

# UNIVERSIDAD POLITÉCNICA SALESIANA

## EL VECINO - CUENCA

**Estudiante:** Gustavo Gualpa

**Profesor:** Ing. Diego Quisi

**Asignatura:** Sistemas Expertos

**Tema:** kNN Classification of members of congress using similarity algorithms in Neo4j

### REQUISITOS

- Neo4j
- Complemento de algoritmos de gráficos Neo4j
- Complemento APOC Neo4j

### Conjunto de datos

Este conjunto de datos incluye votos para cada uno de los congresistas de la Cámara de Representantes de los Estados Unidos sobre los 16 votos clave identificados por la CQA. La CQA enumera nueve tipos diferentes de votos: votó, emparejó y anunció (estos tres se simplificaron a sí), votó en contra, emparejó en contra y anunció en contra (estos tres se simplificaron en contra), votó presente, votó presente para evitar conflicto de intereses, y no votó ni dio a conocer una posición (estos tres se simplificaron a una disposición desconocida).

*LOAD CSV FROM "http://archive.ics.uci.edu/ml/machine-learning-databases/voting-records/house-votes-84.data" as row*

*CREATE (p:Person)*

*SET p.class = row[0],*

*p.features = row[1..];*

### VOTOS FALTANTES

Veamos cuántos miembros del congreso tienen al menos un voto faltante.

*MATCH (n:Person)*

*WHERE "?" in n.features*

*RETURN count(n)*

### RESULTADOS

```
1 MATCH (n:Person)
2 WHERE "?" in n.features
3 RETURN count(n)
```

```
neo4j$ MATCH (n:Person) WHERE "?" in n.features RETURN count(n)
```



Table

count(n)



Text

203



Code

Started streaming 1 records after 2 ms and completed after 71 ms.

Casi la mitad de los miembros del congreso tienen votos faltantes. Eso es bastante significativo, así que profundicemos más. Revisaremos cuál es la distribución de los votos faltantes por miembro.

*MATCH (p:Person)*

*WHERE '?' in p.features*

*WITH p,apoc.coll.occurrences(p.features, '?') as missing*

*RETURN missing,count(\*) as times ORDER BY missing ASC*

**RESULTADOS**



de 430 nodos en nuestro gráfico. Marcaremos 344 nodos como subconjunto de entrenamiento y el resto como prueba

## MARCAR DATOS DE ENTRENAMIENTO

*MATCH (p:Person)*

*WITH p LIMIT 344*

*SET p:Training;*

## RESULTADOS

```
1 MATCH (p:Person)
2 WITH p LIMIT 344
3 SET p:Training;
```

neo4j\$ MATCH (p:Person) WITH p LIMIT 344 SET p:Training;



Table

Added 344 labels, completed after 95 ms.



Code

Added 344 labels, completed after 95 ms.

## MARCAR DATOS DE PRUEBA

*MATCH (p:Person)*

*WITH p SKIP 344*

*SET p:Test;*

```
1 MATCH (p:Person)
2 WITH p SKIP 344
3 SET p:Test;
```

```
neo4j$ MATCH (p:Person) WITH p SKIP 344 SET p:Test;
```



Table



Code

Added 86 labels, completed after 60 ms.

Added 86 labels, completed after 60 ms.

## VECTOR DE CARACTERÍSTICAS

Hay tres valores posibles en los conjuntos de características. Los mapearemos de la siguiente manera:

"Y" a 1

"N" a 0

"?" a 0.5

## TRANSFORMAR A VECTOR DE CARACTERÍSTICAS

```
MATCH (n:Person)
```

```
UNWIND n.features as feature
```

```
WITH n,collect(CASE feature WHEN 'y' THEN 1
```

```
    WHEN 'n' THEN 0
```

```
    ELSE 0.5 END) as feature_vector
```

```
SET n.feature_vector = feature_vector
```

```

1 MATCH (n:Person)
2 UNWIND n.features as feature
3 WITH n,collect(CASE feature WHEN 'y' THEN 1
4                WHEN 'n' THEN 0
5                ELSE 0.5 END) as feature_vector
6 SET n.feature_vector = feature_vector

```

neo4j\$ MATCH (n:Person) UNWIND n.features as feature WITH n,collect(CASE feature WHEN 'y' THEN 1 WHEN 'n' THEN 0 ELSE 0.5 END) as feature\_vector SET n.feature\_vector...

Table Set 430 properties, completed after 128 ms.

Code

Set 430 properties, completed after 128 ms.

## Ejemplo de transformación de conjunto de características a vector de características

| caracteristicas   | característica_vector                                   |
|---|---|
| ["N", "y", "n", "y", "y", "n", "n", "n", "n", "n", "n", "?", "?", "y", "y", "n", "n"] | [0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0.5, 0.5, 1, 1, 0, 0] |

## ALGORITMO CLASIFICADOR KNN

Usaremos la [distancia euclidiana](#) como la función de distancia y el valor top K de 3. Es aconsejable usar un número impar como K para evitar producir casos extremos, donde, por ejemplo, con los dos vecinos superiores y cada uno con una clase diferente, terminamos sin clase mayoritaria, pero una división 50/50 entre los dos.

Tenemos una situación específica en la que queremos comenzar con todos los nodos etiquetados como Prueba y encontrar los tres nodos vecinos principales solo del subconjunto de Entrenamiento. De lo contrario, todos los nodos etiquetados para Prueba también se considerarían parte de los datos de Entrenamiento, que es algo que queremos evitar. Esta es exactamente la razón por la cual no podemos usar una consulta simple como:

*CALL algo.similarity.euclidean(data, {topK: 3, write:true})*

Tendremos que usar `apoc.cypher.run` para limitar los resultados de las coincidencias por fila en lugar de por consulta. Esto nos ayudará a recuperar los 3 primeros vecinos por nodo. Encuentre más en la base de conocimiento .

También usaremos una combinación de [apoc.coll.frequencies](#) y [apoc.coll.sortMaps](#) para recuperar el elemento más frecuente en una colección.

*apoc.coll.sortMaps (apoc.coll.frequencies (colección), '^ count') [- 1])*

## CONSULTA

**MATCH (test:Test)**

**WITH test,test.feature\_vector as feature\_vector**

**CALL apoc.cypher.run('MATCH (training:Training)**

**WITH training,gds.alpha.similarity.euclideanDistance(\$feature\_vector,**  
**training.feature\_vector) AS similarity**

**ORDER BY similarity ASC LIMIT 3**

**RETURN collect(training.class) as classes',**

**{feature\_vector:feature\_vector}) YIELD value**

**WITH test.class as class, apoc.coll.sortMaps(apoc.coll.frequencies(value.classes), '^count')[-**  
**1].item as predicted\_class**

**WITH sum(CASE when class = predicted\_class THEN 1 ELSE 0 END) as correct\_predictions,**  
**count(\*) as total\_predictions**

**RETURN correct\_predictions,total\_predictions, correct\_predictions / toFloat(total\_predictions)**  
**as ratio;**

## RESULTADOS

neo4j\$ MATCH (test:Test) WITH test,test.feature\_vector as feature\_vector CALL apoc.cypher.run('MATCH (training:Training) WITH training,gds.alpha.similarity.euclideanDistance(\$feature\_

Table

Text

Code

| correct_predictions | total_predictions | ratio              |
|---------------------|-------------------|--------------------|
| 78                  | 86                | 0.9069767441860465 |

Started streaming 1 records after 67 ms and completed after 652 ms.