Teoría de Grafos para Big Data

Neo4j

Instrucciones

La parte 1 de esta guía tiene consultas para aprender lo básico de Cypher y Neo4J. Lea y ejecute cada una de las consultas, preocupándose de entender qué hace cada parte de ella.
La parte 2 cuenta con preguntas de otro dataset, para las cuales debe investigar por su cuenta y responder adecuadamente. Esta parte será evaluada en la tarea final.
Para esta actividad, trabajaremos con el sandbox de Neo4j, ubicado en

`https://neo4j.com/sandbox-v2/`

Tras registrarnos y crear una cuenta, podemos acceder a una instancia de Neo4J con un grafo cargado haciendo click en "Launch Sandbox". Inicialmente usaremos el sandbox "Recommendations", que consiste en una base de datos de películas, actores y directores, junto con evaluaciones de usuarios.
Para ver el grafo, en cualquier momento podemos ejecutar la siguiente consulta:

```
MATCH (n) RETURN n LIMIT 100
```

y comprobar que estamos en el grafo correcto.

IMPORTANTE: Es recomendable siempre agregar `LIMIT 300` al final, ya que puede ocurrir que la pestaña del navegador se quede pegada al tratar de dibujar grafos muy grandes.

MÁS IMPORTANTE: En caso de accidentes o problemas, se puede reiniciar el *sandbox* a su estado original. Para esto basta con volver al panel de los sandboxes[1] y en el cuadro de "Recommendations" hacer click en Advanced→Shutdown Sandbox. Luego es cosa de iniciarlo una vez más y estará en su estado original.

Algo de sintaxis y semántica de Cipher

Los patrones básicos de cipher son:

- `(p) --> (q)` y `(p) <-- (q)`, que representan una arista que va de `p` a `q` o de `q` a `p`, respectivamente, y `(p) --> ()`, que representa una arista de `p` a *algún* nodo.
- `(p) -[n:ETIQUETA]-> (q)`, que representa una arista `n` que va de de `p` a `q`, y que además la arista está etiquetada con `ETIQUETA`.

[1] `https://neo4j.com/sandbox-v2/`

- Estos caminos se pueden combinar, por ejemplo `(p) <-- (q) --> (r)` o incorporando una etiqueta: `(p) <-[n:ETIQUETA]- (q) <-- (r)`
- Para retornar usamos `RETURN`. Podemos retornar todo o parte de todo lo que hemos nombrado en nuestra consulta `(p)`, `(q)`, etc. Es posible usar `*` en return para que lo retorne todo.
- Intente distintas combinaciones en el grafo de películas. También pruebe agregando restricciones a los atributos de los nodos y las aristas (nombres de actores, nombres de películas)
- Pruebe con `MATCH c= (p) -[n]-> (q) RETURN *`. ¿Que es `c` en este caso?

La referencia para todos estos problemas es la documentación oficial de Neo4J:

`http://neo4j.com/docs/developer-manual/current/cypher/`

1. Número de Bacon

Esta parte de la guía usa el sandbox "Recommendations" de Neo4J.

1.1. Explorando el grafo

Escriba las siguientes consultas y observe el resultado:

1. Todas las películas en las que ha actuado Kevin Bacon.

   ```
   MATCH (:Actor {name: "Kevin Bacon"})-[:ACTED_IN]->(m:Movie)
   RETURN *
   ```

2. Los actores que han actuado en una película con Kevin Bacon.

   ```
   MATCH (b:Actor {name: "Kevin Bacon"})-->(m:Movie)<--(actor:Actor)
   RETURN *
   ```

3. Los actores que han actuado en al menos dos películas con Kevin Bacon.

   ```
   MATCH (b:Actor {name: "Kevin Bacon"})-->(m1:Movie)<--(a:Actor)
   WITH b, m1, a
   MATCH (b)-->(m2:Movie)<--(a)
   WHERE m1<>m2
   RETURN *
   ```

   ¿Por qué esta consulta no entrega resultados? Reemplace a Kevin Bacon por Tom Hanks y vea qué ocurre.

