



ITESO

Universidad Jesuita
de Guadalajara

Ingeniería en Sistemas Computacionales

Minería de Grafos

Mtro. Víctor Hugo Ortega Guzmán

Mini Proyecto 2

Equipo 3

IS727272 - Marco Ricardo Cordero Hernández

Porcentaje de créditos aprobados: 91%

Tlaquepaque, Jal., 06 de noviembre de 2023

Índice

Introducción.....	1
Desarrollo.....	2
Carga de datos.....	2
Análisis básico.....	3
Recomendaciones nativas.....	7
Recomendaciones con sustento.....	12
Estructura básica de uso de GDS.....	12
Modificación estructural.....	14
Algoritmos y recomendaciones.....	14
Presentación de los resultados	20
Análisis básico	20
Recomendaciones	20
Conclusiones y recomendaciones.....	21
Bibliografía.....	22

Introducción

Pensando en aspectos de la vida personal, y, posiblemente analizando con cierto temor ¿Cuántas de ellos son parte de la cotidianidad solo porque sí? La realidad es que ninguno. De hecho, estos elementos esenciales han sido constituidos como tal a raíz de una costumbre o algún sustento detrás. Pero ni siquiera los pensamientos se crean porque así lo quieren; no se forman de la nada y de un momento a otro, aunque parezca que se obtienen instantáneamente. Es por eso que lo que se hace hoy en día es resultado de toda una línea de desarrollo personal basada en la posible necesidad de resolver un problema, o simplemente porque una persona le comunicó a otra sus descubrimientos y la segunda lo tomó como dogma de fe. A lo que hace referencia lo último no es sino a *recomendaciones*.

En el mundo actual, las recomendaciones se ven en casi todos los rincones y en todo tipo de sectores del día a día, sin embargo, el mismo mundo se ha convertido en un cúmulo de tecnologías interactuando entre sí para facilitar la vida de las personas; de hecho, las personas pasan un promedio de 7 horas *al día* frente a una pantalla (Flynn, 2023), e inadvertidamente usan, sin siquiera notarlo, las tecnologías más recientes para las acciones más burdas y mundanas. Claro está que dentro de este uso se encuentran los servicios de streaming, tales como Netflix, cuya cantidad de usuarios se mantiene dominante ante otras plataformas similares (Moskowitz & Brown, 2023), aún cuando sus costos de suscripción están a la alza constantemente. Dejando de lado el descontento monetario, el objetivo principal de la plataforma es el entretenimiento y seguir proporcionándolo continuamente, porque claro, sin más contenido para los usuarios, el uso baja, los clientes se van, y el dinero junto con ellos. Para atender a esta capitalista problemática, Netflix recientemente ha fungido como casa productora, lanzando contenido de dudosa calidad y del agrado promedio del público general (Tassi, 2022); adicional a esta curiosa táctica, desde los comienzos de su funcionamiento también ha contado con un *sistema de recomendaciones*, en donde múltiples factores son considerados para indicarlo al usuario qué podría ver después, tales como el historial de vistas, calificaciones de contenidos, similitud de preferencias con otros usuarios, etcétera (Netflix, s.f.).

Los sistemas de recomendación pueden ser implementados a través de una amplia gama de técnicas, sin embargo, Le (2019) sugiere 6 pasos esenciales para hacerlo:

1. Entender el negocio/contexto de implementación
2. Obtener los datos
3. Explorar, limpiar y aumentar los datos
4. Implementar modelos predictivos
5. Visualizar los resultados
6. Desplegar el sistema y comenzar de nuevo

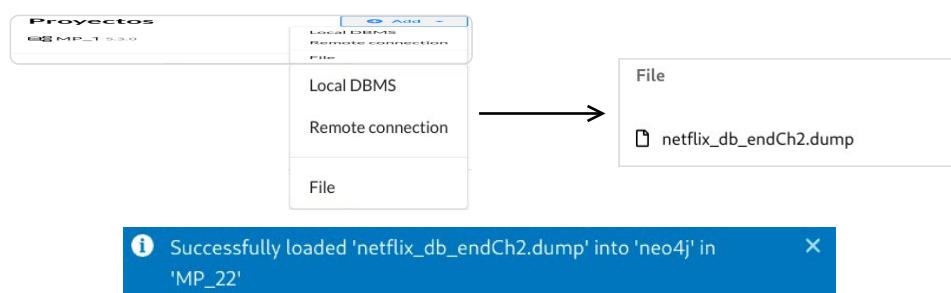
Como puede intuirse, al menos uno de los pasos anteriores involucra aprendizaje máquina, no obstante, esto no es del todo necesario. En este desarrollo, se propone el uso de una base de datos basada en *grafos*: Neo4j (s.f). Las bondades que esta representación de datos ofrece son mayores a las que una base relacional podría brindar (Memgraph, 2023), encontrando ventajas en la creación de relaciones nativas y la aplicación de algoritmos especializados en grafos a través de la extensión *Graph Data Science* [GDS] (Neo4j, s.f.). Estos elementos, en conjunción de algunos adicionales que serán detallados más adelante, harán posible la creación de un sistema primitivo pero funcional para realizar recomendaciones sin demasiada complejidad.

Desarrollo

La siguiente parte proporciona el sustento práctico del propósito de extracción, carga, transformación e interpretación de los datos presentes. Para este propósito, el punto de partida es la carga del respaldo “crudo” de una base de datos basada en grafos con información de películas y su consumo dentro de sus contenidos.

Carga de datos

En este caso, al ser provistos con una base ya transformada contenida dentro de un archivo .dump, el proceso de carga de datos es significativamente más sencillo, evitando los procesos de carga con sentencias de Cypher. Dentro de la interfaz de Neo4j Desktop, al menos para la versión 1.5.8, basta con importar el archivo hacia una nueva base:



Ya que se ha importado el archivo dentro del proyecto, deberá crearse una base desde este archivo importado.



Nota: El archivo proporcionado se comporta de manera inesperada al interactuar con librerías de soporte necesarias para este proyecto en versiones antiguas, por lo tanto, al momento de crear la base deberá seleccionarse *como mínimo* la versión **5.13.0**. De esta forma, se asegura la compatibilidad entre los componentes.



Nota: Se ha añadido un índice adicional para el título de las películas, lo cual servirá en búsquedas futuras.

