Desarrollo e implementación de un sistema de búsqueda, casos y/o similitud en Neo4j

Angel Jadan

Resumen – El siguiente trabajo consiste en realizar los algoritmos de Neo4j, con datos reales de 2 ciudades principales del Ecuador, para ver como se puede aplicar en la realidad y además como conectar con el lenguaje de programación de php, de esta forma de como se puede mostrar en la web estos algoritmos.

I. INTRODUCCION

El avance tecnológico cada día ha ido mejorando cada vez más, pues una de las tendencias de hoy en día que se ha vuelto como más común es la de análisis de datos y mejorar estos datos, para ello las bases de datos también han ido incrementando y una de las cosas mucho más llamativas que se vienen son las de que las maquinas sean capaces de procesar la información de una forma similar a la de las personas, pues eso es lo que trata de hacer una base de datos orientado a grafos, donde cada uno se porta como individual, y esta se pueda relacionar con muchos otros nodos similares o diferentes, pero con algún tipo de relación como son los de las personas un ejemplo que se puede hacer en la relación de grafos y de la realidad es de los humanos, pues por ejemplo una red social se puede mostrar como las personas se relacionan por sus gustos, lugares que suelen frecuentar, conocidos, etc. De esta forma se puede también relacionar en una base de datos orientado a grafos, simulando esta forma.

II. BASE DE DATOS ORIENTADO A GRAFOS.

Una base de datos no relacional o también llamada NoSQL, es una base de datos completamente diferente a la típica base de datos mas conocida, la diferencia principal entre estas dos bases de datos es que en la no relacional se tiene una flexibilidad para tratar los datos.

A. Ventajas

Una de las grandes ventajas de este tipo de bases de datos es su fácil desarrollo, su funcionalidad, y el rendimiento a escala,

B. Desventajas

Realmente como se una nueva novedad de base de datos, no se tiene un lenguaje como las bases de datos tradicional, el lenguaje que mas se utilizar es el lenguaje llamado, Cypher, pero no se tiene un standard, por lo que se vuelve complejo realizar acciones con esta base de datos.

III. NEO4J

Para el desarrollo en las bases de datos no relacionales existen varias bases de datos como son; Cassandra, Hbase, Redis, Mongodb y Neo4j, este último el que utilizaremos.

Neo4j, es un entorno donde podemos manejar de una forma mucho mas realista, pues cada nodo representa una especie de objeto.

IV. ALGORITMO BETWEENNESS CENTRALITY

Este algoritmo lo que hace es detectar los nodos centrales, los que tienen más nodos relacionados a estos.

En el grafo mostrado se puede ver por los diferentes colores como se agrupan por diferentes grupos tiene una similitud a la de Lovain, pero a diferencia, este agrupa por un nodo que es el principal del grupo.

Conclusiones.
Las bases de datos basadas en grafos son de mucha utilidad para el análisis de datos en la inteligencia artificial, pues además de almacenar los datos, nos permite realizar cálculos por medio de los algoritmos que ya vienen disponibles a través de la librería de neo4j.

V. ALGORITMO DE SIMILITUD DE JACCAR.

Estos algoritmos de similitud buscan realizar una comparación la similitud entre dos nodos, en base a algunas métricas o parámetros. [2]

La similitud de Jaccard (coeficiente) , creado por Paul Jaccard, este algoritmo lo que hace es que mide las similitudes entre conjuntos. Se define como el tamaño de la intersección dividido por el tamaño de la unión de dos conjuntos. [2]

Ecuación.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{}{|A|+|}$$

Donde como ya lo indicado la el valor absoluto de la intersección entre A y B se divide para; el valor absoluto de A+B restado el absoluto de la intersección de A con B.

El rango que va para esta similitud va desde 0 hasta 1, donde 0 es cuando no comparten similitudes y 1 cuando comparten similitudes.

