Neo4j / Gephi Assignment

Advanced Analytics in a Big Data World
Prof. dr. Seppe vanden Broucke

Dataset Description

For this assignment, you will work with a graph extracted from TikTok, the popular social short video platform.

Perhaps even more so than other social media platforms, TikTok has been under a lot of scrutiny once the Russian president Vladimir Putin ordered to invade Ukraine. Like every social media platform, TikTok was rapidly overwhelmed by clips, news and videos coming from people in Ukraine, journalists, but also influencers and fake news accounts. More and more analysts are getting worried about the potential dangers of TikTok to be used for misinformation campaigns or propaganda given its large scale. See for instance the following articles on these topics:

- TikTok was 'just a dancing app'. Then the Ukraine war started
  https://www.theguardian.com/technology/2022/mar/19/tiktok-ukraine-russia-war-disinformation
- Ukraine war: False TikTok videos draw millions of views
  https://www.bbc.com/news/60867414
- TikTok and Twitter capture Ukraine war in frighteningly real time
  https://www.latimes.com/world-nation/story/2022-03-31/ukraine-war-social-media-tik-tok
- Watching the World's "First TikTok War"
  https://www.newyorker.com/culture/infinite-scroll/watching-the-worlds-first-tiktok-war
- TikTok doesn't show the war in Ukraine to Russian users
  https://www.nrk.no/osloogviken/xl/tiktok-doesn_t-show-the-war-in-ukraine-to-russian-users-1.15921522

Inspired by these sources, a data set has been collected during April 2022 to see if we can come to some interesting findings of our own.

To create the data set, the following approach was followed:

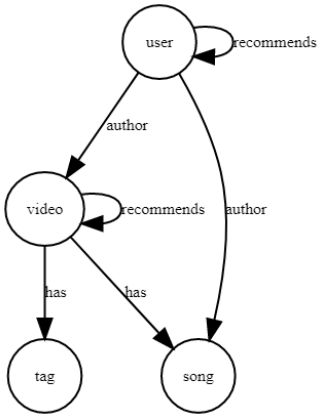
1. We search and collect videos (and corresponding user information) for the following tags:
   - "ukraine", "russia", "ukrainewar", "ukrainerussiaconflict", "ukrainewarrussia", "supportukraine", "ukrainevsrussia", "prayforukraine", "russianarmy", "Z"
2. To expand out the graph:
   - We randomly pick from the videos collected so far and navigate to their page to see which other videos TikTok recommends us ("Browse more For You videos")
   - We randomly pick from the user accounts collected so far and navigate to their page to see which other users TikTok recommends us ("Suggested Accounts")
   - We randomly pick from the user accounts collected so far and collect other videos the user has uploaded
3. Steps 2 and 3 are repeated continuously and kept running for a month or so

For each video, we collect all of its tags, and song. For each video, user, song and tag, as much statistics as possible were gathered without logging into an account. Comments were not included as they are not visible without logging in. https://github.com/davidteather/TikTok-Api and custom Selenium scripts where utilized to set this up.

Note that this is by no means a fully scientific approach. E.g. it was clear during scraping that TikTok was quickly recommending videos either made by Belgian accounts (i.e. based on geolocation) which might impact results. Also, the way how the graph is expanded quickly leads to videos being visited which have nothing to do anymore with Ukraine or Russia but might be just TikTok trying to show something popular, or due to the fact that a certain user's videos leads to a different interest. As such, you will find

“communities” across nations and lots of crazy tags in this data set. (On the other hand, this might make for some interesting analysis.)

