Sistema que ayuda a la búsqueda de patrones de datos para las Elecciones 2021 en Ecuador

Andrés Bermeo, *UPS, Universidad Politécnica Salesiana*

***Abstract*—This document tries to explain how a system works by using graphics algorithms. Using Neo4j's own query language called Cypher. The Neo4j Database is a Graph-oriented database (BDOG) represents the information as nodes of a graph and their relationships with the edges of the graph, so that graph theory can be used to traverse the database. In graph-oriented databases, information is represented by nodes and edges. There are different graph-oriented databases and each one has its peculiarities. This type of database is especially suitable to help us with everything that has to do with relationships. Analyze social networks, identify influencers, Implement recommenders based on similarities and affinities, detect fraud, etc.**

***Index Terms*—Neo4j, Python, Web, LaTeX.**

I. Introduction

EL presente documento trata de explicar como funciona un sistema que mediante la utilización de algoritmos de gráficos. Mediante el propio lenguaje de consultas de Neo4j llamado Cypher. La base de Datos Neo4j es una base de datos orientados a Grafos(BDOG) representa la información como nodos de un grafo y sus relaciones con las aristas del mismo, de manera que se pueda usar teoría de grafos para recorrer la base de datos. En las bases de datos orientadas a grafos la información se representa mediante nodos y aristas. Hay distintas bases de datos orientadas a grafos y cada una presenta sus peculiaridades. Este tipo de bases de datos están especialmente indicadas para ayudarnos con todo lo que tenga que ver con las relaciones. Analizar redes sociales, identificar influencers, Implementar recomendadores en función de similitudes y afinidades, detectar fraudes, etc.

mds Enero 04, 2021

II. API TWITTER

Para la extracción de los datos y poder poblar nuestra base de datos utilizamos la API desarrollada por Twitter, dicha API necesita un mecanismo de Autentificaciones que la podemos realizar dentro de la pagina Developer en Twitter.

La clase API proporciona acceso a todos los métodos API RESTful de Twitter. Cada método puede aceptar varios parámetros y devolver respuestas. Para obtener más información sobre estos métodos.

Cuando invocamos un método de API, la mayor parte del tiempo que se nos devuelve será una instancia de clase de modelo Tweepy. Este contendrá los datos devueltos de Twitter que luego podremos usar dentro de nuestra aplicación. Por ejemplo, el siguiente código nos devuelve un modelo de usuario:

```
user = api.get_user('twitter')
```

Los modelos contienen los datos y algunos métodos auxiliares que luego podemos usar:

```
print(user.screen_name)
print(user.followers_count)
for friend in user.friends():
    print(friend.screen_name)
```

Usamos mucho la paginación en el desarrollo de API de Twitter. Iterando a través de líneas de tiempo, listas de usuarios, mensajes directos, etc. Para realizar la paginación, debemos proporcionar un parámetro de página / cursor con cada una de nuestras solicitudes. El problema aquí es que esto requiere mucho código de placa de caldera solo para administrar el ciclo de paginación. Para facilitar la paginación y requerir menos código, Tweepy tiene el objeto Cursor.

III. Algoritmos de Neo4j

La sintaxis general del algoritmo se presenta en dos variantes:

- Variante de gráfico con nombre: El gráfico sobre el que operar se leerá del catálogo de gráficos.
- Variante de gráfico anónimo: El gráfico sobre el que operar se creará y eliminará como parte de la ejecución del algoritmo.

Cada variante de sintaxis proporciona además diferentes modos de ejecución. Estos son los modos de ejecución admitidos:

- stream: Devuelve el resultado del algoritmo como una secuencia de registros.
- stats: Devuelve un único registro de estadísticas resumidas, pero no escribe en la base de datos de Neo4j.
- mutate: Escribe los resultados del algoritmo en el gráfico en memoria y devuelve un único registro de estadísticas resumidas. Este modo está diseñado para la variante de gráfico con nombre, ya que sus efectos serán invisibles en un gráfico anónimo.
- write: Escribe los resultados del algoritmo en la base de datos de Neo4j y devuelve un único registro de estadísticas resumidas.

