
Introduction to
Neo4j
neo4j

SetUp

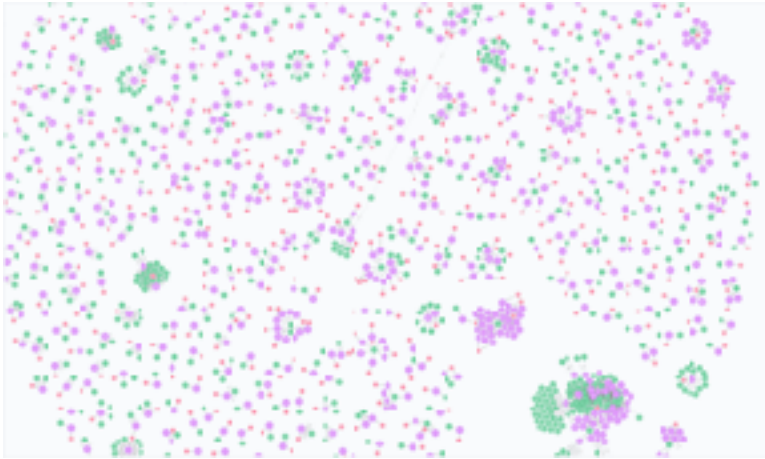
- First go to https://s3.amazonaws.com/neo4j-sandbox-usecase-datastores/v3_5/yelp.db.zip
- Unzip the file
- Create a new Local Graph
- Hit the Manage button followed by the Open Folder icon at the top
- Click on the data folder, followed by databases
- Copy and paste the yelp.db folder into /data/databases
- Change the name from yelp.db to graph.db
- Start the graph
- Click the Graph Algorithms Playground application at the top
- Click the lower left yellow icon
- Load the Game of Thrones and European Roads data into your yelp database which should be active currently

What is Neo4j?

Graph Platform

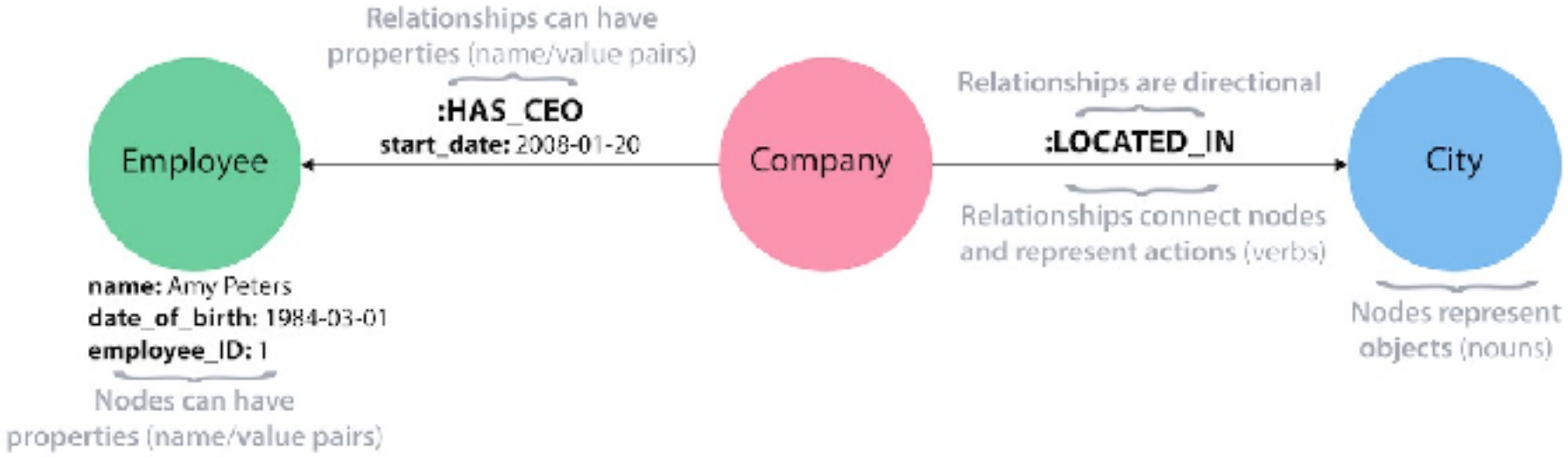
```
MATCH (a:Address)<-[r:REGISTERED_ADDRESS]-(o:Officer)-->(e:Entity)
WHERE a.address CONTAINS "New York"
RETURN *
```

- Database management system (**DBMS**)
- **Property Graph** data model
- **Cypher** query language
- Graph **analytics**
- Data **visualization**
- **Developer tool** for building applications



neo4j.com/

Labeled Property Graph



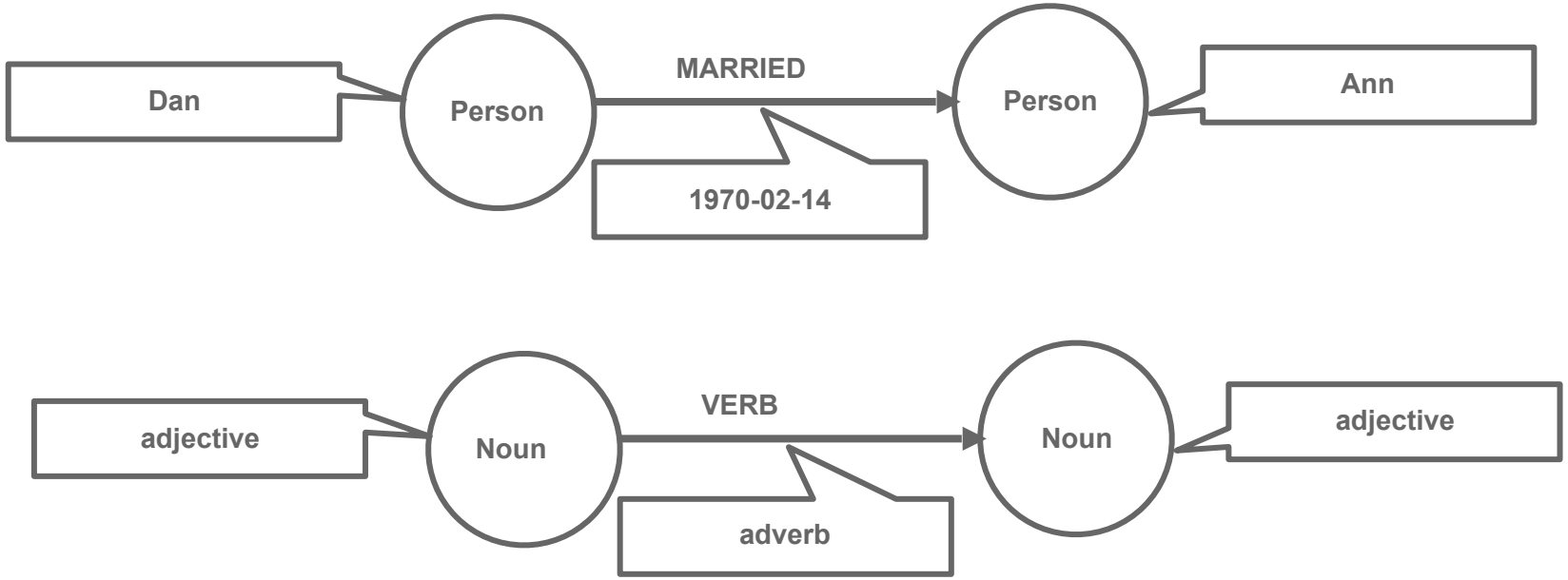
What is Cypher?

- Declarative
- Focuses on **what**, not how
- ASCII art syntax

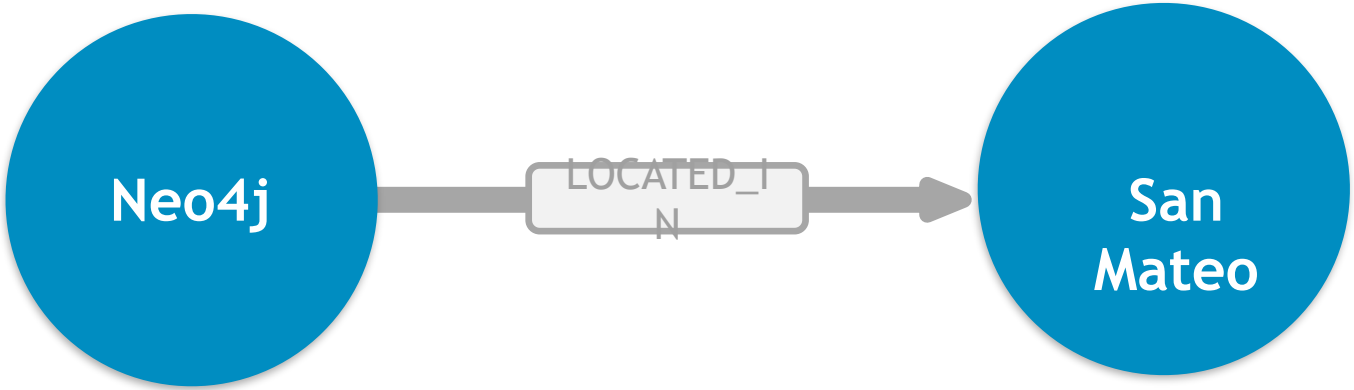

```
MATCH (A)-[:LIKES]->(B)
```


neo4j

Cypher is readable



Cypher Query Language

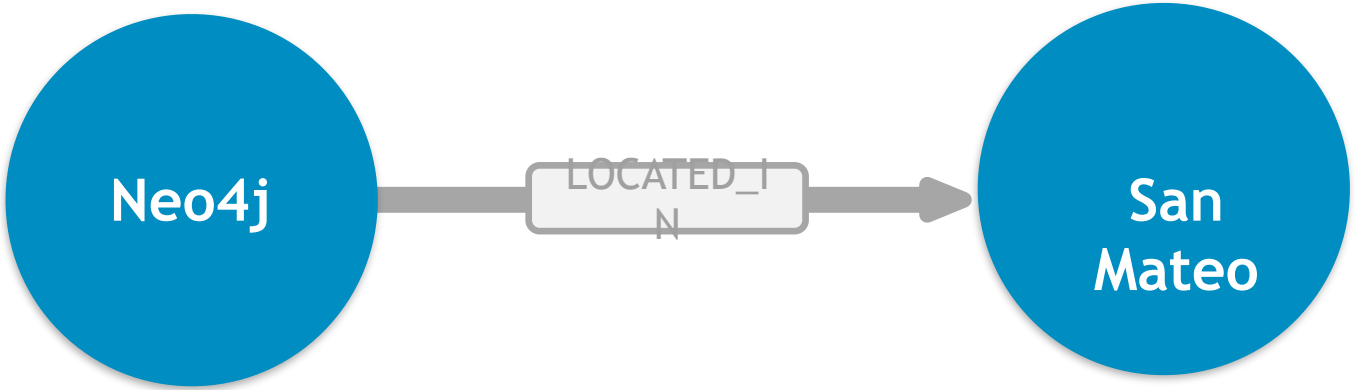


NODE NODE

```
CREATE (:Company { name:'Neo4j'} ) -[:LOCATED_IN]-> (:City { name:'San Mateo'} )
```

LABEL PROPERTY LABEL PROPERTY

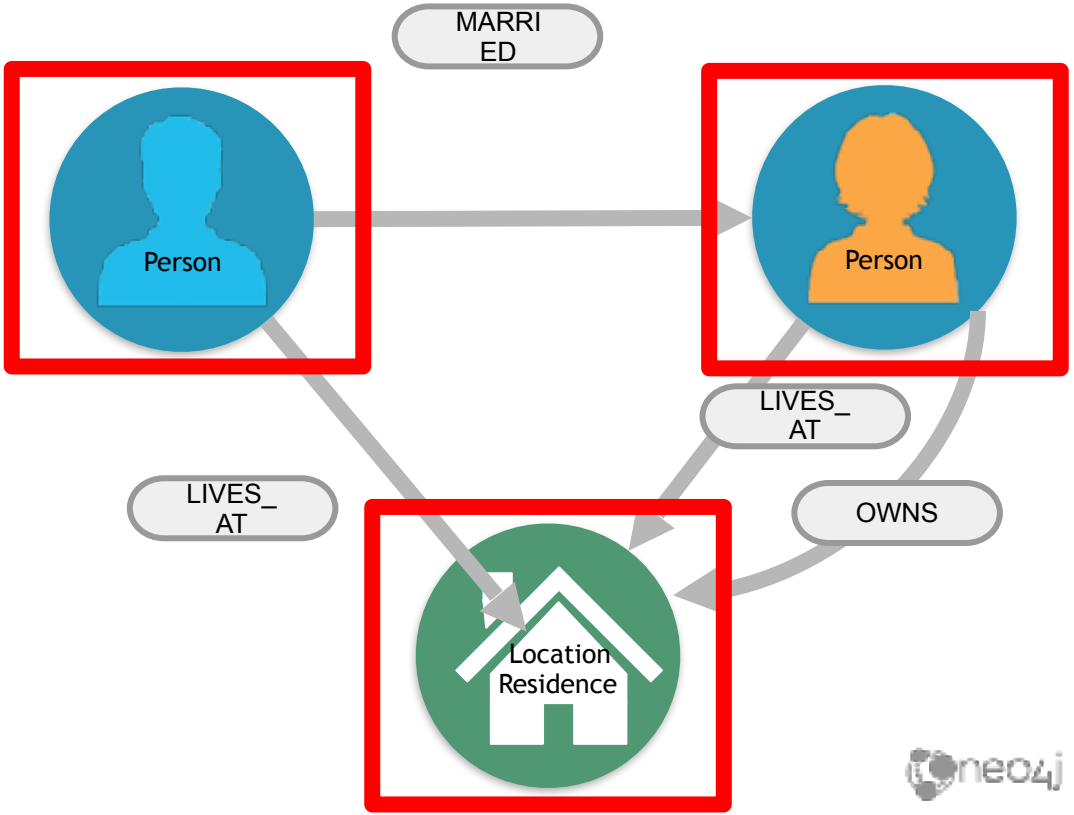
Cypher Query Language




```
MATCH (:Company { name:'Neo4j'} ) -[:LOCATED_IN]->
(place)
RETURN place
```

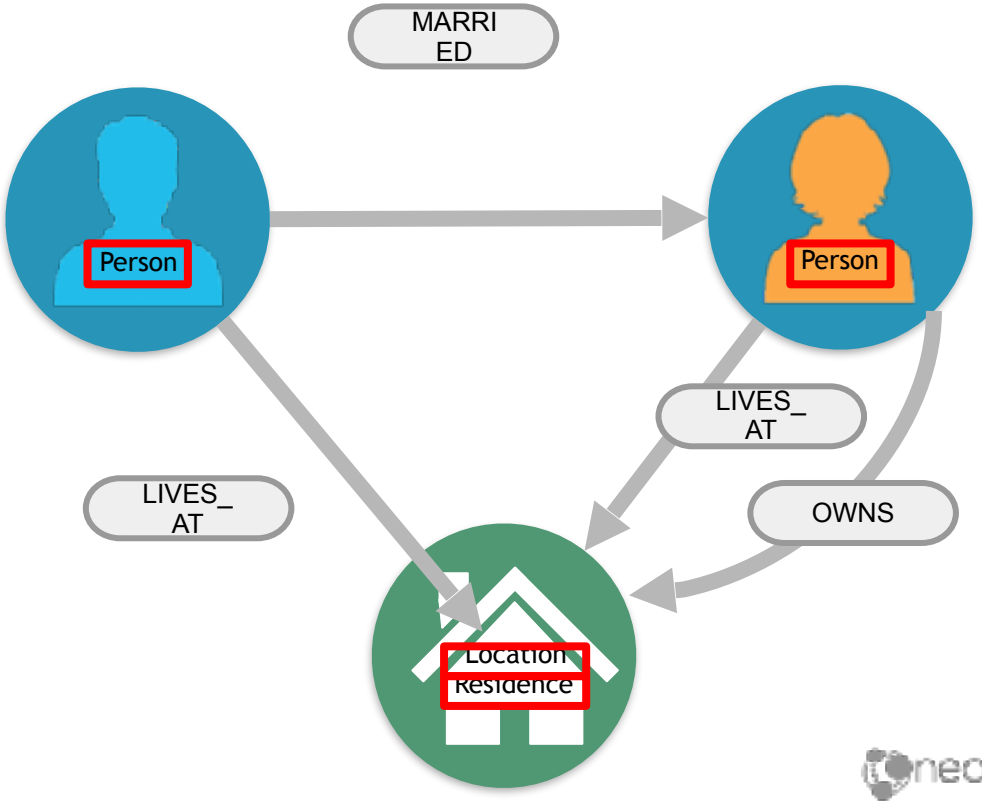
Nodes

```
()
(p)
(l)
(n)
```


MARRI
ED
Person
Person
LIVES_
AT
LIVES_
AT
OWNS
Location
Residence

Labels

```
(:Person)
(p:Person)
(:Location)
(l:Location)
(n:Residence)
(x:Location:Residence)
```

