Technical Report

Knowledge and Data Engineering CS7IS

Group G

Aishwarya Agarwal
Boris Flesch
Sherwin Mascarenhas
Niejun Yin
Rui Xu

Table of contents

Approach to ontology Modelling

Description of Competency Questions that ontology answers

1. What is the role of the player who scored the most goals?

With this question, we aim to find the Role of the player, which is either (“Midfielder”, “Forward”, “Defender” and “Goalkeeper”, who has scored the most number of goals. We chose this question as it gives great insight on which player is at an advantage by his position when it comes to scoring the most goals. Since we are trying to get the max number of goals and the role of the player, our ontology helps us with attributes like hasRole and Goals along with classes like Role and dbpedia’s SoccerPlayer. The datasets that will be used are the Player Scores and the Player Stats datasets.

2. What were the points of the opposition team of the team that faced Russia in the first round ?

With this question, we try to find the Opposition team to the opponents of Russia in the first round. This question was designed to make use of our symmetric property in our ontology which is the isOpponentOf property. The question also made use of the hasHomeTeam, round and points properties along with Classes like dbpedias FootballMatch and Team from our ontology. The Fixtures and teams datasets were used for the question.

3. Which teams have the highest score in the first 3 group stages of the world cup?

With this question, we try to find the teams in the first 3 groups stages of the world cup who have the highest scores. This question helps understand the progression of teams that had the best score in the group stages. This is done by deciding which team won using the score and then seeing which team has the highest number of wins. The team with the highest number of wins in the first three rounds is then filtered out. RoundNumber, hasAwayTeam, hasHomeTeam, HomeTeamScore and AwayTeamScoreq as properties and FootballMatch and Team as classes from our ontology. The datasets used for this are team and fixtures.

4. Who are the goalkeepers of the Semi-Finals teams ?

The query is pretty straight forward as we try to identify the goalkeepers that played in the Semi-Finals. We found that this question was useful enough to be used in relation with other queries to scout goalkeepers or analyzed at later stages to compare stats. Here the classes Team, SoccerPlayer and FootballMatch are used and with properties like playsInTeam, hasRole, RoundNumber from our ontology. The datasets used are Fixtures, Players and Teams

5. How many different clubs are the Spain team players in?

Here we try to see which clubs the Spanish team’s players belong to. This is a fun question that aims to understand whether the players part of such a big national team, play in big clubs as

well. We use the classes Team, Club and properties like hasNationality and playsInClub from our ontology to help with our query. We source the data from Player Scores and the Players datasets for this.

6. Which team has the lowest total number of goals in the season?

Here we try to find the Teams that have scored the least number of goals in a season. This question is not only very intuitive, but it also gives the goals of the team without a direct mapping of team to goals in the dataset but instead through the different players because of a well structured ontology. We use the Team class and hasNationality and Goals from our ontology from this to answer our question. We make use of the Player Scores and the Players dataset for this.

7. What are the nationalities of the top 10 players with the most number of assists in their clubs?

Since we need to find the nationalities for the top 10 players with the most number of assists in their clubs, we first get the assists corresponding to each player and then select the 10 top players and find the nationalities corresponding to each. We decided on this question as it helps get good insight if required in player recruitment. We make use of Country, and Athlete Class and hasNationality and hasRole properties that we have created in our ontology to help with our question. We use the Players Scores and the Players datasets.

8. Which players played in the home team on 23/06/2018 during the Football World Cup?

Here we need to find the players who played in the home team for a certain date in the world cup. Can be used by analyzers to determine which player played or was sick, or was in a legal issue with regard to a ban and still played. The ontology helps us answer the above as we have created a class for HomeTeam and AwayTeam. And since we have a class for players we just query the players that have a property playsInTeam and a Class Match that has property hasHomeTeam to get the information required to answer the question.

9. Which players are composing the teams engaged in semi-final matches?

Here we try and find the players that have played in semi-final matches. This question we felt was pretty informative to recruiters who are scouting for amazing players. To help with this we have FootballMatch and SoccerPlayer Class from our ontology along with properties like Roundnumber , hasAwayTeam and playing team to help get data for our question. To answer this we use 2 datasets namely Players and Fixtures.