The resulting graph is saved in a Neo4j database. The metagraph looks as follows:



- General remarks:
    - All nodes have an “id” attribute which can be used as a unique identifier. This is different from the built-in “<id>” (Neo4j’s Node id)
    - Most of the edges do not have attributes. One exception is “has” between “video” and “tag”, where “position” indicates the position of the tag in the video description
- video:
    - Represents a video
    - createTime: time when the video was uploaded
    - bitrate, codecType, definition, format, width, height, videoQuality: video format information
    - commentCount: number of comments
    - diggCount: how many likes the video got
    - playCount: number of views
    - shareCount: number of shares
    - duration: video duration (seconds)
    - desc: video description
    - stickerText: text placed in “stickers” on the video: this also contains lots of interesting information you can use
- user:
    - Represents a user
    - diggCount: number of likes given by this account
    - followerCount: number of followers
    - followingCount: number of accounts this account is following
    - heartCount: number of likes received
    - videoCount: number of videos
    - Note: statistics can be null in case they couldn’t be obtained (private, deleted or locked account)
    - commerceUserInfo_commerceUser, commerceUserInfo_category: whether this is a commercial (advertiser) user and if so to which commercial category the user belongs to
    - signature: bio signature
    - verified: whether the account is verified
- tag:
    - title: tag name
    - numViews: number of views for videos having this tag (null if the information could not be extracted)
- song:

- album, title: song info
- Note: title says “original sound” in language of user if the user “made” the sound
- authorName: nickname of user who created this sound. If possible, we link the song to the user (see “author” edge)
- original: whether this is an original sound
- duration: in seconds
- numVids: number of videos using this sound (null if the information could not be extracted)

There are some other attributes (as returned by TikTok) which might be interesting to explore.

Note that the recommended videos are probably more related to the internal profile TikTok has build up about the browsing sessions. The recommended users seem more stable and profile-independent.

Assignment

Your task for this assignment is to use Neo4j and Gephi to analyze this data set.

You are free to explore anything you deem interesting and present your findings in your report. The main goal is to get familiar with Neo4j and Gephi, but also to hone your "storytelling" skills. In that sense, try to focus on a single or a few hypotheses or findings you explore in full (with nicely formatted visualizations) and explaining what it says instead of just going for quick filter saying: "here are the three nodes with the most connections" (boring).

Below are some suggestions/tips of what you can go for:

- Start from the tags listed above and go from there. Explore the recommendations and where they lead to. Do you find cliques of misinformation users/tags?
- Use the text of a videos' description and stickers
- Perform community mining on the graph
- Visualize popular accounts based on centrality metrics
- Any other fun idea…

Don't try to go for all of these. Pick something you want to explore and try to work this out in full. You will realize that the data set is most likely too large to do everything at once, so a guided "deep dive" will work much better. Also, trying to visualize the whole graph will most likely not lead to a lot of real insights. Focus on a "small sample with good insights" instead of "big data and discovering nothing".

Finally, keep in mind that the goal of this assignment or data set is not to pick sides between (or put blame on) common Ukrainian and Russian people. Many Russians disagree with the actions of their president, so let's keep that in mind.

Setting up Neo4j and Gephi

First, download and install Neo4j Desktop for your platform:

- https://neo4j.com/download-center/?ref=web-product-database/#desktop

Next, download and install Gephi for your OS:

- https://gephi.org/users/download/

Also download the ".dump" file as instructed on Toledo / elsewhere.

