Individual Assignment 2018

Write a short report on the tasks below. You will present your essay in the oral exam (ca. 20 min).

Max length: 2000 words + ontology/formulas/queries/results

Deadline: 26 November 2018, 07.00 AM

Delivery: E-mail to the course teacher, helper and examiner.

Task 1 RDF Turtle Ontology

Build an RDF(S) ontology for live concerts.

These concerts are given by particular artists/bands from different genres, held at particular locations at particular times. The same repertoire may be used in different concerts at different times and locations, constituting a concert tour. Each artist is described by some properties. Genres are organized into a genre hierarchy. The booking agent may require that the artists play particular songs. Explain the assumptions underlying your ontological choices.

Populate the ontology with 10 concerts (minimum 5 artists, minimum 2 repertoires per artist).

Use the Turtle notation.

Task 2 Querying Ontology

Formulate a SPARQL query that lists all concerts of a given genre and its subgenres that have not yet been held.

Use SPARQL to count and list the songs that two artists share.

We want to know more about the artists than what is included in your ontology. Show how you in SPARQL can combine your ontology with DBpedia to list the birth dates and possibly the children of the artists you added into your ontology.

How can you use SPARQL to find the most similar – as you define it – artist to a given artist.

Task 3 Description Logic

Rephrase/Recreate the ontology in Task 1 in Description Logic (e.g. OWL 2 DL). Explain any additional assumptions you need to make.

Give examples of DL constructions in the ontology that are useful in the ontology, but impossible to include in the RDF(S) ontology. Demonstrate how reasoning works in your ontology.

Task 4 Formalism

Define the Open World and the Closed World assumption. Use your ontology from Task 1 to demonstrate the consequences of the Open World assumption.

Q & A

1. **Hvor står det om Turtle Syntax?**
    a. https://www.w3.org/TR/owl2-primer/. Rull ned til rett før "kap. 2 What is OWL 2". Der kan du velge hvilke syntaks som vises på alle eksemplene på siden.
2. **Is it OK to use OWL?**
    a. Not when it says explicitly that RDF(S) should be used.
3. **Its been a while since my last essay. Any thoughts or visions on how we should write the essay? What should it include and is there any examples on these kind of essays out there?**
    a. Just answer all the questions/tasks one by one, and add your models etc. as an Appendix. We don't have any example essay to copy from, but on the other hand we don't expect it to look like a standard research paper either. Find your own style.
4. **Are we supposed to use Protégé, or create the ontology in turtle by hand? If both are allowed, what is recommended?**
    a. We recommend Protege. You can find the SPARQL-query tab on "Window -> Tabs -> SPARQL Query"
5. **The second line in task 2: "Use SPARQL to count and list the songs that two artists share." - What if I don't have artists that share songs in my ontology?**
    a. Then your SPARQL-query should return 0 and an empty list, until two artists that sing the same song are added.
6. **Task 3 says to recreate the ontology in DL, then point out the constructions in the DL, which are impossible in RDF(S). However, this doesn't make sense since the DL was created *from* the RDF(S) to begin with? Have I misinterpreted the question? Can you rephrase it?**
    a. I found a straightforward explanation on StackOverflow: http://stackoverflow.com/questions/1740341/what-is-the-difference-between-rdf-and-owl
    b. Rephrase: Create the ontology from scratch (in OWL DL), then ...
7. **Is it okay to populate the ontology with two/three actual artists with actual songs and the rest with something as simple as "Artist 1 (Song1, Song2)" etc?**
    a. I think it's OK, but I also think it's more interesting with slightly less generic names.

8. **How are we supposed to submit the rdfs, attach the file or put the code in the pdf we are sending?**
    a. I think you can do both, but if the file is very big, it's better to just keep it as a separate attachment (or zip them together).
9. **Protege is using OWL for NamedIndividuals, Classes etc, has anyone figured out how to make it use RDF and RDFS only?** (Because the same tags are available in RDFS and RDF).
10. **For Task 3, do you mean OWL DL or OWL 2 DL?**
    a. I was thinking about OWL 2 DL.
11. **Hvordan tenker du/dere at eksamen i morgen blir lagt opp? Det er meningen av vi skal presentere essayet vi har skrevet?**

    Får vi ha med hjelpemidler? Type notater eller selve essayet? Burde man lage en presentasjon?

    a. Det blir en presentasjon av essayet deres.
    b. Vi ønsker at alle tar med en utskrift av essayet sitt, men forventer ikke noen annen presentasjon enn muntlig og whiteboard.
12. **Regarding second part of Task 2, i cant seem to get any results querying DBpedia from Protege. Tips anyone?**
    a. Workaround: Use Apache Jena Fuseki to run SPARQL queries. Only need to upload the data set files (.rdf / .ttl) to a local server. Explained in this video https://www.youtube.com/watch?v=JZp70uFsZS0 .
13. **Task 2: "How can you use SPARQL to find the most similar – as you define it – artist to a given artist." Which one is the correct interpretation, a or b?**
    a. We are allowed to post process the result from SPARQL queries to find the most similar. E.g. we run multiple SPARQL queries and use the result from these to rank the most similar
    b. We only use **one** SPARQL query, and from this one query we find the most similar
    Answer: The goal is b - One SPARQL query
14. **SPARQL and DBpedia in Task 2: Does anyone know how you can makes queries to DBpedia? I have tried the workaround in 12. a. , but it does not seem to get any DBpedia endpoints. It works with importing the .ttl file, and querying directly to it, but only not with DBpedia. All suggestions are much appreciated!**
    a. Use a sparql SERVICE statement to access the dbpedia endpoint (<http://dbpedia.org/sparql>).
    b. https://www.youtube.com/watch?v=12YebZvhf-w&t=40s This approach worked for me: Select { where { service <http://DBpedia....> { select {where { statements