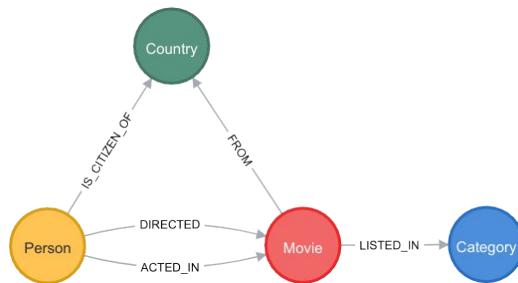
```
CREATE INDEX index_movie_title FOR (m:Movie) ON (m.title);
```

Análisis básico

En ocasiones pasadas se ha denotado la relevancia y evidente importancia del proceso exploratorio de un grafo previamente desconocido. En esta ocasión, al contar con una base nunca antes vista, este proceso tiene cabida dentro del desarrollo presente. Similar a una metodología de trabajo, se contestan estas cuestiones con preguntas previamente definidas. A continuación se presentan los resultados obtenidos.

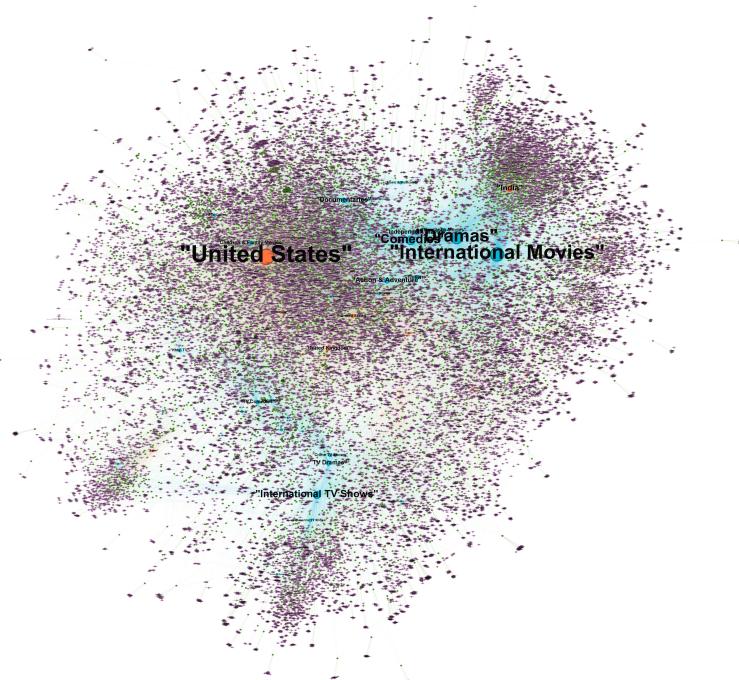
1. Esquema del grafo

```
CALL db.schema.visualization;
```



2. Grafo entero

```
// Hecho en Gephi
```



MDG

3. ¿Cuáles nodos hay?

```
CALL db.labels() YIELD label RETURN COLLECT(label) AS NODOS;
```

NODOS
["Movie", "Person", "Country", "Category"]

4. ¿Cuántos nodos hay?

```
MATCH (n) RETURN LABELS(n) AS tipoNodo, COUNT(DISTINCT n) AS totalNodos
UNION
MATCH (n) RETURN 'Total' AS tipoNodo, COUNT(DISTINCT n) AS totalNodos;
```

tipoNodo	totalNodos
["Movie"]	8807
["Country"]	215
["Category"]	42
["Person"]	40948
"Total"	50012

5. ¿Cuáles atributos tienen los nodos?

```
CALL db.schema.nodeTypeProperties()
YIELD nodeType AS Nodo, propertyName AS Propiedad, propertyTypes AS Tipo;
```

Nodo	Propiedad	Tipo
"Person"	"name"	["String"]
"Country"	"name"	["String"]
"Country"	"point"	["Point"]
"Category"	"name"	["String"]
"Movie"	"id"	["String"]
"Movie"	"dateAdded"	["String"]
"Movie"	"releaseYear"	["Long"]
"Movie"	"title"	["String"]

6. ¿Cuáles son las relaciones?

```
MATCH ()-[r]->() RETURN DISTINCT TYPE(r) AS RELACIONES;
```

RELACIONES
"FROM"
"LISTED_IN"
"DIRECTED"
"IS_CITIZEN_OF"
"ACTED_IN"

7. ¿Cuántas son las relaciones?

```
MATCH ()-[r]->() RETURN TYPE(r) as tipoRelacion, COUNT(r) as totalRelaciones
UNION
MATCH ()-[r]->() RETURN "Total" as tipoRelacion, COUNT(r) as totalRelaciones;
```

tipoRelacion	totalRelaciones
"FROM"	10019
"LISTED_IN"	19323
"DIRECTED"	6977
"IS_CITIZEN_OF"	349
"ACTED_IN"	64124
"Total"	100792

8. ¿Cuáles atributos tienen las relaciones?

```
CALL db.schema.relTypeProperties()
YIELD relType AS Relacion, propertyName AS Propiedad, propertyTypes AS Tipo;
```

Relacion	Propiedad	Tipo
".:FROM:"	null	null
".:LISTED_IN:"	null	null
".:DIRECTED:"	null	null
".:IS_CITIZEN_OF:"	null	null
".:ACTED_IN:"	null	null

(Las relaciones no tienen atributos)

9. Diámetro del grafo*// Hecho en Gephi*

```
Diameter: 2
Radius: 0
Average Path length: 1.6469167571900147
```

La interpretación de este sorprendente resultado indica que el grafo es altamente conexo, bastando el recorrido de tan solo dos nodos desde un origen para poder alcanzar todos los demás. Posiblemente esto se deba a la alta presencia de Estados Unidos como componente dentro del esquema completo.

10. Densidad del grafo*// Hecho en Gephi*

```
Density: 0.000
```

Nuevamente, en un resultado sorprendente pero más bien anticlimático, la densidad obtenida es de 0 para al menos 3 posiciones decimales consideradas. Este resultado, en conjunto de la interpretación previa, indica que aunque los nodos cuenten con una conexión muy relevante, no todos se encuentran conexos entre sí, y, al ser un grafo de dimensiones amplias, el dato resulta lógico.

