COMP8210/COMP7210
Big Data Technologies

Assignment 2

Semester 2, 2023
Macquarie University, School of Computing

Due: Friday, 22 September 5pm
Total Mark: 100
Weighting: 25%

This Assessment Task relates to the following Learning Outcomes:

- Obtain a high level of technical competency in standard and advanced methods for big data technologies
- Understand the current status of and recognize future trends in big data technologies
- Develop competency with emerging big data technologies, applications, and tools

Background. Social data analytics have become a vital asset for organizations and governments. For example, over the last few years, governments started to extract knowledge and derive insights from vastly growing open/social data to personalize the advertisements in elections, improve government services, predict intelligence activities, as well as to improve national security and public health. A challenge in analyzing social data is to model the data in graph models and query the social data graph to understand the hidden insights. In this assignment you will explore Big Data Technologies for organizing and querying data graphs.

Resource: **Beheshti et al., "Social Data Analytics", ISBN 978-1-032-19627-5, CRC Press, 2022.**

Dataset. The Twitter dataset, including 10k tweets, is available on iLearn.

Twitter[1] serves many objects as JSON[2], including *Tweets and Users*. These objects all encapsulate core attributes that describe the object. Each Tweet has an author, a message, a unique ID, a timestamp of when it was posted, and sometimes geo metadata shared by the user. Each User has a Twitter name, an ID, a number of followers, and most often an account bio.

With each Tweet, Twitter generates 'entity' objects, which are arrays of common Tweet contents such as hashtags, mentions, media, and links. If there are links, the JSON payload can also provide metadata such as the fully unwound URL and the webpage's title and description.

So, in addition to the text content itself, a Tweet can have over 140 attributes associated with it. Let's start with an example Tweet:

Figure 1. A sample Tweet.

[1] https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json

[2] JSON is based on key-value pairs, with named attributes and associated values. These attributes, and their state, are used to describe objects.

The following JSON illustrates the structure[3] for these objects and *some* of their attributes:

Figure 2. Illustrating the structure of a Tweet using JSON.

Part 0. Data Curation (0%)

Write a program (in Python) to apply data cleaning/curation on the 10k Tweet Dataset. A sample 50 cleaned Tweets are provided to help you with this step.

Part 1. Initial Graph Data Model (30%)

Create a base graph data model for the 10k Tweet Dataset by importing the JSON data into Neo4j. The initial database should follow the model shown in Figure 3, however the assignment will require that we evolve this data model in stages.

Data can be imported to Neo4j using the APOC support for importing JSON. The process for this is similar to that shown in the lecture for CSV, each JSON document represents a single tweet as a map of values in Cypher. See appendix for an example and the Neo4j documentation.

[3] https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/overview

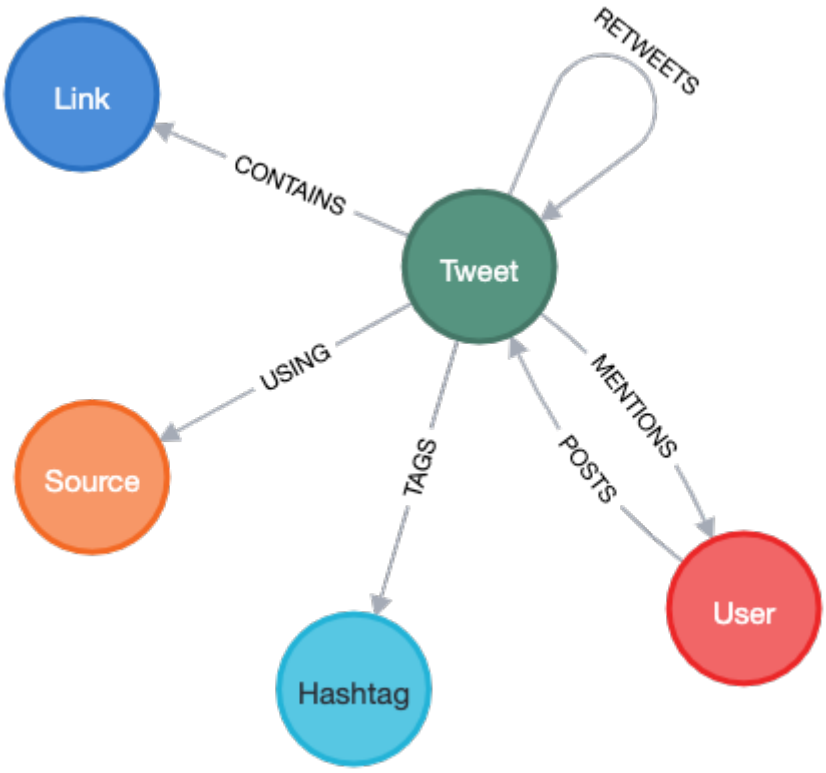

Figure 3. Graph Data Model for Tweets in Twitter.

For creating Links, be sure to find the expanded URL to be stored in the Link node, the twitter data contains both a shortened and expanded version of all links contained in a tweet, however we require the full URL for analysis at a later stage.

To discover which posts are retweets, the twitter data has a verb specifying post or share. A Retweet should be represented as a distinct Tweet node, with its own attributes and source, but also link to the original as shown in the model. A retweet should also replicate the originals tags, links and mentions.

