Proyecto Integrador IA-SE

Jonathan Atancuri
jatancurio@est.ups.edu.ec
Esteban Sibri
esibri@est.ups.edu.ec

***Abstract*—Neo4j created the first business graphics framework for data scientists to improve predictions that drive better decisions and innovation. Neo4j for Graph Data Science incorporates the predictive power of relationships and network structures in existing data to answer previously intractable questions and increase prediction accuracy.**

The Neo4j Graph Data Science ™ library is the analysis engine of this framework, allowing you to address complex questions about system dynamics and group behavior. Data scientists benefit from a flexible, custom data structure for global calculations and a repository of powerful and robust algorithms to quickly calculate results across tens of billions of nodes.

Chart algorithms provide unsupervised heuristics and machine learning methods that learn and describe the topology of your chart. The GDS ™ library includes graphics algorithms hardened with business functions, such as deterministic seeding, for consistent results. And with embedded charts and trained models within the analysis workspace, you can make predictions about your chart from within Neo4j.

Graphical Data Science Algorithms A subset of data science algorithms derived from network science, graphical algorithms allow reasoning about the structure of the network

***Index Terms*—Neo4j**

I. Introduccion

Neo4j creó el primer marco de gráficos empresariales para que los científicos de datos mejoren las predicciones que impulsan mejores decisiones e innovación. Neo4j para Graph Data Science incorpora el poder predictivo de las relaciones y las estructuras de red en los datos existentes para responder preguntas previamente intratables y aumentar la precisión de la predicción.[1]

La biblioteca de Neo4j Graph Data Science ™ es el motor de análisis de este marco, lo que permite abordar preguntas complejas sobre la dinámica del sistema y el comportamiento del grupo. Los científicos de datos se benefician de una estructura de datos personalizada y flexible para cálculos globales y un repositorio de algoritmos potentes y robustos para calcular rápidamente los resultados en decenas de miles de millones de nodos.[1]

Los algoritmos de gráficos proporcionan métodos de aprendizaje automático y heurísticos sin supervisión que aprenden y describen la topología de su gráfico. La biblioteca GDS ™ incluye algoritmos de gráficos reforzados con funciones empresariales, como la siembra determinista para obtener resultados consistentes. Y con incrustaciones de gráficos y modelos entrenados dentro del espacio de trabajo de análisis, puede hacer predicciones sobre su gráfico desde dentro de Neo4j.[1]

Algoritmos de ciencia de datos gráficos Un subconjunto de algoritmos de ciencia de datos que provienen de la ciencia de redes, los algoritmos gráficos permiten razonar sobre la estructura de la red

Las categorías de algoritmos proporcionados con la biblioteca de ciencia de datos de gráficos de Neo4j: Los algoritmos de detección de comunidades agrupan su gráfico en función de las relaciones para encontrar comunidades donde los miembros tienen interacciones más significativas. Esta categoría incluye algoritmos populares, como componentes conectados y modularidad de Louvain. La detección de comunidades ayuda a predecir comportamientos similares, encontrar entidades duplicadas o simplemente preparar datos para otros análisis. Los algoritmos de centralidad revelan qué nodos son importantes según la topología del gráfico. Identifican los nodos influyentes según su posición en la red e incluyen el famoso algoritmo PageRank. Estos algoritmos se utilizan para inferir dinámicas de grupo como credibilidad, vulnerabilidad ondulante y puentes entre grupos. Los algoritmos de integración de nodos transforman la topología y las características de su gráfico en vectores de longitud fija que representan de forma única cada nodo. Las incrustaciones de gráficos son poderosas porque conservan las características clave al tiempo que reducen la dimensionalidad de una manera que se puede decodificar. Las incrustaciones capturan la complejidad y la estructura de un gráfico y lo transforman para usarlo en varias tareas de aprendizaje automático. Los algoritmos de similitud emplean comparaciones de conjuntos para puntuar qué tan parecidos son los nodos individuales en función de sus vecinos o propiedades. Las propiedades y atributos de los nodos se utilizan para puntuar la semejanza entre los nodos. Este enfoque se utiliza en aplicaciones como recomendaciones personalizadas y desarrollo de jerarquías categóricas. Los algoritmos de predicción de enlaces consideran la proximidad de los nodos en un gráfico, así como los elementos estructurales, como los posibles triángulos entre los nodos, para predecir la probabilidad de que se forme una nueva relación en el futuro o que existan conexiones no documentadas. El apego preferencial se incluye en esta clase de algoritmos que tiene muchas aplicaciones, desde la reutilización de drogas y la estimación de la colaboración hasta las investigaciones penales. Los algoritmos de búsqueda de rutas son fundamentales para el análisis de gráficos y encontrar las rutas más eficientes o más cortas para recorrer entre nodos. En esta categoría se incluyen los algoritmos A * y Dijkstra, que se utilizan para comprender dependencias complejas y evaluar rutas para usos tales como logística física y enrutamiento de llamadas o IP de menor costo.