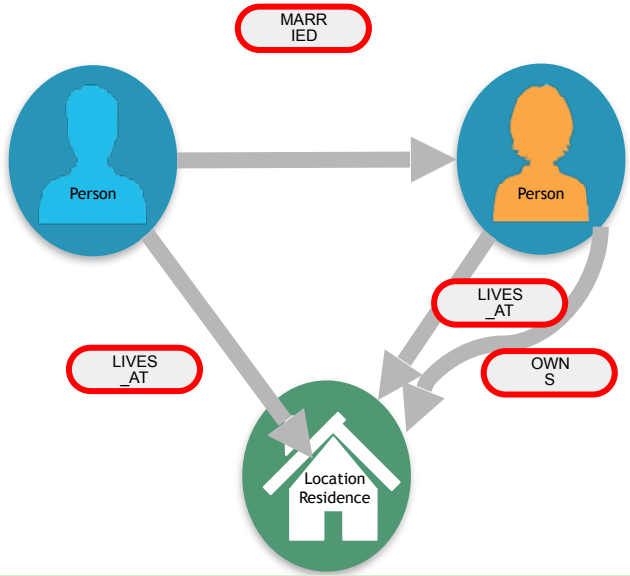


Using MATCH to retrieve nodes

```
MATCH (n) // returns all nodes in the graph
RETURN n
```

```
MATCH (p:Person) // returns all Person nodes in the graph
RETURN p
```

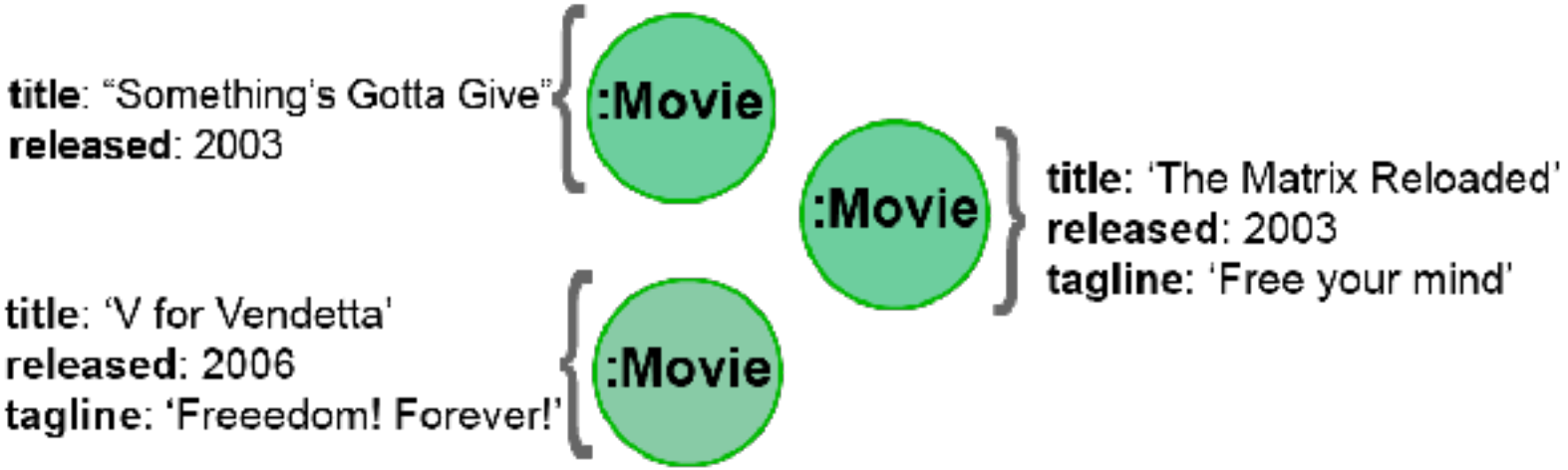
Querying using relationships




```
MATCH (p:Person)-[:LIVES_AT]->(h:Residence)
RETURN p.name, h.address
```

```
MATCH (p:Person)-[:LIVES_AT]-(h:Residence) //undirected
RETURN p.name, h.address
```

Properties


title: "Something's Gotta Give"
released: 2003
:Movie
:Movie
title: 'The Matrix Reloaded'
released: 2003
tagline: 'Free your mind'
title: 'V for Vendetta'
released: 2006
tagline: 'Freeedom! Forever!'
:Movie

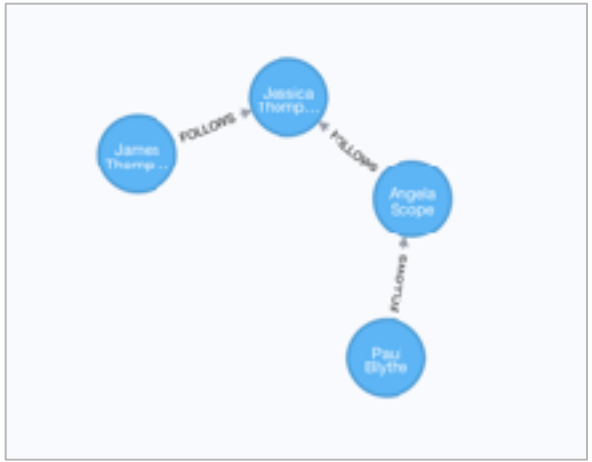
Retrieving nodes filtered by a property value

Find all *people* born in *1970, returning the nodes*:

```
MATCH (p:Person {born: 1970})
RETURN p
```

```
MATCH (p:Person)
WHERE p.born = 1970
RETURN p
```

Using patterns for queries

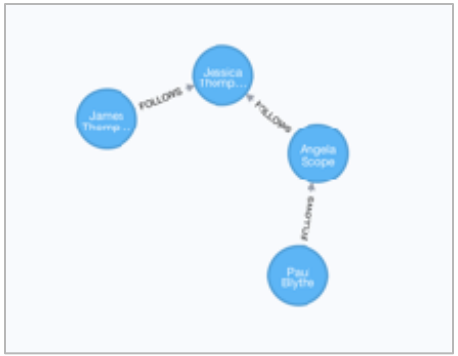


Find all people who *Angela Scope* follows, returning the nodes:

```
MATCH (:Person {name:'Angela Scope'})-[:FOLLOWS]->(p:Person)
RETURN p
```


neo4j

Traversing relationships

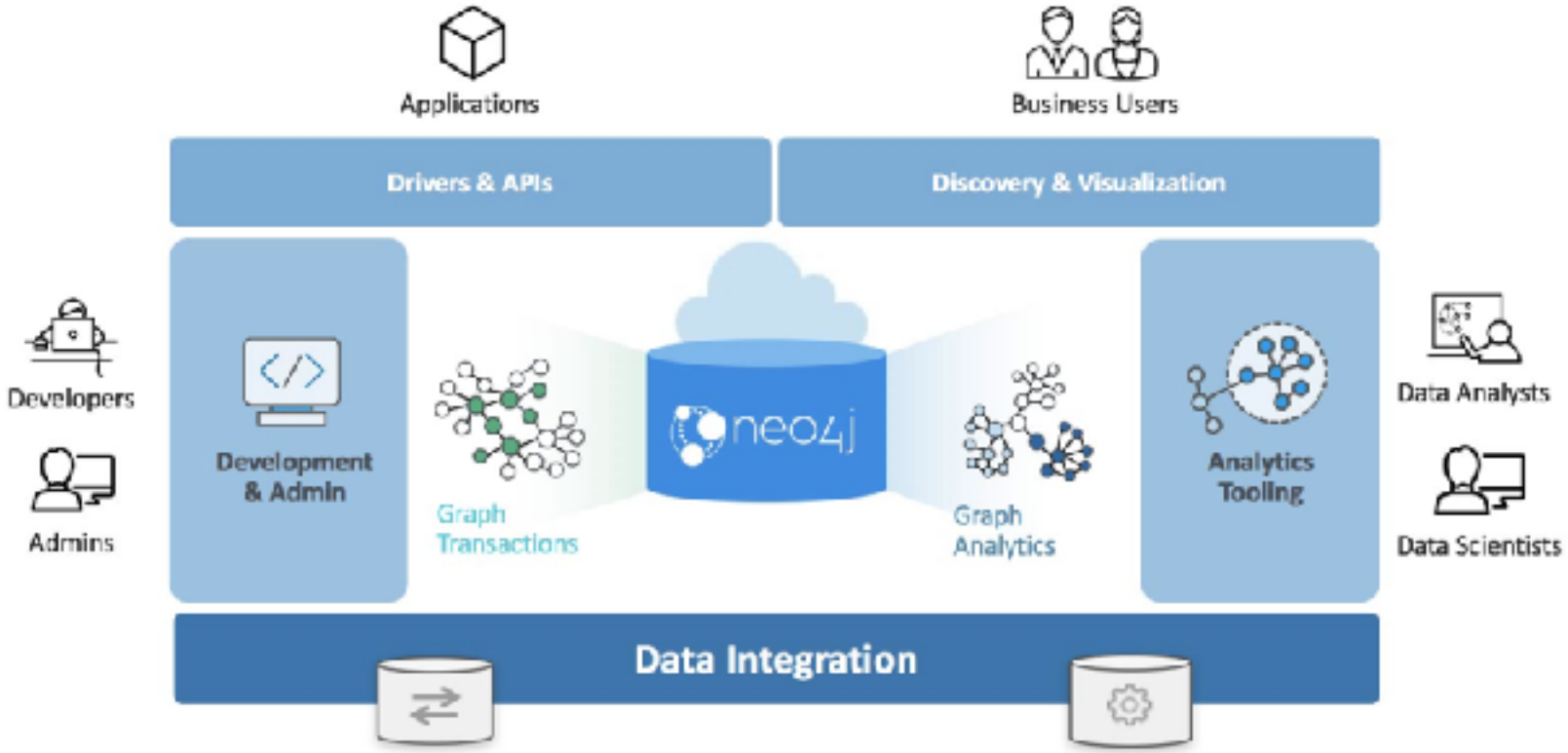


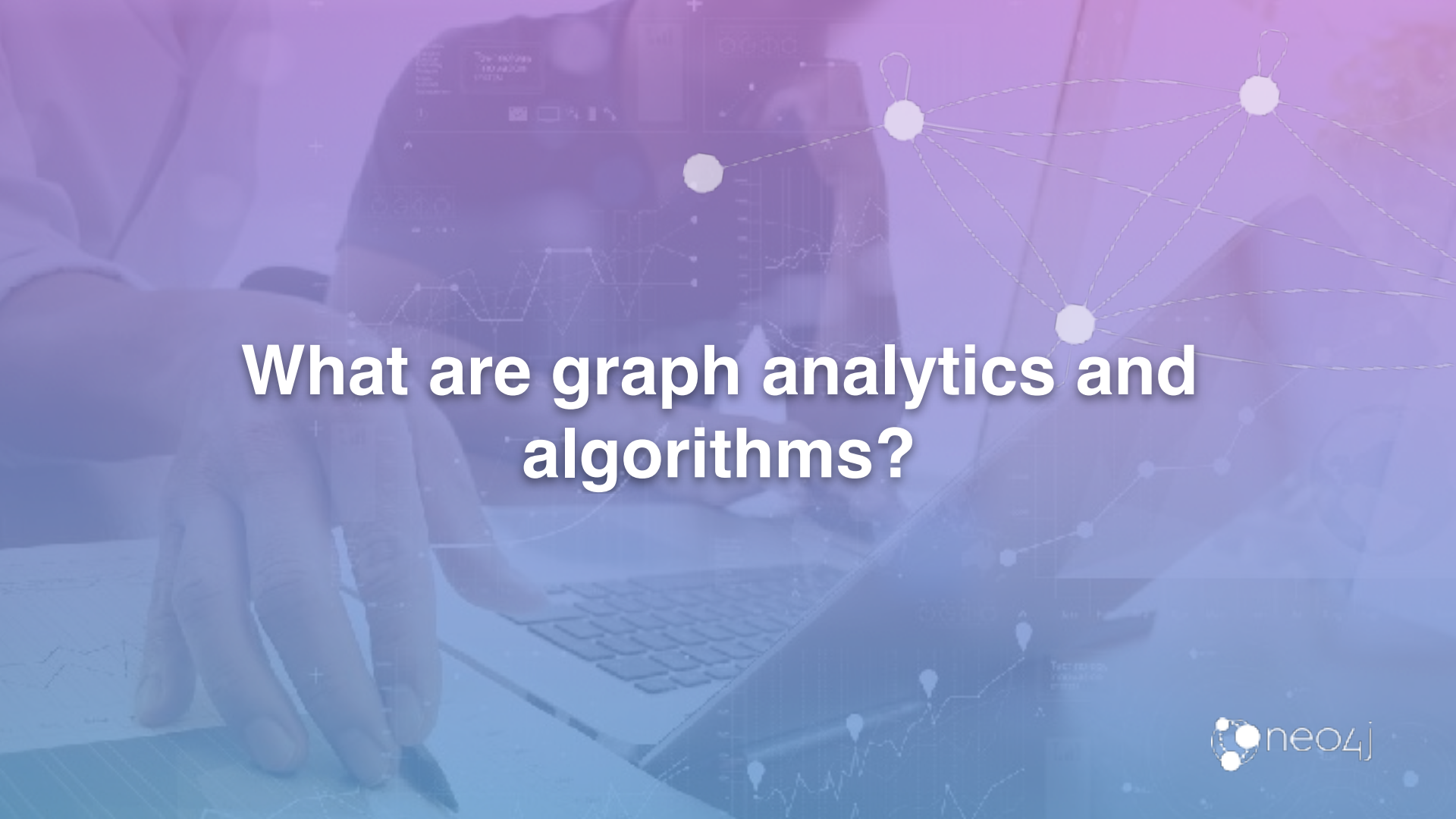
Find persons who follow persons who follow *Jessica Thompson* returning the people as nodes:

```
MATCH   (p:Person)-[:FOLLOWS]->(:Person)-[:FOLLOWS]->
        (:Person {name:'Jessica Thompson'})
RETURN p
```


neo4j

Neo4j Graph Platform architecture




What are graph analytics and algorithms?
neo4j

Graph analytics

Query (e.g. Cypher/Python)

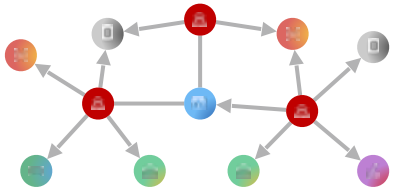
Real-time, local decisioning and pattern matching

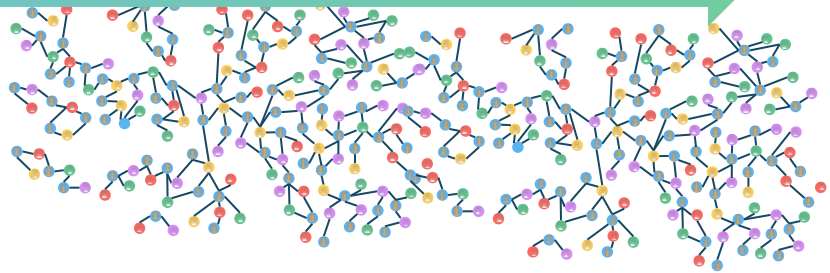
Graph Algorithms libraries

Global analysis and iterations

Local patterns

Global computation





You know what you're looking for and making a decision

You're learning the overall structure of a network, updating data, and predicting

Don’t need graph algorithms to answer . . .

