COMP 4061 - L4 Summative Assignment SSA – Semantic Web

RDF and SPARQL

Objectives

The Semantic Web is both a theoretical and a practical web, which is partially already put in practice, and partially at the forefront of research. This Summative Assignment is giving you the opportunity to put some of the technologies and techniques you have been taught about in the (first part of the) (sub-)module into practice.

These represent also some of the most in-use techniques of the Semantic Web (after XML): RDF, as the basic data representation technology of the Semantic Web, and SPARQL, as the basic query language of the Semantic Web.

The additional purpose of this assignment is to give you a different view about the taught material, as well as a better understanding of how a query language for semi-structural material can be mapped onto concepts you have learned in structural databases, such as SQL.

Moreover, as the Semantic Web is an ever expanding area, with new W3C Recommendations appearing every year, and many of the technologies we are studying, being further improved upon, this is your chance to have a first encounter with the various groups performing Semantic Web research, discussions on the web, special interest groups (SIG) etc.

Finally, this assignment is thus to explore the power of the Semantic Web, and the way different databases across the world can, in principle, be queried and processed together, ultimately, in a semantically coherent way.

Task Specification: implementing joins in SPARQL over RDF resources

We have discussed different methods of using different RDF documents (graphs) in SPARQL, in the two parts of the lecture (SPARQL I and SPARQL II). Your task is as follows:

- Analyse the methods given to you at the lectures, do additional reading on the web, where necessary, and **implement at least two different types of join algorithms in SPARQL over RDF resources, to test them, and to report on the results**, in the wider context of the Semantic Web and its technologies, as well as in the context set by the above 'Objectives' section (power of the Semantic web, research fringe, mapping onto structural database querying, practical side of the implementation).
- The implementation can be based on what you saw in class, or your own invention. The code you will be producing must, however, be your own code.
- Think of different types of joins: inner, outer, etc. and how these might apply to SPARQL as well as think of using different formats, such as FROM, GRAPH.
- Explain the join mechanism you have used, as well as to give some justification to your choice.
- Compare the different join approaches.
- Give some measures of performance of your queries; this will help in answering the aim of justification, above, and possibly address some of the comparison for the join algorithms. In the following, there are some examples of measures; however, you can implement and report on all or any these, or others of your choice: How optimal is your implementation? (in terms of query optimisation); What is the wall-clock running time on test data?; How does it scale if you join more documents? Does it time out?; How does it compare with a similar query in SQL?

In terms of the database used, your algorithms could use the same RDF database, or different databases. You could create your own RDF files and query them, or query resources on the web (such as DBpedia: http://wiki.dbpedia.org/). Please note, however, that if you decide to use DBpedia, or any

other online resource, you will need to ensure to have backups of your queries and retrieved data, as many of these live resources are not always up. A good alternative is to download a portion of the database locally, and have it handy when the online one is not available.

In terms of creating useful code, you can think of using output formats, such as (X)HTML, which could potentially be displayed to a user.

In terms of coding, the only requirements are to use SPARQL and RDF. However, you may wish to use your SPARQL as calls from a library in a different language, such as Python, Java, etc., if that makes more sense to your application.

There is a plethora of information on the web about the Semantic Web. Make good use of discussions and papers on the web etc. When using papers, use Google Scholar (https:/scholar.google.co.uk/) to search for them. Check papers for recency and quality (as additional help, for journals, impact factor can be useful; for both journals and papers, you can use the Core ranking: conferences (http://portal.core.edu.au/conf-ranks/) and journals (http://portal.core.edu.au/jnl-ranks/). Please note however that there is no consensus about impact factors and rankings, and you need to be critical and not just simply trust them.

Generally speaking, discuss your choices, discuss the complexity of your implementation, explain and discuss the tools used for implementation, and the reasons for your choice.

Please note: Only your report will be marked. Why? Because the ability to communicate is vital in whatever you go on to do. Your aim is thus to clearly communicate your experiences. The report must be comprehensible to a non-subject matter expert. The report must be self-contained – the reader should not be expected to visit external websites, etc. Remember that plots may say more than words, so use graphs, examples etc. where possible.

Submission

You must submit the following:

- **A succinct project report in PDF format**: 3 pages (at most 1000 -1500 words), not including references and Annex with the code.
  - Your main constraint will be the page/ words limit. This should be including all figures, main code samples in the 3 pages. You can have as many references as you like, as long as they are referred to in the main text (having references which are not referred to is bad practice, and may lose you points).
  - Your report should have at least 10pt font, as well as “sensible margins” (2cm on all sides).
  - A separate Annex of the full code you have created. This will not be marked, but may be checked if necessary.

Outline format

- Title, name, ID;
- Abstract: executive summary
  - In 200-250 words summarise your project, including algorithms compared, measuring criteria, and the main results;
- Introduction
  - set the scene, provide an overview, giving the motivation behind your work and choices
  - give an overall description of what you did, as well as your main contributions
  - explain how you are comparing algorithms, and how you are measuring the comparison; justify your choices (e.g., based on references)
- Methodology: Description of what you did
  - Describe the algorithms and implementation details
  - Give enough detail to allow someone else to repeat your process
  - Include what measures you will measure the methods on
  - Explain your choices
- Results of the comparison and discussion
  - Work through experimental or analytical comparisons
  - If you include plots/tables, explain each one in the text
- Conclusions/References
  - Include proper references to work you made use of
  - What more could you have done if you had more time (future steps)?
  - What mistakes would you avoid if you did it again?

Evaluation criteria

- Title and Abstract [10%]
  - Is the title appropriate?
  - Is there a clear brief overview in the executive summary explaining what you did, what algorithms you compared, how you are measuring them and what the main results are?
  - Is the abstract of appropriate length?
  - Does it incite interest in the reader and make the reader want to read on?
- Introduction [10%]
  - Did introduction set the scene, and provide overview?
  - Is the motivation clear and the justification appropriate?
  - Is there a clear overall description of what you did, what your main contributions are? Does this description make sense (is it fit for purpose)?
  - Are you comparing algorithms, and is it clear what, why and how you are measuring?
- Methodology [15%]
  - Are the descriptions of the algorithms clear? Is the selection of algorithms well explained? Are there enough details for someone else to repeat the process?
  - Are the measures for the comparison of algorithms well justified, as well as explained?
- Results and discussion/interpretation [50%]
  - Were the results well explained? Were they correct?
  - Did you do a good job of measuring/comparing?
  - Is there good explanation/interpretation of what you found (discussion)?
  - Is there something new/original/interesting in your work?
  - Did you properly consider further steps?
- Quality of Presentation [15%]
  - Was the report well-presented, easy to read, clear? Was the structure appropriate for the purpose? (i.e., should there have been more/less tables/images/text/code etc.)
  - Did the report include appropriate references, and was referencing well used in the text to argue the points made? Were there any extraneous (unnecessary) references?