10. Which players won a match between 16/06/2018 and 22/06/2018 during the Football World Cup 2018?

This question asks for data on the players who won a match during a time period in the 2018 world cup. This question can be converted to a more generic search query over a time period for

different events, showing the functionality of our ontology. Here our ontology along with classes like FootballPlayer, Team, and properties like Home Team Score, AwayTeamScore, hasAwayTeam, hasHomeTeam, scheduled at, and RoundNumber to answer the question. We used the Players and the Fixture table to answer this question.

11. In which country did the first match occur?

This is a very straightforward query, however it helps us draw out the power of our ontology as it makes use of our transitive isin property that helps to go from stadium directly to country. A few properties that were used scheduledAt and tookPlaceIn as isin and a few classes that were used from our ontology were Stadium, City and FootballMatch.

12. What is the population of each opponent's country of the last match? (using dbpedia data)

This question is a peculiar one as it only considers the use of a single of our datasets. However, it also uses information retrieved directly from Dbpedia triplestore as our ontology allows us to bind entities together. We thought it would be interesting to have such a query that uses two triplestores simultaneously. This query retrieves the population (i.e. number of inhabitants) of the countries which teams opposed during the last match of the football world cup.

Description of datasets selected for application

The following datasets have been used in our ontology :

Players

The document is sourced from : https://data.world/sawya/football-world-cup-2018-dataset
This data set contains information on the players nationality and the name. This dataset has 2 columns. The columns are Nationality (of the Player) , Player (Name of the Player).

Players_Scores

The document is sourced from : https://data.world/sawya/football-world-cup-2018-dataset
This dataset contains information on the players individual scores with respect to their club and league. There are a total of 15 columns. These are Rank (Rank of the player in the league), Player (Name of the Player), Club (The club the player is part of), age, Apps (Number of appearances in the matches played in the league), Mins(Minutes Played), Goals, Assists, Yel (number of Yellow Cards Awarded to the player), Red (Number of Red Card awarded to the player), SpG (Shots per Goal), PS (Pass Success), AerialsWon (Number of Aerials Won), MotM (Number of Man of the Match Awarded) and Rating (Overall Player Rating).

Players_Stats

The document is sourced from : https://data.world/sawya/football-world-cup-2018-dataset
This dataset contains Player's KPI's (Key Performance indicators) for the season. There are a total of 9 columns in this dataset. The columns are : Rank (overall Rank of the player), Player (name of the Player), role (Role of the player - can be midfielder, forward, goalkeeper or defender), playedgames (total number of games played) , playedmins (total number of minutes played), perfdef (Defence KPI of the player), perfattack (attack KPI of the player), perfposs (position KPI of the player) and total (Overall Player Performance).

Teams

The document is sourced from : https://data.world/sawya/football-world-cup-2018-dataset
This dataset contains information on the teams that played in the world cup. There are a total of 3 columns. They are : Team (Name of the Team) , Points (Total Points in the world Cup) and Image (Url to the national flag of that team).

Fixtures

The document is sourced from : https://fixturedownload.com/results/fifa-world-cup-2018
This dataset contains Fixture related information that includes the matches, round numbers and scores. The dataset contains 9 columns. They are RoundNumber (Current Round in the Wold Cup), Date (The date the match took place) , Location (Location of the match) , HomeTeam, AwayTeam, Group (The group the teams are in) , Result (Result of the match (Score)), HomeTeamScore (Custom Column that Segregates the Scores Column in to HomeTeamScore and AwayTeamScore) , AwayTeamScore.

Assumptions made

We had a couple of assumptions in the beginning that were quickly resolved as we progressed with the assignment and got help from the TA. The following, however are the assumptions that we still have:
1. The dataset is complete i.e. there is no missing data, and that the dataset has credible information,
2. There is a need for the dataset to be uplifted.
3. There is a need for a general ontology specific to SportsEvents.
4. The Team column in Teams.csv is also the country.

References to sources used/reused e.g. SIOC, FOAF for people

We used dbpedia primarily for all our existing ontology reuse. We found that most of our Classes that we had initially designed already existed on dbpedia. However there were a few that were hard to incorporate into our ontology and we have left them as future scope.