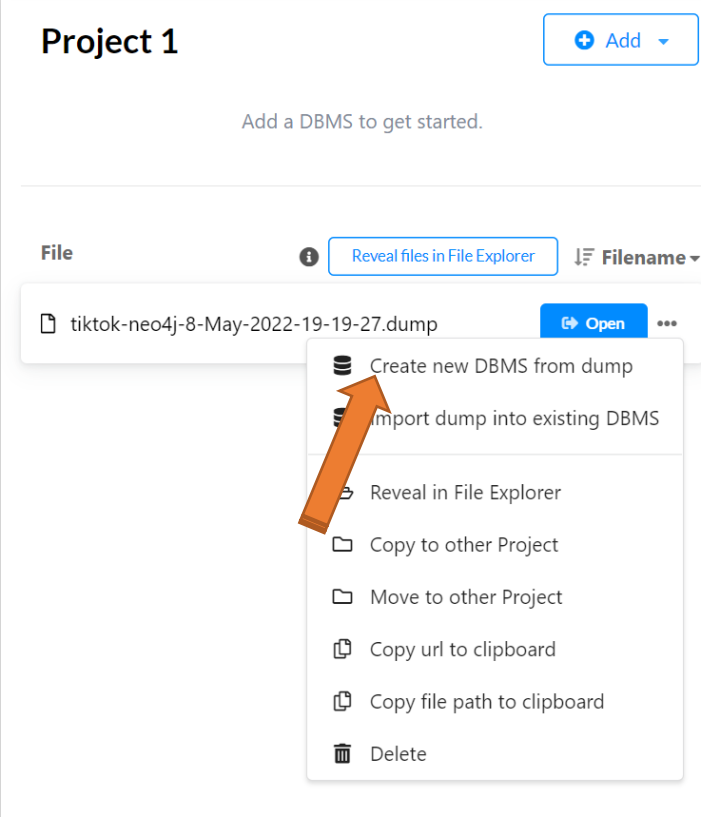
Open Neo4j Desktop and agree to the license. Neo4j will ask you where you want to store your data. You can use the suggested path ("C:\Users\%USERNAME%\.Neo4jDesktop"), but if you have a second hard drive with more space, it might be a good idea to place your data there instead.

Go through the registration steps and install updates when prompted.

Delete the "Example Project" and create a new project (give it any name you like).

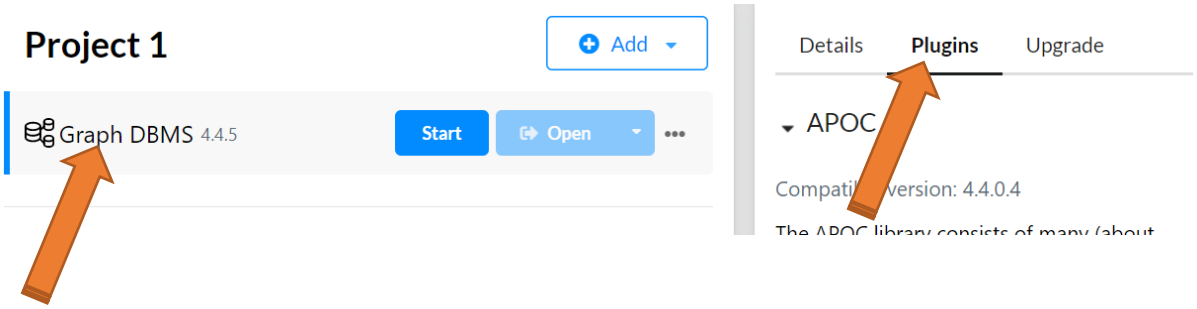
Pick "Add -> File"; and add the ".dump" file.

Next, click the three dots next to the file and pick "Create new DBMS from dump":



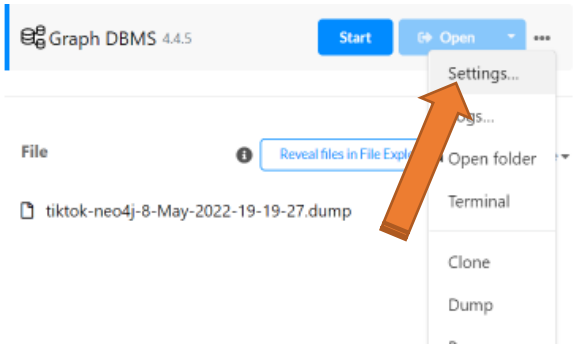
Name this DBMS anything you like (e.g. "tiktok" and set a password).

Click on the newly created DBMS and open the “Plugins” tab in the pane to the right:



Under “Plugins”, install APOC (you can also install the Graph Data Science Library in case you want to play around with it, but this is not required).