4. Los pares de actores que han actuado juntos y con Kevin Bacon.

```
MATCH (b:Actor {name: "Kevin Bacon"})-->(m1:Movie)<--(a1:Actor)
WITH b, a1
MATCH (b)-->(m2:Movie)<--(a2:Actor)-->(m3:Movie)<--(a1)
RETURN *
```

1.2. Agregación

Las siguientes consultas tienen que ver con las funciones de agregación de Neo4J:

1. La cantidad de nodos en el grafo

```
MATCH (n) return COUNT(n)
```

2. La cantidad de películas en las que actúa Kevin Bacon

```
MATCH (:Actor {name: "Kevin Bacon"})-->(m:Movie)
RETURN count(m)
```

3. La cantidad promedio de películas por actor

```
MATCH (a:Actor)-->(m:Movie)
WITH a, count(m) as c
RETURN avg(c)
```

4. El actor que actúa en más películas

```
MATCH (a:Actor)-->(m:Movie)
RETURN a, count(m) as c order by c desc limit 1
```

1.3. Caminos y número de Bacon

Definimos el número de Bacon como sigue:

- Kevin Bacon tiene número de Bacon 0.
- Si un actor aparece en una película con alguien que tiene número de Bacon n, entonces esa persona tiene número de Bacon $n + 1$, a menos que ya tenga un número de Bacon menor.
- Todos los actores que no son alcanzables de esta manera desde Kevin Bacon tienen número de Bacon infinito.

Ejecute las siguientes consultas:

1. Todos los nodos conectados entre sí

   ```
   MATCH (a:Actor)-[:ACTED_IN]-(m:Movie)-[:ACTED_IN]-(b:Actor)
   WHERE a<>b
   RETURN * limit 1000
   ```

   Nótese el `limit 1000`. Esta cláusula evita que el sistema colapse por exceso de nodos.

2. Todos los nodos que estan contectados a Kevin Bacon por un camino de cualquier largo

   Un primer acercamiento es la siguiente consulta:

   ```
   MATCH (:Actor {name: "Kevin Bacon"})-[:ACTED_IN*]-(a:Actor)
   RETURN a
   ```

   Sin embargo, rápidamente vemos que el sistema colapsa. Podemos ahora intentar limitar el largo del camino:

   ```
   MATCH (:Actor {name: "Kevin Bacon"})-[:ACTED_IN*..5]-(a:Actor)
   RETURN a
   ```

   Con esto conseguimos algunos nodos sin que el sistema colapsa, pero con algo de ojo podemos notar que empiezan a aparecer varios caminos entre pares de nodos.

   Esta consulta busca *todos* los caminos entre Kevin Bacon y todos los nodos a los que está conectado, lo que obviamente puede generar caminos arbitrariamente largos. Como nos interesa solamente saber si existe un camino entre Kevin Bacon y el otro nodo, podemos preguntar solamente por el camino más corto:

   ```
   MATCH p=shortestPath((b:Actor {name: "Kevin Bacon"})-[:ACTED_IN*]-(a))
   RETURN p LIMIT 1000
   ```

3. Para un actor determinado (por ejemplo Tom Hanks), verifique si existe un camino entre Kevin Bacon y él, y entre él y Kevin Bacon.

   ```
   MATCH
   p=shortestPath((a:Actor)-[:ACTED_IN*]-(b:Actor))
   WHERE a.name = "Kevin Bacon" AND b.name="Tom Hanks"
   RETURN p
   ```

4. El actor con más caminos a Kevin Bacon

   La siguiente consulta representa la respuesta que buscamos:

   ```
   MATCH p=((a:Actor)-[:ACTED_IN*]-(:Actor {name: "Kevin Bacon"}))
   RETURN a, count(*) AS c ORDER BY c desc limit 1
   ```

   Sin embargo, la consulta se quedará pegada sin importar de cuánta memoria y tiempo dispongamos. ¿Qué está pasando? ¿En qué tipos de grafos esta pregunta se puede responder?

1.4. Modificando el grafo

Ahora vamos a agregar arcos al grafo. En caso de accidentes o problemas, recuerde que puede reiniciar el sandbox.

1. Para todo par de actores que han aparecido en la misma película, agregue en ambas direcciones un arco con etiqueta `ACTUA_CON`.