VI. COMMUNITU DETECTION

1. Strongly Connected Components.

El algoritmo de Componentes Fuertemente Conectados, encuentra conjuntos máximos de nodos conectado en un grafico dirigido. Un conjunto se considera un componente fuertemente conectado si hay una ruta dirigida entre cada par de nodos dentro del conjunto. A menudo se usa el principio de un proceso de análisis de gráficos para ayudarnos a tener una idea de como esta estructurado nuestro gráfico.

Casos de uso.

- En el mundo de los negocios, podemos usar para encontrar un conjunto de empresas en donde haya directa o indirectamente miembros que posean acciones.
- El algoritmo de componentes fuetemente conectados puede ser utilizado en las redes soliciales, para el uso en algoritmos que son solo de gráficos, en una red social existen múltiples conexiones, a esto nos referimos con las conexiones entre amistades, por lo que se puede ver claramente grafos relacionados entre sí.
- Pero no solo se puede realizar algoritmos para estas conexiones, pues también en otras áreas, pues un ejemplo podemos ver en las redes, pues podemos medir la conectividad en los diferentes puntos como son los routers y switchs, y de esta forma medir su rendimiento.

VII. PROYECTO APLICADO.

Para la demostración de estos proyectos se ha utilizado como datos de muestra la información de lugares como; parques, iglesias, estaciones de bomberos de las ciudades de Quito y Guayaquil, para realizar las prácticas de estos algoritmos.

Se utiliza el lenguaje de programación php para la conexión y presentación de estos datos procesados en la base de datos.

Debemos tener instalado composer en nuestro equipo de desarrollo, con la siguiente línea de código podemos instalar la librería de neo4j.

Luego con el siguiente codigo lo que hacemos es realizar la conexión a la base de datos de neo4j.

VIII. PROYECTO

Para poner en practica la teoría, una de las mejores formas de probar cada algoritmo lo vamos a aplicar en un proyecto, para lo cual lo primero que vamos hacer es buscar la información. La información que se va ha utilizar es del candidato Andrés Arauz, candidato a la presidencia del Ecuador en las elecciones del 2021, el mismo que se encuentra como Andrés Arauz (@ecuarauz).

De esta plataforma bajaremos la información utilizando la api de Twitter, que nos facilita la extracción de los datos, para ello lo que vamos hacer es bajar los tweets, relacionados con este usuario y los usuarios que retweetiaron.

Como la cantidad de usuarios es bastante con el uso de Python se descarga estos datos y se le manda a guardar la información en la base de datos de neo4j.

A. Extracción de datos con api de twitter.

Lo primero que tenemos que hacer para extraer la información es registrarnos en twitter, luego en la sección de desarrolladores, registrarnos y obtener nuestro token, pero también necesitaremos los siguientes;

- Consumer_key
- Consumer_secret
- Access_token
- Access_token_secret

En el link podemos obtener lo que requerimos para obtener la información, y en cada librería que usermos la documentación para obtener los datos.

La librería que se utilizo en este proyecto es la librería de tweepy, de Python, para instalar solo debemos poner el comando *pip install tweepy* y se instalara la librería.

El siguiente código es para la conexión a la api de twitter.

```
import tweepy
import json, csv, sys

consumer_key = ""
consumer_secret = ""
access_token = ""
access_token_secret = ""

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)

auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)

#Obtner datos de mi usuario
data = api.me()

print(json.dumps(data._json, indent=2))
```

Es importante poner el parámetro “wait_on_rate_limit=True” y “wait_on_rate_limit_notify=True”, estos dos parámetros son muy útiles para realizar la extracción de grandes cantidades de datos, pues la api de twitter nos permite extraer por cierta cantidad de tiempo, ya luego de esto hay que esperar 15 minutos, entonces estos dos parámetros nos ayuda a que nuestra aplicación, se detenga y consulte luego de este tiempo sin tener que cortarse la consulta y vuelva empezar desde el comienzo, ya que solo continuara en donde se había quedado.

```
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
```

En la variable data, es donde almacenamos toda la información de nuestro perfil en twitter.

```
data = api.me()
```

Esto es muy útil, cuando queremos usar alguno bot o acción para dar likes, retwittear o responder mensajes, ya que administra nuestro perfil, pero para casos de búsqueda de otros usuarios tendremos que usar otro código.

```
data = api.followers(screen_name="ecuarauz")
print(len(data))
```

El codigo anterior, nos ayuda realizar la búsqueda de un usuario en este caso usamos “ecuarauz” que es del candidato Andres Arauz.

