

Universidad Politécnica Salesiana

Nombre: Fernando Sanchez

Materia: Sistema Expertos

Cosine similarity

Fecha: 15/02/2021

Objetivo:

- Consolidar los conocimientos adquiridos en clase de los sistemas expertos.

Importar la API py2neo

Para el ingreso de los datos que se encuentran dentro de la lista

Conexión con Neo4j

Configure la URL de conexión con la base de datos de Neo4j:

```
graph = Graph("bolt://localhost:7687", aut="neo4j",  
password="costo", secure=False)
```

```
In [11]: 1 #IMPORTAR py2neo  
2 from py2neo import Node, Relationship, Graph  
3  
4  
5 # connect to authenticated graph database  
6 graph = Graph("bolt://localhost:7687", aut="neo4j", password="c
```

Creación de los nodos Persona

```
In [16]: 1 graph.run("CREATE (Persona1:Persona{ persona_id: 1, persona_nam
```

```
Out[16]: []
```

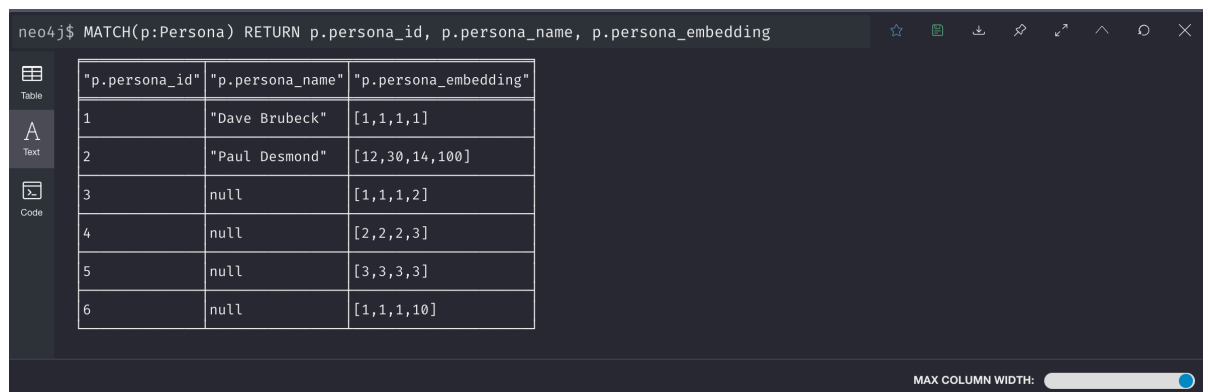
```
In [17]: ("CREATE (Persona2:Persona{ persona_id: 2, persona_name:'Paul Desmor
```

```
Out[17]: []
```

```
In [18]: 1 graph.run("CREATE (Persona3:Persona{ persona_id: 3, persona_nam
2 graph.run("CREATE (Persona4:Persona{ persona_id: 4, persona_nam
3 graph.run("CREATE (Persona5:Persona{ persona_id: 5, persona_nam
4 graph.run("CREATE (Persona6:Persona{ persona_id: 6, persona_nam
5
```

```
Out[18]: []
```

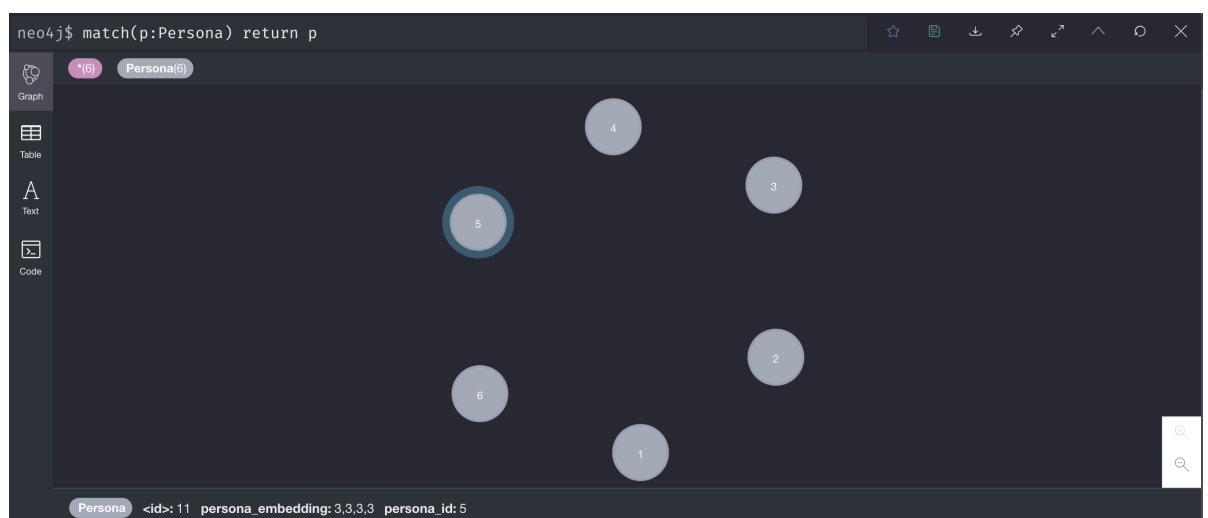
Consultar los nodos creados:



The screenshot shows the Neo4j Desktop interface with a table view of the database. The table has three columns: "p.persona_id", "p.persona_name", and "p.persona_embedding". It contains six rows of data, corresponding to the nodes created in the previous steps. The interface includes a sidebar with icons for Table, Text, and Code, and a top bar with various tool icons.

"p.persona_id"	"p.persona_name"	"p.persona_embedding"
1	"Dave Brubeck"	[1,1,1,1]
2	"Paul Desmond"	[12,30,14,100]
3	null	[1,1,1,2]
4	null	[2,2,2,3]
5	null	[3,3,3,3]
6	null	[1,1,1,10]

Consultar los nodos en grafos:



Crear un gráfico de muestra:

```

1 MERGE (french:Cuisine {name:'French'})
2   SET french.embedding = [0.71, 0.33, 0.81, 0.52, 0.41]
3 MERGE (italian:Cuisine {name:'Italian'})
4   SET italian.embedding = [0.31, 0.72, 0.58, 0.67, 0.31]
5 MERGE (indian:Cuisine {name:'Indian'})
6   SET indian.embedding = [0.43, 0.26, 0.98, 0.51, 0.76]
7 MERGE (lebanese:Cuisine {name:'Lebanese'})
8   SET lebanese.embedding = [0.12, 0.23, 0.35, 0.31, 0.39]
9 MERGE (portuguese:Cuisine {name:'Portuguese'})
10  SET portuguese.embedding = [0.47, 0.98, 0.81, 0.72, 0.89]
11 MERGE (british:Cuisine {name:'British'})
12  SET british.embedding = [0.94, 0.12, 0.23, 0.4, 0.71]

```

Started streaming 21 records in less than 1 ms and completed after 5 ms.

neo4j\$ MERGE (french:Cuisine {name:'French'}) SET french.embedding = [0.71, 0.33, 0.81, 0.52, ...

Added 7 labels, created 7 nodes, set 14 properties, completed after 18 ms.

Added 7 labels, created 7 nodes, set 14 properties, completed after 18 ms.

A continuación, encontrará la similitud entre las cuisines según la propiedad de incrustación:

```

1 MATCH (c:Cuisine)
2 WITH {item:id(c), weights: c.embedding} AS userData
3 WITH collect(userData) AS data
4 CALL gds.alpha.similarity.cosine.stream({
5   data: data,
6   skipValue: null
7 })
8 YIELD item1, item2, count1, count2, similarity
9 RETURN gds.util.asNode(item1).name AS from, gds.util.asNode(item2).name AS to, similarity
10 ORDER BY similarity DESC

```

neo4j\$ MATCH (c:Cuisine) WITH {item:id(c), weights: c.embedding} AS userData WITH collect...

	from	to	similarity
1	"Lebanese"	"Portuguese"	0.9671144333535775
2	"Portuguese"	"Lebanese"	0.9671144333535775
3	"Indian"	"Lebanese"	0.9590440861639105
4	"Lebanese"	"Indian"	0.9590440861639105
5	"Italian"	"Portuguese"	0.9582444106965535
6	"Portuguese"	"Italian"	0.9582444106965535
7	"Indian"	"Mauritian"	0.9464344561993275

Started streaming 21 records in less than 1 ms and completed after 5 ms.

```

1 MATCH (c:Cuisine)
2 WITH {item:id(c), weights: c.embedding} AS userData
3 WITH collect(userData) AS data
4 CALL gds.alpha.similarity.cosine.stream({
5   data: data,
6   skipValue: null
7 })
8 YIELD item1, item2, count1, count2, similarity
9 RETURN gds.util.asNode(item1).name AS from, gds.util.asNode(item2).name AS to, similarity
10 ORDER BY similarity DESC

```

neo4j\$ MATCH (c:Cuisine) WITH {item:id(c), weights: c.embedding} AS userData WITH collect...