The References use
Dbpedia => http://dbpedia.org/ontology/

Class dbo:Athlete => http://dbpedia.org/ontology/Athlete
SubClass : SoccerPlayer => http://dbpedia.org/ontology/SoccerPlayer
Class dbo: Match => http://dbpedia.org/ontology/Match
SubClass : FootballMatch => http://dbpedia.org/ontology/FootballMatch
Class dbo:City => http://dbpedia.org/ontology/City
Class dbo:SoccerClub => http://dbpedia.org/ontology/SoccerClub
Class dbo:Stadium => http://dbpedia.org/ontology/Stadium

The reasons for selecting the entities from dbpedia:
Our original ontology consisted of only Player and Match. However with the Introduction of Athlete and Match along with their subclasses as SoccerPlayer and FootballMatch we managed to build an ontology that was able to be specific to our datasets on the Football world cup as well as provide a base to easily integrate other existing sporting events and classes to our ontology.

We did not select Country and Team from dbpedia because of a conflict that was arising with our custom property mappings. Complete integration can be achieved with dbpedia, which is considered as Future Scope.

Finally we believe that the Role class was a very custom entity that wasn't suitable for mapping with existing ontologies because of the special subclasses like Goalkeeper, Forward, Defender and Midfielder.

Discussion of your data mapping process

To map the data, the first step was creating an ontology in Protégé for a Football Event. In our ontology we defined 14 classes(including sub-classes) which also include classes from existing DBPedia ontology. In our ontology we have created 23 Data Properties and 10 Object Properties(including sub-properties). For all the properties we have defined Domain and Ranges, also we have one property for each of the following characteristics – Inverse, Symmetric and Transitive. In addition to this we have added cardinalities for the classes wherever possible.
Further to uplift data, we created R2RML mapping. Initially we had created the mapping using JUMA but later all the additions and modifications in the mapping were done manually using a Text Editor. In R2RML mapping for each of the Triple Map, we have mapped the Logical Table with one of the csv files of our datasets. Template(rr:template) is used in the subject map to generate subject IRIs from the identifier column for the particular class. For example: for class Team, the subject IRIs are generated from the column Team. Each Predicate Map is mapped with data property name defined in ontology using rr:constant while Object Map is mapped with column name using rr:column. For object properties we have mapped two Triple Maps using join condition. We have also defined data types for all the data properties in R2RML mapping. After R2RML mapping, we uplifted data using R2RML engine.
At last, we uploaded both the files i.e. ontology Model file (OWL file) and Uplifted Data file (TTL File) into Graph DB to validate uplifted data with ontology model and to create SPARQL Queries.

Explanation of use of inverse, symmetric and transitive properties

We implemented the following inverse, symmetric and transitive properties as part of our ontology:

- Inverse property: *se:hasPlayer* inverse of *se:playsIn*
  This property allows us to easily navigate back and forth with a player and the team or club he plays in. For example -
  :Germany :hasPlayer :Manuel Neuer
  further implies
  : Manuel Neuer :playsIn :Germany
- Symmetric property: *se:isOpponentOf* allows us to find opponents of a particular team. For example –
  :Russia :isOpponentOf :Egypt
  further implies
  :Egypt :isOpponentOf :Russia

