Inteligencia Artificial 1.

Examen Practico 27/05/2021

Objetivo:

- Consolidar los conocimientos adquiridos en clase para los métodos de búsqueda y bases de datos orientadas a grafos.

Enunciado:

- Diseñe y desarrolle un sistema recopilador que permita obtener las noticias, Facebook, twitter de los alcaldes dentro de una base de datos orientados a grafos:

  Webscraping es la técnica de extraer datos contenidos en un formato no estructurado en una página web y llevarlos a una estructura fácil de usar.
  Es por ello, que se desea crear nuevos métodos que permitan la recopilación masiva de información para su posterior estudio y correlación en forma de big data.

  En base a ello, vamos a obtener los datos de lo que esta hablando las noticias de los candidatos dentro del Ecuador y almacenar los datos dentro de una base de datos orientadas a grafos.
  - Generar un modelo que permita obtener y almacenar los datos en los grafos.
  - Vincular los datos con el alcalde seleccionado.
  - Se debe tener al menos 1000 nodos generados.
  - Obtener de la noticia: el Link, mensaje, fecha
  - Facebook: Comentarios, Publicaciones, Amigos, Likes, Seguidores, etc.
  - Twitter: Usuario, mensaje, fecha, etc.
  - No se debe repetir los alcaldes.
  - Se puede utilizar cualquier herramienta o procesamiento para el WebScarping.
  - Generar sus análisis, conclusiones y recomendaciones en base a los datos

Documentos de entrega: Se deberá entregar un informe con la base de datos de neo4j, dentro del mismo tener capturas del procesos de neo4j y resultados antes descritos.
Subir los archivos al GIT personal dentro de la carpeta de EXAMEN y ademas subir el archivo fuente de la base de datos de grafos.
El documento va a estar en formato PDF.

Fecha de entrega: 01**/06/2021** – 23**:55.**

Inteligencia Artificial 1.

Examen Practico 27/05/2021

Estudiante Angamarca Rafael

Desarrollo del Examen

Intalamos las librerias necesaria y creamos una api de aplicación para obtener autorizacion de tiwter

```
import logging
from neo4j import GraphDatabase
from neo4j.exceptions import ServiceUnavailable
from facebook_scraper import get_posts
class Neo4jService:
```

Creamos las diferentes variables de busqueda con la conexión a neo4j

```
def __init__(self, uri, user, password):
    self._driver = GraphDatabase.driver(uri, auth=(user, password))

def close(self):
    self._driver.close()

def crear_Alcaldes(self, tx, nombre):
    tx.run("CREATE (:Alcaldes {nombre: $nombre})", nombre=nombre)

def crear_alcalde(self, tx, nombre):
    tx.run("CREATE (:Alcalde {nombre: $nombre})", nombre=nombre)

def crear_publicacion(self, tx, nombre):
    tx.run("CREATE (:Publicacion {nombre: $nombre})", nombre=nombre)

def crear_fecha(self, tx, nombre):
    tx.run("CREATE (:Fecha {nombre: $nombre})", nombre=nombre)

def crear_texto(self, tx, nombre):
    tx.run("CREATE (:Texto {nombre: $nombre})", nombre=nombre)

def crear_likes(self, tx, nombre):
    tx.run("CREATE (:Likes {nombre: $nombre})", nombre=nombre)

def crear_comentarios(self, tx, nombre):
    tx.run("CREATE (:Comentarios {nombre: $nombre})", nombre=nombre)
```

