DB - Study Point Assignment 2

Robert Pallesen, Tobias Linge, Mathias Drejer

March 2023

Store the data, the queries, the application code, snapshots from the database, and the answers of the questions in a Github repository and upload the link to it in Peegrade.

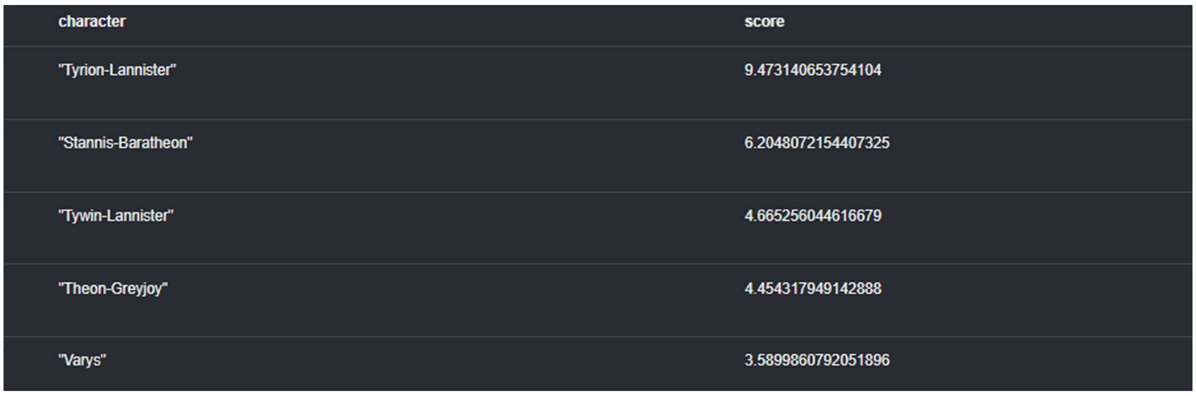
1. Queries

Listing 1: Populate the DB

```
LOAD CSV WITH HEADERS FROM 'https://raw.githubusercontent.com/
    mathbeveridge/asoiaf/master/data/asoiaf-all-edges.csv' AS row
MERGE (source:Character {name: row.Source})
MERGE (target:Character {name: row.Target})
MERGE (source)-[relation:RELATION {type: row.Type, id: row.id,
    weight: toInteger(row.weight)}]->(target)
```

Listing 2: PageRank algorithm

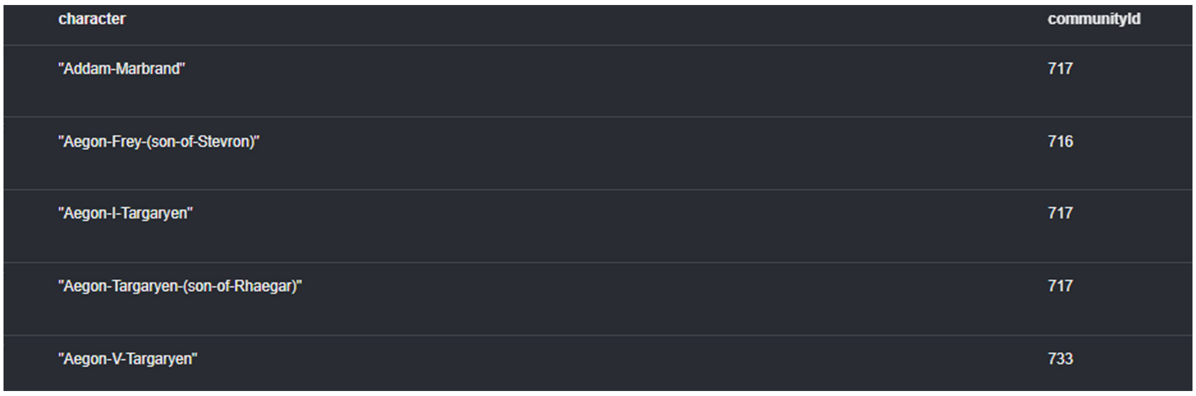
```
CALL gds.pageRank.stream('GoT', { maxIterations: 20, dampingFactor:
    0.85 })
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS character, score
ORDER BY score DESC
LIMIT 5
```



character	score
"Tyrion-Lannister"	9.473140653754104
"Stannis-Baratheon"	6.2048072154407325
"Tywin-Lannister"	4.665256044616679
"Theon-Greyjoy"	4.454317949142888
"Varys"	3.5899860792051896