12. ¿El grafo es homogéneo o heterogéneo?

```
MATCH (n)
WITH COLLECT(DISTINCT LABELS(n)) AS NodeTypes
RETURN NodeTypes, COUNT(NodeTypes) AS TOTAL;
```

```
MATCH ()-[r]->()
WITH COLLECT(DISTINCT TYPE(r)) AS RelantionshipTypes
RETURN RelantionshipTypes, COUNT(RelantionshipTypes) AS TOTAL;
```

NodeTypes	TOTAL	RelantionshipTypes	TOTAL
[{"Movie"}, {"Country"}, {"Category"}, {"Person"}]	1	["FROM", "LISTED_IN", "DIRECTED", "IS_CITIZEN_OF", "ACTED_IN"]	1

Sí es homogéneo, ya que todos los nodos y relaciones tienen las mismas propiedades

Recomendaciones nativas

Una vez que se ha obtenido una vista general del grafo y se ha logrado comprender su estructura superficial, es posible comenzar con el desarrollo del objetivo principal presente: recomendaciones. Ahora bien, a continuación se presentarán una serie de estatutos para realizar recomendaciones basadas en ciertos parámetros de entrada desde Neo4j, es decir, no se implementará un sistema formal en donde las recomendaciones pudieran verse de manera nativa, no obstante, lo siguiente es la parte esencial que daría soporte a un sistema como el descrito. Esto podría verse como la implementación de los componentes del backend para que un desarrollador de frontend luego los tome para integrarlos a su aplicación.

Recalcando lo dicho, a continuación se presentan posibles recomendaciones existentes en un sistema de recomendaciones integral.

1. Recomendar películas basadas en el mismo director de una película de entrada

```
MATCH (m:Movie)←[:DIRECTED]-(p:Person)-[:DIRECTED]→(om:Movie)
WHERE m.title = ''
AND m <⇒ om
RETURN m.title as peliculaInicial,
       COLLECT(om.title) as peliculasRecomendadas
LIMIT 5;
```

Entrada → m.title = Catch Me If You Can

peliculaInicial	peliculasRecomendada
"Catch Me If You Can"	"Indiana Jones and the Kingdom of the Crystal Skull"
"Catch Me If You Can"	"Indiana Jones and the Last Crusade"
"Catch Me If You Can"	"Indiana Jones and the Raiders of the Lost Ark"
"Catch Me If You Can"	"Indiana Jones and the Temple of Doom"
"Catch Me If You Can"	"Jaws"

Esta recomendación es una de las más sencillas, ya que el sustento detrás de la misma se basa en la carrera del director de una película determinada, buscar sobre la lista de materiales dirigidos, y recomendar los mismos a un usuario. El riesgo que corre esta recomendación es la de encontrar la única que película que un director habría dirigido, de forma que no habrían más recomendaciones posibles.

2. Recomendar películas basadas en la misma categoría

```
MATCH (m:Movie)-[:LISTED_IN]→(c:Category)←[:LISTED_IN]-(om:Movie)
WHERE m.title = ''
AND m <⇒ om
RETURN m.title as peliculaInicial,
```

```

    m.title AS peliculasRecomendada,
    c.name AS categoria
ORDER BY peliculasRecomendada
LIMIT 5;

```

Entrada → m.title = La Familia P. Luche

peliculaInicial	peliculasRecomendada	categoria
"La Familia P. Luche"	"#blackAF"	"TV Comedies"
"La Familia P. Luche"	"100 Days My Prince"	"International TV Shows"
"La Familia P. Luche"	"100% Hotter"	"International TV Shows"
"La Familia P. Luche"	"12 Years Promise"	"International TV Shows"
"La Familia P. Luche"	"1983"	"International TV Shows"

Esta recomendación, aunque aún primitiva, cuenta con un mejor sustento que la anterior porque se basa en las categorías más relevantes del contenido de origen. Al basarse únicamente en el director como entrada, es posible que se hagan recomendaciones erróneas, por ejemplo, Steven Spielberg "E.T.", pero también dirigió "La lista de Schindler"; aunque resulte históricamente significativa, la segunda película no sería muy recomendable para audiencias infantiles.

3. Recomendar películas de un actor

```

CALL {
    MATCH (a:Person)-[:ACTED_IN]->(m:Movie)
    WHERE a.name = ''
    RETURN a.name AS Actor, m.title AS movie
    LIMIT 5
} WITH Actor, movie
RETURN Actor, COLLECT(movie) AS Películas;

```

Entrada → a.name = Salma Hayek

Actor
"Salma Hayek"

Películas
["Drunk Parents", "Cirque du Freak: The Vampire's Assistant", "The Pirates! Band of Misfits", "Grown Ups", "Septembers of Shiraz"]

La recomendación actual sirve para fanáticos o interesados en un actor o una actriz en particular de la cual se quiera conocer más acerca de su carrera artística. De esta manera, es más fácil conocer otras obras disponibles sin mayor complicación.

4. Recomendar películas “extranjeras”

```

MATCH (a:Person)-[act:ACTED_IN]->(m:Movie)-[:FROM]->(c:Country)
WITH a, act, m, c
MATCH (m)-[:LISTED_IN]->(ct:Category)

```

```

WHERE c.name <> '' AND TRIM(c.name) <> 'United States'
RETURN DISTINCT m.title AS Película, COLLECT(DISTINCT c.name) AS Países,
       COLLECT(DISTINCT ct.name) AS Categorías, COUNT(DISTINCT a.name) AS totalActores
ORDER BY totalActores DESC
LIMIT 10;
    
```

Película	Países	Categorías	totalActores
"Black Mirror"	["United Kingdom"]	["TV Dramas", "International TV Shows", "British TV Shows"]	50
"Heartbreak High"	["Australia"]	["International TV Shows", "TV Dramas", "Teen TV Shows"]	47
"Creeped Out"	["Canada", "United Kingdom"]	["TV Thrillers", "Kids' TV", "British TV Shows"]	47
"Arthur Christmas"	["United Kingdom"]	["Comedies", "Children & Family Movies"]	44
"Narcos"	["Colombia", "Mexico"]	["TV Dramas", "Crime TV Shows", "TV Action & Adventure"]	42
"Dino Girl Gauko"	["Japan"]	["Anime Series", "Kids' TV"]	33

Porque todas las películas hechas fuera de Estados Unidos son consideradas como extranjeras, esta recomendación atiende la cinéfila necesidad de indicar algunas películas extranjeras basadas en la cantidad de actores asociadas con ellas.

5. Recomendar directores prominentes dentro de la industria

```

MATCH (c:Country)-[:IS_CITIZEN_OF]-(d:Person)-[dr:DIRECTED]->(:Movie)
RETURN d.name AS Director, COLLECT(DISTINCT c.name) AS Nacionalidades,
       COUNT(dr) AS Películas
ORDER BY Películas DESC
LIMIT 5;
    
```

Director	Nacionalidades	Películas
"Michael Simon"	["United Kingdom", "Germany", "France", "Israel", "United States of America"]	30
"Rajiv Chilaka"	["India"]	22
"Clint Eastwood"	["Jamaica", "United States of America"]	14
"Scott Stewart"	["United Kingdom", "Australia", "Canada", "United States of America"]	12
"Walter Hill"	["United Kingdom", "Australia", "United States of America", "United Kingdom of Great Britain and Ireland"]	12

Para algunos amantes del cine, cantidad es mejor que calidad, es por eso que esta recomendación está dirigida a usuarios que busquen consumir más material sin importar su calidad final.

