Sports Facilities in Galway City	https://data.gov.ie/dataset/galway-city-public-sports-facilities/resource/eef4bf96-6a8f-4e24-bc87-1c6dd9a89dc7
Weekend Activities	https://data.gov.ie/dataset/activities

Table 1: Datasets

2.2. Competency Questions

- Is it possible to demonstrate relations between leisure activities (events or local businesses) and other civic structures in Galway?
- Most of them are location-based questions. E.g., How would you like to spend your leisure time in Galway?

2.3. Assumptions made

Except for the geographic coordinates (longitude and latitude), the datasets we use have these columns in common:

- Parks' location and Playgrounds' location
- Parks' names and Sports Facilities' (Pitches') name

They appear to have some values in common so we assume that these two pairs of columns stand for the same meanings.

2.4. References to sources used/re-used

We try to adopt the existing vocabulary of http://schema.org/

- The list of vocabulary used (list in 'false turtle'):
  - @prefix schema: <http://schema.org/> .
  - object properties
  - schema:geo , schema:containsPlace
- Data Properties:
  schema:url , schema:identifier , schema:name , schema:longitude , schema:latitude , schema:address , schema:telephone , schema:openingHours
- Classes
  schema:Thing , schema:Place , schema:Intangible , schema:StructuredValue ,schema:GeoCoordinates , schema:AdministrativeArea , schema:Country , schema:CivicStructure , schema:ParkingFacility , schema:LocalBusiness , schema:FoodEstablishment , schema:BarOrPub , schema:Restaurant , schema:SportsActivityLocation , schema:GolfCourse , schema:Park , schema:Playground

We used the GUI tool Protege (https://protege.stanford.edu/) with the visualization plugin VOWL (http://vowl.visualdataweb.org/protegevowl.html) to design the ontology. They also have the online version (WebProtege: https://webprotege.stanford.edu/ , WebVOWL: http://www.visualdataweb.de/webvowl/).

We also use Widoco (https://github.com/dgarijo/Widoco) to generate documentation. And validate the ontology using OOPS.

We tried to save everything in turtle format (*.ttl), but sometimes with hand written works there will have errors.
We use this turtle validator (https://github.com/IDLabResearch/TurtleValidator), it has an online version and a command line tool.

2.5. Data Mapping Process

We use Juma (http://juma.adaptcentre.ie/juma-editor/) to uplift data which is a user friendly GUI environment.

Juma often crashes and it's difficult to do complex mappings in Juma. We use `grep` to filter the 'Activities' CSV into 4 different CSVs (Bar / Restaurant / Golf Club / Others). The commands are listed below:

- `-w` for a single word
- `-i` ignore the upper cases

- `\|` convert pipe to OR
- cat Activities.csv | grep -i -w "bar\|pub" > Bars_Activities.csv
- cat Activities.csv | grep -i -w "restaurant" | grep -i -w -v "bar\|pub" > Restaurant_Activities.csv
- cat Activities.csv | grep -i -w "golf" | grep -i -w -v "restaurant" | grep -i -w -v "bar\|pub" > Golf_Activities.csv
- cat Activities.csv | grep -i -w -v "golf" | grep -i -w -v "restaurant" | grep -i -w -v "bar\|pub" > Other_Activities.csv
- After uplifting the data using Juma, some operations like replacement (replacing the IRIs with prefixes) were done using code editor (Visual Studio Code, https://code.visualstudio.com/). This step is performed to get a clear TURTLE file.

2.6. Use of inverse, symmetric and transitive properties

- symmetric: :otherPitch
  - if two 'Pitch' have the same NAME, it shows they are in the same park. So pitchA :otherPitch pitchB
- inverse: :locationR
  - location reverse, the inverse of schema:containsPlace.
  - if there is a 'County' inside a 'Country', so have:
    - countryA schema:containsPlace countyA
    - countyA :locationR countryA
- transitive: schema:containsPlace
  - if a 'Country' has a 'County', and that 'County' has a 'Region', so the 'Country' also has that 'Region'
    - countryA schema:containsPlace countyA
    - countyA schema:containsPlace regionA

3. Design Overview

3.1. Application Query Interface

The applications used to develop the query interface are HTML, CSS, bootstrap and Flask framework. We have used Apache Jena Fuseki to query the uplifted '.ttl' file.