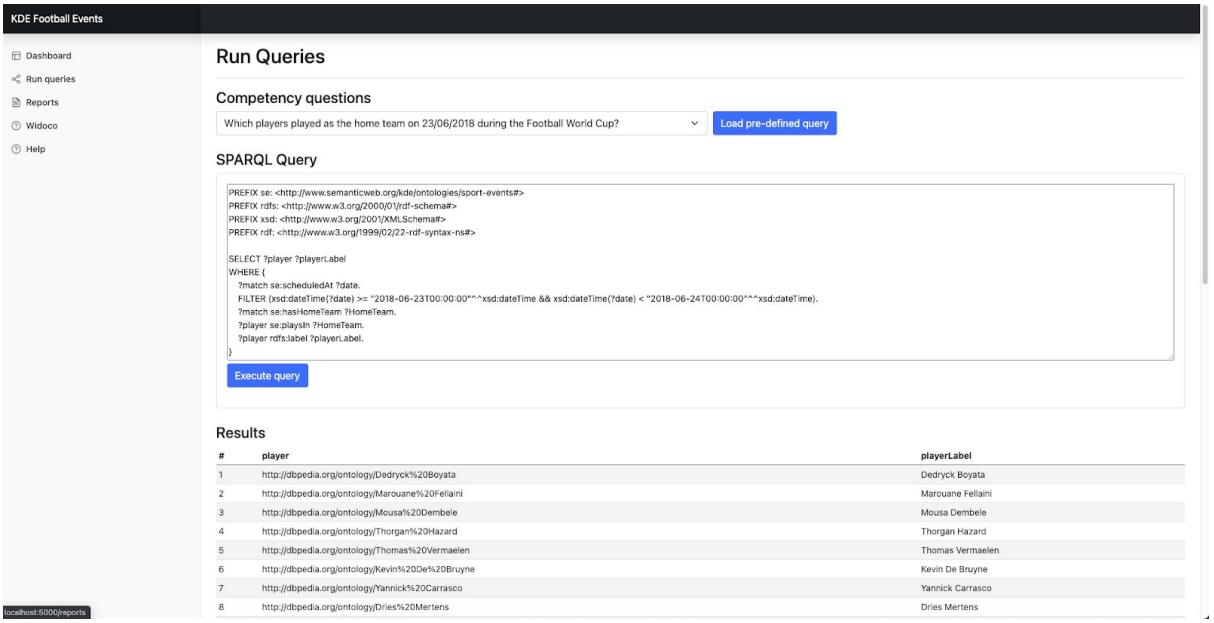
- Transitive property: *se:isIn*
  This transitive property can be used to transitively go from a Stadium to its country, using it as follows: *se:Stadium se:isIn se:City se:isIn se:Country. For example:*
  :Kazan Arena :isIn :Savinovo
  :Savinovo :isIn :Russia
  further implies
  :Kazan Arena :isIn :Russia

Overview of Design

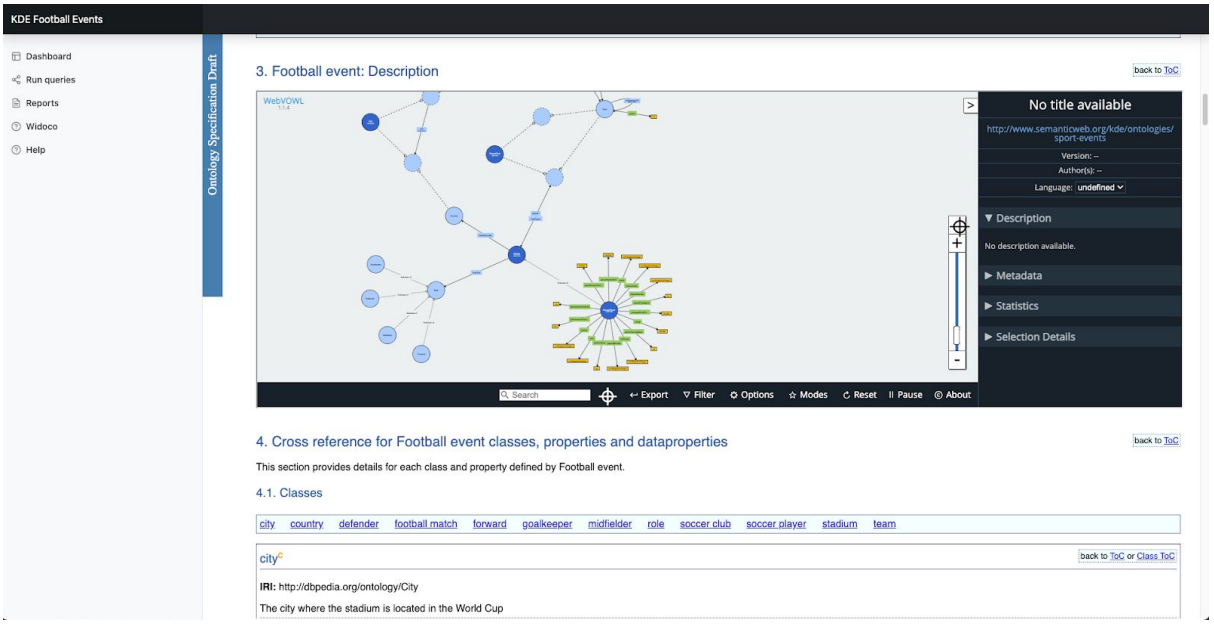
Description of Application Query Interface

