



Social Network Analysis en Neo4j

DRA. KARINA RUBY PÉREZ-DANIEL

FACULTAD DE INGENIERÍA
UNIVERSIDAD PANAMERICANA

ENERO - MARZO, 2020



UNIVERSIDAD
PANAMERICANA

CONTENIDO

- 1 CONSULTAS EN CYPHER
- 2 CASO PRÁCTICO II
- 3 TWITTER API
- 4 RECOLECCIÓN DE DATOS DEL API DE TWITTER
- 5 CYPHER Y TWITTER



SECCIÓN 1

CONSULTAS EN CYPHER



CASO PRÁCTICO

TASK

Consideremos el siguiente grafo en Neo4j

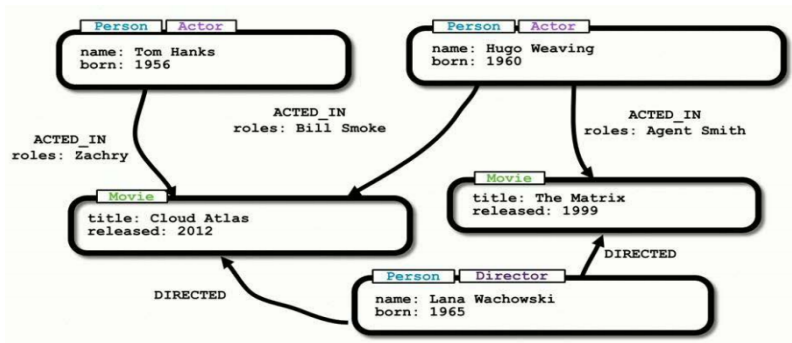


Figura 1. Caso práctico



CONSULTAS TIPO GRAFO EN NEO4J

Componentes de una consulta en Neo4j.

MATCH

```
MATCH (m:Movie)  
RETURN m
```

- MATCH y RETURN son palabras reservadas de Cypher.
- m es una variable
- :Movie es la etiqueta de un nodo.

```
MATCH (p:Person)-[r:ACTED_IN]->(m:Movie)  
RETURN p,r,m
```

- MATCH y RETURN son palabras reservadas de Cypher.
- p, r, m son variables.
- :ACTED_IN es el tipo de relación.



CONSULTAS TIPO GRAFO EN NEO4J

Componentes de una consulta tipo tabla en Neo4j.

```
MATCH path = (:Person)-[ACTED_IN]->(:Movie)
RETURN path
```

- MATCH y RETURN son palabras reservadas de Cypher.
- path es una variable
- :Person y :Movie son etiqueta de nodos.
- :ACTUA_EN es el tipo de relación.



CONSULTAS TIPO GRAFO Y TIPO TABLA EN NEO4J

Componentes de una consulta en Neo4j.

● Consulta tipo grafo

- En este caso la consulta devuelve únicamente los nodos que contengan la etiqueta `Movie`.

```
MATCH (m:Movie)
```

```
RETURN m
```

● Consulta tipo tabla

- `MATCH` guarda en la variable `m` todos los nodos que tienen la etiqueta `Movie`.
- `RETURN` devuelve una tabla con 2 columnas.
- La 1era columna contiene la propiedad `title` de los nodos guardados en `m`.
- La 2da columna contiene la propiedad `released` de los nodos guardados en `m`.

```
MATCH (m:Movie)
```

```
RETURN m.title, m.released
```



CONSULTAS TIPO GRAFO Y TIPO TABLA EN NEO4J

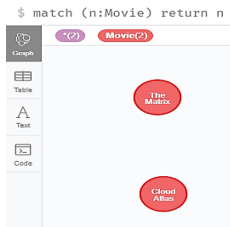


Figura 2. Obtener nodos que cumplan con una etiqueta

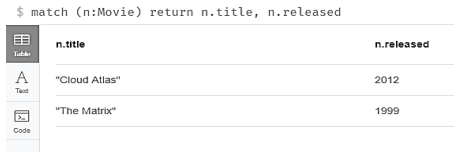


Figura 3. Obtener las propiedades de los nodos que cumplan con una etiqueta



CONSULTAS TIPO TABLA EN NEO4J

Elementos agregados a una consulta en Neo4j.