Finalmente, se puede estimar un modo de ejecución agregando el comando con estimate.

A. Community detection algorithms

Label Propagation

Las redes sociales se han extendido por todo el mundo y están creciendo día a día. Considere una red de medios sociales en la que conozca los intereses de algunas personas y desee predecir los intereses de otras para que podamos orientar las

campañas de marketing. Para este propósito, podemos utilizar la técnica de aprendizaje automático semi-supervisado basada en gráficos llamada Label Propagation.

Ejemplo Teórico

Suponga que tenemos una red de personas como se indica a continuación con dos clases de etiquetas ” interesado en el cricket ” y ” no interesado en el cricket ”. Entonces, la pregunta es, ¿podemos predecir si las personas restantes están interesadas en el cricket o no?.

Para que LPA funcione en este caso, tenemos que hacer una suposición; un borde que conecta dos nodos conlleva una noción de similitud . es decir, si dos personas están conectadas, eso significa que es muy probable que estas dos personas compartan los mismos intereses. Podemos hacer esta suposición ya que las personas tienden a conectarse con otras personas que tienen intereses similares.

- Verde: Personas que les gusta el Cricket.
- Rojo: Personas que no les gusta el Cricket.

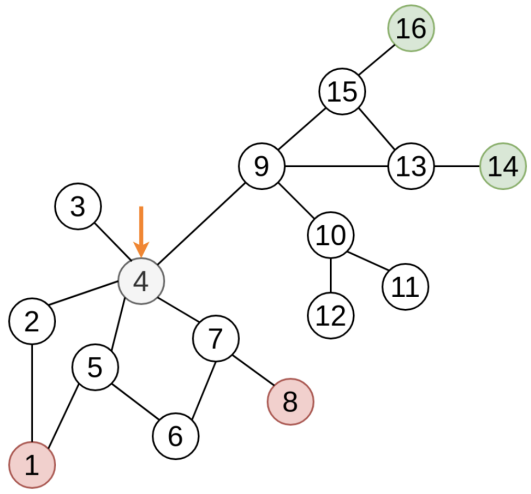

Fig. 1. Implementación del Algoritmo Label Propagation.

Mediante el nodo 4 podemos decir que si llegamos al extremo de cada nodo vamos a poder predecir si llegamos a cada extremo vamos a poder decir que existen personas que les guste el cricket o que no.

B. Similarity algorithms

Node Similarity

El algoritmo de similitud de nodos compara un conjunto de nodos en función de los nodos a los que están conectados. Dos nodos se consideran similares si comparten muchos de los mismos vecinos. Node Similarity calcula las similitudes por pares según la métrica Jaccard, también conocida como Jaccard Similarity Score.

Existe el análisis con base a funciones, que se utiliza normalmente para estudiar la similitud entre pequeños números de conjuntos y adicionalmente se cuenta con la ejecución del algoritmo por procedimientos que esta pensada para analizar similitudes en conjuntos de datos grandes.

Ejemplo Teórico

Mediante el Algoritmo se realizara un predicción acerca de

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

Fig. 2. Ecuación de Jaccard.

que fruta se obtendrá de una determinada canasta.

Se va a estudiar la intersección de los nodos establecidos,

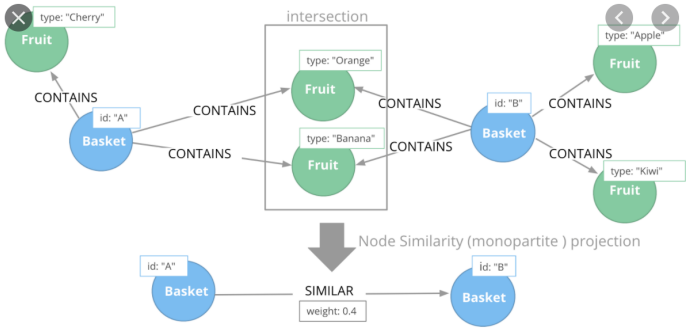

Fig. 3. Algoritmo Node.

mediante la construcción de puntos en este de A y B, se establece una forma mas rápida de llegar, dicho algoritmo nos puede ayudar para establecer las similitudes entre usuarios, ya se para poder encontrar un patrón entre los nodos establecidos.