6. Recomendar películas en base a la nacionalidad de su director

```

MATCH (c:Country)←[:IS_CITIZEN_OF]-(d:Person)-[dr:DIRECTED]->(m:Movie)
MATCH (m)-[:FROM]->(mc:Country)←[:FROM]-(om:Movie)
WHERE d.name = '' AND c.name = mc.name AND m < m
RETURN d.name AS Director, c.name AS País,
       COLLECT(DISTINCT om.title)[..10] AS peliculasNacionales;
    
```

Entrada → d.name = Luis Estrada		Director	País
		"Luis Estrada"	"Mexico"
peliculasNacionales			
["El Chapo", "Lo que la vida me robó", "Rosario Tijeras", "1994", "Monarca", "Miss Dynamite", "Señora Acero", "Juana Inés",			

La excepción a la regla quizás siempre sea Guillermo del Toro, un director mexicano que ha incursionado en el mundo del cine con películas que no son consideradas extranjeras, sin embargo, con directores contemporáneos como Gary Alazraki o el comediante político Luis Estrada, es posible descubrir el mundo cinematográfico que ofrece México, y así con cualquier otro director.

7. Recomendaciones aleatorias dado un año específico

```

MATCH (m:Movie)
WITH m.title AS Película, m.releaseYear AS ry, RAND() AS r
WHERE ry = 0
RETURN Película
ORDER BY r
LIMIT 10;
    
```

Entrada → ry = 2021 (dos iteraciones)

Película	Película
"Tribes of Europa"	"Abla Fahita: Drama Queen"
"City of Ghosts"	"Murder Among the Mormons"
"A Week Away"	"You vs. Wild: Out Cold"
"Good on Paper"	"Under Suspicion: Uncovering the Wesphael Case"
"Prey"	"The Last Mercenary"
"Inside the World's Toughest Prisons"	"The Least Expected Day: Inside the Movistar Team 2019"
"Kitty Love: An Homage to Cats"	"Lee Su-geun: The Sense Coach"
"Mad for Each Other"	"Izzy's Koala World"
"I Care a Lot"	"Prank Encounters"
"Dolly Parton: A MusiCares Tribute"	"The Water Man"

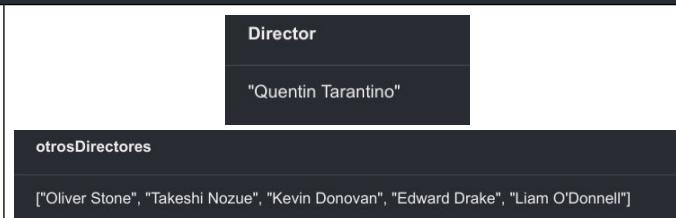
Cuando se ha visto absolutamente todo lo que se piensa que se ha podido ver, el procrastinador promedio hará lo posible por contar con nuevas distracciones. La recomendación actual no tiene ningún parámetro de preferencia más que el año, el cual incluso puede ser removido para hacer sugerencias más generales.

8. Recomendar otros directores del mismo género

```

MATCH (d:Person)-[:DIRECTED]->(m:Movie)-[:LISTED_IN]->(c:Category)
MATCH (c)←[:LISTED_IN]-(om:Movie)←[:DIRECTED]-(od:Person)
WHERE m ≠ om AND d ≠ od
AND d.name = ''
RETURN d.name AS Director,
       COLLECT(DISTINCT od.name)[..5] AS otrosDirectores;
    
```

Entrada → d.name = Quentin Tarantino



Cuando una película gusta, usualmente se quiere explorar más del género o del propio director, pero, cuando el material de este último se agota, es posible que se desee buscar otros directores similares. Una forma de lograr esto es a través del género predominante del director inicial, de forma que se tiene el nombre del mismo y únicamente con este dato se pueden encontrar personajes similares. Dato curioso: Oliver Stone dirigió “Asesinos por naturaleza”, y su guión fue escrito por Quentin Tarantino, lo cual le da sentido a la recomendación mostrada.

9. Recomendar películas de un país

```

MATCH (m:Movie)-[:FROM]->(c:Country)
WHERE c.name = ''
RETURN c.name AS País, COLLECT(DISTINCT m.title)[..10] AS Películas;
    
```

Entrada → c.name = Iceland (Islandia)



Aunque ya se han hecho algunas recomendaciones en donde el país está involucrado, la actual cuenta con este parámetro directo para mostrar películas nacionales sin otro filtro agregado. Esto puede ser útil para una exploración generalizada de contenidos mundiales.

10. Recomendar actores que también han sido directores en sus propias películas

```

MATCH (p:Person)-[:DIRECTED]->(m:Movie)←[:ACTED_IN]-(p)
RETURN COLLECT(DISTINCT p.name) AS actores_directores,
       m.title AS Película
LIMIT 10;
    
```

(Resultado en la siguiente página)

actores_directores	Película
["Funke Akindele"]	"Omo Ghetto: the Saga"
["David de Vos"]	"A Champion Heart"
["Spike Lee"]	"Do the Right Thing"
["Seth Rogen"]	"The Interview"
["Ramzy Bedia", "Eric Judor"]	"2 Alone in Paris"
["David Oyelowo"]	"The Water Man"
["Arvind Swamy", "Gautham Vasudev Menon"]	"Naverasa"
["Clint Eastwood"]	"Space Cowboys"
["Trey Parker"]	"Team America: World Police"
["Briar Grace-Smith"]	"Cousins"

Para la recomendación final se muestra otro componente de lo que podría ser un amplio sistema de recomendaciones, en donde se muestren personas habilidosas que fungen como actores dentro de sus propias películas que estas mismas personas dirigen.