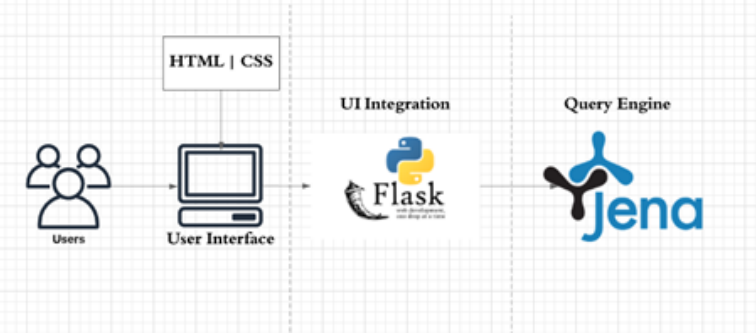

Figure1

3.2. Queries Descriptions

Index	Questions	Approach	Queries

1	List the names of parks which are located near a particular playground.	The location of the playground can be fetched using which locations of nearby parks can be computed.	SELECT DISTINCT ?name WHERE{
			?PARK_obj schema:address ?pg_location .
			?PARK_obj schema:name ?name .
			FILTER regex(str(?PARK_obj) , 'park')
			{
			SELECT ?pg_location ?pg_obj where {
			?pg_obj schema:name "South Park Playground" .
			?pg_obj schema:address ?pg_location .
			}}}

2	What are the facilities provided near a playground?	The data can be fetched from nearby parks which have facilities like Pedestrian Walk, Car/Bus Park, Picnic spots, etc. These facilities will be can be availed by people visiting nearby playgrounds as well.	SELECT ?facilities WHERE{
			?PARK schema:address ?pg_loc .
			?PARK :facilities ?facilities
			{
			SELECT ?pg_loc where {?pg_obj name: "South Park Playground" .
			?pg_obj schema:address ?pg_loc .
			}
			}
			}

3	List the playgrounds in a particular area of the city.	The location of parks with respect to the area of the city (City centre, City west, City east) are specified. The location of the playgrounds in a particular area of the city can be derived from the location of nearby parks.	SELECT ?name where {
			{?X :areaOfCity "City-West" .
			?X schema:address ?name .
			} UNION {SELECT ?name where {
			{?X :areaOfCity "City -West" .
			?X schema:address ?name .
			}
			}
			}

4	Which place has both park and playground?	Distinct locations of parks can be fetched which can then be matched with the respective locations of the playgrounds. The distinct locations along with the names of both playgrounds and parks are displayed.	SELECT Distinct ?Location ?name ?parkname WHERE{
			?PG schema:address ?Location .
			?PG schema:name ?name .
			?PARK_obj schema:name ?parkname .
			FILTER regex(str(?PG) , 'layground')
			{SELECT Distinct ?PARK_obj ?Location where {?PARK_obj schema:address ?Location.
			FILTER regex(str(?PARK_obj) , 'park')
			}
			}
			}

5	Give a list of park names where vehicle parking is within estate.	The csv file of playground consists of parking facilities which are located within the playground's estate. That data can be used to find parks in the same location which would have the same facility for visitors who would like to park their vehicles within estate.	SELECT distinct ?Park_name ?c ?Pitch_type {
			?Pitch_obj :parkName ?Park_name.
			?Pitch_obj :pitchType ?Pitch_type.
			{
			SELECT ?Park_name
			where{
			?Park_obj schema:address 'Cappagh Road, Knocknacarra'.
			?Park_obj schema:name ?Park_name.
			}
			}
			}

6	What are the types of pitches available in a particular location?	The datasets for sports facilities and parks are used for this purpose. From the location of parks, park names are filtered out corresponding to which the pitch types are returned.	SELECT distinct ?Park_name ?Pitch_type {
			?Pitch_obj :parkName ?Park_name.
			?Pitch_obj :pitchType ?Pitch_type.
			{
			SELECT ?Park_name
			where{
			?Park_obj schema:address 'Cappagh Road, Knocknacarra'.
			?Park_obj schema:name ?Park_name.
			}
			}}