Fig. Perfil del usuario.

a. Buscar tweets de una persona.

En el siguiente código usamos un for, para recorrer el arreglo de la información y lo definimos el numero de ítems (5000), pues aquí es donde es importante que hayamos definido que esperar y continuar, mencionado anteriormente, pues de lo contrario, al llegar a los 20 items terminara la consulta.

```
for tweet in tweepy.Cursor(api.search, q="andres arauz", tweet_mode =
"extended").items(5000):
```

b. Extraer información de un tweet.

En el siguiente codigo,

```
item = json.loads(json.dumps(tweet._json, indent=3))
'''Datos de twit'''
iditem = item['id']
fecha = item['created_at']
sms = item['full_text']
dipositivo = item['source']
```

En la variable de item, guardamos toda la informacion del tweet actual en un formato json, ahora solo tenemos que acceder a cada una de las propiedades, en este caso accedemos al ‘id’, ‘la fecha creada’, ‘el mensaje del tweet’ y el dispositivo desde donde fue publicado.

Pero también requerimos obtener la informacion del usuario que publico, para ello vamos a utilizar el siguiente codigo.

```
user = item['user']
useid = user['id']
usename = user['name']
usedescription = user['description']
useaddress = user['location']
usefollowers = user['followers_count']
usefriends = user['friends_count']
```

Ya en le arreglo de item tenemos otro arreglo con la clave de user, esta almacena toda la informacion del usuario que publico el tweet, entonces esto la almacenamos en una variable user, a donde accederemos para obtener cada uno de los datos, como son el identificador ‘id’, el nombre del usuario ‘name’, la descripcion ‘description’, ubicación ‘location’, numero de seguidores ‘followers_count’ y numero de amigos ‘fiends_count’, esto es muy útil para almacenar en la base de datos de neo4j, representándola por nodos y poder sacar una similitud en base a estos datos. [1]

B. Conexión a neo4j con Python.

Una herramienta muy útil para almacenamiento y proceso de la base de datos es la conexión a neo4j, existen algunas librerías para esta conexión como son: neo4j, Py2neo y Neomodel.

Para el proyecto se ha considerado utilizar la librería Neomodel, pues es muy útil, ya que se maneja mediante modelos, pues de esta forma podemos declarar modelos en Python y recoger los datos para mandar a guardar en la base de datos.
Para instalar lo único que se tiene que hacer en la consola de nuestro sistema operativo es:

pyp install neomodel

1. Conexión

```
from neomodel import (config, StructuredNode, StringPro
    UniqueIdProperty, RelationshipTo, RelationshipFrom)
```

Fig. En la imagen podemos ve que la conexión se realiza de una forma muy sensilla, pues, solo tenemos que poner el nombre del usuario(neo4j), la clave(password), el dominio(localhost) y el puerto(7678)

2. Declaración de modelos.

```
class Candidat(StructuredNode):
    nameCan = StringProperty(unique_index=True)
    partido = StringProperty(unique_index=True)

class Twee(StructuredNode):
    idtweet = UniqueIdProperty()
    date = StringProperty(unique_index=False)
    messaje = StringProperty(unique_index=False)
    device = StringProperty(unique_index=False)
    usuario = RelationshipTo('Usuario' , 'TWEET_
    candidato = RelationshipTo('Candidat' , 'TWE

class Usuario(StructuredNode):
```

Fig. En la imgagen podemos ver la declaracion de las clases que vamos a utilizar estos son lo modelos para insertar en la base de datos, pero la clase debe de heredar de una clase maestro como es StructureNode, que trae la librería.

3. Inserccion a la base de datos.

```
tw = Twee(idTeet=iditem, date=fecha, messaje=sms,device=dispositi
us =  Usuario(userid=useid, name=usename, description=usedescrip
            followers=usefollowers, friends=usefriends).save()
```

Fig. En esta imagen, se realiza la inserccion de los datos en la base de datos, luego de extraer la informacion de la api de twiter en una formato json.