II. Metodologia

Los algoritmos inteligentes que vamos a usar en noe4j son:

- Weakly Connected Components:
  Este algoritmo de Componentes débilmente conectados, o Búsqueda de unión, encuentra conjuntos de nodos conectados en un gráfico no dirigido donde cada nodo es accesible desde cualquier otro nodo del mismo conjunto.
- Pre-computed counts:
  El algoritmo de coeficiente de agrupación local calcula el coeficiente de agrupación local para cada nodo del gráfico. El coeficiente de agrupamiento local C n de un nodo n describe la probabilidad de que los vecinos de n también estén conectados.
- Eigenvector Centrality:
  Es un algoritmo que mide la influencia transitiva o la conectividad de los nodos.
  Las relaciones con los nodos de puntuación alta contribuyen más a la puntuación de un nodo que las conexiones con los nodos de puntuación baja. Una puntuación alta significa que un nodo está conectado a otros nodos que tienen puntuaciones altas.
- Node2Vec
  Es un algoritmo de incrustación de nodos que calcula una representación vectorial de un nodo basándose en recorridos aleatorios en el gráfico. El vecindario se muestrea a través de caminatas al azar. Utilizando una serie de muestras de vecindad aleatorias, el algoritmo entrena una sola red neuronal de capa oculta. La red neuronal está entrenada para predecir la probabilidad de que ocurra un nodo en una caminata en función de la aparición de otro nodo.
- Euclidean Similarity:
  La distancia euclidiana mide la distancia en línea recta entre dos puntos en un espacio n-dimensional.
  Podemos utilizar el algoritmo de distancia euclidiana para calcular la similitud entre dos cosas.
  La función Distancia euclidiana calcula la similitud de dos listas de números.
- Resource Allocation:
  Es un algoritmo que mide la influencia transitiva o la conectividad de los nodos.
- Depth First Search:
  El algoritmo Depth First Search es un recorrido de gráfico que comienza en un nodo determinado y explora en la medida de lo posible a lo largo de cada rama antes de retroceder
  Hay múltiples condiciones de terminación admitidas para el recorrido, basadas en alcanzar uno de varios nodos de destino, alcanzar una profundidad máxima, agotar un presupuesto dado de costo de relación atravesada o simplemente atravesar todo el gráfico. El resultado del procedimiento contiene información sobre qué nodos se visitaron y en qué orden.

III. Descripcion del problema

Se desea generar metodos inteligentes para encontrar infomación basada en Twitter de los candidatos presidenciales, para ello se debe registrar la informacion que permita realizar un analisis de datos, por ejemplo, fechas de publiciones, hashtags, geolocalización, usuarios, mensajes, clicks de visualización, follows, fuentes, etc.

Pasos a seguir:

1. Realizar una extración de datos de Twitter en base a palabras claves de los partidos politicos y/o presidentes:
   - Unes: Arauz – Asambleistas
   - Creo: Laso – Asambleistas
   - Pachacutig: Yaku – Asambleistas
   - Juntos: Carrasco – Asambleistas
   - Sociedad Patriotica: Gutierrez – Asambleistas
   - Alianza Pais: Peña
   - Construye : Juan Velasco
   - Avanza: Romero

2. Migrar esta información a la base de grafos Neo4j, en virtud de ello, se deberá tener al menos 15000 nodos.
3. Implementar los algoritmos inteligentes dentro de la base de grafos.
4. Generar un sistema inteligente para la recomendación o toma de desiciones en base al algoritmo aplicado.
6. Agregar conclusiones y recomendaciones.

