CMPUT 690: Assignment 3

Peng Xu, *pxu4@ualberta.ca*

1. Observe the RDF Graph

The first thing I do when I started working on this assignment is to observe the given graph and figure out what I should do in this assignment. Via some basic queries, I found there are 12 predicates in the given graph:

- fbk:wikipedia.en_id
- fbk:wikipedia.en
- fbk:wikipedia.en_title
- fb:type.object.key
- fb:soccer.foot_ball_team_management_tenture.manager
- fb:soccer.foot_ball_team_management_tenture.team
- fb:soccer.foot_ball_team_management_tenture.from
- fb:soccer.foot_ball_team_management_tenture.to
- fb:sports.sports_team_roster.team
- fb:sports.sports_team_roster.player
- fb:sports.sports_team_roster.from
- c690:hasDocument

Given these predicates, we can already answer the second question – "Who plays for which teams?" In order to answer the rest questions, we still need to extract some other relations:

- Relations between teams and their stadiums
- Relations between stadiums and where their names come from
- Relations between player and their nationalities

In the subsequent sections, I will illustrate how I extract these relations.

2. Resources Used in this Assignment

In this assignment, I use the *Python* as the programming language. Following package will be used:

- *rdflib*: parse, construct and query the RDF graph
- *SPARQLWrapper*: query the DBpedia
- *NTLK*: used in previous assignment to identify entities in the RDF Graph

In addition, I will also leverage the information on both Freebase and DBpedia to answer the given questions.

3. Identify Entities in the RDF Graph

Via *fb:sports.sports_team_roster.team* and *fb:sports.sports_team_roster.player* the

two relations in the given RDF Graph, it's convenient to write sparql query to obtain the player and the team entities in the graph. The sparql query is stored in "src/q2.rq".

Likewise, via *fb:soccer.foot_ball_team_management_tenure.manager* and *fb:soccer.foot_ball_team_management_tenure.team* the two relations in the given RDF Graph, we can obtain the relation between the teams and their corresponding coaches in the graph. The sparql query is stored in "src/coach.rq".

The case of identifying entities of stadium will be a little bit different, because there is no explicit relation involved with stadium. As a result, we need to extract them from the corpus in the graph. First, we get the entities with a document and their corresponding documents via the relation *c690:hasDocument*. The sparql query is stored in "src/withDoc.rq". Then, we can use the program in the assignment 1 to identify stadium entities.

The source code can be found in "src/prepare.py". The results are stored in files: "output/player.txt", "output/team.txt", "output/coach.tsv" and "output/stadium.tsv".

4. Extract Necessary Relations from DBpedia

After a little testing, it's not hard to find that there is only a small portion of entities that have a textual description. As a result, we cannot get all the necessary information just from the given graph, for example the player's nationality. However, the graph gives us real ids to the Freebase, and we probably can leverage the information on Freebase. Unfortunately, it seems that there is no available API to access Freebase. So we need to seek for another approach.

On the page of DBpedia, we find there is a mapping from DBpedia to Freebase through the predicate *owl:sameAs*. In addition, we can query DBpedia conveniently. Then, we can extract the nationality of players and the teams' stadiums from the DBpedia easily.

The source code can be found in "src/player2nation.py" and "src/team2stadium.py". The results are stored in "output/nation.tsv" and "output/team2stadium.tsv".

5. Answer the Questions

After above process, we can answer the given questions now. But before that we need to add some necessary tuples to the existing RDF graph. The followings are what I add:

- ?player *fbk:nation* ?nation
- ?team *fbk:player_list* ?player
- ?team *fbk:coached_by* ?coach

All the information required in this task is extracted in the previous section.

- Q1: Which stadiums are used by which clubs?
  - Already extracted in "team2stadium.tsv". Just need to match the stadium entities in DBpedia to the given graph.
- Q2: Who plays for which teams?
  - Easy to write query to obtain the results.
  - The query can be found in "q2.rq".
- Q3: Who coaches a team with Spanish players?
  - Leveraging the *fbk:nation* relation extracted from DBpedia, we can answer the question with very simple sparql query.
  - The query can be found in "q3.rq".
- Q4: Which clubs have stadiums named after former presidents?
  - There is no direct relation about where the stadiums' names come from on DBpedia.
  - We try to extract this information from the stadiums' document in the graph. The methods is to find the following patterns in the document:
    - … named … after/in honour of … president/chairman …
  - If we can find such a pattern in a single sentence of the document, we consider this stadium is named after the club's former presidents.
- Q5: Which teams have the most nationalities amongst their roster?
  - The sparql query is a little bit more complicated to answer this question. In this case, we need to use nested selected along with GROUP BY, ORDER BY and COUNT.
  - The query can be found in "q5.rq".

The source code can be found in "src/main.py". Also, there are five scripts to answer each questions separately: "src/q1.py", "src/q2.py", "src/q3.py", "src/q4.py", "src/q5.py".

6. Issues Encountered

- The Freebase cannot be accessed easily. As an alternative, use DBpedia.
- There is no direct relation about where the stadiums' names come from on DBpedia. So we need to extract this relation from the graph itself.
- How to write a sparql query to obtain the count of distinct nationalities in a team's roster. After some search on Google, I learn how to use nested select and aggregate functions to solve this problem.