Our application query interface is a web-based application running on Python Flask on the back-end and Bootstrap on the front-end. The SPARQLWrapper Python library allows us to query our triplestore from the back-end.

The “Run query” section of the web application is meant to answer predefined questions that can be answered using related SPARQL queries. Selecting a question and clicking on “Load pre-defined query” will execute the SPARQL query and show the query and its results to the user. To provide more flexibility, we decided to opt for a text zone where the user can modify the SPARQL query and therefore execute custom queries. Results are displayed in a table containing SPARQL variables as columns and entries (i.e. results) as rows:



We also decided to include as much help/information as possible directly in this web-application, for example with “Widoco” documentation:



Description of Queries

All our queries have been listed and described as part of the Widoco documentation, section 6 “SPARQL Queries Documentation” (see folder *documentation* at the root of our submitted archive).

Discussion of challenges

While ontology modelling or creating queries and mappings

Even though we managed to pull off and succeeded in creating a thorough ontology as well as being able to answer our SPARQL queries, we faced many challenges throughout.

Our first major hurdle was our confusion with ontology and our R2RML mappings and the uplifting of our data. We were not able to figure out that the disconnection of our ontology and our mappings gets resolved when we store them in the triple store. We managed to finally figure that out with the help of Albert Navarro Gallinad (Teaching Assistant).
We also had troubles understanding how to “join” our datasets when creating the R2RML mapping. At first, it was confusing to understand how to use it to create the relation between our original data and our ontology.

Finally, it has been quite challenging to bind our ontology to an existing one. It took us some time to figure out how to do that and we had to refactor some of the properties and relations of our ontology. While iterating to adjust our ontology, R2RML mapping also required further changes which slowed down the iteration process until everything started to work well together.

Conclusions

Self reflection of group on strengths/weakness of ontology model, queries & interface

Our ontology model has been created to be suitable for football sport events. To do that, we used datasets from the 2018’s Football World Cup. We spent several days working and reflecting on the ontology and tried to make it as flexible as possible. Instead of “mapping our ontology to our datasets”, we created an ontology with classes and properties that seemed relevant to us and processed our data accordingly so that it can fit our ontology (i.e. splitting columns, editing the format of some values, adding new columns if relevant, etc.).
We also tried to bind as many entities of our ontology as possible with already existing ones, such as Dbpedia ontology which offers several classes and properties related to the area of sports. This linking makes our ontology stronger as it takes advantage of external entities that are already part of concrete use cases.

However, we already focused on Dbpedia and think that more existing ontologies could have been mapped to our. One of the weaknesses of our ontology model is its inability to map, as of now, with Country and Team classes from Dbpedia; this is due to the fact that we built particular relationships between these entities. For eg. a Team, in our ontology, is a National Team and is represented by a Country: that allows us to provide more flexibility for relationships between Players, Countries and Teams.
Another weakness is that the players’ roles can only be used as instances as it is not possible to map a subclass to an instance of player. On further updates of the ontology, it would be interesting to evaluate whether the best option is to keep classes and subclasses for Roles or just using them as Players’ properties instead.

We are overall satisfied with the queries we have been able to write; these queries answer the competency questions we defined by making use of our ontology and uplifted data. The only query that could probably be written in a more elegant and efficient way is the one using Goalkeeper subclass of Role, for the reasons mentioned above (i.e. using a class as an instance itself).

On top of that, we built an interface based on Python using the Flask framework. This interface is pretty simple to use: it is intuitive, efficient and provides all the necessary documentation and help to the users. However, this interface is currently quite limited in terms of features and can either answer predefined queries or needs knowledge in SPARQL to be flexible. Providing more built-in features such as filters and navigation through entities would make it much more user friendly.