4. Resultado con extracción de datos de twitter.

```
#q <- es la busqueda por texto el tweet que se desee
c=0
cand = Candidat(nameCand="Andres Arauz", partido="UNES").save()
for tweet in tweepy.Cursor(api.search, q="andres arauz", tweet_mode =
    '''Para mostrar todos los datos'''
    print("************************"+str(c))
    #print(type(tweet))
    item = json.loads(json.dumps(tweet._json, indent=3))
    '''Datos de twit'''
    iditem = item['id']
    fecha = item['created_at']
    sms = item['full_text']
    dipositivo = item['source']
    print(fecha)
    print(iditem)
    print(sms)
    print(dipositivo)

    '''Datos de usuario'''
    user = item['user']

    useid = user['id']
    usename = user['name']
    usedescription = user['description']
    useaddress = user['location']
    usefollowers = user['followers_count']
    usefriends = user['friends_count']
    print(useid)
    print(usename)
    print(usedescription)
    print(useaddress)
    print(usefollowers)
```

C. Nodos en neo4j.

Para almacenar los datos en neo4j, lo hemos hecho en 3 tipos de nodos, en donde esta el candidato a la presidencia “Andrea Arauz”, el usuario que twittea y el tweet. [2]

Fig. Grafo del paritodo UNES(Andres Arauz).

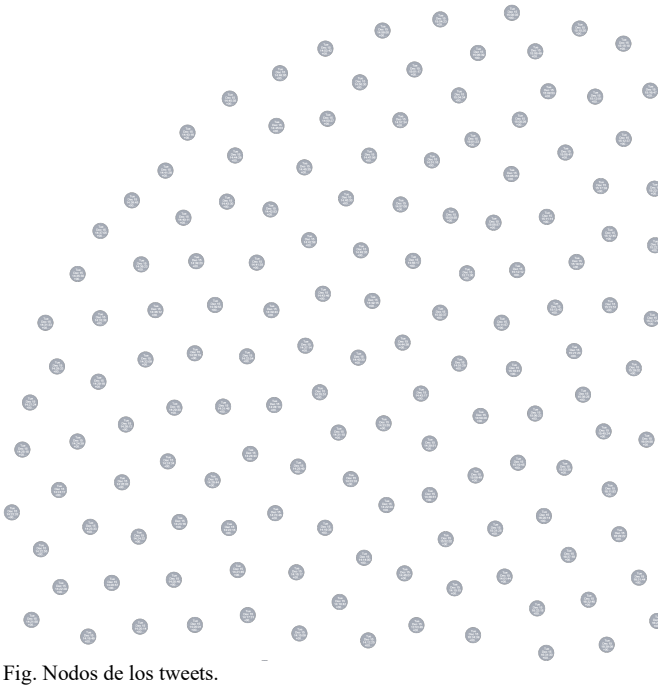


Fig. Nodos de los tweets.