● Consulta tipo tabla con elementos agregados

- `MATCH` guarda en la variable `p` todos los nodos que tienen la etiqueta `Person` y en `m` todos los nodos que tienen la etiqueta `Movie`.
- `RETURN` devuelve una tabla con 2 columnas.
- La 1era columna contiene la propiedad `name` de los nodos guardados en `p`.
- La 2da columna contiene la propiedad **agregada**. `count(*)` cuenta todas las relaciones de `p` con la etiqueta `ACTED_IN` que llegaron a algún nodo `Movie` y las guarda en la variable `numberOfMovies`, mediante el comando `AS`.

```
MATCH (p:Person)-[:ACTED_IN]-> (m:Movie)
RETURN p.name, count(*) AS numberOfMovies
ORDER BY numberOfMovies DESC
```



OTRAS FUNCIONES AGREGADAS

Aggregation	
<code>count(*)</code>	The number of matching rows.
<code>count(variable)</code>	The number of non-NULL values.
<code>count(DISTINCT variable)</code>	All aggregation functions also take the <code>DISTINCT</code> modifier, which removes duplicates from the values.
<code>collect(n.property)</code>	List from the values, ignores NULL.
<code>sum(n.property)</code>	Sum numerical values. Similar functions are <code>avg</code> , <code>min</code> , <code>max</code> .
<code>percentileDisc(n.property, {percentile})</code>	Discrete percentile. Continuous percentile is <code>percentileCont</code> . The <code>percentile</code> argument is from 0.0 to 1.0.
<code>stdev(n.property)</code>	Standard deviation for a sample of a population. For an entire population use <code>stdevp</code> .

Figura 4. Lista de algunas otras funciones agregadas



WHERE

El condicional WHERE.



```
1 MATCH (p:Person)-[:ACTED_IN]-> (m:Movie)
2 WHERE p.name = "Tom Hanks"
3 RETURN p,m
```

\$ MATCH (p:Person)-[:ACTED_IN]-> (m:Movie) WHERE p.name = "Tom Hanks" RETURN p,m

Graph

Table

Text

Code

*(3) Actor(1) Person(1) Movie(1)

*(1) ACTED_IN(1)

Figura 5. Caso práctico

```
MATCH (p:Person)-[:ACTED_IN]-> (m:Movie)
WHERE p.name = "Tom Hanks"
RETURN p,m
```