Next, open “Settings…”:



Change the following properties (you can press CTRL+F to search):

```
dbms.security.procedures.unrestricted=jwt.security.*,apoc.*,apoc.meta.*,gds.*
```

And add these two to the end:

```
apoc.import.file.enabled=true
apoc.export.file.enabled=true
```

(If you have memory to spare, you can also change the existing `dbms.memory.heap.max_size=1G` line to allow Neo4j to use more memory (the exact setting depends on how much RAM you have available, so make a conservative choice); the default setting should work well for this reasonably-sized graph, however.)

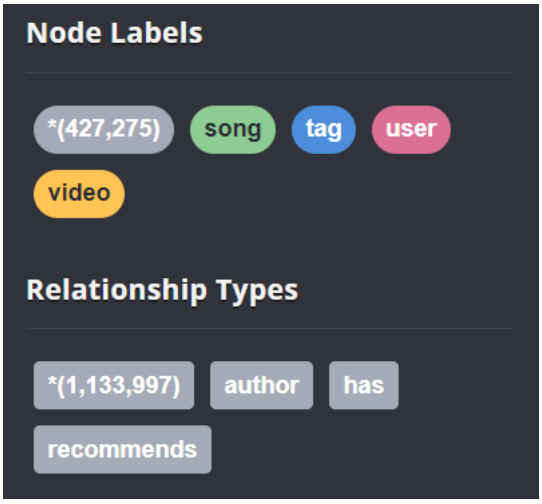
Do not forget to press “Apply” to save the configuration.

You’re now ready to start the database:

Working with the Graph

Press “Open” to bring up Neo4j’s query window.

You can take a look at the Database Information to get a feel for the types of nodes, edges, and properties in the graph:



Feel free to play around with some Cypher queries to explore this graph, e.g.:

```
match (v:video)--(u:user)
where u.id = 'cbsnews'
return v, u
```

Or:

```
match (v:video)--(u:user)
where v.desc =~ '.*ukraine.*'
return v, u
```

Or:

```
match (t:tag)--(v:video)
where t.title = 'ukraine'
return *
```

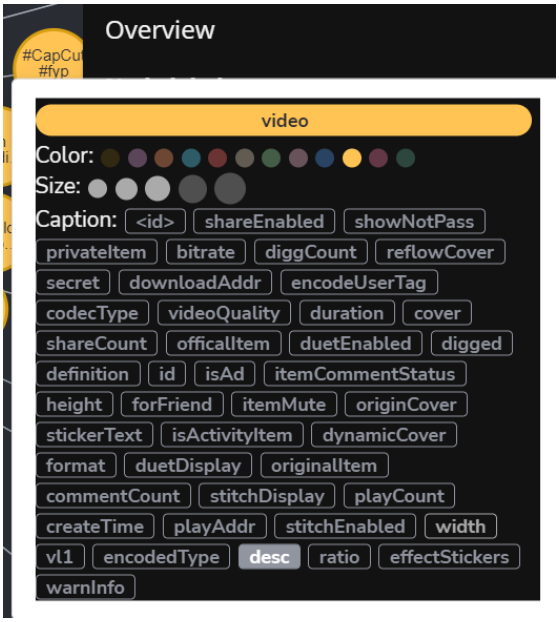
Or:

```
match (a:tag)-[:has*1..2]-(b:tag)
where a.title = "ukraine"
merge (a)<-[e:tag_connected]-(b)
return a,e,b
```

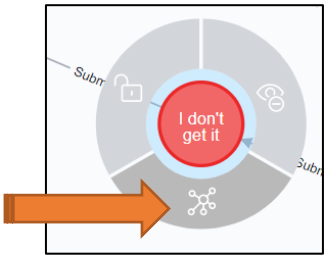
and then (to delete the edges created in the query above):

```
match (:tag)-[e:tag_connected]-(:tag)
delete e;
```