7	Find parking slots near sports facilities.	Visitors going to sports facilities can avail vehicle parking located within parks which are nearby. The location of parks can be retrieved from the Parks csv file, from which the names of parks and sports facilities located nearby can be fetched	SELECT ?parking ?parkname where {
			?playground_obj schema:address ?location.
			?playground_obj schema:containsPlace ?parking.
			{
			SELECT ?location ?parkname where{?park_obj schema:address ?location.
			?park_obj schema:name ?parkname.
			FILTER contains(?location,"Mervue")
			FILTER regex(str(?park_obj), 'park')
			}
			}
			}

8	Find toilet facilities near a particular park.	The playground csv file has listed if toilet facilities are available or not near the respective playgrounds. With reference to that, the list of parks in the same neighborhood can be retrieved which will have toilet facility in the same radius.	SELECT ?name ?toilet_availability
			WHERE{
			?PARK_obj address: ?pg_location .
			?PARK_obj name: ?name .
			?PARK_obj co:toilet ?toilet_availability.
			{
			SELECT ?pg_location ?pg_obj where {
			?pg_obj name: ?name .
			?pg_obj address: ?pg_location .
			FILTER contains(?name,"Cappagh")
			}}}

9	List playgrounds which are also considered as local neighbourhood parks.	Whether a playground is located near a local neighborhood park or a city park can be found out by using the playground and park csv files. The parks have descriptions about the category they belong to. Comparing this information with the locations where both parks and playgrounds are situated, the list of playgrounds near local neighborhood parks can be retrieved.	SELECT distinct ?pg_name where {
			?park_object schema:address ?location.
			?park_object schema:name
			?park_name.
			?park_object :description
			?local_neighbourhood_park
			FILTER regex(str(?park_object) , 'park')
			FILTER
			contains(str(?local_neighbourhood_park) ,'Local')
			{
			SELECT ?location ?pg_name where
			{?object schema:address ?location.
			?object schema:name ?pg_name
			filter regex(str(?object),"playground")
			}
			}
			}

10	List of places along with their phone numbers Using Latitude and long in Galway.(using sports facilities along with activity)	Ireland is famous for the enthusiasm they possess for sports. Crowds often visit pubs or restaurants post a football match hosted in a stadium. This particular query compares the geographical coordinates of food or drink stalls with those of nearby sports facilities and returns their location as well as their contact numbers for easy access by people.	-

11	List the type of surface available near a park.	From park and playground csv files, the locations of parks and playgrounds can be matched and retrieved using which the surfacetype available near a park can be returned.	SELECT ?name ?surface
			WHERE{
			?PARK_obj address: ?pg_location .
			?PARK_obj name: ?name .
			?PARK_obj :surface ?surface.
			{
			SELECT ?pg_location ?pg_obj where {
			?pg_obj name: ?name .
			?pg_obj address: ?pg_location .
			FILTER contains(?name,"Cappagh")
			}}}

Table 2: Description of Queries

4. Challenges

4.1. Ontology Modelling

- For Protege, we tried to adopt the existing vocabulary from schema.org. But once we import the ontology file in Protege and save. All the vocabulary from schema.org will be exported to our ontology turtle file. So for the ontology file we wrote it manually using code editor and then validate it with:
  - Protege
  - IDLabResearch/TurtleValidator.
- For Widoco, the final visualization does not work on the generated webpage, but it works fine on the WebVOWL

4.2. Creating Queries

- Challenges faced in using longitude and latitude in construction of queries

4.3. Mappings

- In Juma, one mapping cannot be linked to itself. So the symmetric linking is done in the code editor. (40+ instances so it is fine). After that we need to validate the TURTLE file.
- In Juma, there seems to be a limitation on the number of mappings. So we tried to separate the CSVs in 2 mapping projects in Juma.
- Complex filtering using command line tools.

5. Conclusions

As a group, we all had the opportunity to walk through ontology, queries and the User Interface of the application. Some of the insights which we gathered from this project are as below:

- Building ontologies involve essential relationships construction between concepts. This enables automated reasoning about data and deploys systematicity in retrieving information.

- Ontologies provide a more coherent and easy navigation between concepts, thereby making it more user readable.
- Protégé not always adapts standard ontology editors to terminological purposes.
- Involved work can be time consuming.