C. Centrality algorithms

Betweenness Centrality

La centralidad de intermediación es una forma de detectar la cantidad de influencia que tiene un nodo sobre el flujo de información en un gráfico. A menudo se utiliza para encontrar nodos que sirvan de puente entre una parte de un gráfico y otra.

El algoritmo calcula las rutas más cortas no ponderadas entre todos los pares de nodos de un gráfico. Cada nodo recibe una puntuación, basada en el número de rutas más cortas que pasan por el nodo. Los nodos que se encuentran con mayor frecuencia en las rutas más cortas entre otros nodos tendrán puntuaciones de centralidad de intermediación más altas.

Es una medida dentro de la Teoría de Grafos que determina cuán importante es un vértice dentro de una red en particular. Esto puede ayudar a entender desde cuan importante es una persona dentro de una red social, importancia de una avenida dentro de una ciudad, etc…

Ejemplo Teórico

Veamos un ejemplo práctico de aplicación de este tipo de algoritmo dentro de una red social. Con el se pretenderá determinar dentro de un grafo, la centralidad de intermediación de un numero determinado de usuarios para conocer la capacidad de influenciar en el entorno.

IV. Descripción del Problema

Mediante el uso de la API de Twitter logramos poblar la base de datos orientadas a Grafos utilizando los tweets

emitidos por los usuarios basados en las palabras claves del candidato a la presidencia del Ecuador: Guillermo Lasso.

Estos tweets serán procesados utilizando los diferentes algoritmos de Machine Learning que utiliza Neo4j y estadísticamente llegaremos a un análisis de los diferentes resultados.

V. Resolucion

1) Obtener los datos de Twitter.
   Implementamos la librería tweepy, neomodel y neo4j para la conexión y la correspondiente extracción.

Fig. 4. Librerías

Utilizamos la librería de neo4j para poder validar la conexión a la base de datos.
Para poder realizar la extracción utilizamos la librería de Tweepy, primero tenemos que establecer la autentificacion establecida para poder utilizarla.

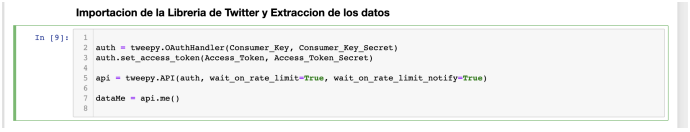

Fig. 5. Tweepy

2) Creación de Nodos en Neo4j.
   Mediante la librería NeoModel creamos los modelos de los nodos, que van a ser Candidato, Tweet y Usuario.

Fig. 6. Nodos

Se establecen las relaciones en las clases determinadas para establecer las aristas de los nodos.

Mediante un for que se establece dentro de un cursor, y se crea un JSON que vamos a ir recorriendo sacando los tipos de datos que necesitamos guardar dentro de los nodos.

3) Aplicación del Algoritmo Label Propagation.
   Utilizando la Base de datos tratamos de buscar un tipo de comunidad.

Fig. 7. Programación

```
:param limit => ( 42);
:param config => ({
  nodeProjection: 'Candidato',
  relationshipProjection: {
    relType: {
      type: 'Tweet_del_Candidato',
      orientation: 'UNDIRECTED',
      properties: {}
    }
  },
  relationshipWeightProperty: null
});
:param communityNodeLimit => ( 10);
```

Fig. 8. Label Propagation

Se utiliza la relación tweet del Candidato con el Candidato para ver que tipo de comunidad se forma. Se realiza la llamada al nuevo Nodo mediante la librería GDS de Neo4j

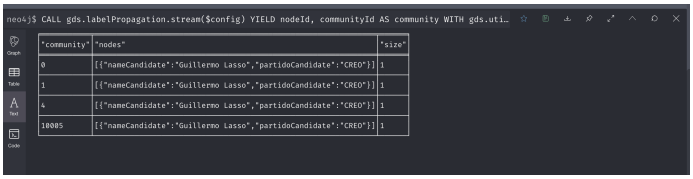

Fig. 9. Llamada del Algoritmo

Se calculo cuando de procesamiento se utilizo para poder llegar al resultado.