- Questions with just a few connections or flat (not nested).
- Questions solved with specific, well-crafted queries.
- Simple statistical results (sums, averages, ratios).
- Example:
  - Regular reporting based on defined criteria and well-organized data

Common types of graph algorithms

Pathfinding & Search

Centrality / Importance

Similarity

Link Prediction

Community Detection+

Basic stats

Decomposition, covering & coloring

Network flow & percolation

Subgraph & isomorphism

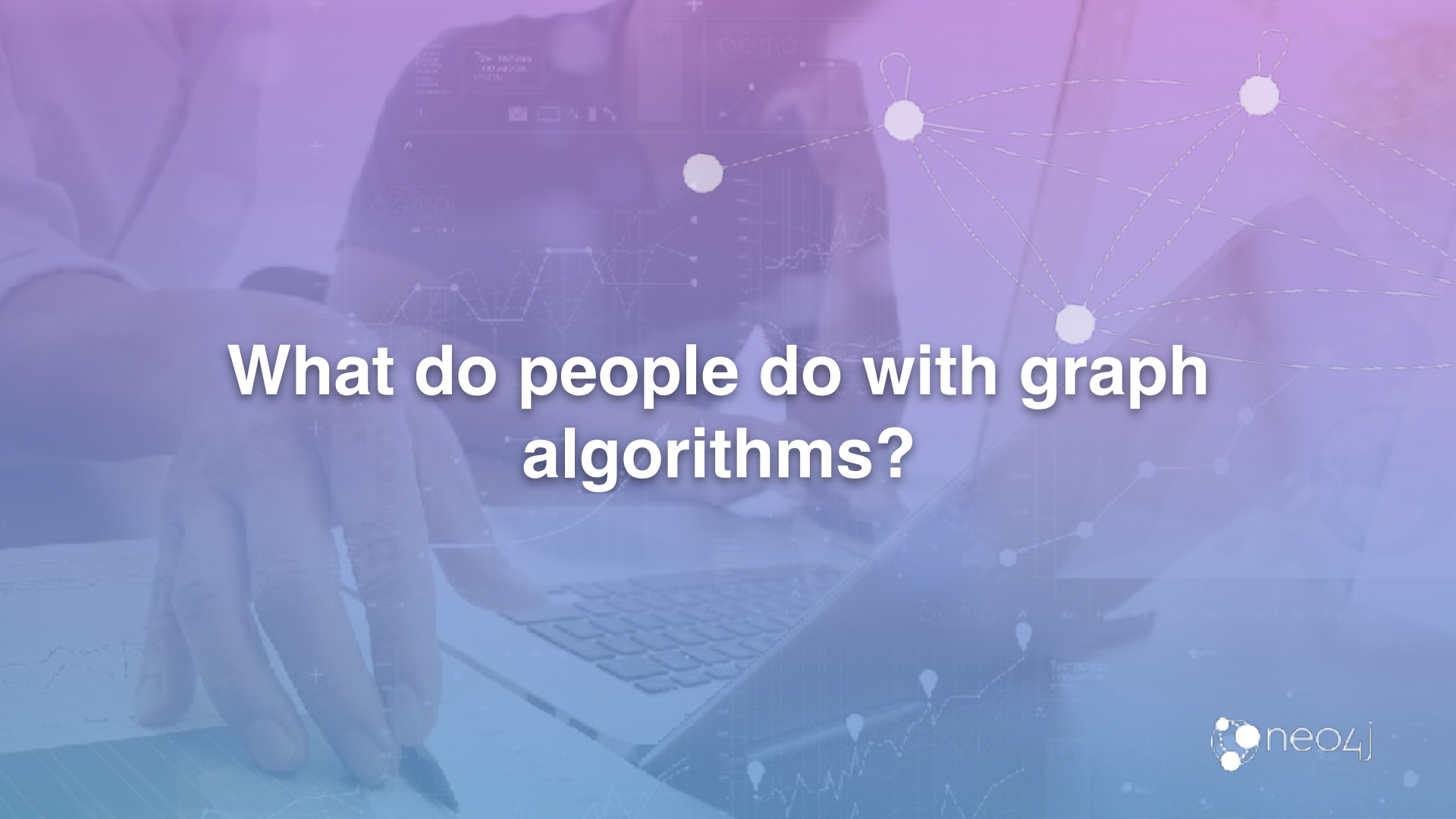
ML Workflow

Assortative mixing

Classic graph algorithms categories

Other common categories

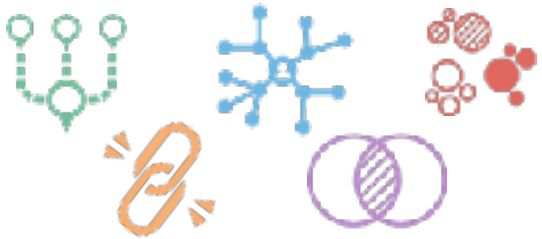
So many others!


What do people do with graph algorithms?
neo4j

Using graph algorithms

Explore, plan, measure

Find significant patterns and plan for optimal structures



Score outcomes and set a threshold value for a prediction

Machine learning

Use the measures as features to train an ML model

1st node	2nd node	Common neighbors	Preferential attachment	Label
1	2	4	15	1
3	4	7	12	1
5	6	1	1	0

Understand and predict complex behavior

Propagation pathways

Flow & dynamics

Interactions & resiliency

Requires understanding relationships and structures

Common graph data science applications

Financial crimes

Drug discovery

Recommendations

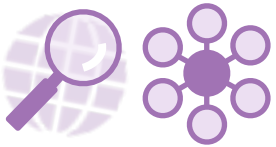
Customer segmentation

Cybersecurity

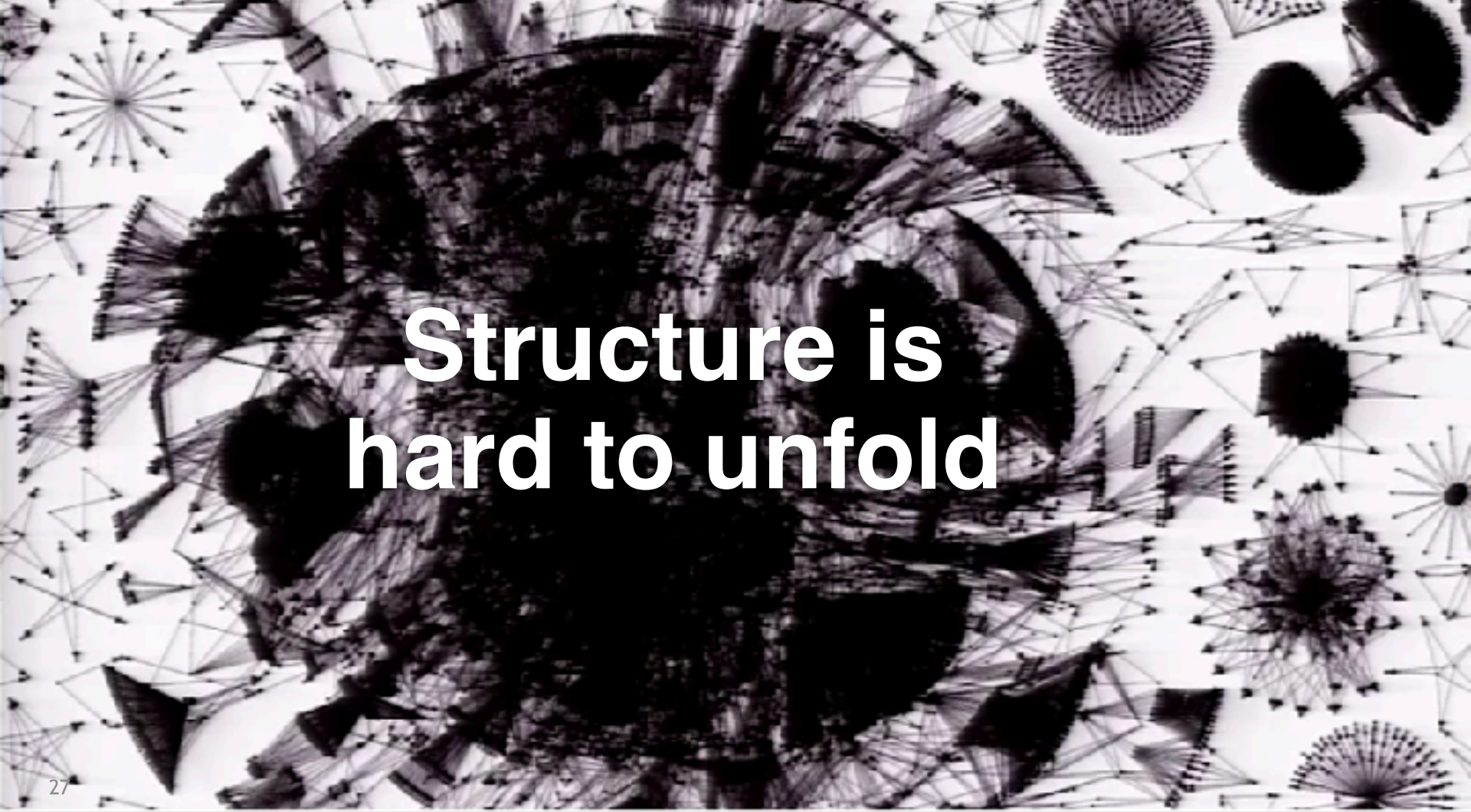
Churn prediction

Predictive maintenance

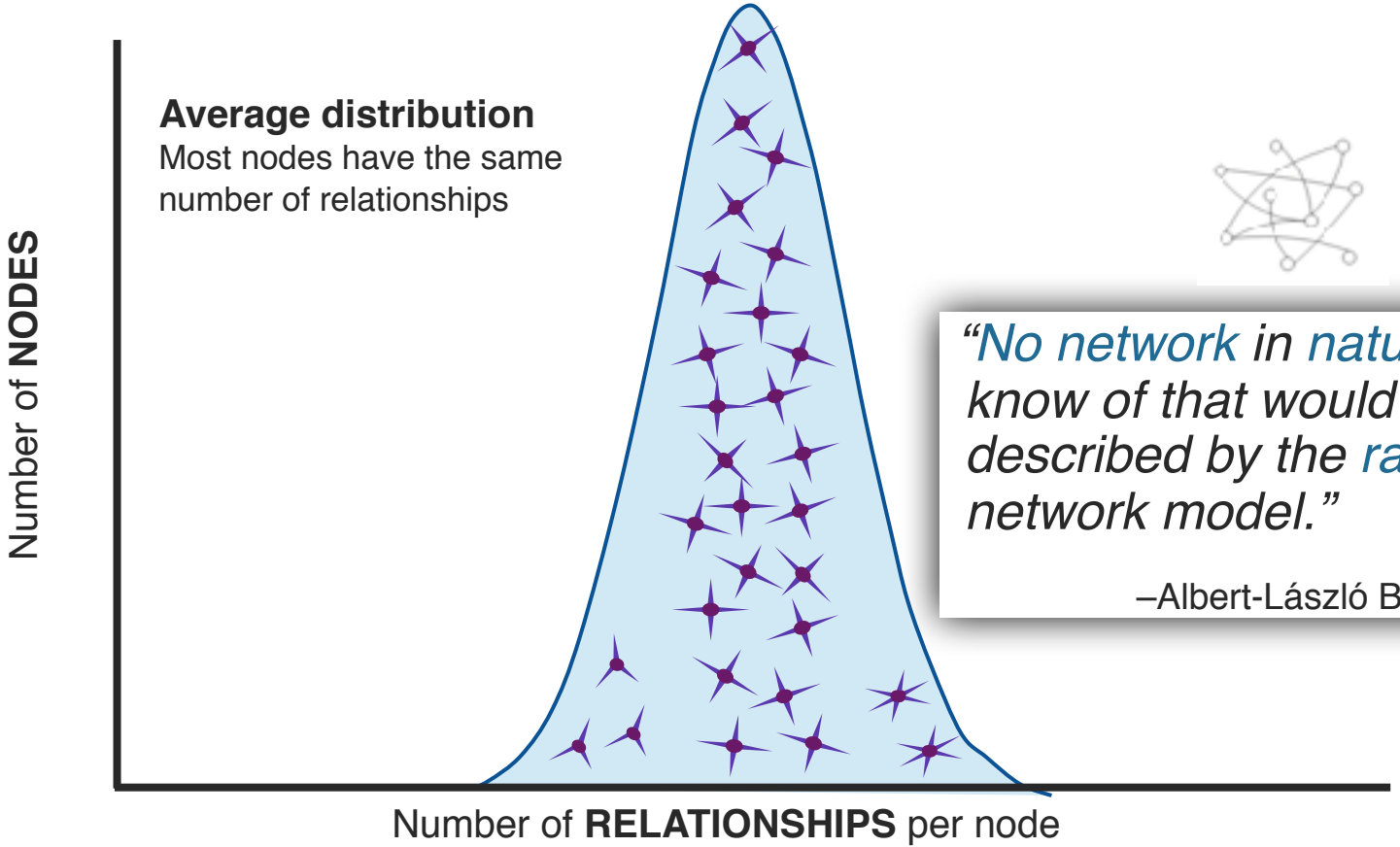
Search & MDM




Why not use other types of analytics?
neo4j


Structure is
hard to unfold

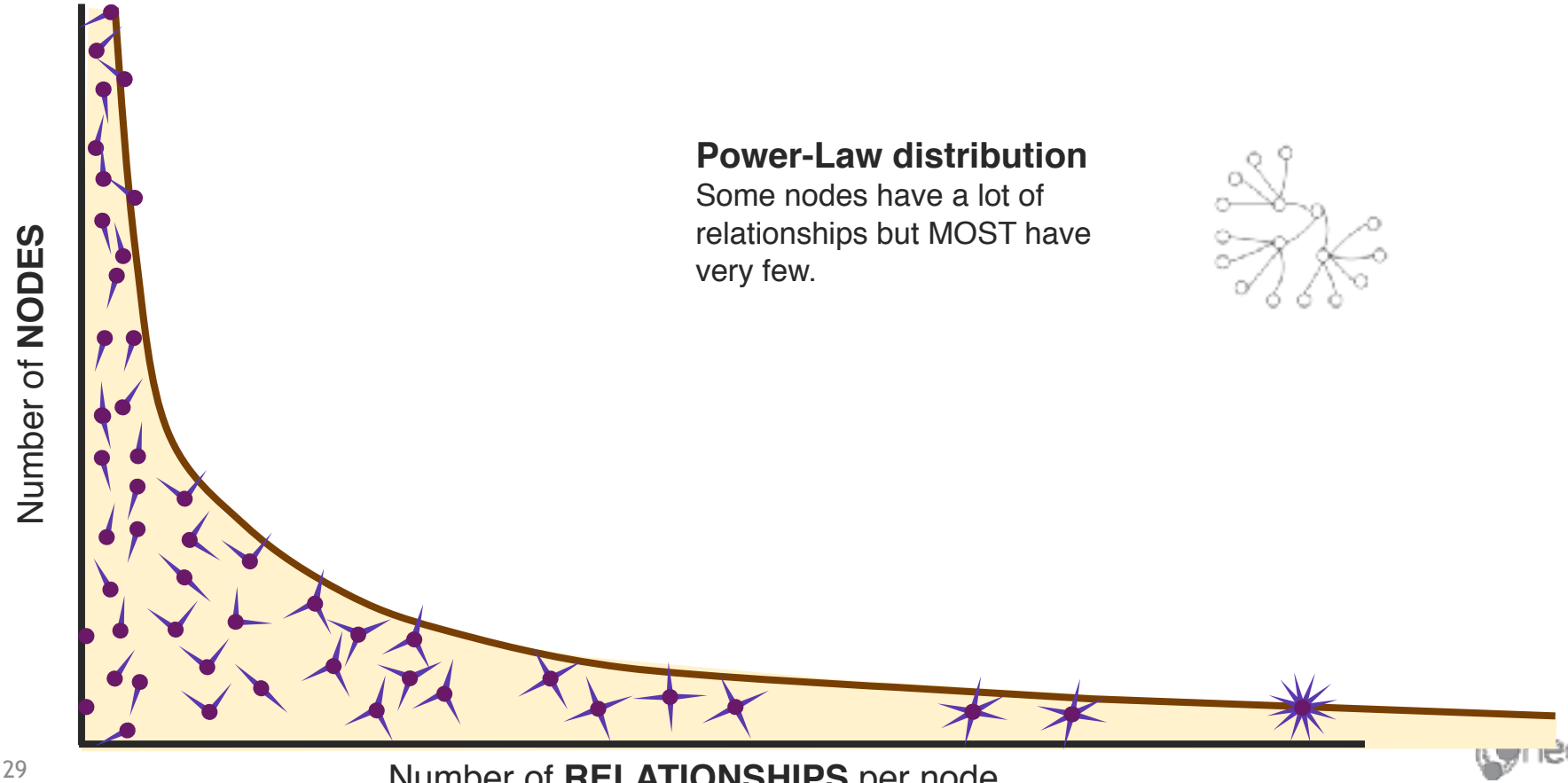
Random network



"No network in nature that we know of that would be described by the random network model."