   ```
   MATCH (b:Actor)-->(m:Movie)<--(a:Actor)
   CREATE (a)-[:ACTUA_CON]->(b)
   ```

2. Para el nodo "Kevin Bacon", fije el atributo `numero_bacon` en 0. Hint: La sintaxis es igual a `CREATE`, pero usando `SET` en vez de `CREATE`.

   ```
   MATCH (b:Actor {name: "Kevin Bacon"}) SET b.numero_bacon=0
   ```

3. Escriba una consulta que al ejecutarla, encuentre los nodos sin número de Bacon (`WHERE NOT EXISTS(n.numero_bacon)`) que están conectados a un nodo con número de Bacon conocido, y les cree su número de Bacon.

   ```
   MATCH (a:Actor)-[:ACTUA_CON]-(b:Actor)
   WHERE EXISTS(a.numero_bacon) AND NOT EXISTS(b.numero_bacon)
   WITH b, min(a.numero_bacon) AS m
   SET b.numero_bacon=m+1
   RETURN *
   ```

   Ojo con la tercera línea. Esa línea hace que siempre se fije el número de bacon de un nodo a partir del vecino con el número de bacon más bajo. Si no usamos esa línea y solamente fijamos el número de bacon del nodo a partir del número de bacon de un vecino cualquiera, podemos estar eligiendo un camino más largo y fijando el número en un valor mayor al real.

4. Ejecute la consulta anterior 5 veces. ¿Cuáles son los nodos con número de Bacon menor o igual a 5? ¿Qué pasa con los otros nodos?

   ¿A qué algoritmo visto en clases se parece este procedimiento?

5. Cree ahora una consulta que automáticamente llene el número de Bacon de todos los nodos que son alcanzables desde Kevin Bacon, usando el largo del camino más corto desde Kevin Bacon. Hint: use la función `length`.

   ```
   MATCH
   p=shortestPath((b:Actor {name: "Kevin Bacon"})-[:ACTUA_CON*]-(a:Actor))
   SET a.numero_bacon = length(p)
   RETURN a, length(p)
   ```

6. Encuentre el número de actores que NO son alcanzables desde Kevin Bacon.

   Como fijamos el número de bacon de todos los actores alcanzables desde Kevin Bacon en el paso anterior, los actores no alcanzables son aquellos que no tienen número de bacon:

   ```
   MATCH (n:Actor) WHERE NOT EXISTS(n.numero_bacon)
   RETURN n
   ```

2. Donald Trump

Ahora cambiamos de Sandbox. Vuelva a la página anterior e inicie el sanbox "Trumpworld".

2.1. Estadísticas

Encuentre:

1. Los caminos más cortos entre Donald Trump y Vladimir Putin.
2. Las 5 personas con mayor grado entrante, saliente y bidireccional en el grafo. Escriba una consulta para cada tipo de grado.
3. Los 5 bancos (nodos de tipo Bank) más importantes en el grafo, y la gente conectada a ellas directamente o a través de una organización.
4. La gente conectada a través de una organización a Donald Trump.

2.2. Nodos críticos

1. Definimos el diámetro (o geodésico) de un grafo como el camino más largo de los caminos más cortos entre pares de nodos de la red. Encuentre el diámetro de la red
2. Dados dos nodos a y b, decimos que el nodo c es un pivote entre a y b si se encuentra presente en todos los caminos más cortos entre a y b. Encuentre todos los nodos pivotes de la red

2.3. Pagerank

Como vimos en clase, podemos aplicar PageRank a cualquier grafo para obtener una métrica de importancia de los nodos en nuestra red. Neo4J trae implementado este algoritmo junto con otros.
Para esto, podemos ejecutar la siguiente consulta:

```
MATCH (c:Person)
WITH collect(c) AS people
CALL apoc.algo.pageRank(people) YIELD node, score
RETURN node.name AS name, score ORDER BY score DESC
```

1. ¿Quiénes son las personas más relevantes según Pagerank?
2. ¿Cómo se relacionan estas personas con las encontradas por grado?
3. Seleccione las 5 personas con mayor pagerank. ¿Qué nodos conectan a estas personas?