Recomendaciones con sustento

Como se mencionaba en la introducción, dentro de este proyecto también se hará uso de algoritmos orientados a grafos, los cuales sirven para proporcionar información que no puede verse a simple vista con un análisis básico de la estructura en este caso presente. Ahora bien, no todos los algoritmos presentes dentro de la librería GDS serán utilizados, por lo cual antes de ingresar a su implementación vale la pena mencionar que estos se dividen en varias denominaciones particulares que atienden a propósitos distintos dentro de un mismo grafo. Únicamente se hará uso de los algoritmos encontrados en dos de estos subgrupos: algoritmos de detección de comunidad y algoritmos de similaridad. Ambas herramientas aplicadas brindarán un sustento adicional hacia las recomendaciones para lograr resultados potenciados y *posiblemente* mejores que las sugerencias nativas vistas con anterioridad.

Estructura básica de uso de GDS

Para muchos de los algoritmos que se implementarán, se seguirá una estructura en común que se presenta a continuación, sin embargo, si se desea analizar todas las implementaciones, es posible encontrarlas en el documento “gds_calls.cql” adjunto a este reporte o a través de GitHub.

El modelo base de ejecución es el siguiente:

```
// Creación de subgrafo/proyecto
CALL gds.graph.project(
    'nombre_del_proyecto',
    'TIPO_DE_NODO',
```

```

    {
      NOMBRE_DE_RELACION: {
        orientation: 'ORIENTACION_DE_LAS_RELACIONES [NATURAL | REVERSE | UNDIRECTED]',
        properties: 'PROPIEDAD(ES)_DE_LAS_RELACIONES'
      }
    }
);

// Cálculo de memoria requerida (Opcional y aplicable únicamente hacia algoritmos con
// calidad de producción)
CALL gds.ALGORITMO_A_APPLICAR.write.estimate(
  'nombre_del_proyecto',
  {writeProperty: 'PROPIEDAD_A_CALCULAR'}
)
YIELD nodeCount, relationshipCount, bytesMin, bytesMax, requiredMemory;

// Ejecución del algoritmo
CALL gds.[CALIDAD_DEL_ALGORITMO.]ALGORITMO_A_APPLICAR.stream('nombre_del_proyecto')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).PROPIEDAD_ORIGINAL_DEL_NODO As Name, score
ORDER BY score DESC, Name ASC;

// Creación de atributo con resultado individual
CALL gds.[CALIDAD_DEL_ALGORITMO.]ALGORITMO_A_APPLICAR.write(
  'nombre_del_proyecto',
  {writeProperty: 'PROPIEDAD_A_ESCRIBIR'}
)
YIELD centralityDistribution, nodePropertiesWritten
RETURN
  centralityDistribution.min AS MinScore,
  centralityDistribution.max AS MaxScore,
  centralityDistribution.mean AS MeanScore,
  nodePropertiesWritten;

// Resultado
MATCH (n:TIPO_DE_NODO)
RETURN n.PROPIEDAD_DEL_NODO AS NODO_X, PROPIEDAD_A_ESCRIBIR AS ALGORITMO_A_APPLICAR
ORDER BY n.ALGORITMO_A_APPLICAR DESC
LIMIT X;

```

Modificación estructural

Previo a la implementación de los algoritmos, existe un problema que debe ser atendido: relaciones faltantes. Para la mayoría de las ejecuciones planeadas es necesario contar con relaciones que vayan desde un tipo de nodo hacia el mismo, por ejemplo, una relación arbitraria *A* que vaya desde **PELÍCULA_A** hacia **PELÍCULA_B**. Como se vio en el análisis básico, este tipo de relación es inexistente hasta este punto del desarrollo, sin embargo, esto puede ser resultado tanto para películas como para dentro del grafo personas, ambos tipos de nodos a los cuales se aplicarán los algoritmos planeados.

Una *possible* implementación del código sería la siguiente:

```
// Personas
MATCH (pa:Person)-[:DIRECTED | ACTED_IN | IS_CITIZEN_OF]-(:Movie | Country)-[:IS_CITIZEN_OF | ACTED_IN | DIRECTED]-(pb:Person)
WHERE pa <~> pb
MERGE (pa)-[:COLLEAGUE_OF]-(pb);

// Películas
MATCH (ma:Movie)-[:FROM | LISTED_IN | DIRECTED]-(:Country | Category | Person)-[:DIRECTED | LISTED_IN | FROM]-(mb:Movie)
WHERE ma <~> mb
MERGE (ma)-[:RELATIVE_TO]-(mb);
```

Esta implementación puede considerarse rudimentaria e incluso deja fuera relaciones valiosas y las que crea no tienen peso. A pesar de lo mencionado, las relaciones que se crearían serían útiles para la mayoría de los procedimientos siguientes, pero, al involucrar productos cartesianos y complejidades de espacio y tiempo considerablemente grandes con crecimiento exponencial, se necesitaría una mejor planeación en conjunto de un mejor equipo de cómputo para concretar su ejecución. Es por esto que la siguiente sección comenzará con los algoritmos de similaridad antes que los de comunidad, puesto que los primeros serán los que proporcionen la modificación estructural exitosa.

Algoritmos y recomendaciones

Con la plantilla predefinida de GDS y la modificación integral del grafo tomada en consideración, se propone la aplicación de los siguientes algoritmos, donde (C) denota *Comunidad* y (S) denota *Similaridad*:

1. Weakly Connected Components (C): Este algoritmo contempla posibles entornos bidireccionales para relaciones no dirigidas donde no las hay, sin embargo, la estructura del grafo presente no permite usarlo como debería hacerse, por ende, se usará para proporcionar un agrupamiento básico.

2. Strongly Connected Components (C): Anteponiéndose a la idea principal del anterior, este algoritmo buscará nodos con fuerte relación entre ellos como lo pueden ser películas del mismo género o director sin tener que hacer uso de más nodos en la recomendación.
3. Louvain (C): Al ser uno de los algoritmos básicos de comunidad, este puede implantarse dentro del grafo para proporcionar un nivel adicional de seguridad ante las comunidades de nodos generadas.
4. Label Propagation (C): Otro método de revisar comunidades alternas es la de etiquetar a los nodos con un mecanismo alterno que pudiera revelar otro tipo de relaciones previamente desconocidas.
5. Node Similarity (S): El primero de los algoritmos de similaridad opera a través de dos tipos de nodos (los anteriores solo lo hacen con uno), encontrando un nivel semejanza entre ellos en función de los vecinos que tienen en común. Este posiblemente sea el mejor de los algoritmos para el propósito del proyecto actual.
6. K-Nearest Neighbors (S): El último algoritmo a utilizar, el cual nunca había sido implementado anteriormente, genera nuevas relaciones en base a la similitud de las propiedades de los nodos.

La implementación y traducción a lógica de negocio de estos algoritmos se presenta a continuación a través de un formato similar a las recomendaciones anteriormente mostradas.

