COMP38211: Documents and Data on the Web: Lab Manual - 2

This document describes the laboratory work that is to be carried out in the latter part of the semester. The laboratory focuses on supporting functionality associated with “Data on the Web” by investigating how to index RDF documents. In carrying out the laboratory exercise, you will refine your understanding of techniques and issues relating to search from “Documents on the Web”, and obtain some additional experience using MapReduce.

There is a single exercise in this laboratory, with several deliverables, for which support is available during 4 Laboratory Sessions in the normal timetabled slots for the unit (in which there will be academics and Teaching Assistants present). This lab requires some thought to explore a design space, so please make the most of these lab sessions. The laboratory sessions are as follows:

Date	Activity
27th November	1 hour lab session: Exercise 2.1: Analysis of given solution
4th December	1 hour lab session: Exercise 2.2: Design your own solution
11th December	1 hour lab session: Exercise 2.3: Implementation of your solution
18th December	3 hour lab session: Finalising the lab; working on report

Lab 2 Specification

Indexing Data on the Web

Aim

To provide students with an opportunity to explore the opportunities presented for searching using the information available in RDF documents, by designing an index for RDF data, implementing that index using MapReduce, and analyzing the strengths and weaknesses of the result.

Learning Outcomes

A student who successfully completes this exercise will be able to

- Access RDF documents from a program.
- Make decisions relating to the design of an index for Data on the Web.
- Implement a MapReduce program for populating such an index.
- Evaluate the impact of design decisions on the implementation and applicability of the index.

Note that although this is a laboratory, this is not primarily an implementation activity – the exploration of the design space and evaluation of the result are key features.

Getting Started

You should carry out the following steps for this lab:

1. Following the approach from Laboratory 1, use Eclipse to compile and run the supplied RDF MapReduce index building program by following the instructions in: *How To Compile and Run Hadoop Programs in Eclipse*. The RDF for this laboratory is available on Blackboard as Exercise2.zip. If you would rather not use Eclipse, you can compile and run using Ant directly, as described in: *How To Build and Run Projects Using Ant*, or run the relevant Ant scripts from another IDE (though we may not be able to help you much with that).
2. Use Eclipse to browse the source of the program, and in the file system look at the input and the output. The program uses Jena to access RDF data; the online tutorial on Jena should provide the information required to understand what is going on: http://jena.apache.org/tutorials/rdf_api.html.

The following steps outline a waterfall-style approach of analysis, design and implementation. Of course, a more iterative approach may well be appropriate. Note, however, that it does seem important to have carried out a systematic analysis of the existing approach before starting on the (perhaps more iterative) design and implementation phases.

Exercise 2.1: Analysis

You have been supplied with a preliminary implementation of an index building program for supporting searches over RDF data. The program indexes some RDF data on the Simpsons; you should study this data in the input folder.

As in documents on the web, the idea is that, given terms extracted from the (in this case RDF) document, you should create an index of the form (*term* $\rightarrow$ *resource*). In an index for documents on the web, the *term* is a word from the document and the *resource* is the URL of the web page. In the lab for documents on the web, it was fairly obvious what the terms should be. However, when indexing RDF, you have to ask yourself “what would be suitable terms”.

Answer the following questions to yourself in relation to the index provided. Note that the index manifests several design decisions that might be considered to be rather controversial.

1. What are the main design decisions reflected in this program? Specifically, given that the index is of the form (*term* $\rightarrow$ *resource*), how was the term obtained from the RDF, and what is the resource?
2. What searches would the index support? What does the user need to provide as input to a search over the index? How easy is it for the user to provide this information?
3. What searches would the index not support? You might like to ask yourself what you think someone might want to search for in an index of Simpsons episodes. Likely you will find that at least some of these are not directly supported by the index.
4. How useful is the provided index?

Exercise 2.2: Design

Design an index for the Simpsons data set. Designing the index involves addressing questions such as the following.