Creamos las relaciones para asociar la busqueda de datos

```
  def crear_relacion_Alcs_Alc(self, tx, nombre1, nombre2):
      tx.run("MATCH (a:Alcaldes {nombre: $nombre1}) "
             "MATCH (b:Alcalde {nombre: $nombre2}) "
             "MERGE (a)-[:Alcaldess_Alcalde]->(b)",
             nombre1=nombre1, nombre2=nombre2)

  def crear_relacion_alc_publicacion(seelf, tx, nombre1, nombre2):
      tx.run("MATCH (a:Alcalde {nombre: $nombre1}) "
            "MATCH (b:Publicacion {nombre: $nombre2}) "
            "MERGE (a)-[:Alcalde_Publicacion]->(b)",
            nombre1=nombre1, nombre2=nombre2)

  def crear_relacion_publi_fecha(self, tx, nombre1, nombre2):
      tx.run("MATCH (a:Publicacion {nombre: $nombre1}) "
             "MATCH (b:Fecha {nombre: $nombre2}) "
             "MERGE (b)-[:Publicacion_Fecha]->(a)",
             nombre1=nombre1, nombre2=nombre2)

  def crear_relacion_public_texto(self, tx, nombre1, nombre2):
      tx.run("MATCH (a:Publicacion {nombre: $nombre1}) "
             "MATCH (b:Texto {nombre: $nombre2}) "
             "MERGE (b)-[:Publicacion_Texto]->(a)",
             nombre1=nombre1, nombre2=nombre2)

  def crear_relacion_public_like(self, tx, nombre1, nombre2):
      tx.run("MATCH (a:Publicacion {nombre: $nombre1}) "
             "MATCH (b:Likes {nombre: $nombre2}) "
             "MERGE (b)-[:Publicacion_Like]->(a)",
             nombre1=nombre1, nombre2=nombre2)

  def crear_relacion_public_comen(self, tx, nombre1, nombre2):
      tx.run("MATCH (a:Publicacion {nombre: $nombre1}) "
             "MATCH (b:Comentarios {nombre: $nombre2}) "
             "MERGE (b)-[:Publicacion_Comentarios]->(a)",
             nombre1=nombre1, nombre2=nombre2)
```

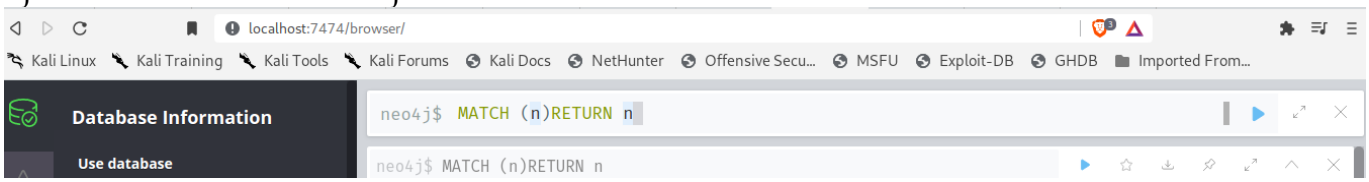
Autentificamos la url de neo4j y mostramos resultados

```
neo4j = Neo4jService('bolt://localhost:7474', 'neo4j', 'admin')
with neo4j._driver.session() as session:
        contador = -1
        session.write_transaction(neo4j.crear_Alcaldes, "Alcaldes")
        session.write_transaction(neo4j.crear_alcalde, "Pedro Palacios")
        session.write_transaction(neo4j.crear_relacion_Alcs_Alc , "Alcaldes" , "Pedro Palacios")
        for post in get_posts ('jorgeyundamachado', pages=52, timeout=1):
            contador = contador + 1
            publicacion = "publicaciones de PALACIOS N°" +str(contador)
            fecha = "Fecha de la publicacion N°" +str(contador)+ " es " +str(post['time'])
            text = "El texto de la publicacion N°" +str(contador)+ " es " +str(post['text'])
            likes = "Los likes de la publicacion N°" +str(contador)+ " es " +str(post['likes'])
            comentarios = "El numero de comentarios de la publicacion N°" +str(contador)+ " es " +str(post['comments'])
            session.write_transaction(neo4j.crear_publicacion, publicacion)
            session.write_transaction(neo4j.crear_fecha, fecha)
            session.write_transaction(neo4j.crear_texto, text)
            session.write_transaction(neo4j.crear_likes, likes)
            session.write_transaction(neo4j.crear_comentarios, comentarios)
            session.write_transaction(neo4j.crear_relacion_alc_publicacion, "Jorge Yunda", publicacion)
            session.write_transaction(neo4j.crear_relacion_publi_fecha, publicacion, fecha)
            session.write_transaction(neo4j.crear_relacion_public_texto, publicacion, text)
            session.write_transaction(neo4j.crear_relacion_public_like, publicacion, likes)
            session.write_transaction(neo4j.crear_relacion_public_comen, publicacion, comentarios)
            print(publicacion)
```

