Issue Report

Issue Link: https://github.com/RDFLib/rdflib/issues/949

Description:

Issue was regarding the semantical equivalence of the two queries. These following example queries are mentioned in the given issue.

```
SELECT ?x WHERE { ?x foaf:haha ?k .\n ?person foaf:knows ?x .}
SELECT ?s WHERE { ?person foaf:knows ?s .\n ?s foaf:haha ?k .}
```

Solution For the given issue: We did some enhancements in the library in which we added a module named as queryDetails.py. To use this file first import as:

```
import rdflib.plugins.sparql.queryDetails
```

In this module we implemented some methods to solve the above issues as following:

1. **printFormalizedQuery ()**: It takes an argument of rdflib.plugins.sparql.prepareQuery and print the formatted query. We used library's basic function to print the formatted query.

How to use above method to resolve this issue: By using above function we can print both the query and by observing these queries we can find whether each query is equal or not but this solution takes human intervention which is very time consuming when there are so many queries. So we need an automated solution to handle this situation which can prevent human resources.

2. **QueryEquivalenceScore()**: This function takes two arguments of the rdflib.plugins.sparql.prepareQuery and calculates semantical similarity score. Range of this score is between the range of [0-1].

How to use above method to resolve this issue: After providing two queries it gives score. To get similarity, we can use this score as following:

a). Score=1: Then both queries are equivalent.

b). 0<Score<1: Then some similarity is existing in between the given queries. And after observing the score we can analyze how much similar are given queries. More the score means more the similar queries.

c). Score=0: Zero score shows that both queries are not similar at all.

Methodology of QueryEquivalenceScore() function: This function takes 2 query arguments of rdflib.plugins.sparql.prepareQuery type. Then we generated the query algebra From this query algebra content we filtered the query content by removing different variable names and other noise. Because these variable names can't help to find semantical similarity means if we have different variable names, then also queries can be semantically similar.

After finding this filtered content of both of the input queries we made two lists of it. Now we took all the distinct terms of both of the queries and by using these we made two vectors in which I stored the frequency of distinct words with respect to each query.

Now I length normalize both of the vector and take dot product of it by using following formula:

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Here A and B are two vectors.

This formula gives scaler value for the similarity. But there is problem that all the SPARQL query algebra has some terms always common. Because of this we always got score more than 0.7 irrespective of whether query has similarity or not so we rescaled this score in the range of 0.7-1.0 by using min-max normalization by using following formula.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Here Xmin is the minimum and Xmax is maximum value of score. By using those we are normalizing our score X.

Conclusion: By manually checking is the costly and time-consuming task and it takes a lot of human time if query is huge or a lot of queries are there. By using the above approach this process becomes fast and a lot of resources can be prevented. For the validity of the answer if two queries are exactly similar it always gives correct result as score 1. But by having different queries it gives appropriate scores and we can easily find how much similarity in the input queries.