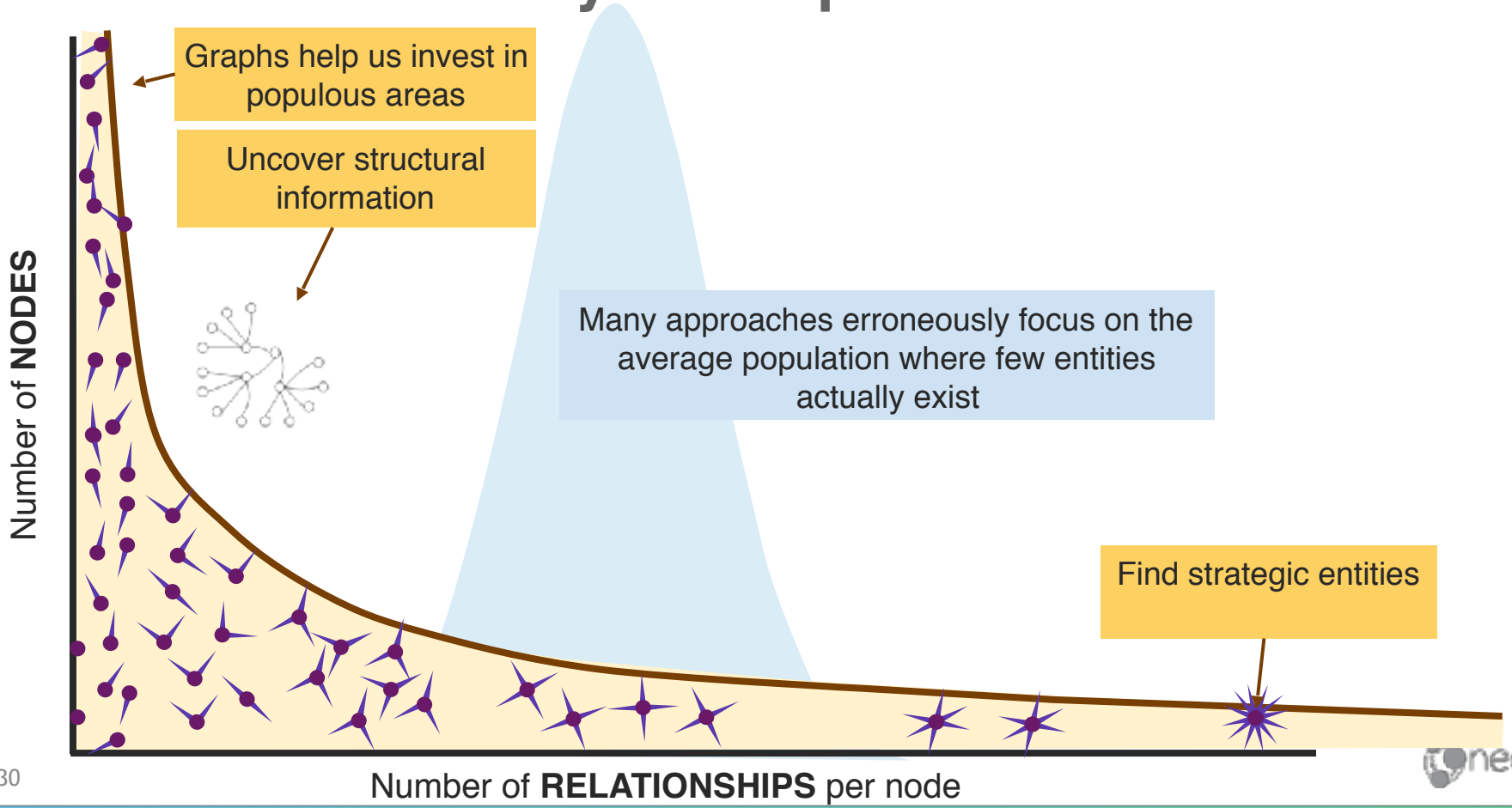
–Albert-László Barabási

Structured network

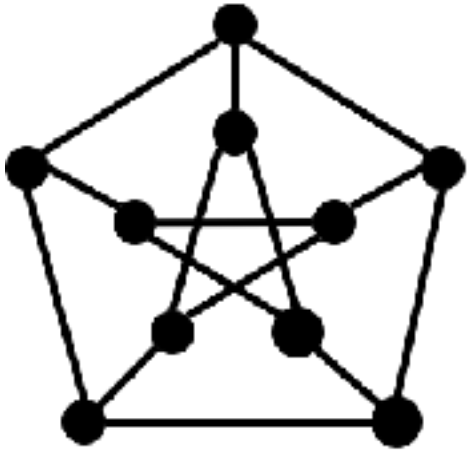


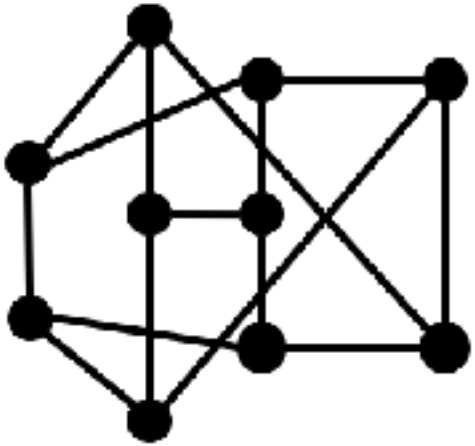
Power-Law distribution
Some nodes have a lot of relationships but MOST have very few.

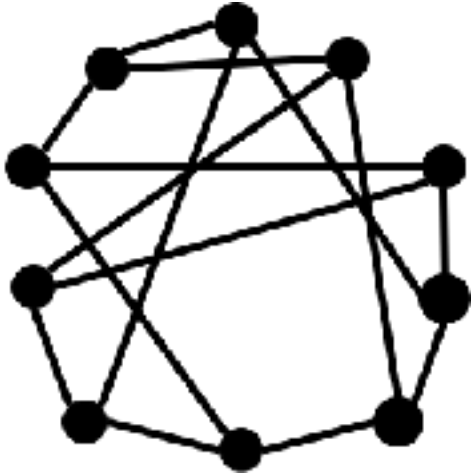
Structure and density are important



What do these graphs have in common?

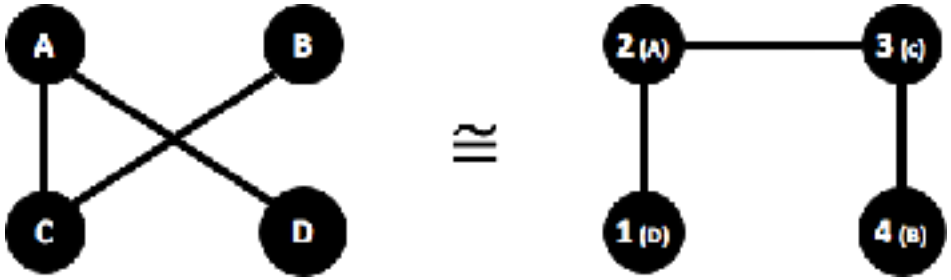




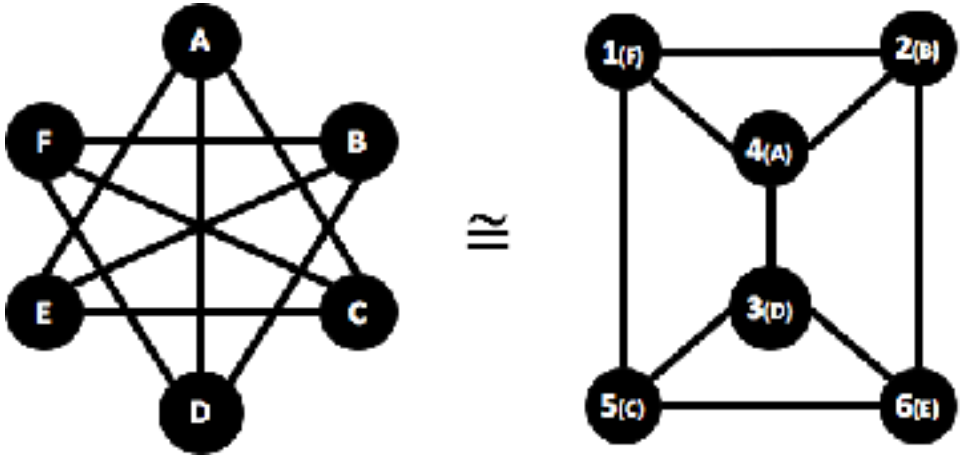


neo4j

Understanding graph complexity



Simple isomorphisms can be visualized.

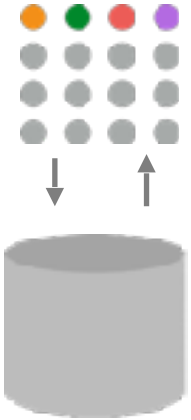


Anything more complicated requires graph algorithms to find.

Native graph platforms are designed for connected data

TRADITIONAL PLATFORMS

Store and retrieve data

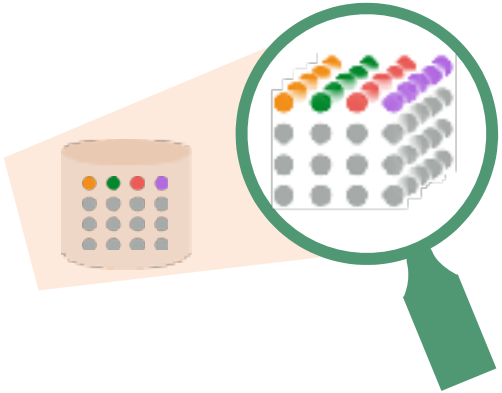


Real time storage & retrieval

Max # of hops ~3

BIG DATA TECHNOLOGY

Aggregate and filter data



Long running queries aggregation & filtering

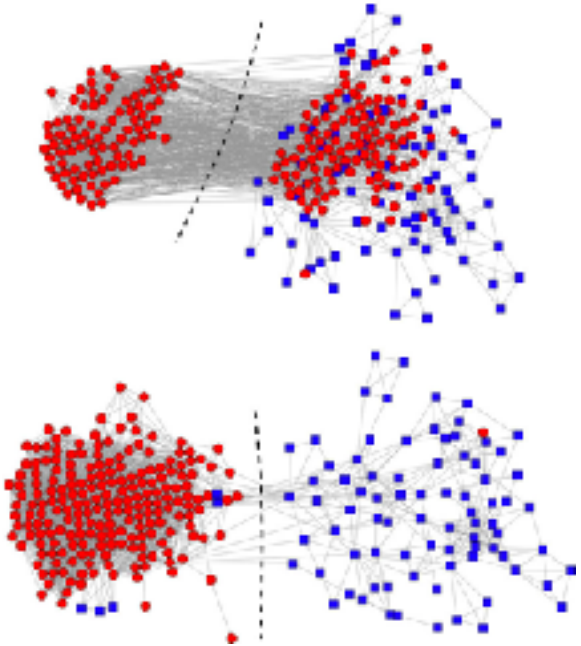
neo4j

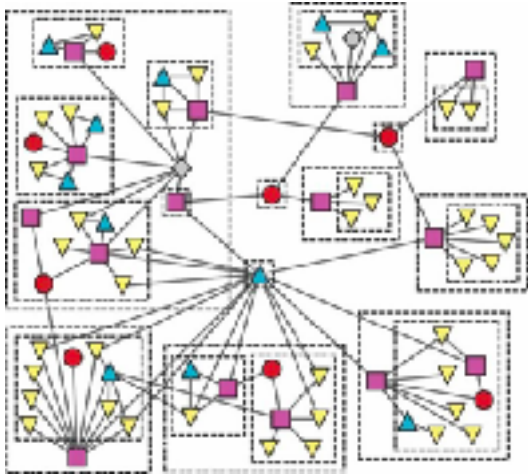
Connections in data

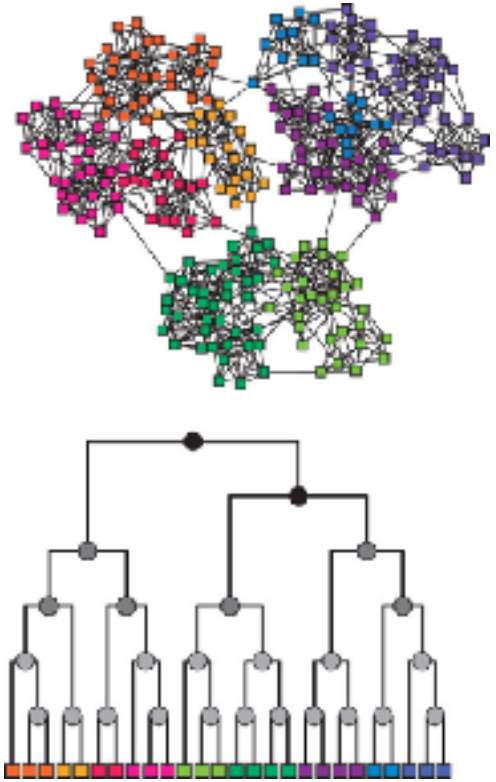
Millions

Real-Time Connected Insights

Graph algorithms:
Extract structure and infer behavior

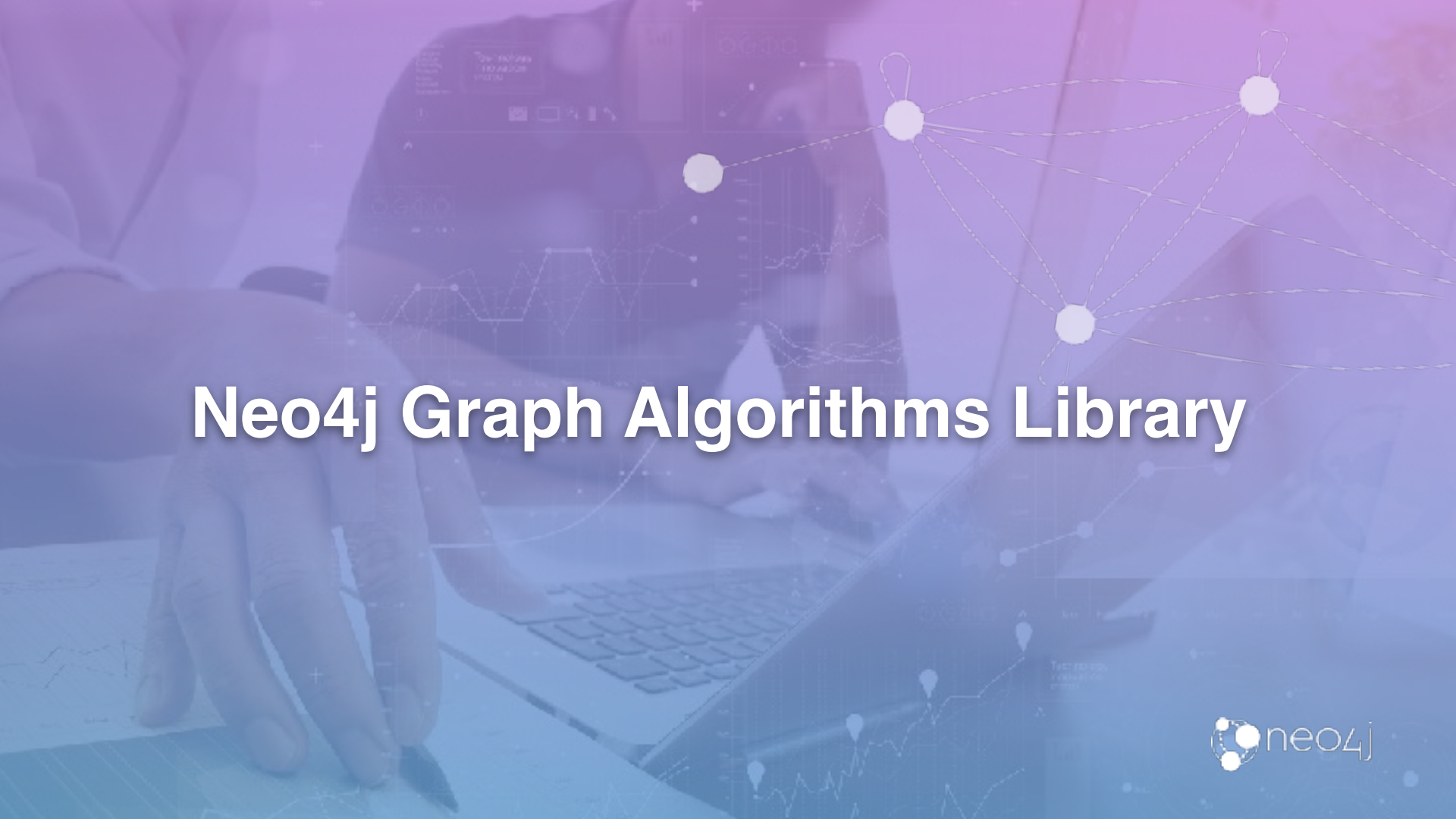






Source: “Communities, modules and large-scale structure in networks“ - Mark Newman
Source: “Hierarchical structure and the prediction of missing links in networks”; ”Structure and inference in annotated networks” - A. Clauset, C. Moore, and M.E.J. Newman.

neo4j


Neo4j Graph Algorithms Library
neo4j

+35 Graph & ML algorithms in Neo4j

Pathfinding & Search

Finds optimal paths or evaluates route availability and quality.

Centrality / Importance

Determines the importance of distinct nodes in the network.

Community Detection

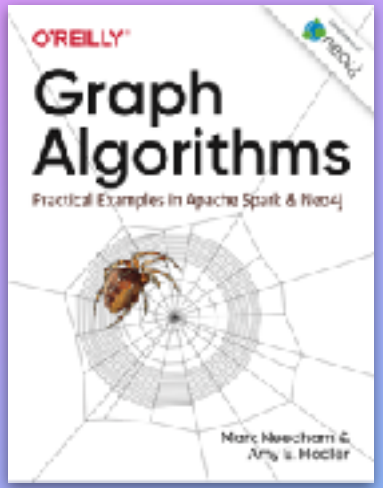
Detects group clustering or partition options.

Estimates the likelihood of nodes forming a future relationship.

Link Prediction

Evaluates how alike nodes are.

