COMP8210/COMP7210
Big Data Technologies

Assignment 2

Semester 2, 2022
Macquarie University, School of Computing

Due: 9 September 2022 (Friday) at 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 social data graphs.

Resource: Beheshti, A., Ghodratnama, S., Elahi, M., & Farhood, H. (2022). Social Data Analytics (1st ed.). CRC Press. https://doi.org/10.1201/9781003260141

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

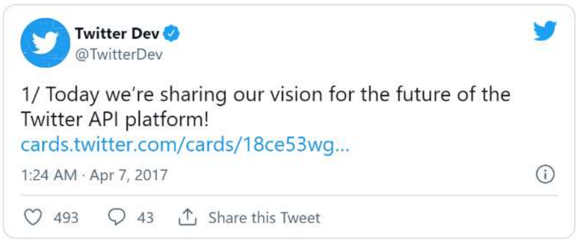
Twitter[1] serves many objects as JSON[2], including *Tweets and Users*. These objects all encapsulate core attributes that describe the object. Each Tweet has an author, a message, a unique ID, a timestamp of when it was posted, and sometimes geo metadata shared by the user. Each User has a Twitter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

[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.



Twitter Dev
@TwitterDev

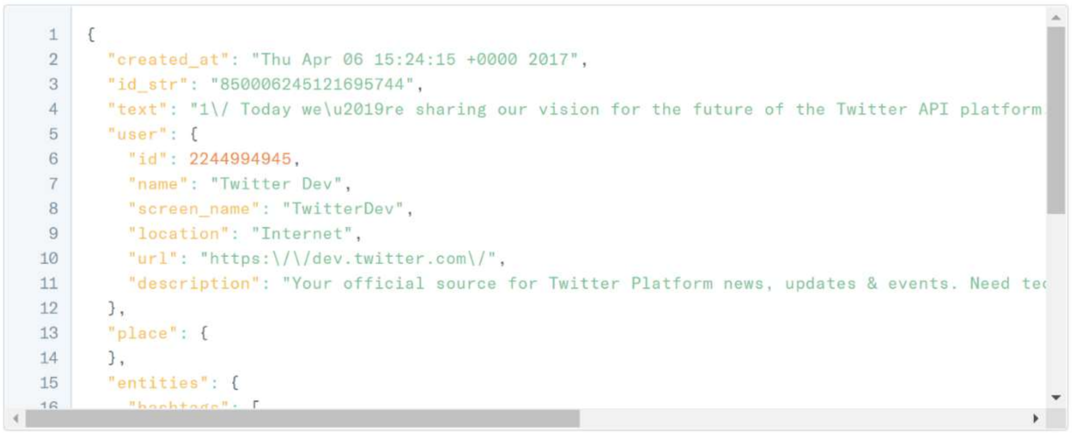
1/ Today we're sharing our vision for the future of the Twitter API platform!
cards.twitter.com/cards/18ce53wg...

1:24 AM · Apr 7, 2017

493   43   Share this Tweet

Figure 1. A sample Tweet.

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



```
{
  "created_at": "Thu Apr 06 15:24:15 +0000 2017",
  "id_str": "850006245121695744",
  "text": "1\/ Today we\u2019re sharing our vision for the future of the Twitter API platform
  "user": {
    "id": 2244994945,
    "name": "Twitter Dev",
    "screen_name": "TwitterDev",
    "location": "Internet",
    "url": "https:\/\/dev.twitter.com\/",
    "description": "Your official source for Twitter Platform news, updates & events. Need tec
  },
  "place": {
  },
  "entities": {
    "hashtags": [
```

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

Part 1. Data Curation (10%)

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 2. Graph Data Model (40%)

Considering the following Graph Data Model for Tweets in Twitter, in Figure 3, Write a program (in Python) to generate an RDF[4] graph model from the 10k Tweet Dataset.

Note that the data required to create the "FOLLOWS" relationship shown in this model does not exist in the source dataset and will be inferred as an exercise during Part 3. For the remaining relationships, data can be imported to Neo4j using the APOC support for importing JSON. See appendix for information.

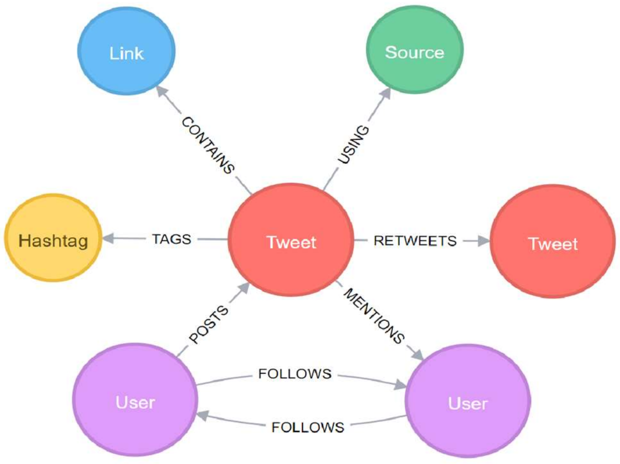


Figure 3. Graph Data Model for Tweets in Twitter.

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

[4] https://www.w3schools.com/xml/xml_rdf.asp

Part 3. Query (50%)

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 source name and the number of tweets sent via that source.

  Example result row :

  "name_of_source" , 2385

- Problem 2 (10%):

  Find the top five most used hashtags across all tweets on each day between 26th and 31st March 2016 (inclusive). For each day, return the date and a list of the five top hashtags in order of popularity on that day.

  Example result row :

  "2016-03-26" , ["Tag1", "Tag2", "Tag3", "Tag4", "Tag5"]

- Problem 3 (10%):

  Find all users that use any of the same hashtags as user "m_mrezamm". This query must exclude any retweets since these posts would automatically contain common tags. The query must return the user name and the number of hashtags that were used in their tweets that are also used by "m_mrezamm". Order results by the number of hashtags used in common.

  Example result rows :

  "user_x" , 5

  "user_y" , 3

- Problem 4 (10%):

  The original dataset does not contain information about which users follow each other. For this exercise we will infer that any user that MENTIONS another user in a tweet FOLLOWS that user. Write a Cypher expression which creates FOLLOW relationships based on this assumption, for example if UserA mentions UserB in a tweet, then it is assumed that UserA FOLLOWS UserB. Each FOLLOWS relationship added to the graph should also have a 'weight' property with a value of 1.

- Problem 5 (10%):

  Using the FOLLOWS relationship derived in Problem 4, use the Neo4j Graph Data Science library to calculate the most popular nodes using Degree Centrality from the FOLLOWS subgraph. Then, find the top five users with the highest Degree Centrality score. (consider using NATURAL orientation and the weight property for the graph projection).

  Example result rows :

  "user_x" , 22

  "user_y" , 17

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) 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.
- 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 will be required to complete the import and create the data model as shown for the model as shown in Part 2.