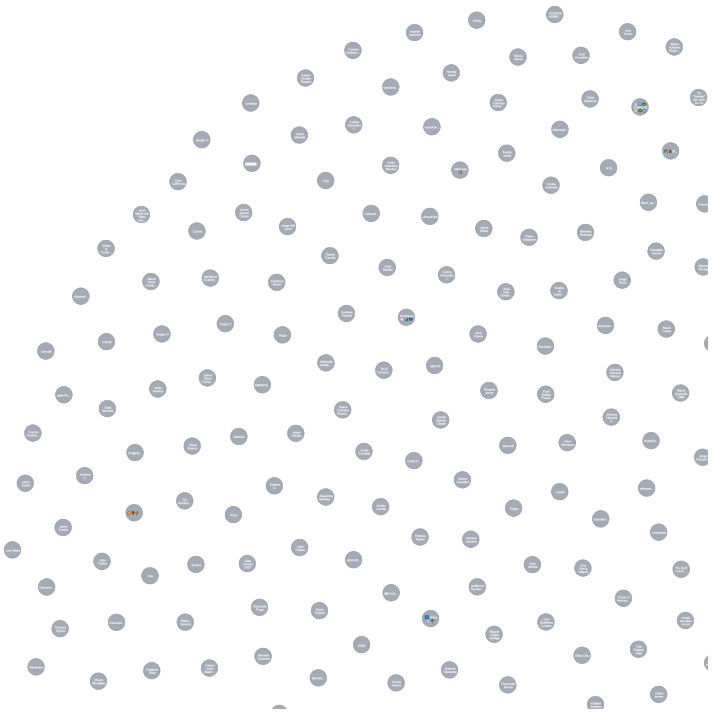


Fig. Nodos de usuarios registrados.

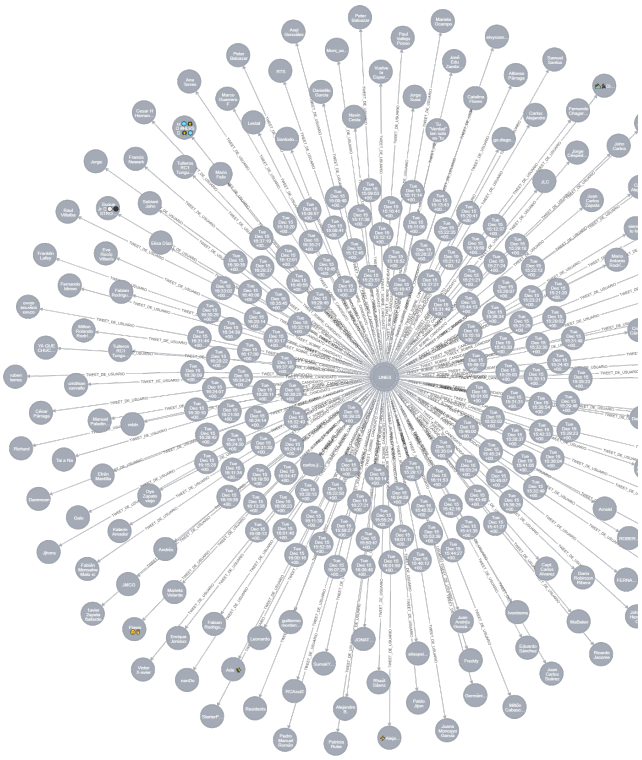
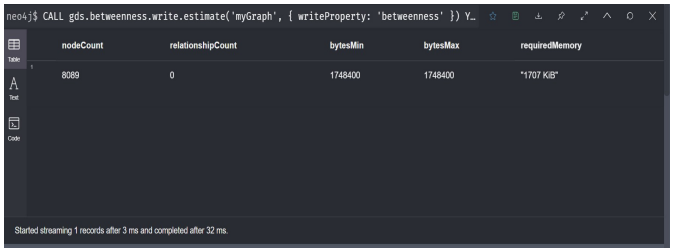


Fig. Todos los nodos, tenemos dos nodos, pues en la mas amplia esta la que contiene la mayor cantidad de nodos, luego esta una secundaria, que nos sirvió para probar con los 20 datos que a principio la api de twitter nos permite bajar.

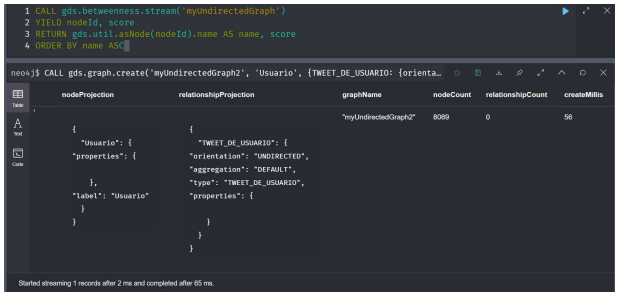
D. Corrida de algoritmos.