Similarity



neo4j.com/graph-algorithms-book/

Graph and ML algorithms in Neo4j

Pathfinding & Search

- Parallel Breadth First Search & Depth First Search
- Shortest Path
- Single-Source Shortest Path
- All Pairs Shortest Path
- Minimum Spanning Tree
- A* Shortest Path
- Yen's K Shortest Path
- K-Spanning Tree (MST)
- Random Walk

Centrality / Importance

- Degree Centrality
- Closeness Centrality
- CC Variations: Harmonic, Dangalchev, Wasserman & Faust
- Betweenness Centrality
- Approximate Betweenness Centrality
- PageRank
- Personalized PageRank
- ArticleRank
- Eigenvector Centrality

Community Detection

- Triangle Count
- Clustering Coefficients
- Connected Components (Union Find)
- Strongly Connected Components
- Label Propagation
- Louvain Modularity – 1 Step & Multi-Step
- Balanced Triad (identification)

Similarity

- Euclidean Distance
- Cosine Similarity
- Jaccard Similarity
- Overlap Similarity
- Pearson Similarity
- Approximate Nearest Neighbors

Link Prediction

- Adamic Adar
- Common Neighbors
- Preferential Attachment
- Resource Allocations
- Same Community
- Total Neighbors

Updated October 2019

neo4j.com/docs/graph-algorithms/current

How to use graph algorithms

1. **Pass in specification (Node Label, Relationship, or Cypher Queries) and configuration.**
2. **Return results:**
   - **Stream procedure returns a lot of results on large graphs - usually one record per node in your graph**

```
CALL algo.<name>.stream('Label','TYPE',
{conf})
YIELD nodeId, result
```

   - **Non-stream procedure can write results to graph and returns statistics.**

```
CALL algo.<name>('Label','TYPE',{conf})
```

Pathfinding & Search

Community Detection

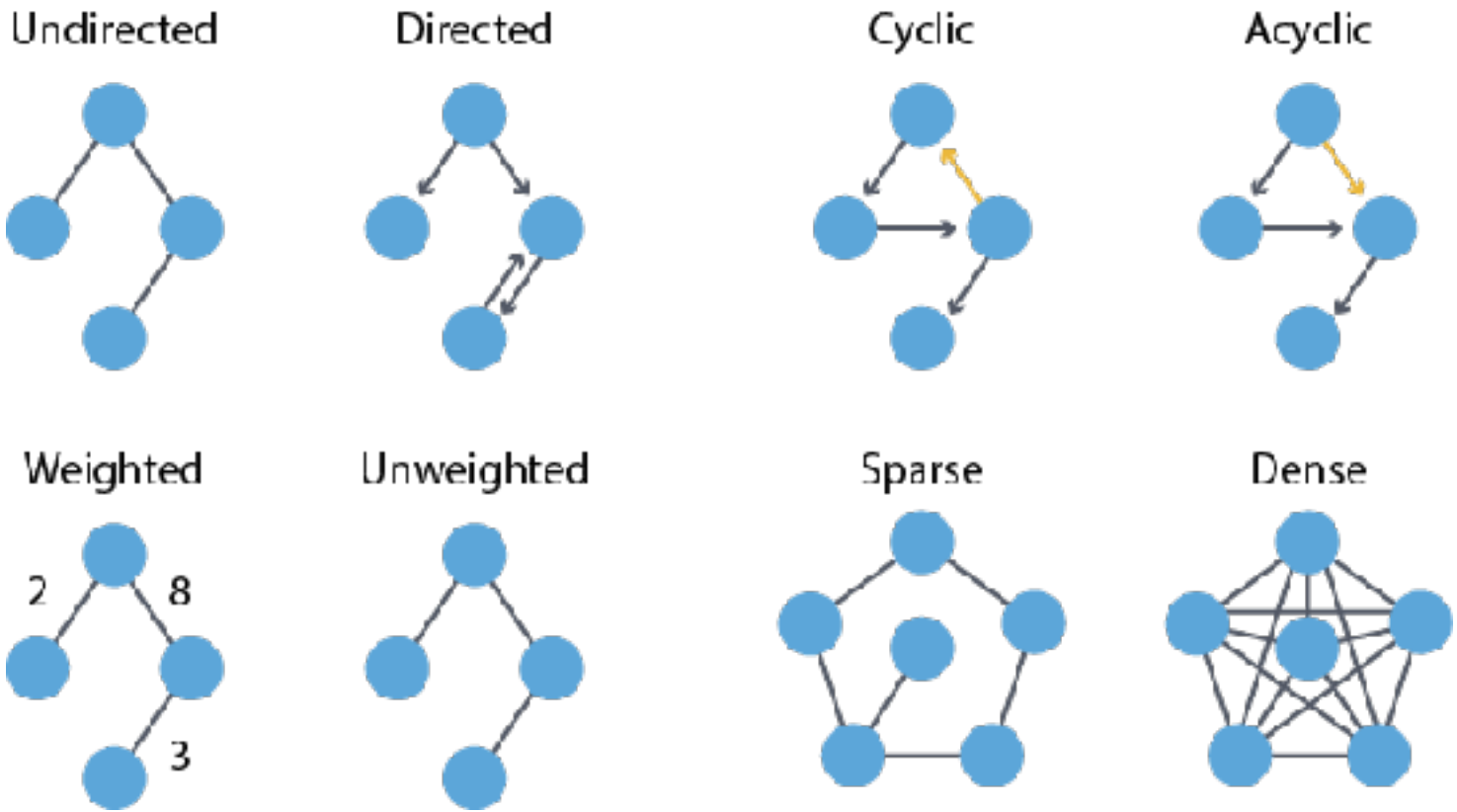
Centrality / Importance

Similarity

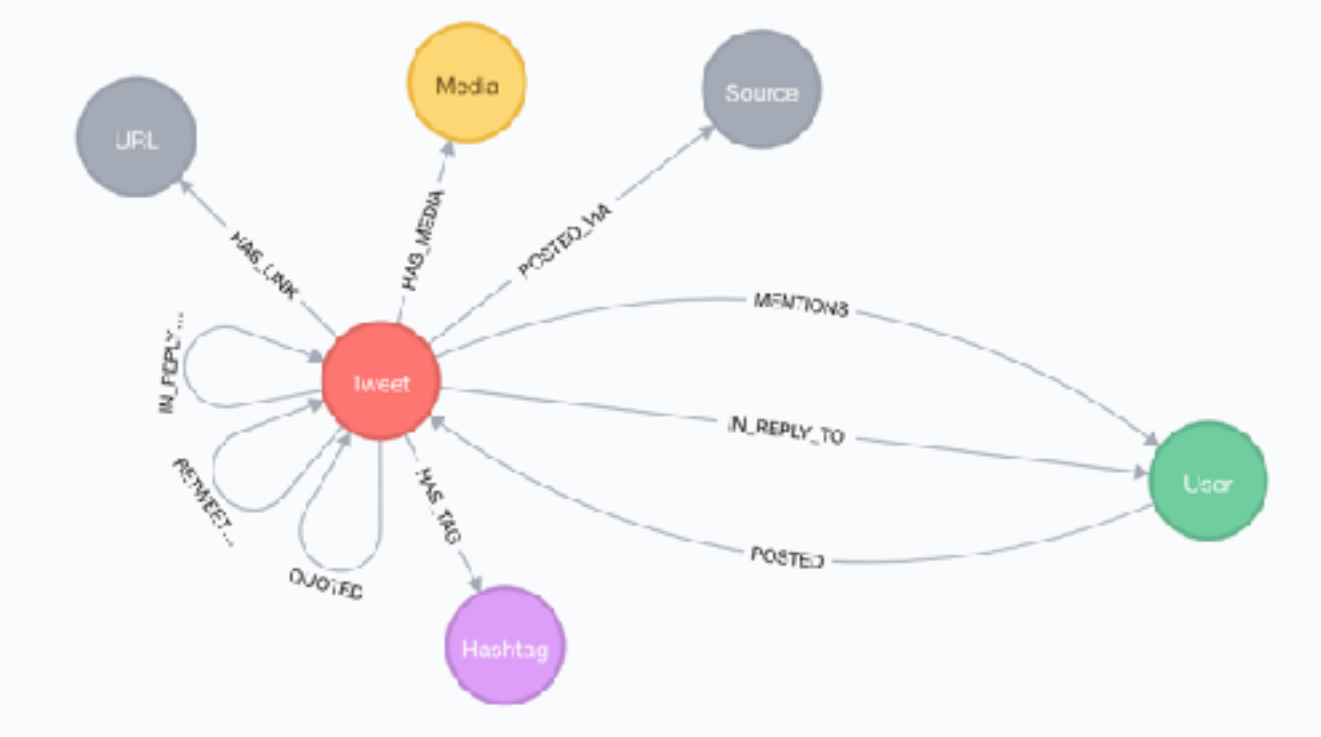
Link Prediction

neo4j

Types of graphs considerations

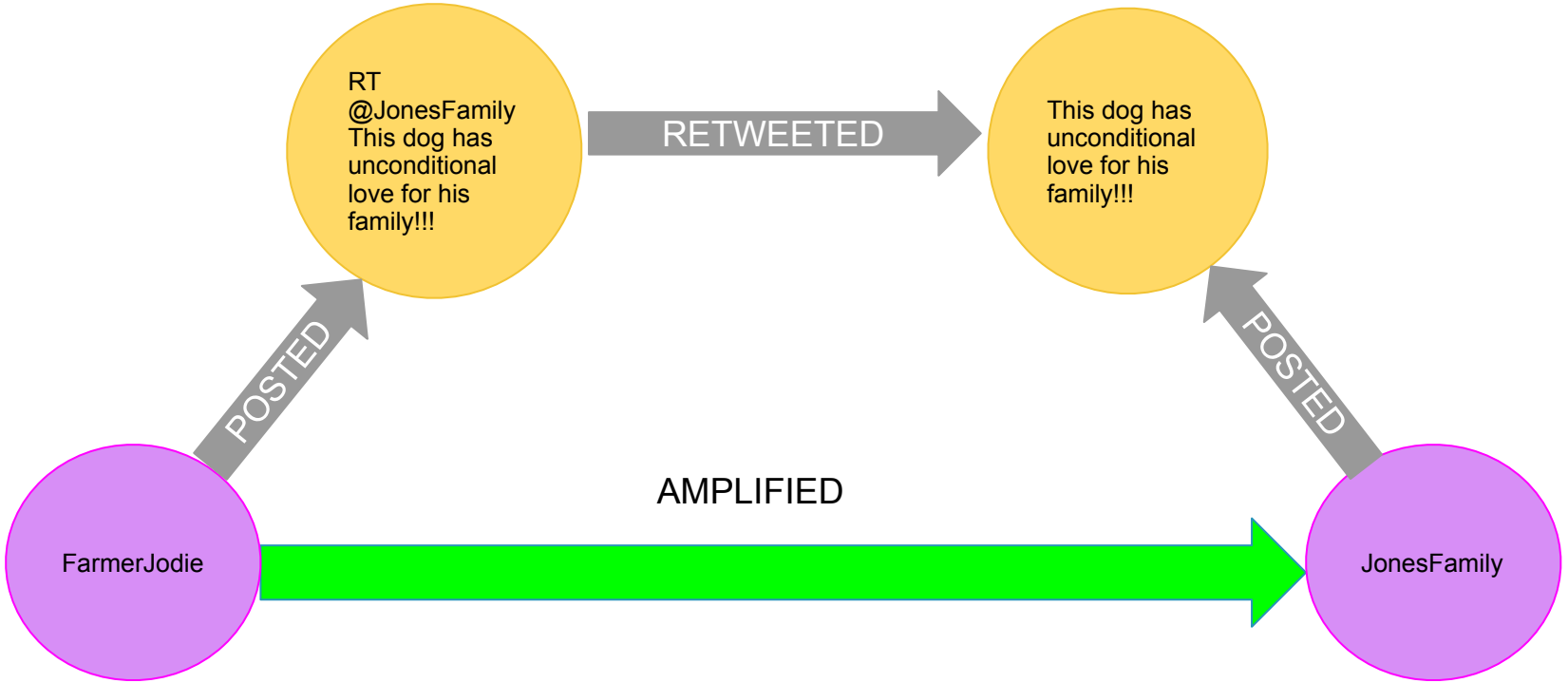


Cypher projection example: Twitter Trolls - 1

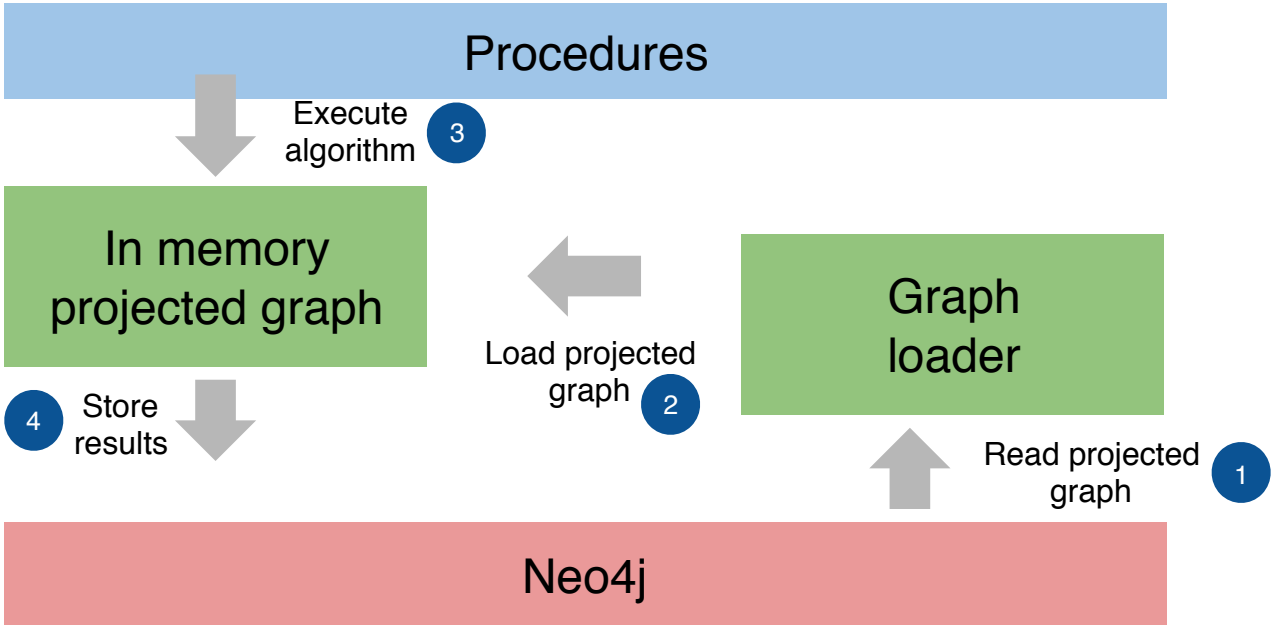


Inferred relationships

```
MATCH (r1:Troll)-[:POSTED]->(t1:Tweet)<-[:RETWEETED]-(t2:Tweet)<-[:POSTED]-(r2:Troll)
```



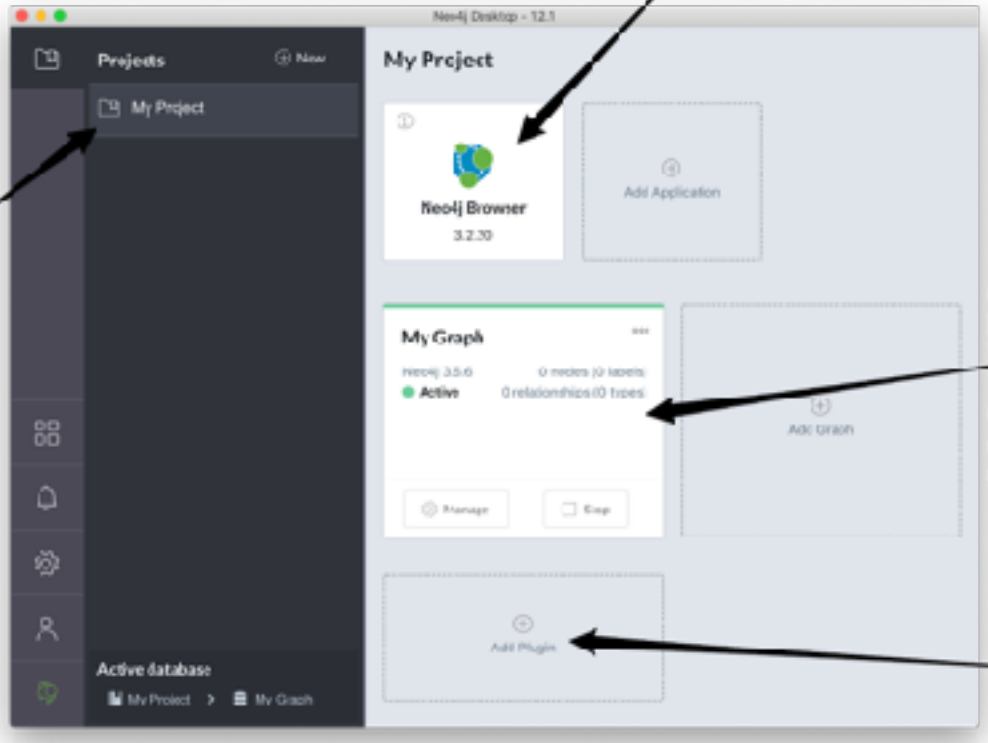
How do graph algorithms work?