SECCIÓN 2

CASO PRÁCTICO II



CASO PRÁCTICO II, IMPORTANDO DATOS DESDE CSV

NEO4J PUEDE IMPORTAR INFORMACIÓN ALMACENADA EN ARCHIVOS CVS, YA SEA QUE ESTEN ALMACENADOS DE FORMA LOCAL O DESDE UNA URL

- LOAD CSV WITH HEADERS FROM 'https://.../file.csv'
AS line
RETURN line

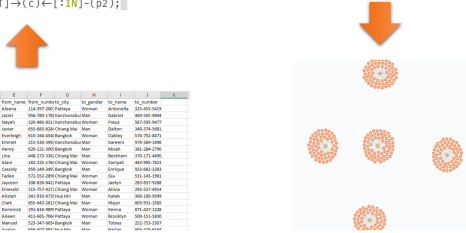
	A	B	C	D	E	F	G	H	I	J	K
1	from_dt	to_dt	from_city	from_gende	from_name	from_numbr	to_city	to_gender	to_name	to_number	
2	2019-01-01T	2019-01-01T	Pattaya	Woman	Aleena	114-397-200	Pattaya	Woman	Antonella	325-453-5419	
3	2019-01-01T	2019-01-01T	Pattaya	Man	Jaziel	956-780-1788	Kanchanabul	Man	Gabriel	469-505-9894	
4	2019-01-01T	2019-01-01T	Hua Hin	Woman	Nayeli	320-480-8311	Kanchanabul	Woman	Freya	587-595-9477	
5	2019-01-01T	2019-01-01T	Bangkok	Man	Javier	655-683-6284	Chiang Mai	Man	Dalton	340-374-5681	
6	2019-01-01T	2019-01-01T	Pattaya	Woman	Everleigh	610-164-4544	Bangkok	Woman	Oakley	574-752-8471	
7	2019-01-01T	2019-01-01T	Chiang Mai	Man	Emmet	215-530-3993	Kanchanabul	Man	Kareem	974-369-1496	
8	2019-01-01T	2019-01-01T	Pattaya	Man	Henry	620-121-3005	Bangkok	Man	Micah	381-284-2790	
9	2019-01-01T	2019-01-01T	Kanchanabul	Woman	Lina	448-273-3383	Chiang Mai	Man	Beckham	370-171-4490	
10	2019-01-01T	2019-01-01T	Pattaya	Woman	Alani	183-225-1764	Chiang Mai	Woman	Zaniyah	443-995-7423	
11	2019-01-01T	2019-01-01T	Chiang Mai	Woman	Cassidy	959-149-3493	Bangkok	Man	Enrique	923-682-3283	
12	2019-01-01T	2019-01-01T	Hua Hin	Man	Tadeo	572-552-2896	Chiang Mai	Woman	Gia	531-143-1961	
13	2019-01-01T	2019-01-01T	Hua Hin	Man	Jayceon	108-826-6422	Pattaya	Woman	Jaelyn	283-837-9288	
14	2019-01-01T	2019-01-01T	Hua Hin	Woman	Emerald	315-757-4272	Chiang Mai	Woman	Alivia	295-527-4914	
15	2019-01-01T	2019-01-01T	Kanchanabul	Man	Alistair	341-910-6735	Hua Hin	Man	Kaleb	366-100-9549	
16	2019-01-01T	2019-01-01T	Bangkok	Man	Clark	655-643-2811	Chiang Mai	Man	Major	805-931-1585	
17	2019-01-01T	2019-01-01T	Pattaya	Man	Dominick	291-616-9895	Pattaya	Woman	Kenna	871-427-1228	
18	2019-01-01T	2019-01-01T	Bangkok	Woman	Aileen	413-665-7064	Pattaya	Woman	Brooklyn	508-151-5830	
19	2019-01-01T	2019-01-01T	Hua Hin	Man	Manuel	523-347-6054	Bangkok	Man	Tobias	252-753-3307	
20	2019-01-01T	2019-01-01T	Hua Hin	Woman	Avalon	656-507-8937	Hua Hin	Man	Marlan	866-376-6184	



IMPORTANDO UN ARCHIVO CSV

Neo4j puede importar información almacenada en archivos CVS, ya sea que estén almacenados de forma local o desde una URL

```
1 LOAD CSV WITH HEADERS FROM 'https://vbatushkov.bitbucket.io/log_of_calls.csv' AS line
2 MERGE (c1:City { name: line.from_city })
3 MERGE (p1:Person { name: line.from_name, number: line.from_number, gender: line.from_gender })
4 MERGE (p1)-[:FROM]-(c1)
5 MERGE (c2:City { name: line.to_city })
6 MERGE (p2:Person { name: line.to_name, number: line.to_number, gender: line.to_gender })
7 MERGE (p2)-[:FROM]-(c2)
8 CREATE (c:Call { from: datetime(line.from_dt),
9               to: datetime(line.to_dt),
10              duration: duration.between(datetime(line.from_dt), datetime(line.to_dt)).minutes })
11 CREATE (p1)-[:OUT]-(c)←[:IN]-(p2);
```