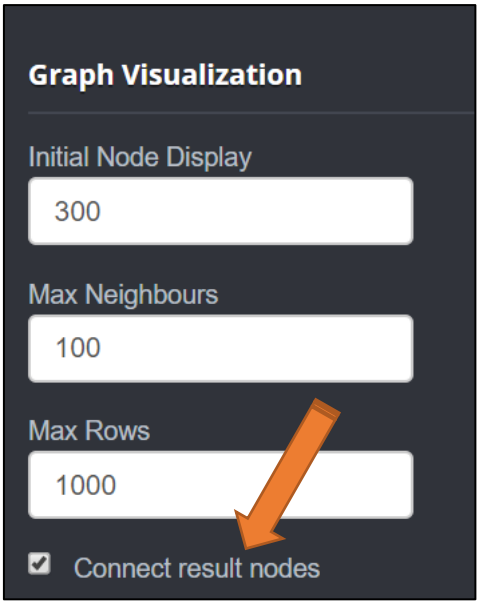
You can change size and color of the node types by clicking on them. You can also change which attribute is used as the node text label:



You can expand nodes by selecting them:



Also take a look at the settings pane. Note that Neo4j connects result nodes together even if you don't explicitly return edges by default. Other settings prevent the result window from becoming overloaded (though it is a good idea to nevertheless provide a LIMIT clause whilst exploring to prevent the browser from crashing on large result sets):

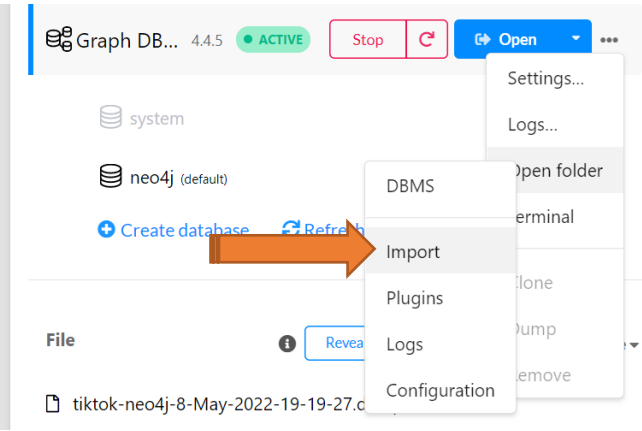


Using Gephi

To load the data in Gephi, we must get it out of Neo4j in some manner. There is a Neo4j Gephi plugin, but it has not received much attention lately. We could use neo4j-shell-tools or Gephi's "Graph Streaming" plugin together with Neo4j's "apoc.gephi" procedures to directly stream data from Neo4j to Gephi, but since this is a bit cumbersome, we're going to use a simpler approach. Using APOC, we can directly export the full graph to a GraphML file:

```
CALL apoc.export.graphml.all("export.graphml",
  {useTypes: true, storeNodeIds: true, readLabels: true})
```

The resulting "export.graphml" file will be placed under the "import" folder in your database's installation folder:



(If you get a "Neo.ClientError.Procedure.ProcedureCallFailed: Failed to invoke procedure `apoc.export.graphml.all`: Caused by: java.lang.RuntimeException: Import from files not enabled, set apoc.import.file.enabled=true in your neo4j.conf" error. Make sure that you have changed your settings according to the instructions above.)

You can also export the result of a Cypher query using ".query":

```
CALL apoc.export.graphml.query("
  match (t:tag)-[e]-(v:video)
  where t.title = 'ukraine'
  return *
", "subset.graphml",
   {useTypes: true, storeNodeIds:true, readLabels:true})
```

Important: you need to explicitly name the edges in your query and return them in order for them to be included in the GraphML export!

You can then open these GraphML files with Gephi. The goal here is that you use Gephi after having found an interesting subgraph using Cypher queries to further visualize, explore and annotate.