The University Of Sheffield.

COM3504/COM6504 Intelligent Web

Assignment 2015-2016

This assignment is primarily concerned with applying the ideas that are being presented in the module on methods for accessing the Web and making sense of its content. Some of the basic algorithms needed for it have already been introduced during the module, and can be obtained from MOLE.

Organisation of the Assignment

The assignment is divided into two parts:

1. **Part 1**: Building a system to query twitter.com controlled via a Web interface.
    a. No marks: after feedback the revised Part 1 will have to be submitted as part of the Final Assignment. Penalty for not handing in a reasonable solution: up to 25 marks taken off the final marks.
    b. Deadline: **Monday 11.4.2016 at 23:59:59**
2. **Final Assignment**: tbd

All submissions are electronic via MOLE.

Workload

The whole assignment is intended to account for between 20 and 30 hours of each person's work towards the module as a whole. Experience has demonstrated that it is not really possible for one person to do all the work for the assignment in that time, unless they are an exceptionally competent programmer with previous knowledge of Web programming. Therefore, the assignment is organised on the basis that people should work on it in **groups**. Groups must be composed of a **maximum of 3 members.** Students are allowed to do the assignment in pairs or on their own.

Please note!
You can only have groups within the same module code, i.e. members must be either all in COM3504 or in COM6504.
PhD students should ask the lecturer before joining a group.
You must register your group at
https://drive.google.com/open?id=1hZ2VRFcGg55Z7icDs8Xqml5c48Bm9UIgNsV7R7D1Emg&authuser=0

Given a group, you will normally be expected to keep with that group for the whole of the assignment. However, if any problem arises that makes this difficult, you should notify the lecturer **immediately**, so that appropriate action can be taken. See lecture 1 slides on this topic.
It is important that you make clear what contribution each group member has made. It is required that each person contributes to each stage of the assignment, but it is up to each group to decide how to divide the work up between individuals. This must be explicitly stated in the report, as explained below.

Deadlines

The deadline is **absolutely fixed**. You should therefore plan your work to aim at handing the report in at least a few days before the deadline – do not leave it until the deadline, just in case any minor thing goes wrong and you then find that you are late.

Note that the Computer Science department applies fairly severe penalties for handing coursework in late. If you want to look at the details you can find them on the University's web site.

Material Provided

Lecture notes, lab class examples and websites/books to use.
NOTE: no third party code can be used in the assignment, except what has been **explicitly** provided in the lectures. For example you are allowed to use code given in the lecture slides but you are not allowed to download any code from the Web or to use any other software that will perform a considerable part of the assignment. **Unauthorised re-use of third party software will be considered plagiarism.** In case of doubt ask the lecturer for permission before using any third party code. Libraries allowed despite not being mentioned in the lecture notes are: JQuery, Angular, Bootstrap, Socket.io. For other libraries, please ask before using.

1. Scenario

The goal of the first part of the assignment is to build a program able to query the social Web through the available APIs to track people's movements and the topic of their discussions. The field chosen is Football.

1.1. Querying the social Web (40% of marks)

You must build a program that enables querying social media (at least Twitter.com –additional marks can be awarded for additional sources) through their APIs using a browser interface and one (or more) node.js server(s). The goal is to perform the following queries:

1. Tracking discussions: track discussions concerning a football team using the SEARCH API. For example following discussions about Manchester United on Twitter will require tracking/searching
   i. @ManUtd (as an author and as a retweeted author and/or a team mentioned in a message (tweet), AND/OR
   ii. their players (using their Twitter screen name e.g. @WayneRooney),
   iii. AND/OR some related hashtags (e.g. #ManUtdVArsenal) AND/OR
   iv. specific keywords.

   The interface must allow the users to choose how to track the discussion about the team (i.e. the interface should allow setting the query in a flexible way). All the terms in a query must be either in OR or in AND (i.e. it is not necessary to allow queries containing both ANDs and ORs). When the search API is used, make sure to return at least 300 tweets (if available). For each message, provide
      i. the link to the original message and
      ii. a link to the author's page
      iii. time, date, any original author (if retweeted)

   Example:

   Input:

   QUERY: autor:@maunutd ***OR*** *mention::@maunutd* ***OR*** *author:*@WayneRooney **OR** *mention:*@WayneRooney **OR** tag:#ManUtdVArsenal

   Output:

   - *Author:* ***Lizzie Cundy*** *@lizziecundy date: 16:02 Sun 28 Feb 2016*
     Tweet: "What a game !! And King Louis is the best at falling over !! " ***#ManUtdvArsenal*** *#football*
   - *Author:* ***Wayne Rooney*** *@WayneRooney 16:03 Sun 28 Feb 2016*
     Retweet: @McIlroyRory
     Tweet: "Both top fighters mate but @scottquigg for me. How about the loser pays the bill next time your in Manchester?"
     …

2. Frequency: given the tweets retrieved in 1), display:
   i. the most frequent 20 words contained in the collection.

   Example:

   Losers: 50
   ManU: 30
   Rooney: 12
   …

   ii. the 10 most active users and the main keywords associated with them

   Example:

< profile picture> @WayneRooney 25 tweets - most frequent words: pal(100), mate(100), ball(11), player(8)…
<profile picture> @lizziecundy tweets: 12 - most frequent words: game(12), what(11), gorgeous(8) …

3. The locations: what areas are involved in the discussion? For each query, show all the geo-located tweets on a map.

1.2. Storing information (15% of marks)

Information retrieved by the queries above must be stored in a Web accessible database (e.g. MySQL). For complex objects (e.g. photos), storing a link is fine. The design of the database schema is left to the student. Please note that the quality of the database schema carries marks.
The database must be accessible using the same Web interface used for direct connection to Twitter (and hence the same queries must be possible).
It must be possible to use the database

- as a source of data per se (i.e. a source that can be queried on its own – put a tick box in the interface saying "query database only") OR
- as a cache to reduce the number of queries sent to the Twitter API. This means that in case of repeated queries, the database must provide the data already made available by the past queries and the Twitter API must be used to retrieve only the additional data that should have become available since the previous queries were issued.
  Example:
  Query1, sent at 10.45 on the 23rd of February: "author=@WayneRooney"
  Result returned by Twitter API: 25 tweets
  New tweets stored into the database: 25
  Query2, sent at 2pm on the 28th of February: "author=@WayneRooney"
  Results returned by the database: 25 tweets
  Results returned by the Twitter API: 3 tweets
  these 3 tweets were issued between the at 10.45 on the 23rd of February and 2pm on the 28th of February
  New tweets stored into the database: 3

In your report you are requested to explicitly discuss the details of the interaction between the database queries and the live queries to Twitter, etc. In particular you must explain in what ways you are using the database to save Twitter calls.

1.3. Web Interface (10% of marks)

The system must be implemented as an HTML/Javascript set of files (for the interface).
The program will be controlled through a browser-based interface. The interface must enable the user to perform the queries described above. This interface must be implemented in an HTML/Javascript file called `queryInterface.html`. The interface must be served by a node.js server. All the interaction with the social web must also be done through a node.js server.

1.4. Quality of the solution (10% of marks)

The assignment per se is quite simple. However it is very important that you focus on the quality of the solution. For example, while a simple architecture calling the node.js server directly from the HTML form via POST would be ok, a far better solution (i.e. attracting far more marks) would e.g. use:

- Javascript to check on the client that the input is properly provided;
- Ajax to leave the client active while the server retrieves the answer;
- a JSON-based exchange of information between client and server;

These are just examples. The quality of the solution will attract more or less marks. I suggest that

you start with a simple solution, check that it works and then add more sophistication.

1.5. Additional features (25% of marks)

Additional features will give you additional marks: for example you could add pictures of teams, people or places taken from Google or from Flickr, list of games retrieved from the Web, etc. Or you could add the capability to perform the same queries using the Streaming API.
Another possibility would be to create mixed queries: what users have been discussing about in the last *X>1* days in a radius of *Y* km around a specific geographic point: this will require querying geo-tagged messages (e.g. tweets) and extract the contained keywords as per 2) above.

2.Marking schema

Each part described in subsections 1.1-1.5 will carry marks divided as follows:

36% for the documentation in the project report. More details on the detailed marking schema can be found in appendix A.

39% for the implementation of your solution (quality of code and Javadoc-like documentation). See appendix B for further details on the specific marking schema.

25% for the correctness of results.

Please note

- the direct consequence of the marking schema is that providing a program returning the correct solution is not enough to get a pass mark. You will need to implement the correct strategies and document/discuss them properly! Quality of documentation and code are very important issues for computer scientists.
- the weight assigned to section 1.5 makes so that getting marks over 75% requires to be creative and go beyond what strictly required by the assignment: we expect that only exceptional solutions will get marks above that threshold.

3. Handing in

Your solution must be contained in a self-contained directory called *IntelligentWeb* (`<MainDirectory>` in the following) compressed into a zip file submitted through MOLE.
The directory must contain:

1. The code of the solution (please note that we will both inspect and run the code).
   (a) All code must be **developed in HTML /Javascript**. We must be able to run your solution without problems on a standard lab machine. It should not need any application to be installed in order to run. If in doubt about using a library or not, be sure to ask the lecturer for instructions.
   (b) You are required to use node.js and to make sure that your solution runs on the lab computers.
   (c) All the external libraries must be included in your solution
   (d) Source code: please provide:
      1. All the code should be in the directory **`<MainDirectory>/solution.`** Please note that the quality of the code carries a relevant portion of marks, so be sure to write it properly.
      2. The database schema (e.g. obtained with mysqldump --no-data) must be stored in **`<MainDirectory>/databaseschema`**
2. A report documenting your work. The report must be contained in the directory **`<MainDirectory>/report/.`** An outline and set of requirements for the report are provided in Appendix A.