	A	B	C	D	E	F	G	H	I	J	K
1	from_dt	to_dt	from_city	from_gender	from_name	from_number	to_city	to_gender	to_name	to_number	
2	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Paraguay	Women	Alexsne	114-297-2007	Paraguay	Woman	Antoniella	323-453-5429	
3	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Paraguay	Man	Lisset	916-700-1783	Kanchanaburi	Man	Adriana	689-355-6884	
4	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Thailand	Woman	Nayeli	328-489-8111	Kanchanaburi	Woman	Fraja	587-585-9877	
5	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Bangkok	Man	Jerrie	655-685-630	Chung	Man	Dustin	342-354-5681	
6	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Paraguay	Woman	Freelighth	602-366-8342	Bangkok	Woman	Cailey	176-753-8671	
7	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Chung	Man	Ernest	215-539-3991	Kanchanaburi	Man	Kareem	979-389-1886	
8	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Paraguay	Man	Henry	626-521-3007	Bangkok	Man	Natash	581-284-2790	
9	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Kanchanaburi	Woman	Lina	448-275-1385	Chung	Man	Betham	170-171-4490	
10	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Paraguay	Woman	Alexa	186-225-176	Chung	Man	Zoraghi	483-980-7523	
11	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Chung	Man	Cassidy	959-349-3491	Bangkok	Man	Krisque	913-683-3283	
12	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Thailand	Man	Taylor	575-532-2093	Chung	Man	Gia	311-143-2961	
13	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Thailand	Man	Jayven	338-826-8422	Paraguay	Woman	Jeffery	283-677-9388	
14	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Thailand	Woman	ErinMall	325-753-4272	Chung	Man	Alfina	295-127-4954	
15	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Bangkok	Man	Alfonso	345-933-9722	Asia	Man	Katad	360-380-5540	
16	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Bangkok	Man	Clark	600-440-2812	Chung	Man	Major	805-791-1383	
17	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Paraguay	Man	Dominick	290-438-9899	Paraguay	Woman	Karene	875-027-1228	
18	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Bangkok	Woman	Adriana	412-682-7869	Paraguay	Woman	Brendley	508-515-5830	
19	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Thailand	Man	Marned	523-347-8059	Bangkok	Man	Tahara	252-753-3387	
20	2019-05-07T07:20:51-01:00	2019-05-07T07:20:51-01:00	Thailand	Woman	ErinMall	496-2475-9887	Asia	Man	Heather	846-175-2414	

Figura 7. Caso práctico II



WHERE

WHERE

Consideremos la siguiente consulta

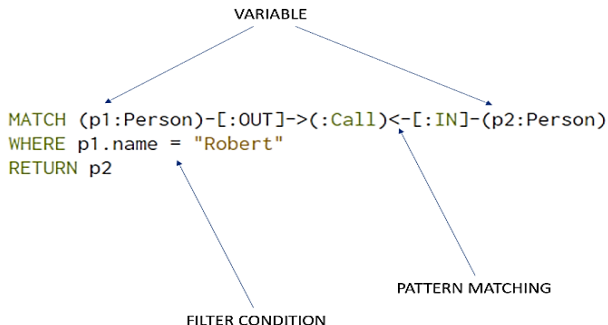


Figura 8. Consulta



WHERE

WHERE

Consideremos la siguiente consulta

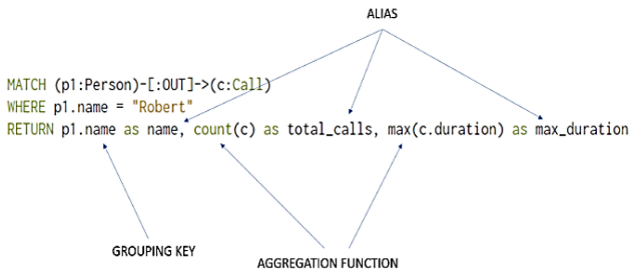


Figura 9. Consulta



EJERCICIOS EN NEO4J

TASK

Obtengamos las siguientes inferencias a partir del grafo anterior.

- 1 ¿Cuántas llamadas iniciaron en Mayo?
- 2 ¿A qué personas llamó Caleb?
- 3 ¿Cuántas llamadas se hicieron desde Bangkok?
- 4 ¿Quiénes son las 10 personas que reciben más llamadas desde Bangkok?
- 5 Mostrar todas las personas que recibieron llamadas de 10 minutos en Julio y que además hicieron llamadas de 5 minutos.
- 6 Encuentra el nombre del hombre que recibió una llamada de Tiffany en Mayo.