1	
Encontrar nodos similares dentro del grafo en base a otro tipo de nodos (Node Similarity).	Hacer recomendaciones de películas basadas en categorías en común.
<pre> MATCH (m:Movie)-[s:SIMILAR]->(om:Movie) WHERE m.title = '' AND m <--> om AND s.score >= 0.75 RETURN m.title as peliculaOriginal, COLLECT(DISTINCT om.title)[0..5] as peliculasRecomendadas; </pre>	
Entrada → m.title = Catch Me If You Can	
peliculaOriginal	peliculasRecomendadas
"Catch Me If You Can"	["Cousins", "The Book of Henry", "My Girl", "The Best of Enemies", "The Life of David Gale"]

2	
Encontrar nodos similares dentro del grafo aplicando un algoritmo alterno al anterior (K-Nearest Neighbors).	Hacer recomendaciones de películas similares en base a su año de lanzamiento.
<pre> MATCH (m:Movie)-[ks:KSIMILAR]->(om:Movie) WHERE m.title = '' AND m <--> om </pre>	

```

AND ks.score >= 0.75
RETURN m.title AS peliculaOriginal,
       COLLECT(DISTINCT om.title)[0..5] AS peliculasRecomendadas;
    
```

Entrada → m.title = Catch Me If You Can

peliculaOriginal	peliculasRecomendadas
"La Familia P. Luche"	["Jim Gaffigan: Mr. Universe"]

3

Encontrar similaridad entre otro tipo de nodos dentro del grafo (Node Similarity).	Hacer recomendaciones de directores o actores basadas en su nacionalidad.
--	---

```

MATCH (d:Person)-[s:SIMILAR]->(p:Person)
WHERE d.name = ''
      AND d <> p
      AND s.score >= 0.33
RETURN d.name AS personaInicial,
       COLLECT(DISTINCT p.name)[0..5] AS personasRecomendadas;
    
```

Entrada → d.name = Luis Estrada

personalInicial	personasRecomendadas
"Luis Estrada"	["Mark Raso", "Francisco Ruiz Velasco", "Manolo Caro", "Yulene Olaizola"]

4

Encontrar conjuntos de nodos similares con una comprobación holgada sin demasiada complejidad (Weakly Connected Components).	Recomendar colecciones de películas pertenecientes a un mismo grupo (sin importar sus categorías).
--	--

```

MATCH (m:Movie)
RETURN m.WCC AS Categoria, COLLECT(m.title)[1..5] AS Peliculas;
    
```

Categoría	Películas
0	["My Little Pony: A New Generation", "The Starling", "Je Suis Karl", "Confessions of an Invisible Girl"]
1	["Ganglands", "Jailbirds New Orleans", "Kota Factory", "Midnight Mass"]
7	["Ankahi Kahaniya", "LSD: Love, Sex Aur Dhokha", "La diosa del asfalto", "I missed you: Director's Cut"]
14	["Myth & Mogul: John DeLorean", "I AM A KILLER", "Sophie: A Murder in West Cork", "Murder Maps"]
24	["The Secret Diary of an Exchange Student", "Slay", "Flower Girl", "Kambili: The Whole 30 Yards"]
35	["Shadow Parties", "Anjaam", "Gurgaon", "Agatha Christie's Crooked House"]

Esta recomendación puede ser útil para configuraciones iniciales de un servicio de streaming,

en donde un usuario seleccionaría algún grupo de su preferencia con contenidos que le llamen su atención para posteriormente realizar más recomendaciones con especialización puntual.

5

Encontrar conjuntos de nodos fuertemente conectados entre sí (Strongly Connected Components).

Realizar recomendaciones de grupos de personas del mismo gremio dentro de la industria.

```
MATCH (p:Person)
RETURN p.SCC AS Gremio, COLLECT(p.name)[1..8] AS Colaboradores
ORDER BY Colaboradores DESC;
```

312	[“Rano Karno”, “Dian Sastrowardoyo”, “Ifa Isfansyah”, “Jason Iskandar”, “Lucky Kuswandi”]
321	[“Neri Parenti”, “Gabriele Muccino”]
278	[“Kim Tae-hyung”, “Lee Kae-byeok”]
0	[“Julien Leclercq”, “Mike Flanagan”, “Robert Cullen”, “Haile Gerima”, “Theodore Melfi”, “Christian Schwochow”, “Adam Salky”]

Esta recomendación podría descubrir colaboraciones recurrentes entre un grupo de directores y actores, contando con posibles casas productoras.

6

Utilizar las relaciones creadas con K-Nearest Neighbors para agrupar un tipo de nodos (Louvain).

Recomendar grupos de películas de manera general sin filtros adicionales.

```
MATCH (m:Movie)
RETURN m.louvain AS Grupo, COLLECT(m.title)[1..5] AS Películas;
```

Grupo	Películas
2959	[“I’m Glad I Did”, “The New Legends of Monkey”, “Grey’s Anatomy”, “Deadwind”]
304	[“Heroes of Goo Jit Zu”, “Barbie Big City Big Dreams”, “Post Mortem: No One Dies in Skarnes”, “Lady Boss: The Jackie Collins Story”]
442	[“Johnny Test”, “The Dig”, “Mighty Little Bheem: Kite Festival”]
1210	[“Lee Su-geun: The Sense Coach”, “The Rational Life”, “Love Alarm”, “Paper Lives”]
996	[“Chhota Bheem”, “Kim’s Convenience”, “Ray”, “The Sons of Sam: A Descent into Darkness”]
1437	[“Jaguar”, “The Father Who Moves Mountains”, “Eden”, “Cinema Bandi”]

La recomendación actual puede ser usada cuando se termine de ver algún contenido digital y nuevas películas o series son sugeridas sin ningún tipo de filtro. En la práctica, evidentemente esto necesitaría alguna refinación adicional, ya que su uso no supervisado puede resultar en recomendaciones inesperadas e indeseadas.

7

Provisionar a los nodos con etiquetas indistintas para una rápida agrupación (Label Propagation).