Part 2. Initial Queries (30%)

Write Cypher statements, in Neo4j, for each of the following problems:

- *Problem 1 (10%):*

  Find the top five most used sources (app or site) to post or share a tweet. For each source return the number of posts and the number of users that created tweets from that source.

  Example result row:

  source num_posts *num_users*

  "name_of_source" 345 229

- *Problem 2 (10%):*

  Find top 3 users that have the highest number of tweets with a retweetCount greater than 50. For each of these users show the number of popular tweets they have and the top 2 hashtags present in all their tweets in order of occurrence.

  Example result row:

  user_name num_pop_posts *top_hashtags*

"user 1" 15 ["tag 1","tag 2"]

- *Problem 3 (10%):*

  Find the shortest path connecting the User 'luckyinsivan' and the hashtag 'imsosick' using any relationship type except :USING. Submit a picture of the path from neo4j browser graph view and the length of this path in the follow format :

  Example result row:

  path_length 13

Part 3. Refactoring the Model (40%)

Now we have a base model and dataset, it is common to refactor the model to either enhance query performance or accommodate new use cases. It has been decided that we would like to perform more detailed analysis on the links/urls contained in tweets. Specifically, certain site owners such as realestate.com.au would like a service to analyse links to anything in their domain.

- *Part A (10%):*

  Suggest a modification to the base data model shown in Figure 3, that would make it easier to answer questions such as *"Show me the top 5 users posting links from realestate.com.au"*. Discuss any other options you considered.

- *Part B (10%):*

  Write the Cypher statements to implement your changes to the model. This should process existing data in the model to create new nodes and relationships from the data you already have.

- *Part C (10%):*

  Using elements of your extended data model, answer the following question with a Cypher query. Which user(s) post the most links from linkedin, that is links from the domain "www.linkedin.com" or "lnkd.in"

  Example result rows:

  user_names num_posts

  ["user_x","user_y"] 5

- *Part D (10%):*

  In part C, we can see that an organisation might have multiple domains associated with it. Can you design a model to also allow for this so the query in Part C could be just focused on the company linkedin itself. We would also like to able to use the graph to analyse these links by industry, extend the model again to allow for this.

  Only the data model is required, you can use the arrows tool to design your new graph model or make the changes to your database and use neo4j procedure db.schema.visualization() to generate a diagram similar to that shown in Figure 3.

Evaluation and Marking

- This is an individual assignment worth 25%.

- All assignments will be submitted using iLearn. The results of all assignments will be available via iLearn.
- You will need to **create a video (max 10 minutes) to demonstrate your solution**, and upload it on YouTube. Then, share the YouTube link in your assignment submission.
- The submission will be a zip file including the source code for Part 1 and 2, the YouTube Link, and the queries for part 3. You do not need to include the RDF Graph in the submission file.
- **Students will demonstrate their assignments during week 8 and week 9**.
- No late submissions will be accepted, unless a Special Consideration is Submitted before the assessment submission deadline, and Granted.Your assignment will be evaluated by the tutor independently.

Appendix 1 - Importing JSON data into Neo4j with APOC

For part 2, JSON data must be imported into a new Neo4j database to support the queries found in part 3. JSON data can be imported using the Neo4j APOC libraries using the following APOC procedures :

https://neo4j.com/labs/apoc/4.4/overview/apoc.import/apoc.import.json/

https://neo4j.com/labs/apoc/4.4/overview/apoc.periodic/apoc.periodic.iterate/

While this is similar to the LOAD CSV process taught in the Neo4j lectures, it uses a direct APOC library call which was not explicitly covered in the lecture. As an example, the following Cypher statement may be used to create the Tweet nodes in the database :

```
CALL apoc.periodic.iterate(
'CALL apoc.load.json("file:///10000_tweets_1.json") YIELD value',
'WITH
  value.id AS id
 ,datetime({ epochMillis: apoc.date.parse(value.postedTime, "ms",
            "yyyy-MM-dd\'T\'HH:mm:ss.SSS\'Z\'")}) AS postedTimestamp
 ,value.text AS text
 ,value.twitter_lang AS language
 ,value.retweetCount AS retweetCount
 ,value.favoritesCount AS favoritesCount
 MERGE (t:Tweet{id:id})
 ON CREATE SET
  t.postedTimestamp = postedTimestamp
 ,t.text = text
 ,t.language = language
 ,t.retweetCount = retweetCount
 ,t.favoritesCount = favoritesCount
',
{batchSize:500})
YIELD * ;
```

Subsequently, the file can be reprocessed to extract the links and connect them to the Tweet nodes via the CONTAINS relationship in the following way:

```
CALL apoc.periodic.iterate(
'CALL apoc.load.json("file:///10000_tweets_1.json") YIELD value',
'WITH
  value.link AS link
 ,value.id AS id
 MATCH(t:Tweet{id:id})
 MERGE (l:Link{link:link})
 MERGE (t)-[:CONTAINS]->(l)',
{batchSize:500})
YIELD * ;
```

Several further steps may be required to complete the import and create the data model as shown for the model as shown in Part 2.