SECCIÓN 3

TWITTER API



TWITTER API

● Para realizar la conexión al API de tweeter es necesario:

- Contar con una cuenta de developer (<https://developer.twitter.com/en/apps>).
- Crear un app en twitter developer para obtener las credenciales de usuario y los tokens necesarios para autenticarnos durante la búsqueda de datos.
- Guardar las llaves de usuario y los tokens de acceso.
- Instalar tweepy y py2neo



CREAR UNA APP EN TWITTER PARA LA OBTENCIÓN LLAVES

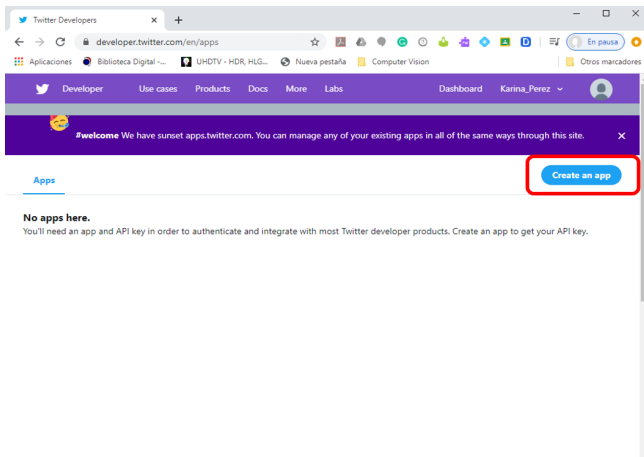


Figura 10. Paso 1



CREAR UNA APP EN TWITTER PARA LA OBTENCIÓN LLAVES

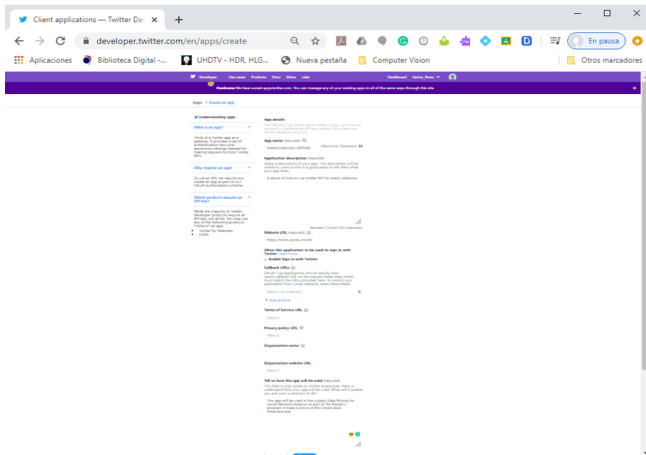


Figura 11. Paso 2



CREAR UNA APP EN TWITTER PARA LA OBTENCIÓN LLAVES

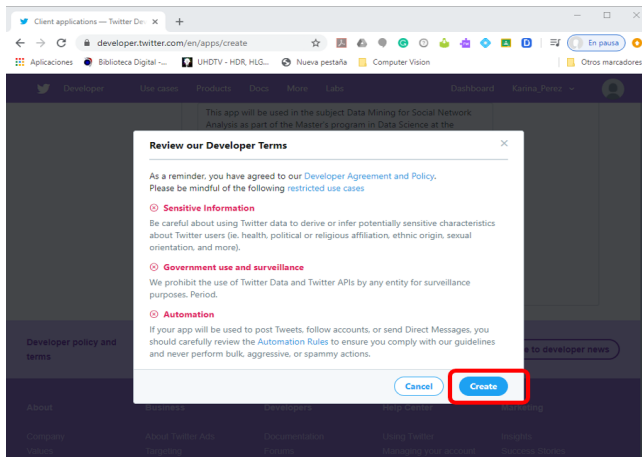


Figura 12. Paso 3



CREAR UNA APP EN TWITTER PARA LA OBTENCIÓN LLAVES

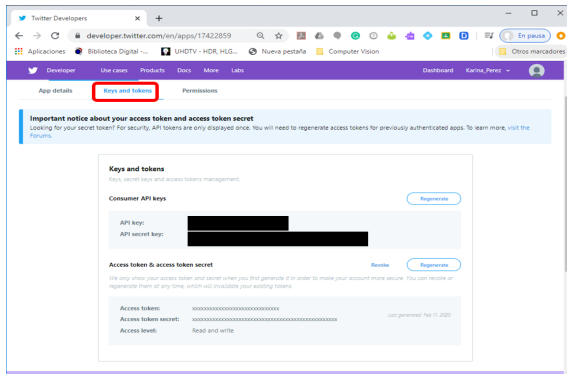


Figura 13. Paso 4

Guarda las 4 llaves obtenidas:

- Consumer API key,
- Consumer API secret key,
- Access token,
- Access token secret.



SECCIÓN 4 DE TWITTER

RECOLECCIÓN DE DATOS DEL API



OBTENCIÓN DE DATOS DEL API DE TWITTER

Tweepy es una biblioteca de Python que se usa para recuperar datos del API de Twitter.

Se pueden realizar diversas tareas usando `tweepy`, tales como.

- Buscar tweets de un usuario en particular,
- Buscar tweets con un hashtag o texto específico,
- Obtener datos de ubicación, nombre, número de amigos, etc.,
- Posteo de tweets del usuario autenticado,
- Información de trends,
- Búsqueda y envío de DMs del usuario autenticado,
- Obtención de status en el timeline, etc.

NOTA: Los datos que se pueden recuperar cambian constantemente.



OBTENCIÓN DE DATOS DEL API DE TWITTER

Actualmente el API de **twitter** tiene 3 versiones, la *Standard*, la *Premium* y la *Enterprise*.

La versión estándar permite recuperar tweets de los últimos 7 días.

Conexión al API de twitter.

● Instalando las llaves y los tokens de acceso