IV. Descripcion de la solucion y pasos seguidos

- Vamos a usar la herramienta de Python para la extraccion de los datos de tweeter de partido politico, presidente, vicepresidente,asambleistas,partidos aliados de la lista Creo21-Guillermo Lasso, y la herramienta neo4j para poder graficar los datos buscados.
- Primero se debe tener la autorizacion de twitter para la extraccion de los datos , esto se realiza una solicitud indicando para que vamos a usar el api para consumir.
- Una ves aprobada la solicitud de twitter,vamos a usar consumer key ,consumer secret,access token,access token secret
- Los datos de los tweets se mando a guardar en un archivo json
- Despues de obtener los datos se prosigue a subir al neo4j
- Hay que tener en cuenta que el twitter nos limita por tiempos para poder extraer mas datos, dependiendo de la cantidad de actividad de twitter que tenga
- Para poder aplicar los algoritmos inteligentes se desarrollo una investigacion para poder aplicar a nuestro grapho de neo4j

1) Extraccion de los datos de twitter de partidos politicos, presidente , vicepresidente , asambleista, partidos aliados :

```
# URL parameters.
q = urllib.parse.quote_plus(os.environ.get("TWITTER_SEARCH","@LassoGuillermo OR #AsambleistasDelCAMBIO OR #CREO21 OR #CapacidadP
count = 100
result_type = "recent"
lang = "es"
since_id = -1
```

Fig. 1. Tweets a buscar con palabras claves

2) Subir los tweets al grapho de neo4j :

3) Implementar los algoritmos inteligentes dentro de la base de grafos.: Node2Vec MATCH (t1:Tweet name: 'Guillermo Lasso')-[tweet1:MENTIONS]-¿(mentioned) MATCH (t2:Tweet name: "Andrea Rodriguez Carranza")-[tweet2:MENTIONS]-¿(mentioned) RETURN t1.name AS from, t2.name AS to, algo.similarity.euclideanDistance(collect(tweet1.score), collect(tweet2.score)) AS similarity
Euclidean Similarity MATCH (t1:Person name: 'Guillermo Lasso')-[tweet1:MENTIONS]-¿(mentioned) MATCH (t2:Person name: "Andrea Rodriguez Carranza")-[tweet2:MENTIONS]-¿(mentioned) RETURN t1.name AS from, t2.name AS to, algo.similarity.euclideanDistance(collect(tweet1.score), collect(tweet2.score)) AS similarity
Resource Allocation MATCH (t1:Tweet name: 'Guillermo Lasso') MATCH (t2:Tweet name: 'Andrea Rodriguez Carranza') RETURN algo.linkprediction.resourceAllocation(t1, t2) AS score
Depth First Search

CALL gds.alpha.dfs.stream(graphName: String, configuration: Map) YIELD // general stream return columns startNodeId: Integer, nodeIds: Integer, path: Path CALL gds.graph.create('myGraph', 'Node', 'REL', relationshipProperties: 'cost')

```
# Pass dict to Cypher and build query.
query = """
UNWIND $tweets AS t
WITH t
ORDER BY t.id
WITH t,
     t.entities AS e,
     t.user AS u,
     t.retweeted_status AS retweet
MERGE (tweet:Tweet {id:t.id})
SET tweet.text = t.text,
    tweet.created_at = t.created_at,
    tweet.favorites = t.favorite_count
MERGE (user:User {screen_name:u.screen_name})
SET user.name = u.name,
    user.location = u.location,
    user.followers = u.followers_count,
    user.following = u.friends_count,
    user.statuses = u.statusus_count,
    user.profile_image_url = u.profile_image_url
MERGE (user)-[:POSTS]->(tweet)
MERGE (source:Source {name:t.source})
MERGE (tweet)-[:USING]->(source)
FOREACH (h IN e.hashtags |
  MERGE (tag)-[:TAGS]->(tweet)
)
FOREACH (u IN e.urls |
  MERGE (url:Link {url:u.expanded_url})
  MERGE (tweet)-[:CONTAINS]->(url)
  MERGE (tweet)-[:pertenece]->(partidopolitico)
)
FOREACH (m IN e.user_mentions |
  MERGE (mentioned:User {screen_name:m.screen_name})
  ON CREATE SET mentioned.name = m.name
  MERGE (tweet)-[:MENTIONS]->(mentioned)
)
FOREACH (r IN [r IN [t.in_reply_to_status_id] WHERE r IS NOT NULL] |
  MERGE (reply_tweet:Tweet {id:r})
  MERGE (tweet)-[:REPLY_TO]->(reply_tweet)
)
FOREACH (retweet_id IN [x IN [retweet.id] WHERE x IS NOT NULL] |
    MERGE (retweet_tweet:Tweet {id:retweet_id})
    MERGE (tweet)-[:RETWEETS]->(retweet_tweet)
)
"""
```