1. What types of question are to be supported, and why? In general terms, your index will be of the form (*term* → *resource*), but you need to decide what the terms are like, and what you should store about the resource. As you are indexing RDF, which is structured, one question is: can the index exploit the structure of the RDF in some way. In the given index, there is an attempt to exploit the structure, using predictate and object information together to create index terms. The way in which this was done created some problems, but you should ask yourself if structural information can be exploited in the search.
2. What terms are to be indexed, and why? As you are indexing RDF, information from the term must come from one or more of the <subject, predicate, object> components of RDF triples. Here you should ask yourself questions like: which of these components might provide suitable terms? How should terms be obtained from URIs?
3. What is stored in each index entry, and why? In an index of the form (*term* → *resource*), it is sometimes useful to store more than just the identifier of the resource. What extra information (if any) might it be useful to store in an index for RDF search? What information is needed to support the types of question identified at (1).

All of these questions have multiple plausible (and no doubt multiple rather implausible) answers. As such, you should think carefully about your design before committing it to code.

Exercise 2.3: Implement

Implement your design from Exercise 2.2. Of course, you can follow an iterative approach with a growing design and implementation, but at least the basics of the design should be clear before you start on the implementation. Overall, having completed the Documents on the Web coursework, the implementation here should not be especially challenging; for this coursework, the bulk of the challenge is in the design.

If you choose, you could develop an implementation that can build indexes in which decisions configurable, to allow measurements to be taken with different design decisions.

Deliverables

You should submit the following in an archive (i.e., a zip file): (1) your source code including your RDFInvertedIndex.java program, and (2) a Word or PDF file that includes:

1. A report of not more than 1000 words that discusses questions such as the following. Note that quite a few marks require you to go beyond bookwork. What are the specific features of *your* implementation, how does it do with The Simpsons test data set, etc. These questions can be used to structure your report.
    a. What searches can be performed successfully, and what design decisions were made to support them?
    b. What terms have been indexed, and why?
    c. What is stored in the index, and why?

    In discussing questions such as these, you might consider topics such as:

   i. The strengths and weaknesses of the index in terms of functionality and/or search effectiveness (e.g. what searches are not supported).
   ii. Any benefits (or difficulties) that derive from searching over RDF compared with web pages.
   iii. Improvements that might be achieved using information from the Web of Data that is not available in the test data sets on The Simpsons.
   iv. The performance of the index, in terms of processing costs and size.
2. A (not necessarily contiguous) sample of the output of your program that occupies not more than 50 lines, and that includes complete index entries for terms that illustrate well the properties of your index. You can refer to this sample from (1).

Assessment

This exercise is worth 40% of the overall mark for this course unit. Credit will be given taking into account: (i) the amount of functionality supported; (ii) the elegance and effectiveness of the design and implementation; and especially (iii) the clarity and depth of insight in the report. The majority of the marks will be assigned on the basis of the report.

A first class level submission will provide comprehensive functionality, implemented in an elegant way, with an insightful discussion of the strengths and weaknesses of decisions made.

An upper second level submission will provide significant functionality, with few design issues, implemented in a generally appropriate way, with a reasonable discussion of the strengths and weaknesses of the features supported.

A lower second level submission will provide reasonable functionality, but may well include some implausible design decisions, implemented in a way that provides significant opportunities for improvement, with a rather superficial discussion of the features supported.

Below lower second class honours, there is likely to be limited functionality, which likely contains design errors, with significant weaknesses in the implementation, and a discussion that rarely goes beyond bookwork.

Note that this lab is intended to complement the first one, so here you are not expected to focus on topics such as stemming or design patterns that were covered in the first lab unless you have something specific that relates to the Web of Data context.

Submission Deadline: **Friday 15th January at 18:00**.

This is a hard deadline; extensions will only be granted as a result of formally processed Mitigating Circumstances (http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=4272). Marks for late submissions will be reduced in line with the university's rather punitive policy (http://documents.manchester.ac.uk/display.aspx?DocID=24561).

Submission Guidelines

The deliverables should be submitted using Blackboard.