```
1 | import tweepy
2 |
3 | # consumer keys and access tokens, used for OAuth
4 | consumer_key = 'YOUR-CONSUMER-KEY'
5 | consumer_secret = 'YOUR-CONSUMER-SECRET'
6 | access_token = 'YOUR-ACCESS-TOKEN'
7 | access_token_secret = 'YOUR-ACCESS-TOKEN-SECRET'
8 |
9 | # OAuth process, using the keys and tokens
10 | auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
11 | auth.set_access_token(access_token, access_token_secret)
12 |
13 | # creation of the actual interface, using authentication
14 | api = tweepy.API(auth)
```



OBTENCIÓN DE DATOS DEL API DE TWITTER

● Recuperando informacion del usuario autenticado

```

1 | user = api.me()
2 |
3 | print('Name: ' + user.name)
4 | print('Location: ' + user.location)
5 | print('Friends: ' + str(user.friends_count))

```

● Recuperando informacion de otros usuarios

```

1 | #Recuperando informacion de otro usuario
2 | user = api.get_user('userName')
3 |
4 | print('Name: ' + user.name)
5 | print('Location: ' + user.location)
6 | print('Friends: ' + str(user.friends_count))

```

● Recuperando los amigos del usuario autenticado

```

1 | #Recuperando amigos del usuario autenticado
2 | user_friends = user.friends()
3 | for friend in user_friends[:5]:
4 |     print (friend.screen_name)

```



OBTENCIÓN DE DATOS DEL API DE TWITTER

- **Búscando 5 tweets que contengan el texto “México”, escritos en inglés**

```
1 || search = tweepy.Cursor(api.search, q="Mexico", lang="en").items(5)
```

- **Búscando los 5 tweets más recientes que contengan la palabra “México”, sin considerar el idioma y mostrar los resultados recuperados**

```
1 || search = tweepy.Cursor(api.search, q="Mexico", result_type="recent").items(5)
2 ||
3 || #Mostrar los resultados recuperados
4 || for item in search:
5 ||     print (item.text)
```

- **Para publicar un nuevo tweet desde la cuenta actual**

```
api.update_status('El mensaje')
```