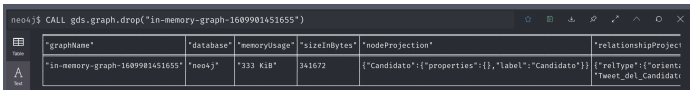

Fig. 10. Estimación de Procesamiento.

4) Aplicacion del Algoritmo Node Similarity.
   Se busca la similitud de mensajes entre los tweets de los seguidores de Guillermo Lasso.

   No se encontraron algun similitud entre los seguidores de Guillermo Lasso.

5) Aplicacion del Algoritmo Betweenness Centrality.
   Se utiliza la centralización de provincias entre el

```
:param limit ⇒ ( 42);
:param weightProperty ⇒ ('messageTweet');
:param config ⇒ ({
  nodeProjection: '*',
  relationshipProjection: '*',
  writeProperty: 'score',
  writeRelationshipType: 'SIMILAR_COSINE',
  similarityCutoff: 0.1,
  degreeCutoff: 0
});
:param communityNodeLimit ⇒ ( 10);
```

Fig. 11. Node Similarity.

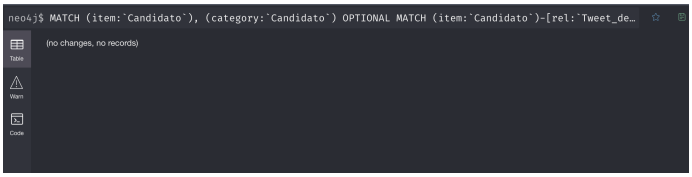

Fig. 12. Resultados

Candidato y la Direccion del Tweet,

Como podemos observar en los resultados no existen

```
:param limit ⇒ ( 42);
:param config ⇒ ({
  nodeProjection: 'Candidato',
  relationshipProjection: {
    relType: {
      type: 'Direccion_del_Tweet',
      orientation: 'NATURAL',
      properties: {}
    }
  }
});
:param communityNodeLimit ⇒ ( 10);
```

Fig. 13. Betweenness Centrality.

una similitud de provincias entre los mensajes de los usuarios seguidores de Guillermo Lasso.

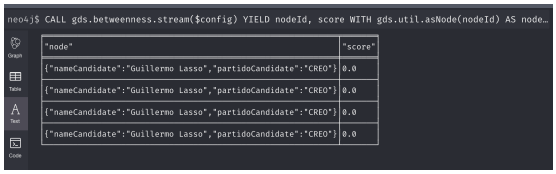

Fig. 14. Llamada al Algoritmo.

Pero encontrar una similitud entre los tweet que tiene que ver en su mayoría con las provincias de Esmeraldas.

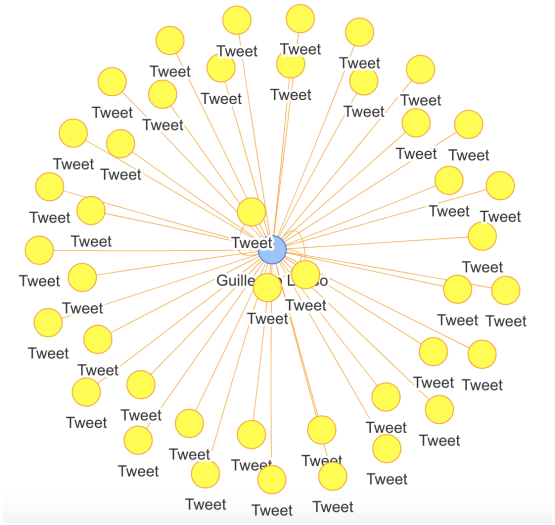

Fig. 15. Tweets que tienen similitud.

VI. Conclusión

Mediante la utilización de diferentes algoritmos hemos logrando mostrar el comportamiento de un comunidad, la búsqueda de las diferentes similitudes establecidas y la centralización de un comportamiento dentro de un nodo.