Categorizar a directores y actores dentro de grupos determinados para su muestreo en una plataforma web.

```
MATCH (p:Person)
RETURN p.labelPropagation AS Grupo, COLLECT(p.name)[3..8] AS Personas
ORDER BY Personas DESC;
```

Grupo	Personas
9069	[“Saket Chaudhary”, “Lijo Jose Pellissery”, “Rahul Rawail”, “Nagesh Kukunoor”, “Vidhu Vinod Chopra”]
9259	[“Marcelo Piñeyro”, “Jorge Blanco”, “Jaume Balagueró”, “Francisco Ruiz Velasco”, “Luis Estrada”]
9262	[“Kenneth Gyang”, “Michelle Bello”, “Seyi Babatope”, “Pascal Atuma”, “Mahmood Ali-Balogun”]
9068	[“Jane Campion”, “Phillip Noyce”, “Anthony Minghella”, “Simon Wincer”, “Martin Campbell”]
9407	[“Ifa Isfansyah”, “Jason Iskandar”, “Lucky Kuswandi”]
9084	[“Hirotugu Kawasaki”, “Toshiyuki Tsuru”, “Tensai Okamura”, “Yoshiyuki Tomino”, “Yoshikazu Yasuhiko”]

Dejando de lado por un momento a los usuarios de las plataformas en donde estas recomendaciones pudieran residir, también pueden ser usadas como herramientas auxiliares a diseñadores y desarrolladores de las mismas páginas, de forma que la estructura visual de la misma puede ser acomodada de acuerdo a la cantidad de grupos que los datos hasta el momento se tendrían.

8

Aplicar Node Similarity nuevamente para un par distinto de nodos.

Encontrar películas similares en base a sus directores; estas recomendaciones no necesariamente deben ser otras películas dirigidas por los mismos directores.

```
MATCH (m:Movie)-[sd:SIMILAR]->(om:Movie)
WHERE m.title = ''
AND m <-> om
AND sd.score >= 0.75
RETURN m.title as películaOriginal,
       COLLECT(DISTINCT om.title)[0..5] as películasRecomendadas;
```

Entrada → m.title = Planet Hulk

películaOriginal	películasRecomendadas
“Planet Hulk”	[“Batman: The Killing Joke”, “Thor: Tales of Asgard”]

9

Repetir nuevamente Node Similarity sobre el mismo conjunto anterior pero con una relación distinta.	Encontrar películas similares en base a los actores que han participado en las mismas.
---	--

```

MATCH (m:Movie)-[sa:SIMILAR_MA]->(om:Movie)
WHERE m.title = ''
AND m <--> om
AND sa.score >= 0.75
RETURN m.title as peliculaOriginal,
       COLLECT(DISTINCT om.title)[0..5] as peliculasRecomendadas;
    
```

Entrada → m.title = Django Unchained

peliculaOriginal	peliculasRecomendadas
"Django Unchained"	["Mars", "American Son", "Away", "The Unicorn", "187"]

10

Encontrar la similaridad entre nodos con una combinación alterna de pares y relaciones sin dirección (sin restricción).	Recomendar películas basadas en sus países de origen.
---	---

```

MATCH (m:Movie)-[sc:SIMILAR_MC]->(om:Movie)
WHERE m.title = ''
AND m <--> om
AND sc.score >= 0.75
RETURN m.title as peliculaOriginal,
       COLLECT(DISTINCT om.title)[0..5] as peliculasRecomendadas;
    
```

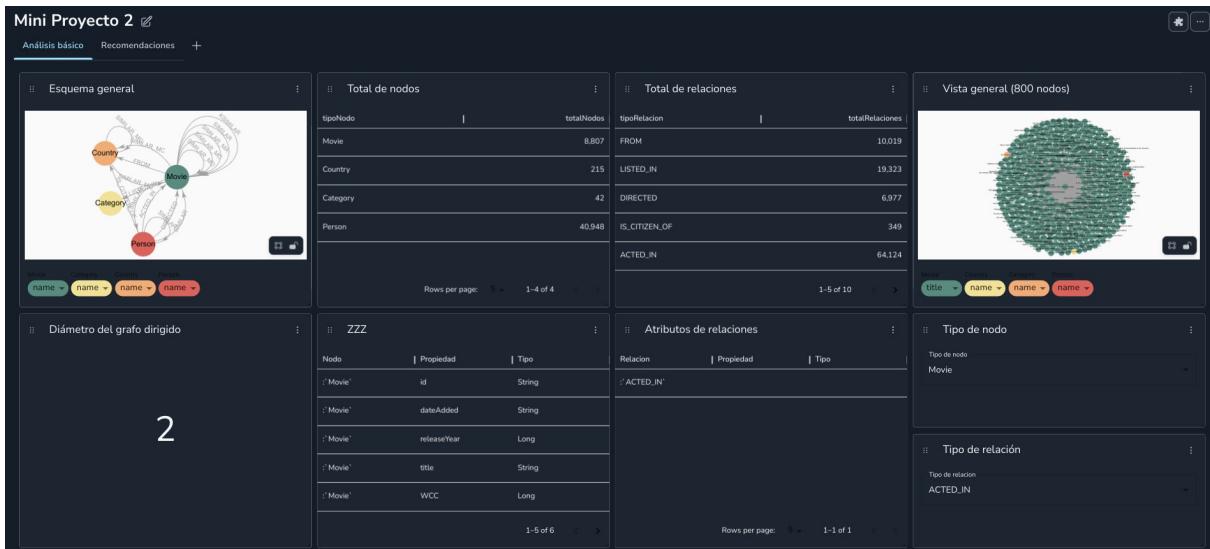
Entrada → m.title = Pulp Fiction

peliculaOriginal	peliculasRecomendadas
"Pulp Fiction"	["Jaws", "Jaws 3", "The Starling", "Jaws 2", "Dark Skies"]