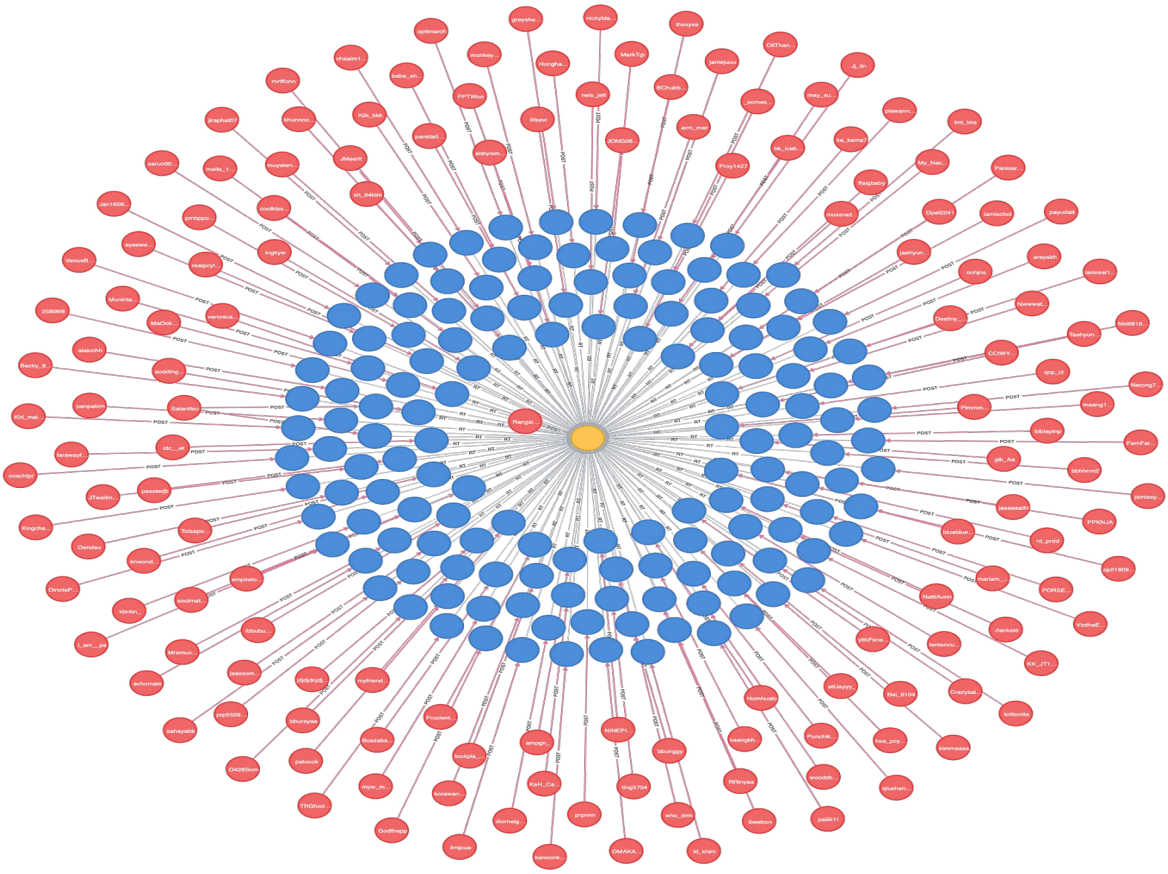
Inteligencia Artificial 1.

Examen Practico 27/05/2021

Ejecucion del servicio Neo4j



Grafos



Concluciones

Para poder recopilar enorme cantidad de informacion es importante simplificar el metodo de busqueda con ayda de apis o modulos.

Recomendaciones:

conocer los conceptos de Base de datos Garfo,

instalar herramientas necesarias para su ejecuciion,