Everything is concurrent

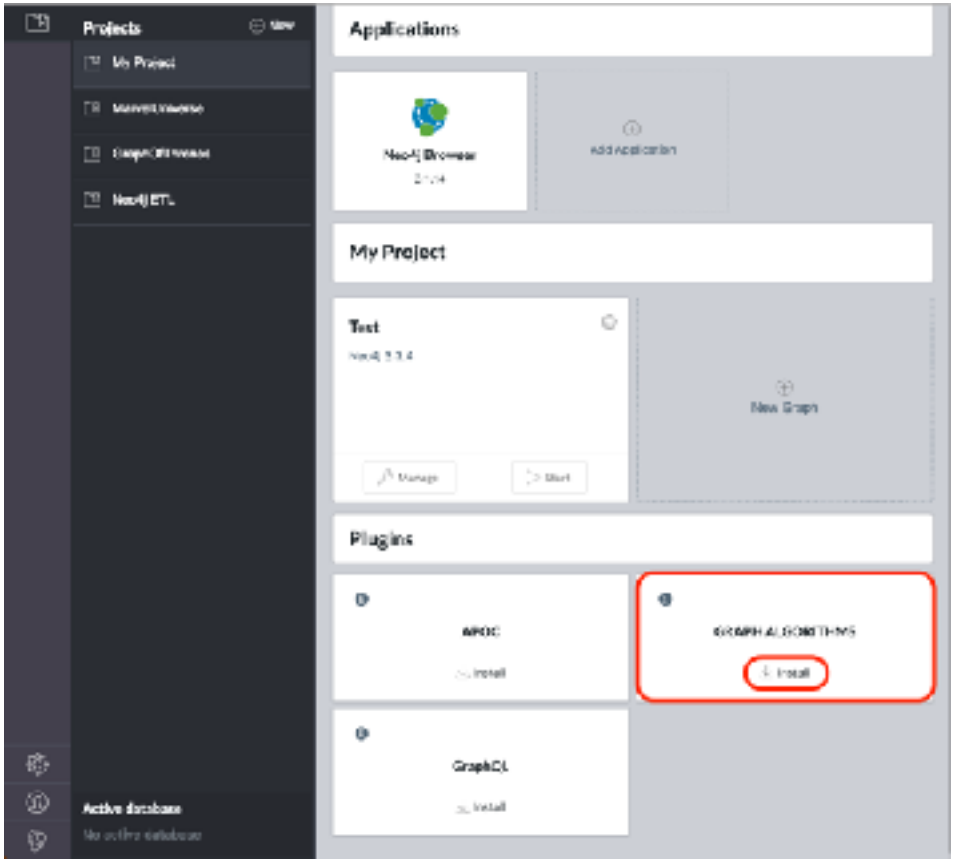
BREAK
and Installation

Neo4j Desktop - “Mission Control”



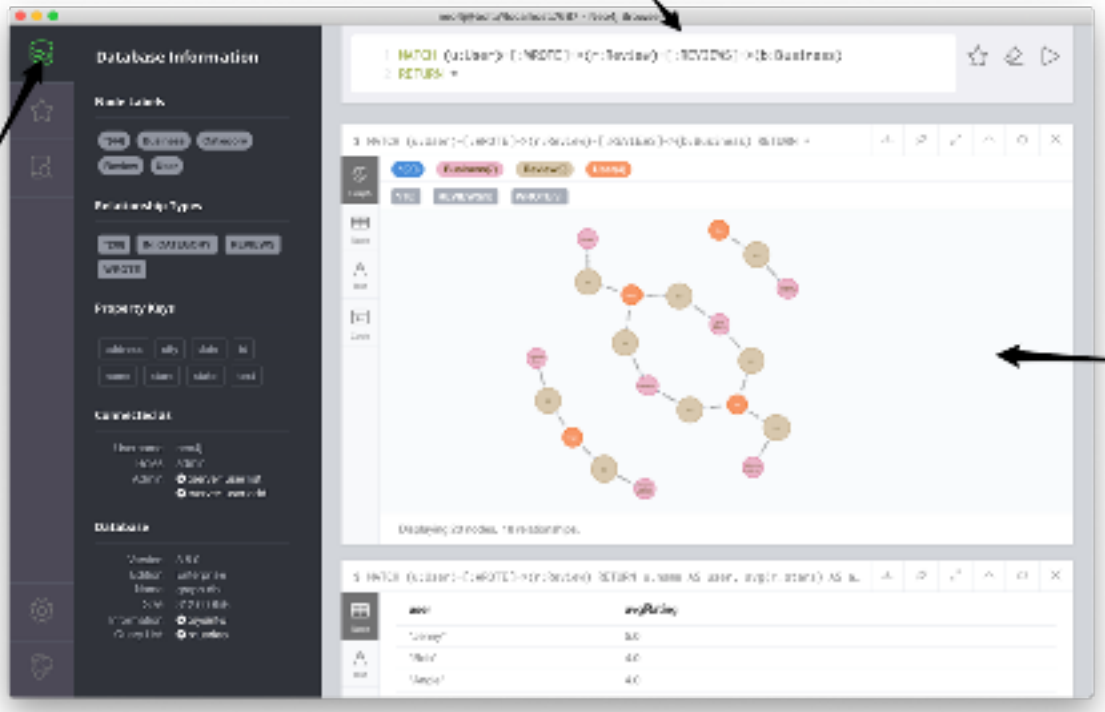
neo4j.com/download

One-click install in Neo4j Desktop



Neo4j Browser - “Query Workbench”

Use the Cypher query editor to execute Cypher statements.



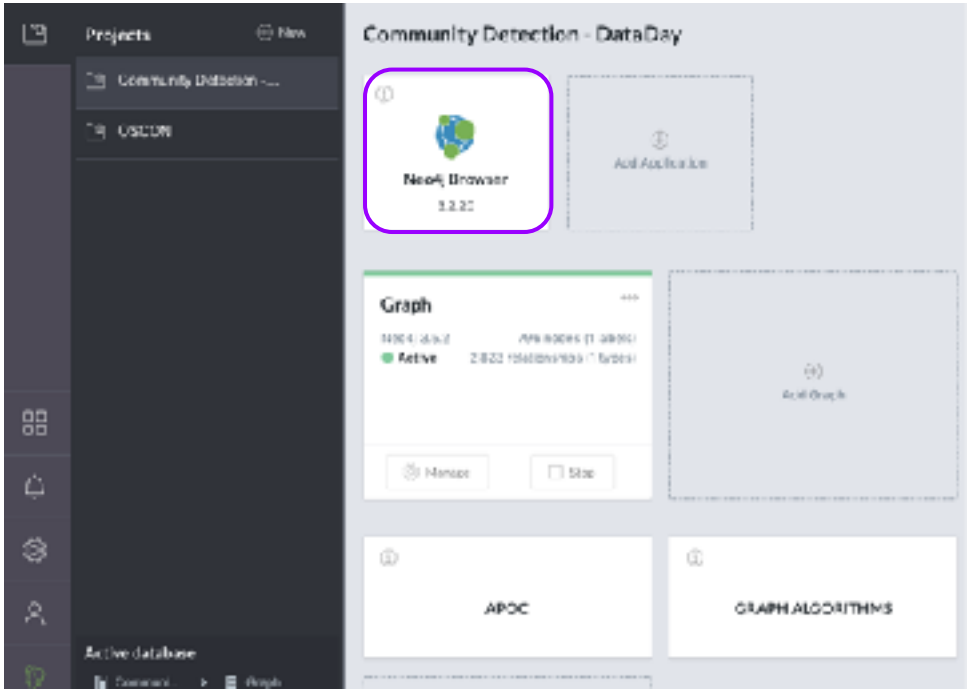
The database information drawer shows the node labels, relationship types, and property keys stored in the currently active database as well as other information about the database.

A result frame is created for each executed Cypher statement.

Results can be shown as a graph visualization or as a table.

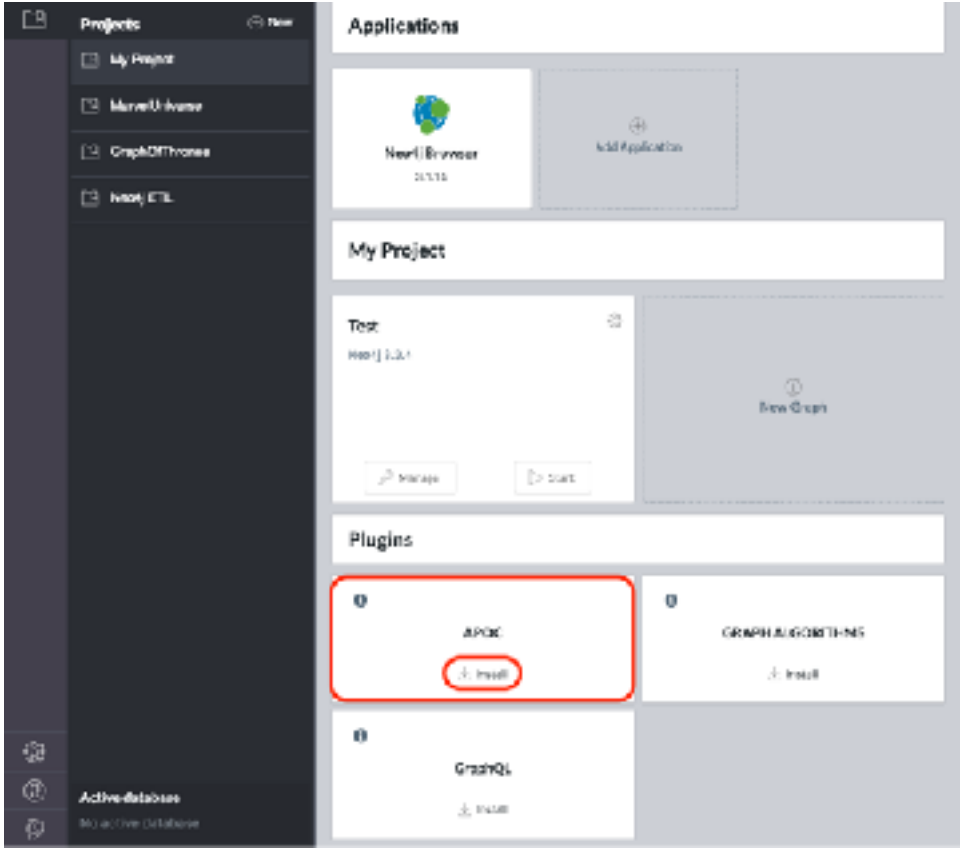
Cypher & Neo4j 101

```
:play gotintro.html
```

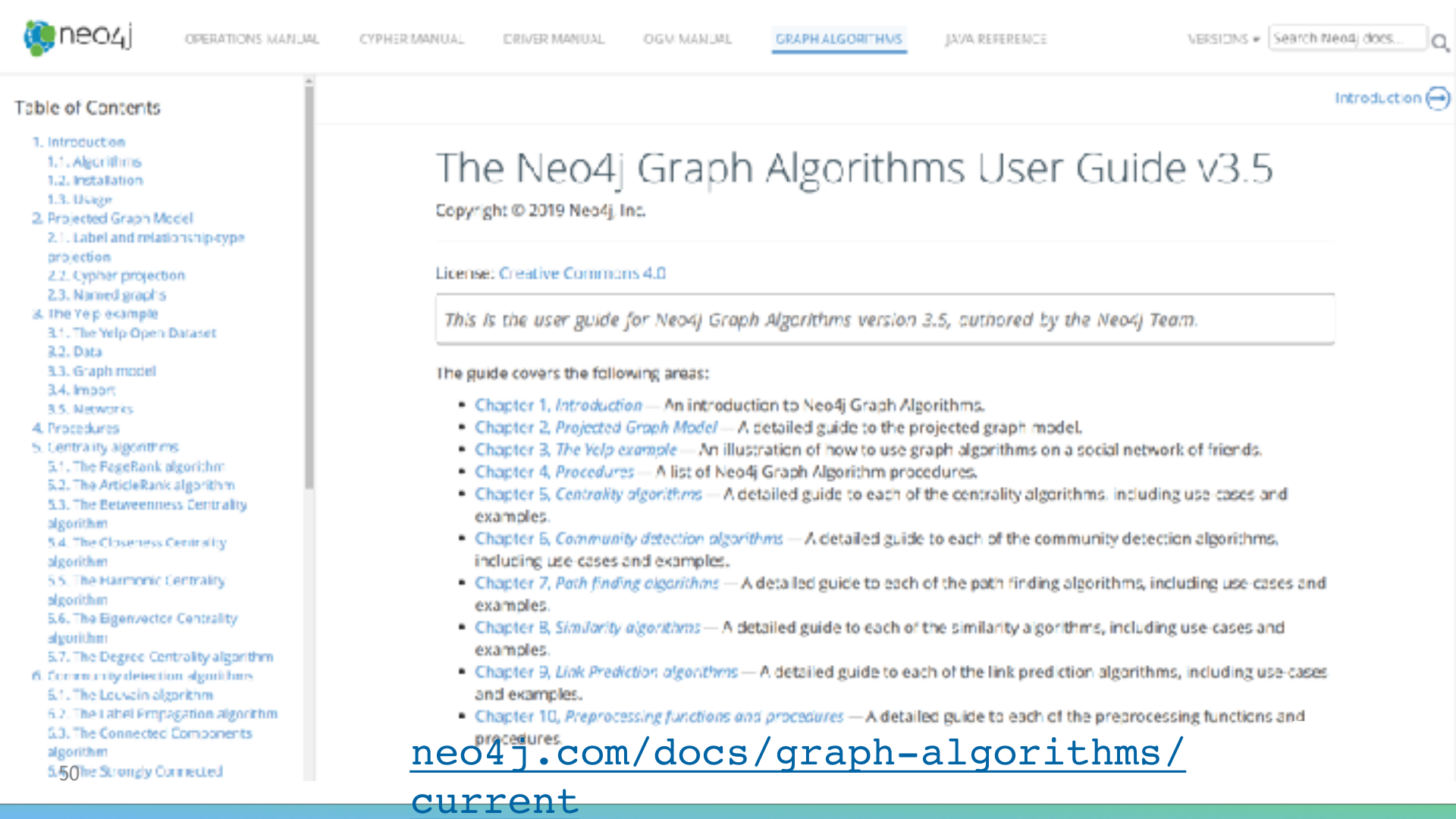


OR web browser url: **localhost:7474**

One-click install in Neo4j Desktop

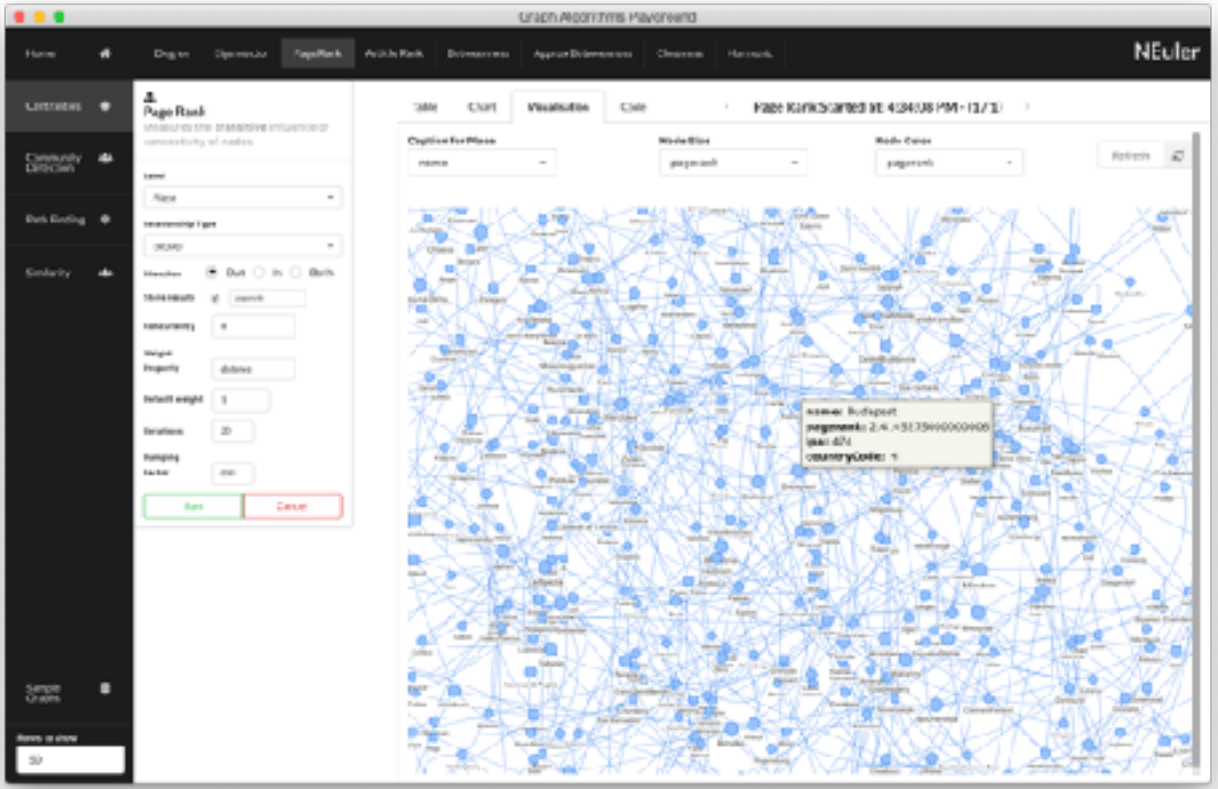


CALL dbms.procedures()

