Clasificación de Votos utilizando algoritmos de similitud en Neo4j

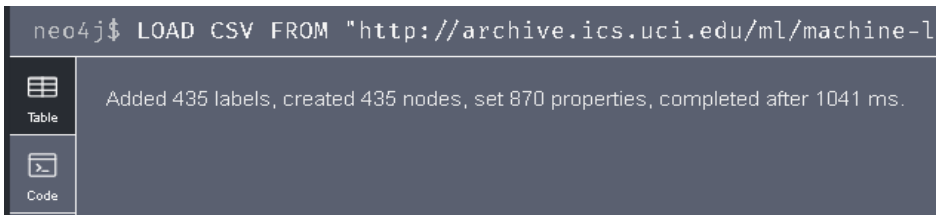
Introducción

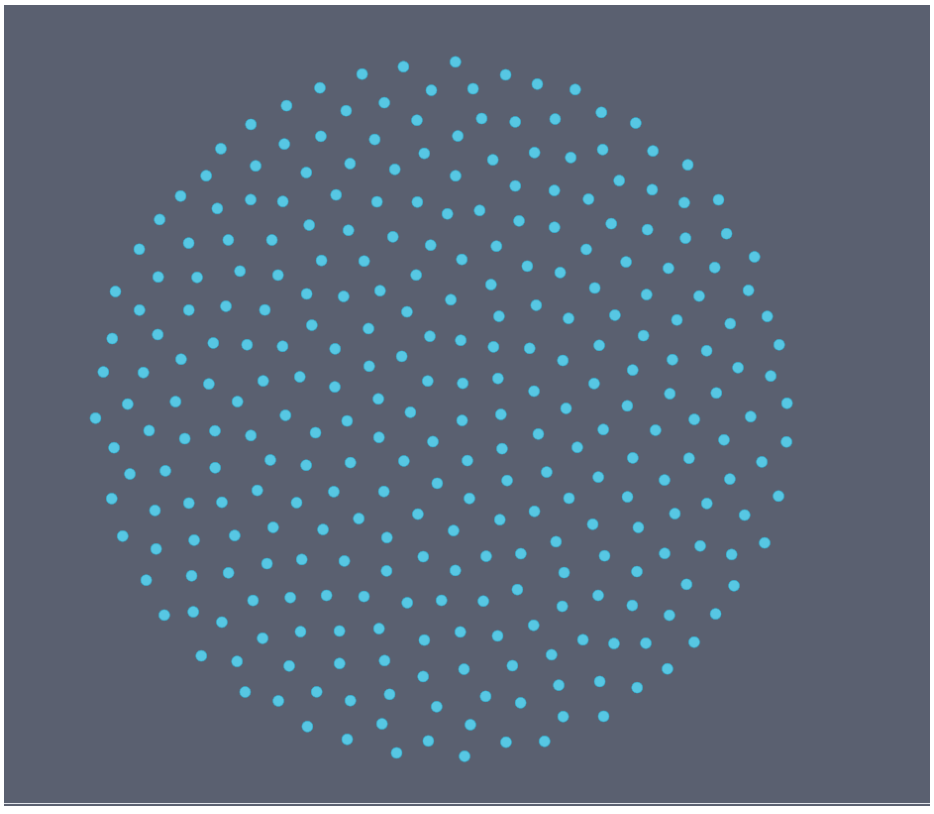
Para el siguiente trabajo procederemos a realizar el análisis de votos que diferentes miembros de l congreso realizaron. Para esto implementaremos el algoritmo de similitud K-NN de neo4j el mismo que nos permitirá calcular la distancia euclidiana y la similitud de cada unos de los casos existentes.

Desarrollo

Para el análisis de los votos que se realizaron, procederemos a cargar los nodos con sus respectivos votos los mismo que se encuentran dentro del un repositorio. Neo4j nos permite cargar archivos csv con diferentes datos desde un repositorio o de forma local.

