Individual Project ICS2205/2230
Web Intelligence

Charlie Abela, Joel Azzopardi

November 26, 2020

This document contains the details for the individual ICS 2205/2230 project which is marked out of 100%, however it is equivalent to 60% of the total mark for this unit. While discussions between individual students are considered as healthy, the final deliverable needs to be that produced by you and not plagiarised in any way.

The **deadline** for this project is **12:00pm Friday 29th January 2021**. Deliverables and attached plagiarism form must be uploaded on the VLE. Projects submitted late will be penalised or may not be accepted.

1 Specifications

This project consists of **TWO** tasks, with each task being assigned 50% of the mark for this project. You are to address **BOTH** tasks and will need to create a Jupyter[1] notebook for each task. The notebooks should also contain inline comments and markdown detailing important aspects of the code. These will be the main deliverables. In the following sections we provide the details for each of the two tasks.

1.1 Task 1: Graph Analytics

The aim of this task is to use the given Twitter dataset and to perform some analyses on the underlying *mention* graph (explained below). The dataset is presented in a raw format and as a *tab-separated-values* file and is a subset of a larger twitter dataset that was collected between June 2009 and December 2009. Each line contains a tweet and consists of three tab-separated fields. These include a timestamp, the alias of the user that sent the tweet and the tweet content itself.

This is a typical entry in the file:
2009-06-11 17:07:59 drew @jack Whens it coming to the iPhone? Or is it?.

[1] `https://jupyter.org/`

Users might *mention* other users in their tweets by using the @ symbol followed by the name of the other user, such as *@jack* in the example. Multiple other users might be mentioned. The *mention* graph is therefore a directed graph where the vertices are the users and the edges represent the mentions. Thus $userA - [mentions]- > userB$. The number of times that $userA$ mentions $userB$ can be transformed into a weight. You do not need to consider the time when creating the *mention* graph.

You need to do the following (marks are allocated to each part out of 100%):

1. Extract the mention graph from the Twitter data (**marks 20%**):
   - parse the *.tsv* file using Python to get the individual tweets;
   - generate the adjacency list for each user based on the alias. A user is adjacent to another user if he/she is mentioned in a tweet. Keep track of the number of times that some $userA$ mentions $userB$;
   - use the adjacency list to create the *mention* graph in NetworkX. Use the counts as weights on the edges;
   - in the jupyter notebook provide information about any challenges that you might have encountered when parsing and cleaning the data and when creating the graph, and how you solved them. Discuss as well any decisions that you might have made.
2. To start analysing the underlying structure of the graph, compute the following statistics for the graph (**marks 30%**):
   - number of nodes and edges;
   - indegree and outdegree;
   - degree distribution;
   - average path length;
   - global clustering coefficient.

   In the jupyter notebook provide information about any challenges that you might have encountered when computing these statistics, and how you solved them. Is the graph connected? Are there "giant" components[2] in the graph? What can you say about the graph based on the computed statistics?
3. Determine the top 10 users (**marks 30%**): based on the degree, closeness and betweenness centrality measures. You need to deal with directionality to facilitate computation. Discuss how you did this.

[2]A giant component is a connected component of a network that contains a significant proportion of the entire nodes in the network. Typically as the network expands the giant component will continue to have a significant fraction of the nodes.

4. Visualise the graph (or the most important components) (**marks 20%**): use some size-color-valuation scheme, whereby nodes with higher centrality (say betweenness) will have a specific color and size. The user's alias can be used as the node label and the edge width can represent the weight (based on mention counts).

This task is being allocated a total of **50 marks**.

1.2 Task 2: Information Retrieval

For this task, you are to use the *WES* dataset available from `http://pikes.fbk.eu/ke4ir` to build a simple Information Retrieval engine that uses the Vector Space model to find documents related to a user query.

You need to use the following files from the dataset:

- The document collection – `https://knowledgestore.fbk.eu/files/ke4ir/docs-raw-texts.zip`; and
- The set of queries – `https://knowledgestore.fbk.eu/files/ke4ir/queries-raw-texts.zip`.

Both of these files are being made available on the VLE.

An IR engine consists of TWO parts – the document indexing part, and the querying component.

For the document indexing part, you need to implement these process steps:

1. Parse the document to extract the data in the XML's $< raw >$ tag;
2. Tokenise the documents' content;
3. Perform case-folding, stop-word removal and stemming;
4. Build the term by document matrix containing the *TF.IDF* weight for each term within each document.

For the querying component, you need to implement the following process steps:

- Get a user query – note that it can be set within the notebook, directly into a variable named *query*;
- Preprocess the user query (tokenisation, casefolding, stop-word removal and stemming);
- Use cosine similarity to calculate the similarity between the query and each document returned in the previous step;
- Output the list of documents as a ranked list.

Note that you can use *NLTK* or any other library to help in the tokenisation and preprocessing of the text. However, you **need** to implement your own **TF.IDF** weighting and **Cosine Similarity** measures.

This task is being allocated **50 marks**.

The marks are distributed as follows:

- Indexing part:
  - Document parsing – **5 marks**;
  - Extraction of index terms – **10 marks**;
  - Weighting of index terms (term-by-document matrix) – **15 marks**;
- Querying part:
  - Query processing – **2 marks**;
  - Calculated similarity between the query and each document – **13 marks**;
  - Building and outputting the ranked list of relevant documents – **5 marks**.

2 Summary of deliverables

T1:	Graph Analysis	50 marks
T2:	Information Retrieval	50 marks

You will need to submit the notebooks (and any other relevant files/documents) as zipped files on the VLE.

3 Helpful resources

- Beginner's tutorial for Jupyter Notebook – `https://www.dataquest.io/blog/jupyter-notebook-tutorial/`
- Using NLTK – `http://www.nltk.org/book/`

4 Final Remarks

Final suggestion: if you have difficulties do not hesitate to contact us. Any issues, including technical difficulties should be identified and highlighted as early as possible to ensure timely resolution.

Good luck!!