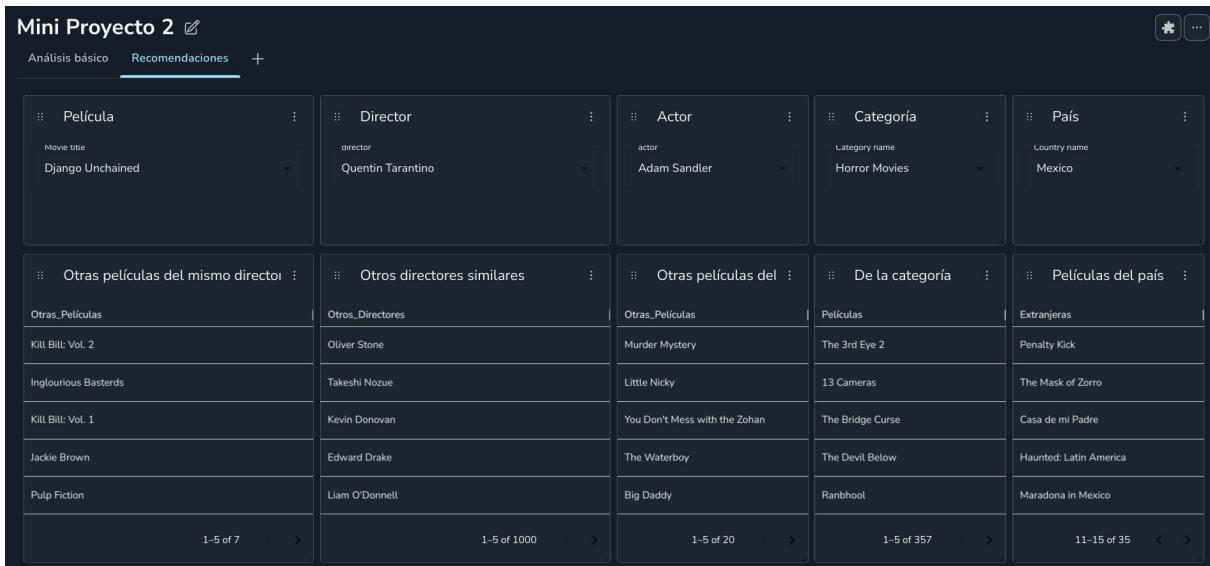
Presentación de los resultados

Una excelente herramienta visual para mostrar los resultados anteriores de forma dinámica es el uso de NeoDash (de Jong et al., s.f.), una utilidad agregada a Neo4j la cual permite construir tableros de información sin requerir demasiadas configuraciones iniciales para su implementación. Lo que se presenta a continuación cuenta con el soporte de las consultas mostradas anteriormente, de forma que, aunque no sea visible, el código es el mismo que se ha usado.

Análisis básico



Recomendaciones



Conclusiones y recomendaciones

Como único elemento implementador de este proyecto y por la parte técnica de toda la esencia, la primera y más fuerte recomendación que debe de hacerse es la de agregar peso a las relaciones. Esto puede lograrse de muchas maneras, por ejemplo, si un país visualiza mucho un contenido, una nueva relación “WATCHES” o algo similar se crearía con este mismo dato embebido como propiedad; a mayores vistas, mayor peso. Este sencillo mecanismo de adición de datos aumentaría exponencialmente la facilidad con la que se analizarían los datos, ya que la base proporcionada cuenta con un relativo nivel bajo de calidad, optando más bien por la cantidad.

La siguiente recomendación es la de mostrar a los usuarios como consumidores de contenido. Aunque el conjunto analizado es suficiente para comenzar a plantear el sistema inicial de recomendaciones, sería algo invaluable el contar con métricas de visualización y tiempo de consumo de los servicios de streaming. Muchas compañías anteponen barreras para llegar al consumo de estos datos, y parte de la decepcionante realidad siendo usuario es que la falta de moral y ética finalmente impulsarán la venta masiva de esta información, solo habrán procesos hastiantes para hacer que esto parezca como algo que no se da. La cuestión es que, dejando de lado la legalidad de estas acciones, si ya se tienen esos datos ¿Por qué no aprovecharlos? Si fuera el caso, solo existirían dos relaciones más, ambas salientes del nodo de tipo persona: una hacia nodos de tipo película para catalogar los contenidos que consume, y otra hacia su país de origen. Con la simple presencia de este par de nuevas relaciones, todo un mundo analítico emerge casi como si se tratase de un milagro y las posibles recomendaciones serían aún más atinadas que las revisadas.

Ahora bien, para dar final a este proyecto, y a manera personal, en algún tiempo pasado existía la pregunta de cómo se implementaría algo como lo revisado en una plataforma web real. La respuesta en aquellos entonces fue muy sencilla: usar Neo4j como base de datos. No se había logrado una comprensión de la simplicidad detrás de la solución, ya sea por ignorancia tecnológica o por estar encadenado a lo que dictaban los ejemplos de libro. En esta ocasión, aunque el material previo que dio sustento al desarrollo, así como algunas bases también fueron revisadas con anterioridad, la realidad es que no hubo una senda predefinida para dar solución a los requerimientos del proyecto, sino que existió una libertad creativa para la entrega, algo que muchas veces no es posible hacer en entornos reales pero que definitivamente se agradece.

Al tomar la materia en un semestre avanzado, es posible comenzar a hacer abstracciones de los conceptos aprendidos para implementar sistemas mucho más complejos pero con las bases puras de lo visto en esta ocasión. Pudieran incluso implementarse árboles léxicos dentro del mismo Neo4j para contar con otra herramienta adicional para brindar recomendaciones en contextos generales, pero sustentados por el mismo motor a través de la misma tecnología.

Con algo de suerte, el siguiente proyecto será igual de interesante y útil para el futuro profesional, pero quizás con menos requerimientos. Siempre ha sido válido soñar.

Bibliografía

Flynn, J. (2023). *18 Average Screen Time Statistics [2023]: How Much Screen Time Is Too Much?*. Recuperado de <https://www.zippia.com/advice/average-screen-time-statistics/>.

Moskowitz, R., Brow, K. A. (2023). *Celebrate National Streaming Day with the most popular streaming services out there.* Recuperado de <https://www.usatoday.com/story/tech/reviewed/2023/05/19/national-streaming-day-most-popular-streaming-services-ranked-netflix-disney/70235316007/>.

Tassi, P. (2022). *Netflix's Quality Control Problem Is Getting Worse, Not Better.* Recuperado de <https://www.forbes.com/sites/paultassi/2022/08/30/netflixs-quality-control-problem-is-getting-worse-not-better/?sh=18f51e43a505>.

Le, J. (2019). *Recommendation System Series Part 1: An Executive Guide to Building Recommendation System.* Recuperado de: <https://towardsdatascience.com/recommendation-system-series-part-1-an-executive-guide-to-building-recommendation-system-608f83e2630a>.

Neo4j. (s.f.). Neo4j Graph Database. Recuperado de <https://neo4j.com/product/neo4j-graph-database/>.

Memgraph. (2023). *Graph Database vs Relational Database.* Recuperado de <https://memgraph.com/blog/graph-database-vs-relational-database>.

Neo4j. (s.f.). *The Neo4j Graph Data Science Library Manual v2.4.* Recuperado de <https://neo4j.com/docs/graph-data-science/current/>.

Needham, M., & Hodler, A. E. (2021). Graph algorithms: Practical examples in Apache Spark and neo4j. O'Reilly.

de Jong, N., Conjeaud, M., Agudelo Ramirez, H., et al. (s.f.). NeoDash - Dashboard Builder for Neo4j. Recuperado de <https://neo4j.com/labs/neodash/>.