3. The documentation in the Javascript/HTML file must be of very high quality. We expect the same type of quality that would enable generating a complete Javadoc documentation from a Java file. Please note that this documentation carries a relevant portion of marks, so be sure to write it properly. For more information on guidelines for this type of documentation see http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html
4. An example of results of 3 queries for each query in 1. must be stored in **`<MainDirectory>/queryexamples`**
5. The filled self assessment form.

3.1. How to submit

Everything must be submitted electronically. Nothing is to be handed in at the reception! **Use MOLE**. Store your solution in a .ZIP file that when unzipped will generate the directory organisation described above. As emergency measure (and only in that case!), if any last minute issue should arise in handing in electronically, please send your solution by email to the lecturer (cc to demonstrators) in a self contained .ZIP file. http://wit.shef.ac.uk:8080/
Make sure never to leave your solution in the public_html (or any other public directory) as it could be copied by others. Solutions not working on the departmental computers (e.g. working only on personal computers) will not be considered.

3.2. Anti-cheat measures

Please note that measures are in place for detecting plagiarism and in general cheating.

4. Queries about this assignment?

Should you have any queries about the assignment, feel free to contact

Fabio Ciravegna, Jie Gao, Suvodeep Mazumdar:
{fabio, j.gao, s.mazumdar}@ shef.ac.uk

Appendix A: Report Outline and Marking Schema

It is important that you produce a high quality report. Be sure not to leave the report at the very last minute. It is importance that you do not ignore techniques and examples provided to you during the lectures and lab classes. Referring back to them during your planning/implementation stages will give you the base from where to start, as well as a point of comparison/discussion where your ideas differ from what already presented to you (i.e., do not reinvent the wheel – if it has been done before, reuse it and complement it where it lacks functionality).
You may use diagrams to aid your explanation in these sections, but please note that a diagram alone is not acceptable and must be accompanied by an explanation/discussion.

Your report must be organised according to the following outline, which is designed to help you ensuring that all the required information is provided. **Length: maximum 4 pages.**

Index Table

[No marks – OPTIONAL] - You may or may not want to include an index table on your report. This is completely up to you. This does not count against the page limit.

Introduction

[no marks] - Introduce the project/assignment and the different tasks that you have performed and which you have not. This should show that you understand what is to be done, and give the marker a guide as to what to expect from your report. Please make the marker's work easier by being honest.

For each task in section 1

Create a subsection in your report for each of the requirements in the sections 1.1-1.5 above. It is very important that they are documented individually by explicitly following the organisation below.

- **Issues**

[10% of the report marks for this section] **-** Introduce the task this section refers to and the challenges that you were faced with.

- **Design choices**

Design and its Motivations: [55% of the report marks for this section] explain how your solution works and explain why you chose to design your solution in this particular way? Does it have advantages/disadvantages over other design choices? Here is where we expect you to explain and motivate e.g. what social media API feature you use for each type of query and if you query the database or the live stream.

Requirements: [20% of the report marks for this section] how does this design comply with the requirements specified in the original assignment sheet? Are you meeting all requirements?
Limitations: [15% of the report marks for this section] have you thought about exceptional situations that may limit your solution? Is your solution extensible? Can it be easily adapted for other requirements? Remember that no design is flawless!

Conclusions

[no marks] - You should include here any relevant conclusions you've collectively arrived at regarding the process of designing the solution for this part of the assignment as well as any lessons learned.

Division of Work

[No marks – MANDATORY] – Have all the group members shared the workload in a balanced way? In what way? E.g. what has each member provided – be precise!

Extra Information

[No marks – MANDATORY] – This section should include any extra details needed to run your code. If no extra configuration is needed, please explicitly say so in this section.

Bibliography

[no marks] **-** Do not forget to cite any sources and reference these within the text where appropriate (e.g., "(...) *we have used the techniques as per [1].*").Make sure to use a standard format style. The lecture notes must indicate the lecture number (e.g. week 1)

Appendix B: Marking schema for the Code

The quality of the code of your submission for each part of the assignment will account for quite a large part of the marks. It will be marked according to the following schema:

[65% of the code marks] – Code functionality: how the code meets the requirements set in the assignment description.

[5% of the code marks] - Code documentation. This includes

- in-line commenting to make your code intentions clearer to someone reading
- Javadoc-like documentation: higher level comments on the code and its use. It is important that you note the different level of detail generally included in Javadoc-like comments as compared to the kind of comments that are included within your code!

[15% of the code marks] - We must be able to run your code without any problems using the lab computers.

[5% of the code marks] - Your code should follow a consistent format regarding class and variable naming as well as indentation, this is can be easy achieved with the use of an IDE (Eclipse, IntelliJ, Netbeans, etc.).

[5% of the code marks] - Proper use and handling of Exceptions in your code.

[5% of the code marks] - Code organisation (e.g. readability, use of modules in node.js, Javascript files, etc.).