Listing 3: Louvain algorithm

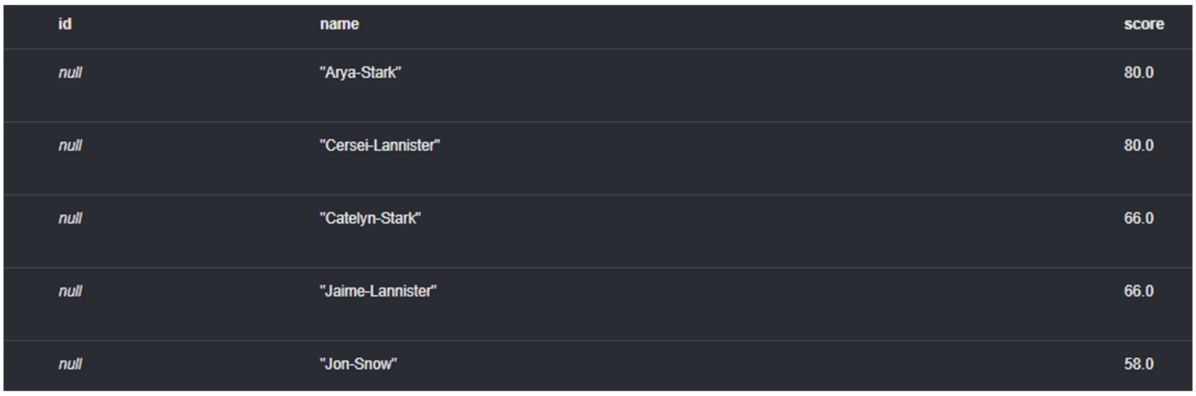
```
CALL gds.louvain.stream('GoT')
YIELD nodeId, communityId
RETURN gds.util.asNode(nodeId).name AS character, communityId
LIMIT 5
```



character	communityId
"Addam-Marbrand"	717
"Aegon-Frey-(son-of-Stevron)"	716
"Aegon-I-Targaryen"	717
"Aegon-Targaryen-(son-of-Rhaegar)"	717
"Aegon-V-Targaryen"	733

Listing 4: Centrality Degree Algorithm

```
CALL gds.degree.stream('GoT')
YIELD nodeId, score
WITH gds.util.asNode(nodeId) AS node, score
RETURN node.id AS id, node.name AS name, score
ORDER BY score DESC
LIMIT 5
```



id	name	score
null	"Arya-Stark"	80.0
null	"Cersei-Lannister"	80.0
null	"Catelyn-Stark"	66.0
null	"Jaime-Lannister"	66.0
null	"Jon-Snow"	58.0

2. Application code

Listing 5: Python application code

```
from neo4j import GraphDatabase

uri = 'bolt://localhost:7687'
username = 'neo4j'
password = '12345678'

def get_most_active_character():
    query = '''
    CALL gds.degree.stream('GoT')
    YIELD nodeId, score
    WITH gds.util.asNode(nodeId) AS node, score
    RETURN node.name AS name, score
    ORDER BY score DESC
    LIMIT 1
    '''
    driver = GraphDatabase.driver(uri, auth=(username, password))
    with driver.session() as session:
        result = session.run(query)
        record = result.single()
        most_active_character = record['name']
        degree_centrality = record['score']
    driver.close()
    print(f"The most active character is {most_active_character},
       with a degree centrality of {degree_centrality}.")

def get_character_degree_centrality(character_name):
    query = f'''
    CALL gds.degree.stream('GoT')
    YIELD nodeId, score
    WITH gds.util.asNode(nodeId) AS node, score
    WHERE node.name = '{character_name}'
    RETURN node.id AS id, node.name AS name, score
    ORDER BY score DESC
    LIMIT 1
    '''
    driver = GraphDatabase.driver(uri, auth=(username, password))
    with driver.session() as session:
        result = session.run(query)
        record = result.single()
        degree_centrality = record['score']
    driver.close()
    print(f"The degree centrality of {character_name} is {
       degree_centrality}.")

get_most_active_character()
get_character_degree_centrality("daenerys-targaryen")
```