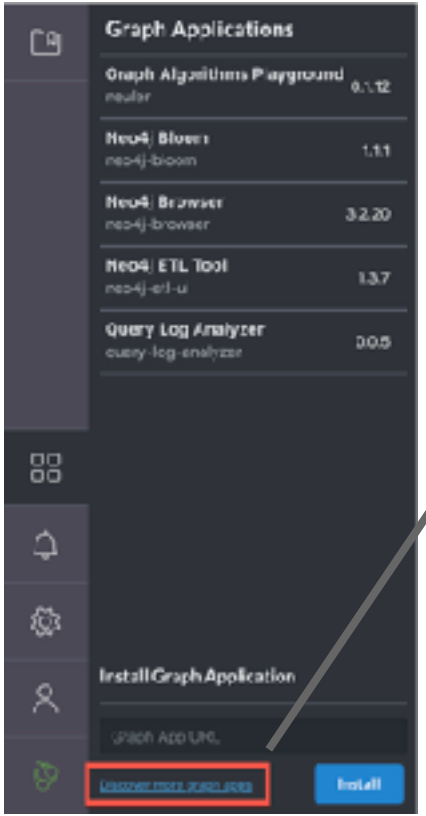


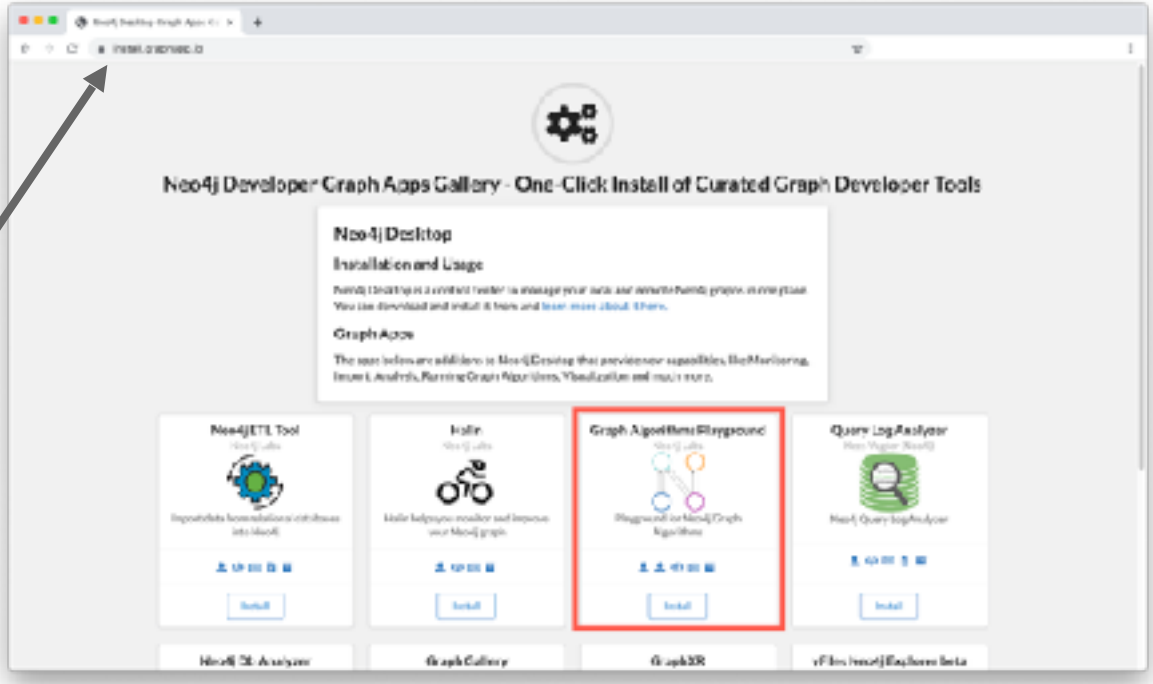
neo4j.com/docs/graph-algorithms/current

Enter the NEuler

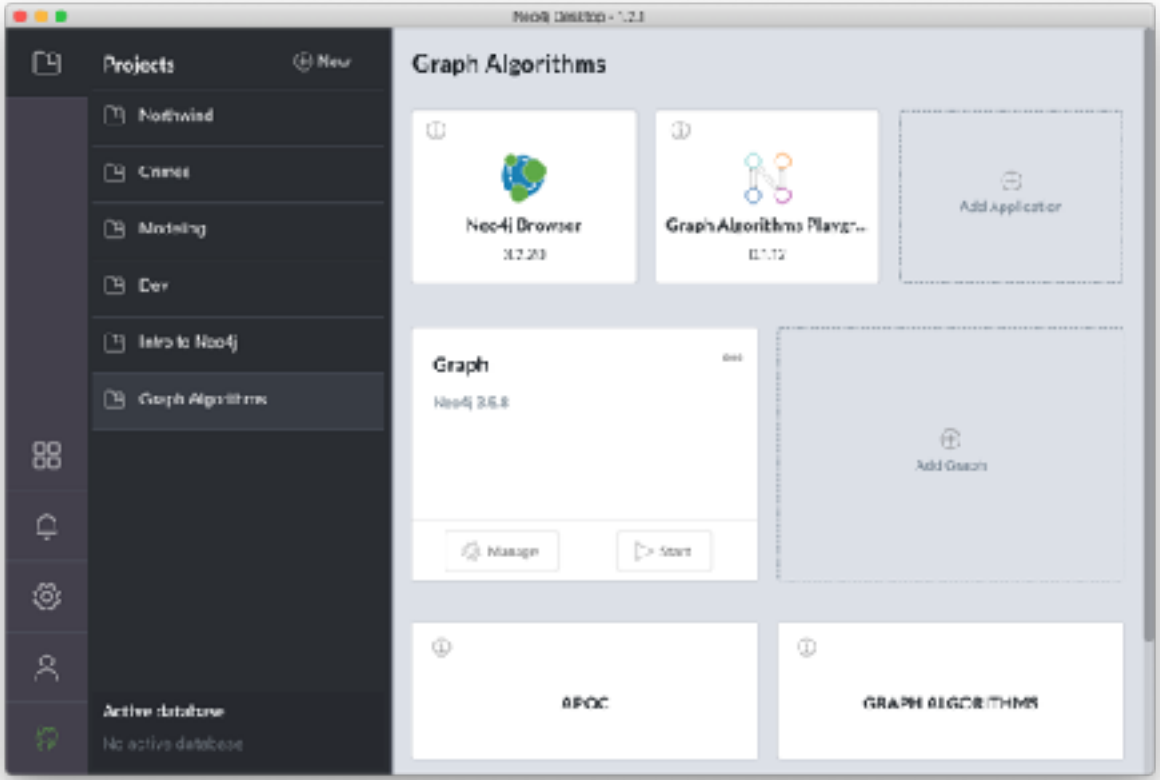


Install NEuler (Graph Algorithms Playground): install.graphapp.io

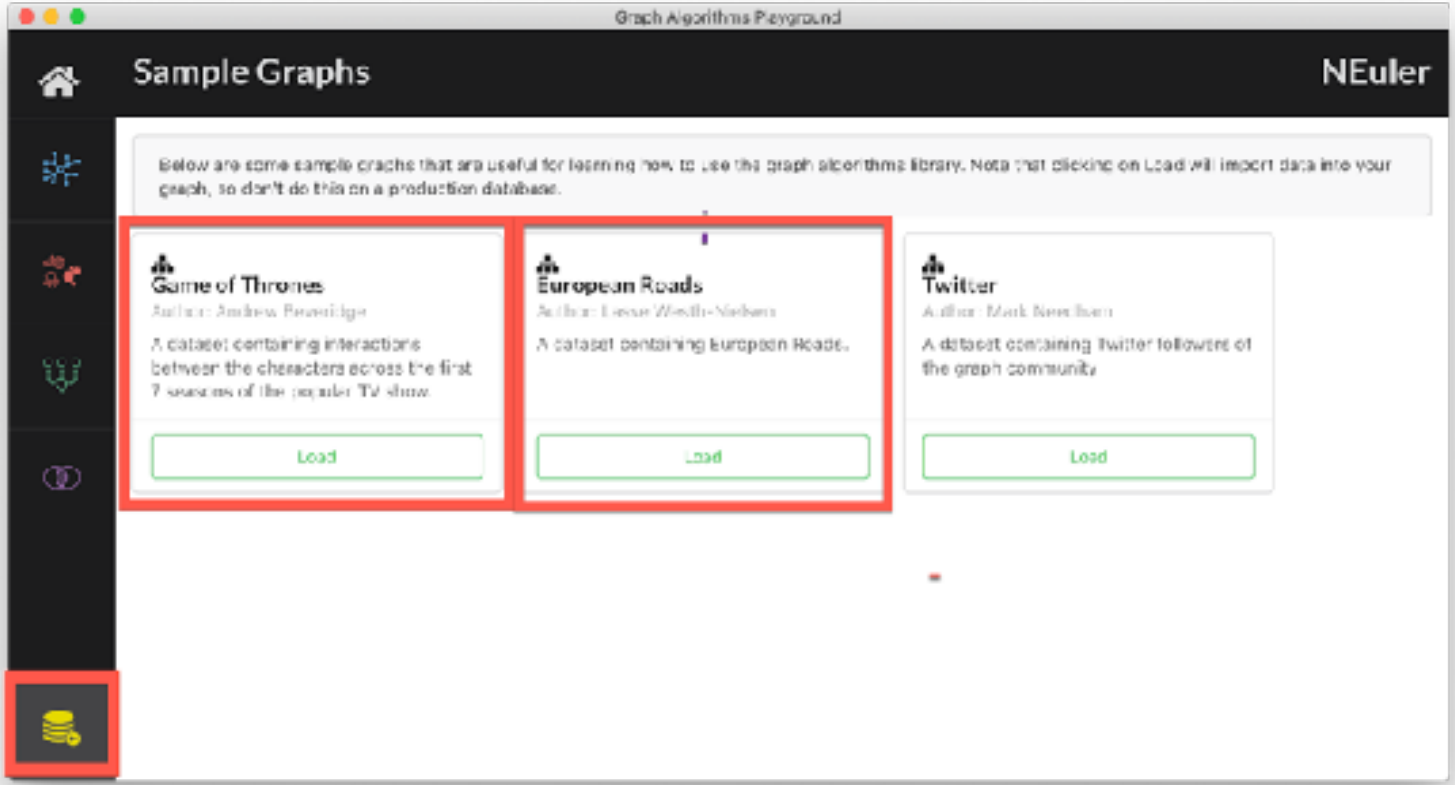




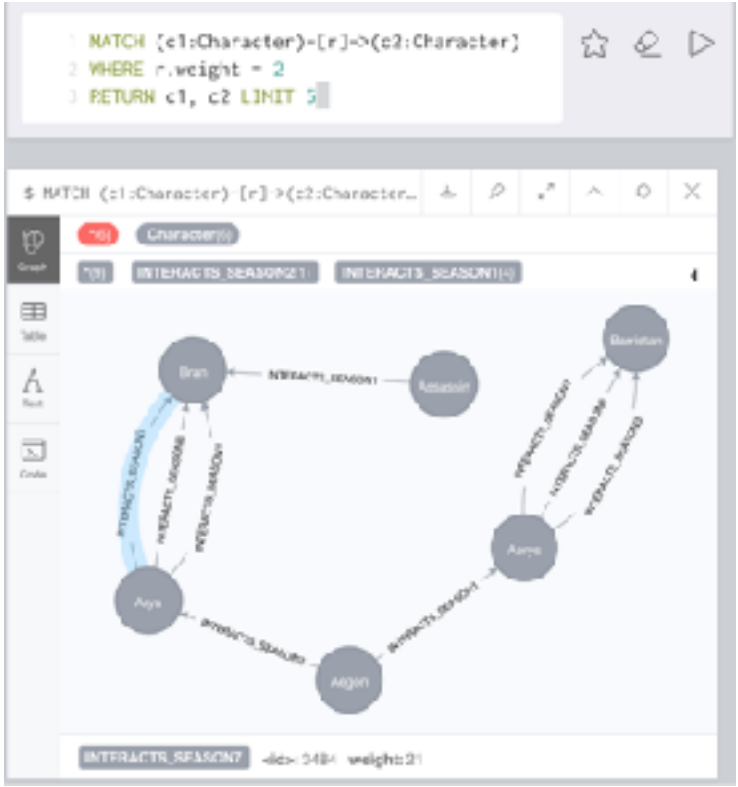
Graph Algorithms Playground (NEuler) installed



Loading datasets from NEuler



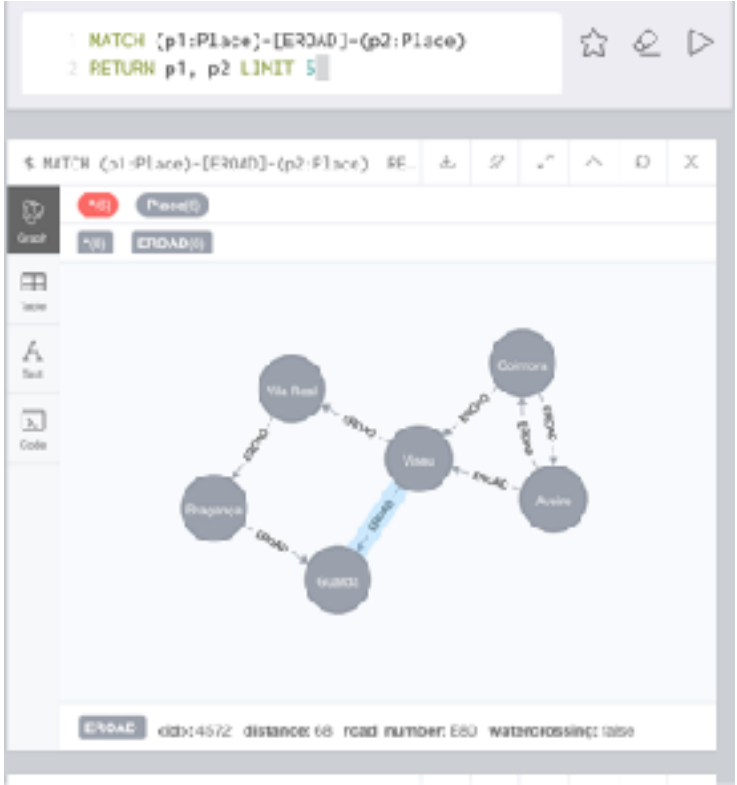
Game of Thrones (GOT) data model



The **INTERACTS_SEASONx** relationship has a property, **weight** to indicate the number of interactions.

The **Character** node has two properties, **name** and **id** where **id** is the capitalized value of **name**.

European Roads data model



The **EROAD** relationship has three properties, **distance**, **road_number**, and **watercrossing**.

The **Place** node has multiple properties, **name** and **countryCode**.

Exercise 1: Set up your Development Environment

Estimated time to complete this exercise:

10 minutes

Before coming to this training, you should have:

1. Installed Neo4j Desktop.
2. Downloaded the Yelp dataset. (Note: This database is 1G and will take a while to download) https://s3.amazonaws.com/neo4j-sandbox-usecase-datastores/v3_5/yelp.db.zip and unzip.
3. Created a project and a local graph, providing a password you will remember. (Do not start the database!)
4. Installed APOC, Graph Algorithms, and Graph Algorithms Playground in the project.
5. Copied the Yelp database to your local graph:
    a. Click the Manage button for the local graph you just created.
    b. Click Open Folder.
    c. Navigate to data/databases.
    d. Copy the yelp.db folder that you unzipped to databases.
    e. Rename the yelp.db folder to graph.db.
6. Started the database.

:play intro-graph-algos-exercises (Set up your Development Environment)


Community Detection algorithms
neo4j

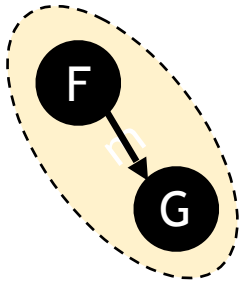
Community Detection Algorithms

Evaluates how a group is clustered or partitioned

Different approaches to define a community

Weakly Connected Components

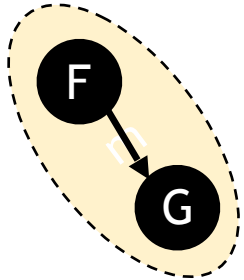
Weakly Connected Components (UNION FIND)



- All nodes can reach each other when disregarding direction.
- Find disconnected subgraphs or nodes in common and preprocess data.
- Optionally write unionFind value to each node for the analysis.