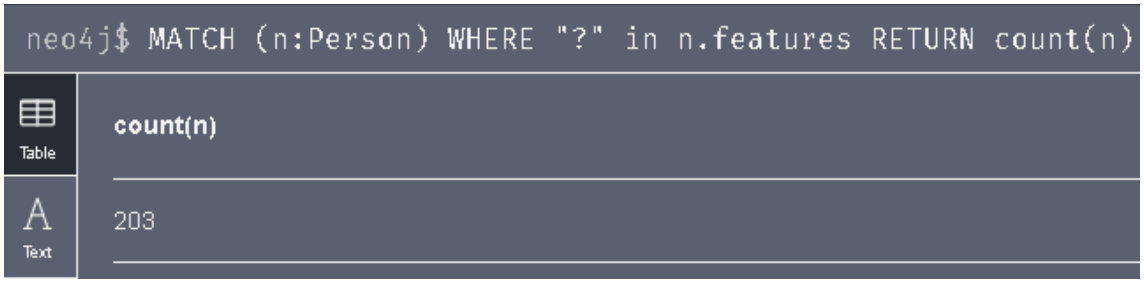
```
LOAD CSV FROM "http://archive.ics.uci.edu/ml/machine-learning-databases/voting-records/house-votes-84.data" as row

CREATE (p:Person)

SET p.class = row[0],

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





Para contar los nodos ingresado realizamos la siguiente consulta. Esta consulta permite contar las personas que no han realizado votos. Cada persona tiene un voto dentro del nodo marcado como “ ? “,

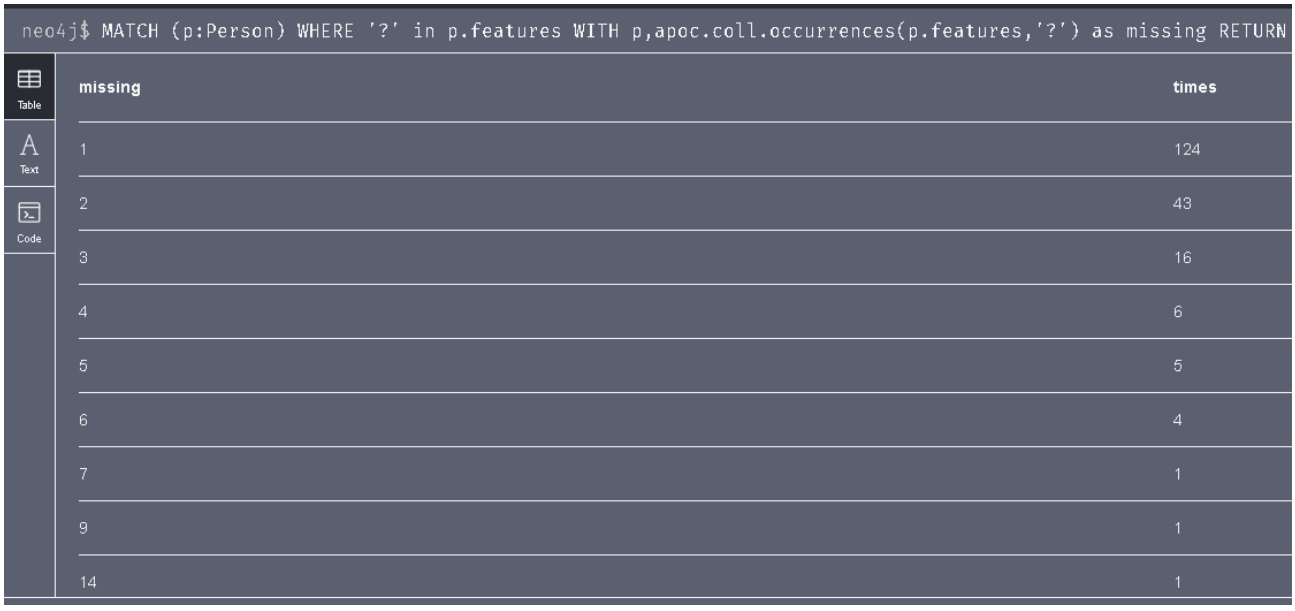
```
MATCH (n:Person)
WHERE "?" in n.features
RETURN count(n)
```



Para visualizar cada persona y la participación con votos de cada uno realizamos la siguiente consulta:

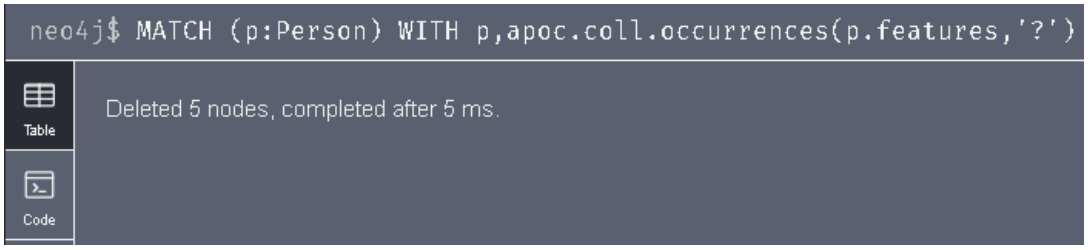
```
WHERE '?' in p.features
WITH p,apoc.coll.occurrences(p.features,'?') as missing
RETURN missing,count(*) as times ORDER BY missing ASC
```

Esta consulta nos permite visualizar con detalle la participación de cada persona con los votos que realizo.



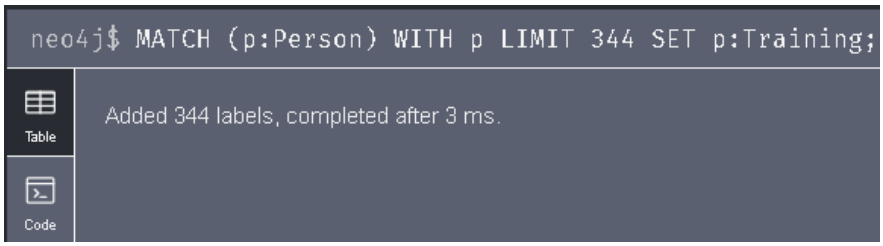
Se procede a borrar aquellas personas que tienen mas de 6 botos faltantes para que permita reducir la separación entre el análisis de cada persona. Para eso se implementa la siguiente sentencia:

```
MATCH (p:Person)
WITH p,apoc.coll.occurrences(p.features,'?') as missing
WHERE missing > 6
DELETE p
```



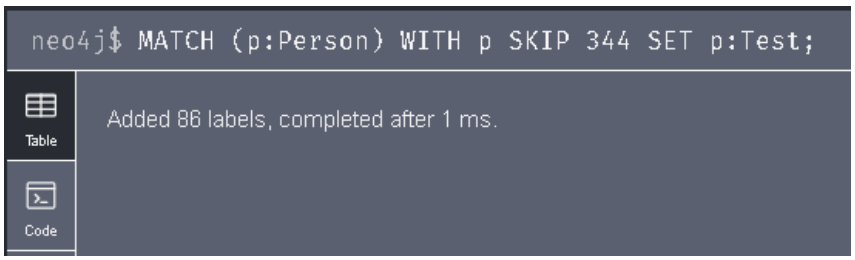
Luego se procede a realizar la asignación de los nodos que serán utilizados como entrenamiento y como prueba. Para ello se implementa el 80% de los nodos que contiene el grafo los cuales serán utilizados para entrenamiento y el 20% de los nodos serán utilizados para pruebas. Para marcar los nodos de entrenamiento utilizaremos la siguiente consulta:

```
PARTIDO (p: Persona)
CON p LIMIT 344
SET p: Entrenamiento;
```



Para los nodos de prueba utilizaremos la siguiente consulta:

```
PARTIDO (p: Persona)
CON p SKIP 344
SET p: Prueba;
```



Luego procedemos a construir nuestro vector el mismo que contiene los diferentes datos que serán 3 posibles valores

“Y” se asignará el valor de 1, lo que significa que a realizado una votación positiva.

“N” se asignará el valor de 0, lo que significa que realizado una votación negativa

“?” se asignara un valor de 0.5 lo que significa que no a realizado una votación.

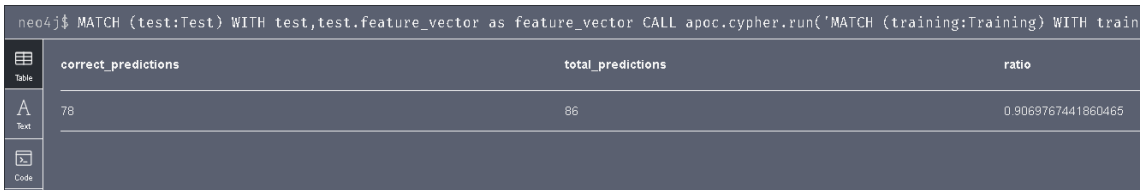
Para la construcción de nuestro vector realizamos la siguiente consulta:

```
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
```

Luego procedemos a realizar el entrenamiento con los nodos que precisamente se marcaron como entrenamiento que son el 70% de los datos dentro del grafo y buscamos los 3 primeros nodos que coincidan de nuestro subconjunto de datos marcados como prueba. Se utiliza los 3 primeros datos encontrados ya que se recomienda que se un numero impar para definir una mejor aproximación dentro de la búsqueda ya que si es numero par no se tendrá una aproximación correcta ya que los nodos se pueden dividir en 50% positivos y 50% negativos.

```
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
```



Resultado

Se obtiene que los nodos predichos correctamente son 78 de un total de 86 con una aproximación de 0.91, este valor podrá ir entre 0 y 1 donde 1 será el valor más próximo que se tendrá o que coincidirán. Se puede ver que el algoritmo realiza una clasificación satisfactoria entre los datos que se marcaron como prueba utilizando los datos marcados como entrenamiento.

Bibliografía

kNN Clasificación de miembros del congreso utilizando algoritmos de similitud en Neo4j. (2018). Recuperado 22 de mayo de 2020, de wordpress website: https://tbgraph.wordpress.com/2018/11/25/knn-classification-using-similarity-algorithms-in-neo4j/