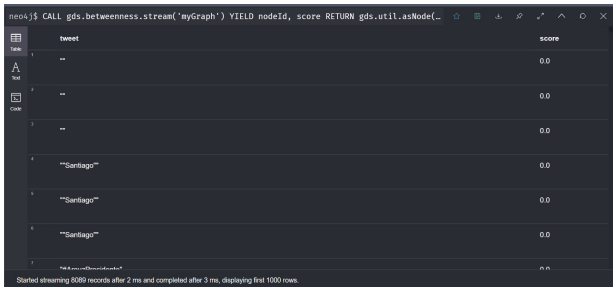
a. Betweenness Centrality.



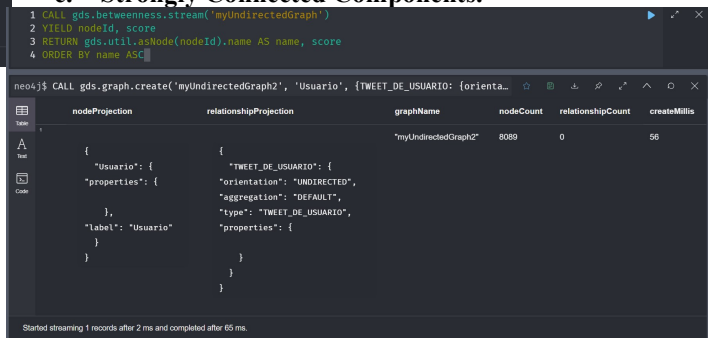
b. Jaccard Similirity.

```
MATCH (u1:Usuario {name: 'Wilmer Alexis Suare'})-
[:TWEET_DE_USUARIO]->(twee1)
WITH u1, collect(id(twee1)) AS u1Usuario
MATCH (u2:Usuario {name: "Carlos leonardo"})-
[:TWEET_DE_USUARIO]->(twee2)
WITH u1, u1Usuario, u2, collect(id(twee2)) AS u2Usuario
RETURN u1.name AS from,
       u2.name AS to,
       gds.alpha.similarity.jaccard(u1Usuario, u2Usuario) AS
similarity
```

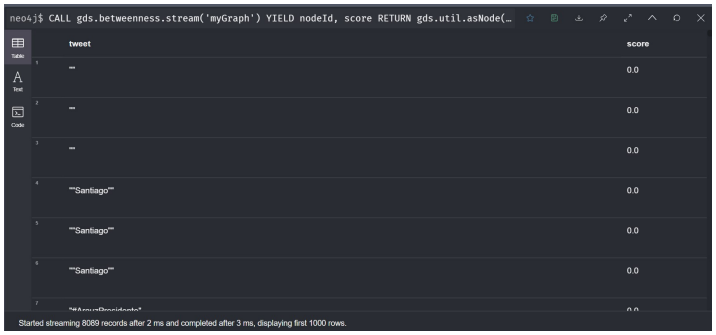




c. Strongly Connected Components.



[3] tweepy, «tweepy,» tweepy, 12 2020. [En línea]. Available: http://docs.tweepy.org/en/latest/. [Último a



IX. CONCLUSIONES.

Las bases de datos orientadas a grafos nos dan una gran ventaja para realizar lo que son estudios de datos, por lo que al unir con los lenguajes de programación nos son de mucha utilidad para utilizar en nuestra vida cotidiana, pues en combinación con otras tecnologías podemos realizar grandes proyectos que serían de gran utilidad en la vida diaria de las personas. Un ejemplo con el proyecto realizado es que podemos analizar datos, con minería de datos o consumo de servicios o microservicios podemos correr diferentes algoritmos y sacar resultados para diferentes estudios.

X. BIBLIOGRAFÍA

[1] B. S. López, «Ingeneria Industrial,» 12 6 2019. [En línea].
Available: https://www.ingenieriaindustrialonline.com/investigacion-de-operaciones/algoritmo-de-la-ruta-mas-corta/. [Último acceso: 1 6 2020].

[2] neoej, «neo4j algoritms,» [En línea]. Available:
https://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&tl=es&u=https://neo4j.com/docs/graph-data-science/current/algorithms/&usg=ALkJrhjjckY1o4lkvqMKyrxjmzIPLgf5Nw. [Último acceso: 1 06 2020].