Fig. 2. Metodo para manda que Pasa a Cypher y compila la consulta

4) Sistema inteligente para la toma de desiciones en base a los algoritmos aplicados.:

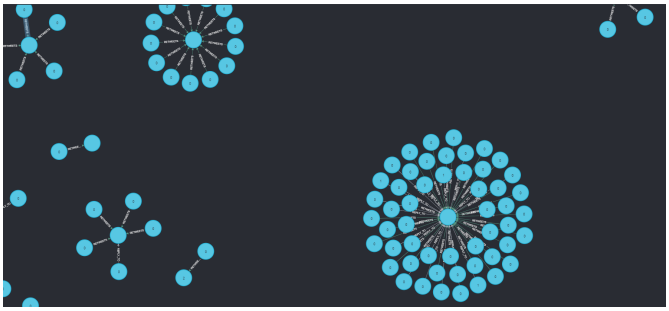

Fig. 3. Tweets

Fig. 4. Usuarios

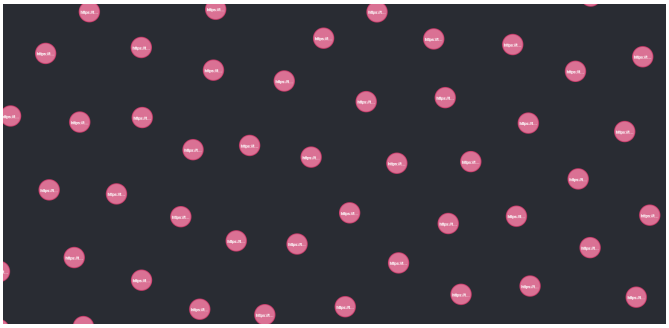

Fig. 5. Links de cada tweet

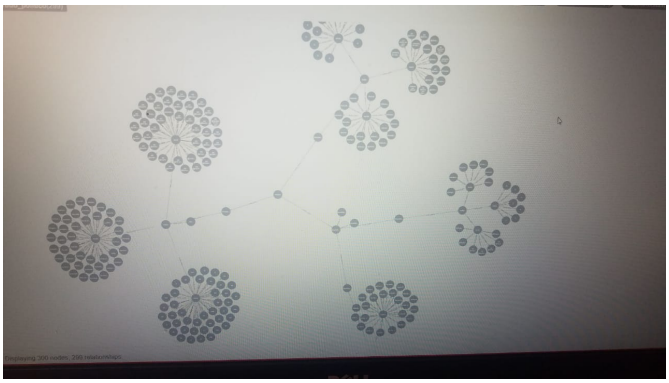

Fig. 6. Datos en Presidente ,Vicepresidente, asambleistas, partidos politicos aliados

V. Conclusion

La realizacion de este trabajo nos pudo mostrar las diferentes aplicaciones y usos que se le puede dar a la aplicacion de

twitter asi como la utilizacion de la base de datos neo4j para el analisis de datos obtenidos por el webscraping realizado de los diferentes candidatos a sambleistas, presidente y vicepresidente del partido creo, los diferentes algoritmos implementados nos ayudan a encontrar una solucion a problemas como saber cual es el mas activo en redes sociales o quien genera mas noticias a nivel provincial o nacional, a mas de poder conseguir recomendaciones por popularidad de un candidato. Nos ayuda mucho para recolectar datos masivos el websraping asi tambien poder interpretar estos datos

VI. Usos

- Node2Vec se usa en la Valoracion de peliculas basados en lenguaje natural y Deep Learning
- Euclidean Similarity Se a usado evaluar los sistemas con el corpus basado en Wikipedia distribuido en el concurso de similitud de textos

References

[1] S. Schafer, M. Litchfield, A. Zai, Z. Popović, and C. Campbell, "X-Band MMIC GaN Power Amplifiers Designed for High-Efficiency Supply-Modulated Transmitters," in *IEEE MTT-S Digest, IMS 2013*, Seattle, WA, 2013, pp. 1–4.