OBTENCIÓN DE DATOS DEL API DE TWITTER

- **Buscando los 5 tweets más recientes que contengan la palabra “México”, en inglés y mostrar los resultados recuperados, así como otros parámetros.**

```

1  search = tweepy.Cursor(api.search, q="Mexico", result_type="recent", lang="en").items(5)
2
3  for item in search:
4      # print tweet text
5      print (item.text)
6
7      # print tweet created date
8      print (item.created_at)
9
10     # print retweet count of the tweet
11     print (item.retweet_count)
12
13     # print the username who published the tweet
14     print (item.user.name)
15
16     # print the location of the user who published the tweet
17     print (item.user.location)
18
19     # print the language code of the tweet
20     print (item.metadata['iso_language_code'])
21
22     # print the search result type
23     print (item.metadata['result_type'])
24
25     # print the device/source
26     print (item.source)

```



OBTENCIÓN DE DATOS DEL API DE TWITTER

● Imprimir la lista de hashtags presentes en el tweet

```

1 | for item in search:
2 |     # print list of hashtags present in the tweet
3 |     hashtags = item.entities['hashtags']
4 |     if hashtags:
5 |         ht = [ht['text'] for ht in hashtags]
6 |         print ht
7 |     else:
8 |         print hashtags      # print empty list when no hashtag is present

```

● Imprimir la lista de usuarios mencionados en el tweet

```

1 | for item in search:
2 |     # print list of users mentioned in the tweet
3 |     user_mentions = item.entities['user_mentions']
4 |     if user_mentions:
5 |         mentions = [user['screen_name'] for user in user_mentions] # name, id, screen_name
6 |         print mentions
7 |     else:
8 |         print user_mentions # print empty list if no user is mentioned in the tweet

```



TWITTER API

Para más detalles sobre Twitter API:

- <https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets>
- <https://developer.twitter.com/en/docs/tweets/data-overview/intro-to-tweet-json>
- <https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets>
- <https://developer.twitter.com/en/docs/tweets/data-overview/tweet-object>



MINERIA DE DATOS EN TWITTER

- **Entendimiento del problema**
- **Procesado de los tweets y limpieza**
- **Observación de los datos**
- **Extracción de características**
- **Modelado: análisis de sentimiento**



SECCIÓN 5

CYPHER Y TWITTER



CONEXIÓN CON CYPHER

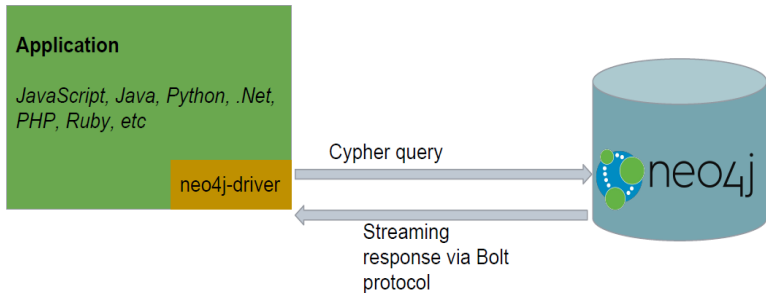


Figura 14. Clientes de Cypher



CONEXIÓN CON CYPHER

Nodes:

- **User** nodes – represents a Twitter user (handle and number of followers)
- **Tweet** nodes – represent a tweet (text, number of likes)
- **Hashtag** nodes – represent a hashtag
- **Country** nodes – represent a country (country name, country code)

Relationships:

- **TWEETED** relationship – in between a **User** and a **Tweet**; indicates that this user is the author of the tweet; also indicates the date at which it was tweeted
- **RETWEETED** relationship – in between a **User** and a **Tweet**; indicates this user retweeted this tweet; also indicates the date at which it was retweeted
- **HAS_HASHTAG** relationship – in between a **Tweet** and a **Hashtag**
- **USED_HASHTAG** relationship – in between a **User** and a **Hashtag**
- **MENTIONED** relationship – in between two **Users**
- **FROM** relation – in between a **User** and a **Country**

Figura 15. Nodos y Relaciones en Neo4j



Gracias por su atención

Fin