Table

	from	to	similarity
	"Mauritian"	"Indian"	0.9464344561993275
9	"French"	"Indian"	0.9414524820541921
10	"Indian"	"French"	0.9414524820541921
11	"Portuguese"	"Mauritian"	0.92092461331529
12	"Mauritian"	"Portuguese"	0.92092461331529
13	"Lebanese"	"Mauritian"	0.9192477665074964
14	"Mauritian"	"Lebanese"	0.9192477665074964

Code

Started streaming 21 records in less than 1 ms and completed after 5 ms.

```

1 MATCH (c:Cuisine)
2 WITH {item:id(c), weights: c.embedding} AS userData
3 WITH collect(userData) AS data
4 CALL gds.alpha.similarity.cosine.stream({
5   data: data,
6   skipValue: null
7 })
8 YIELD item1, item2, count1, count2, similarity
9 RETURN gds.util.asNode(item1).name AS from, gds.util.asNode(item2).name AS to, similarity
10 ORDER BY similarity DESC

```

neo4j\$ MATCH (c:Cuisine) WITH {item:id(c), weights: c.embedding} AS userData WITH collect...

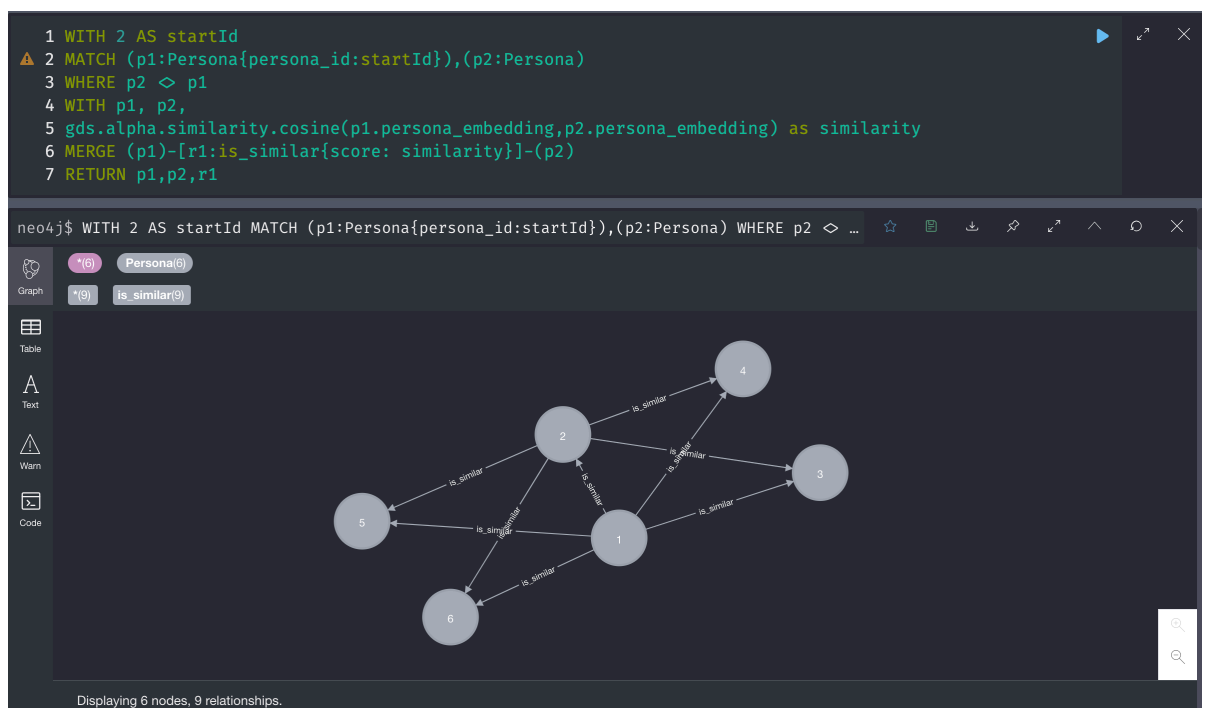
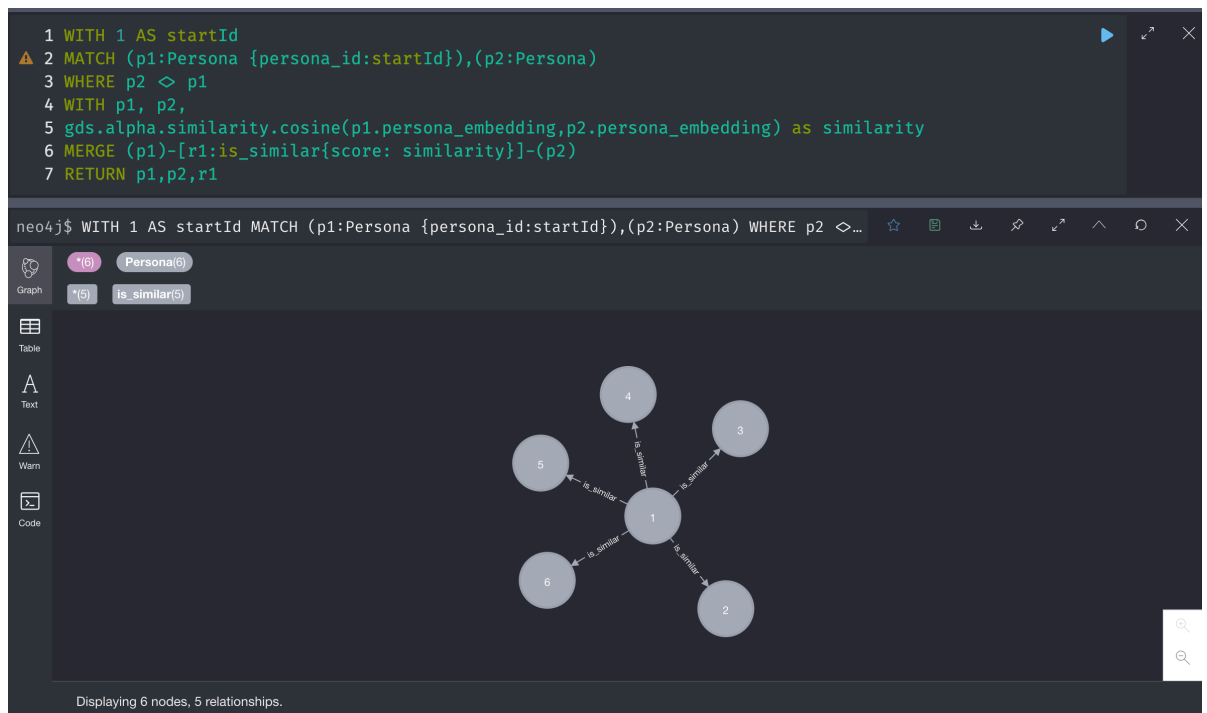
Table

	from	to	similarity
16	"French"	"Mauritian"	0.8913504022120791
17	"French"	"Lebanese"	0.8853775767967607
18	"Italian"	"French"	0.8778358577197801
19	"British"	"French"	0.8384644973081824
20	"British"	"Indian"	0.7717276859027897
21	"British"	"Lebanese"	0.7393113934601527

Code

Started streaming 21 records in less than 1 ms and completed after 5 ms.

Realizar la siguiente consulta para calcular la similitud y crea relaciones is_similar entre el nodo p1 y todos los demás nodos Person que no son p1:



Consultar la similaridad con el nombre de la persona 'Dave Brubeck':

```
1 MATCH (p1: Persona{persona_name:"Dave Brubeck"})-[r:is_similar]-(p2:Persona)
2 WHERE r.score > 0.8
3 RETURN p1.persona_id, r.score, p2.persona_id
```

neo4j\$ MATCH (p1: Persona{persona_name:"Dave Brubeck"})-[r:is_similar]-(p2:Persona) WHERE...

	p1.persona_id	r.score	p2.persona_id
1	1	1.0	5
2	1	0.9819805060619657	4
3	1	0.944911182523068	3

Started streaming 3 records after 1 ms and completed after 2 ms.

In []:

1