Weakly Connected Components (UNION FIND)

Uses
Early graph analysis to understand graph structure Understanding how graph structure has changed with updates (scales well) Seeing how connected a graph is Looking for new nodes in common between groups



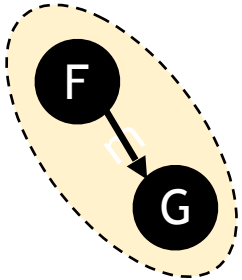
Weakly Connected Components (UNION FIND)

Examples

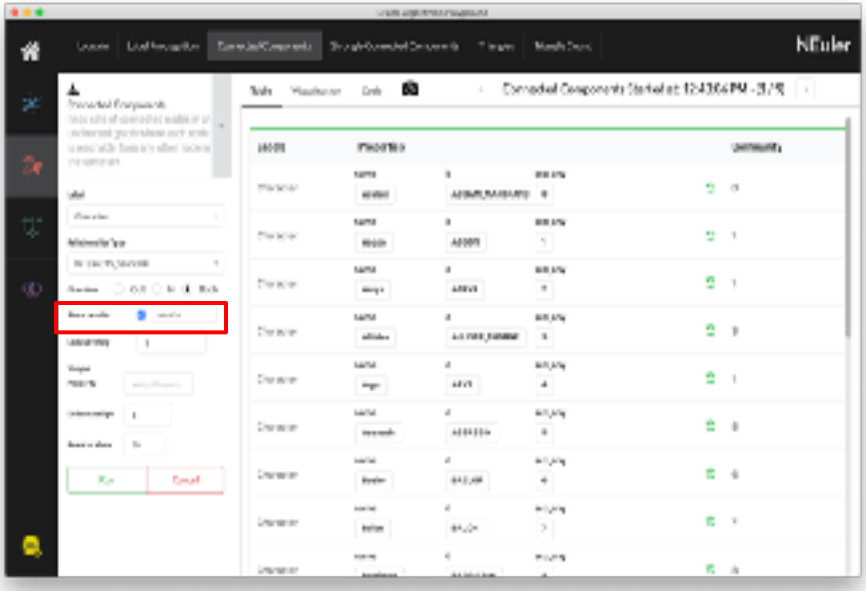
MDM

- tracking clusters of database records for deduplication process

Citation networks

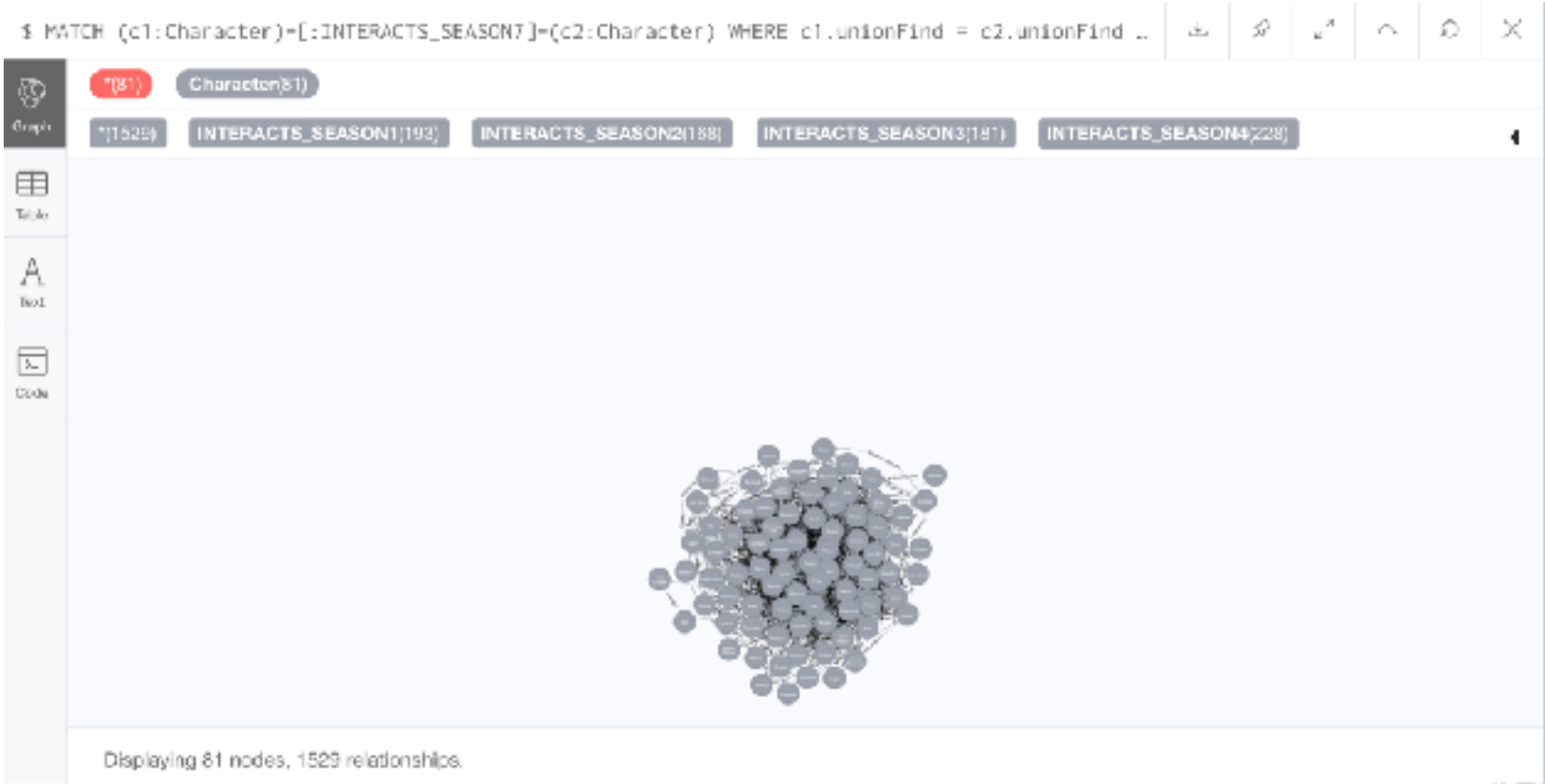


Example: Connected Components - GOT

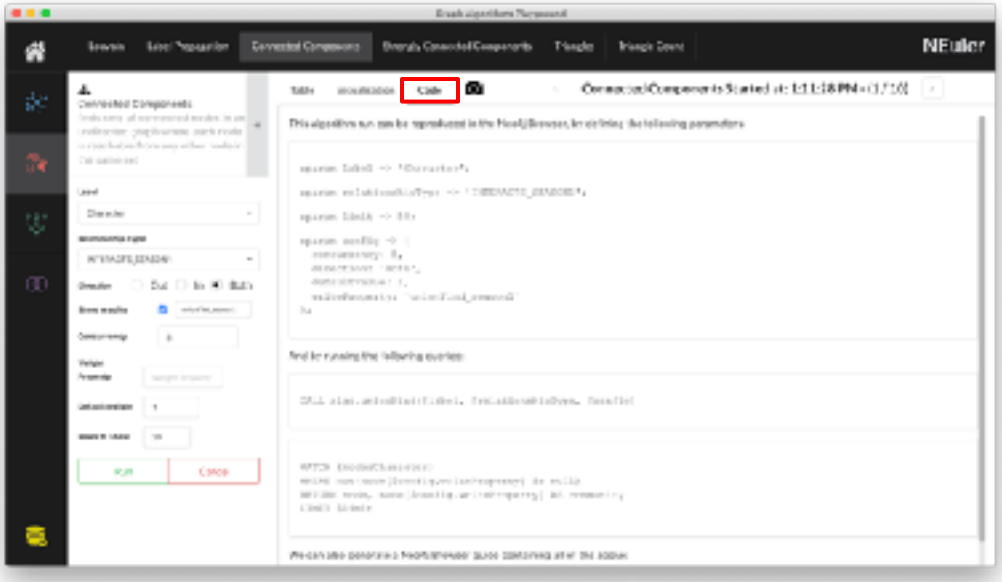


Any node that has a unique value for unionFind will not be connected to any other node in the graph for the season 1 relationship.

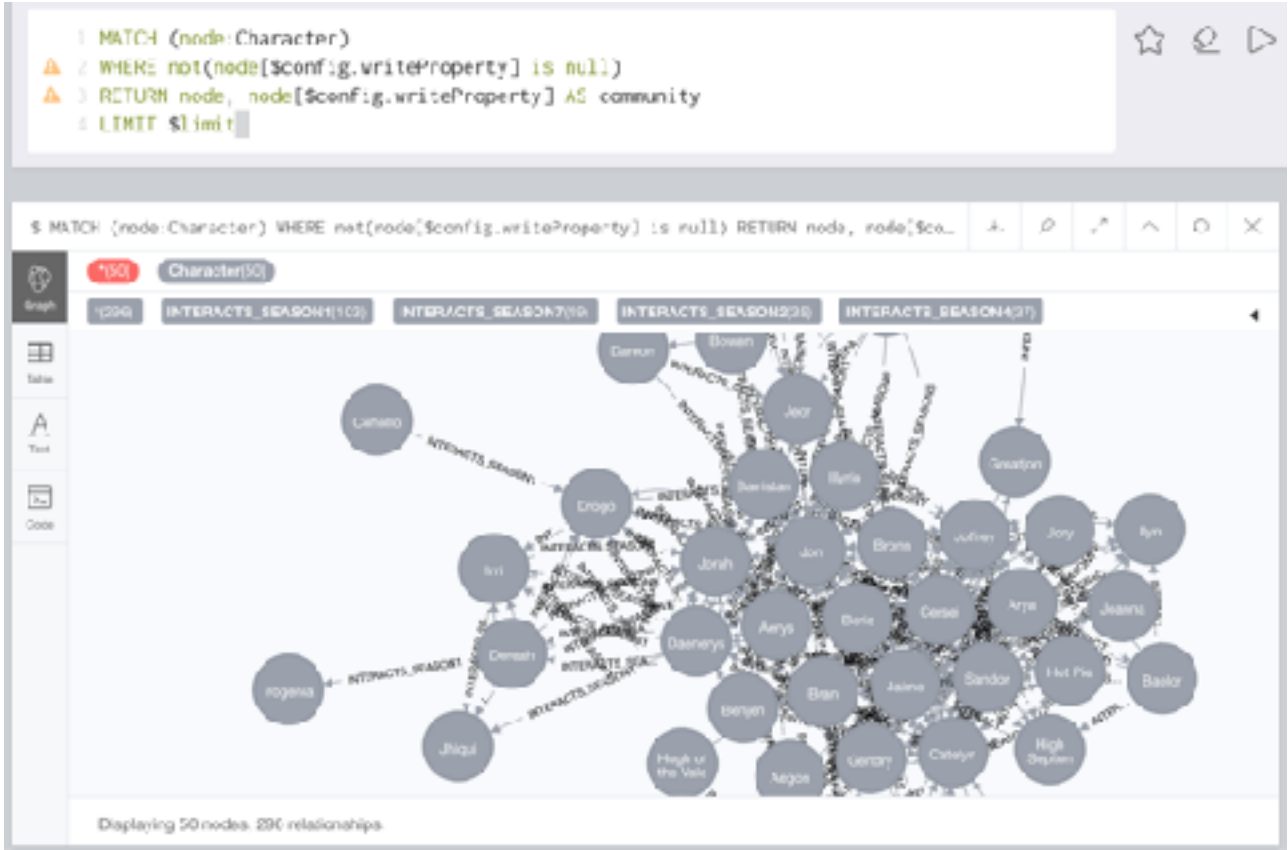
Connected Components - GOT data



Connected Components in Cypher



Weakly Connected Components - GOT data, season 1



Exercise 2:
Weakly Connected Components

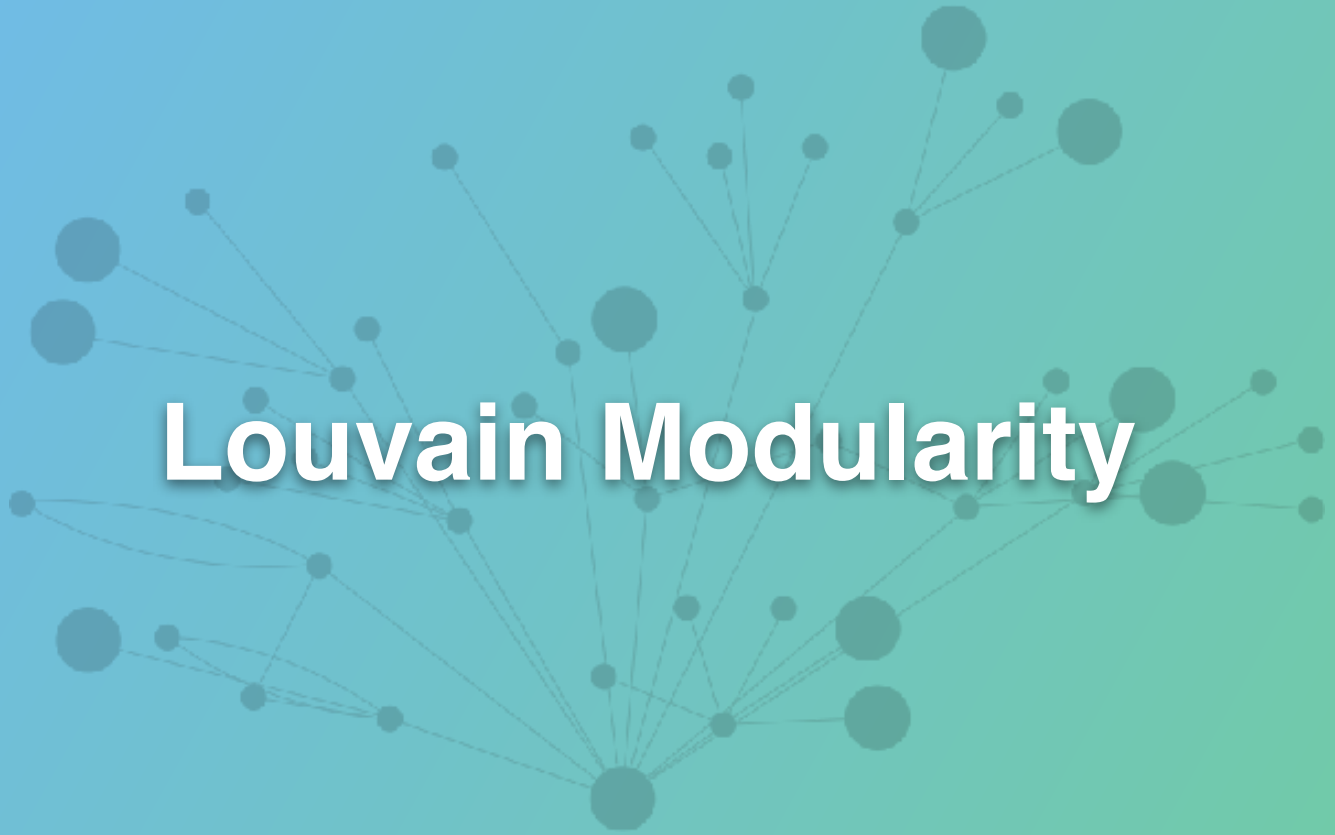
Estimated time to complete this exercise:

20 minutes

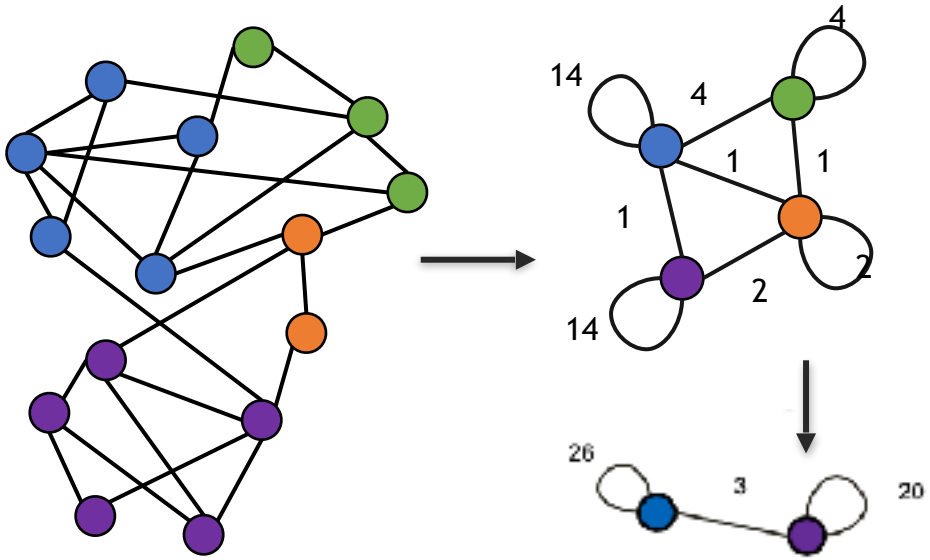
1. Start or restart NEuler so that all loaded data can be seen.
2. In NEuler:
   a. Find all Connected Person nodes writing the unionFind_helps property.
   b. Find all Connected Characters for Season 3 writing the unionFind_season3 property.
      i. Advanced: How many nodes are connected for Season 3?
   c. Do the same for any relationship, writing the unionFind_any value.
3. In Neo4j Browser:

:play intro-graph-algos-exercises (Weakly Connected Components)

Louvain Modularity



Louvain Modularity



Continually maximizes the modularity by comparing relationship weights and densities to an estimate /average.

Tip / Caution

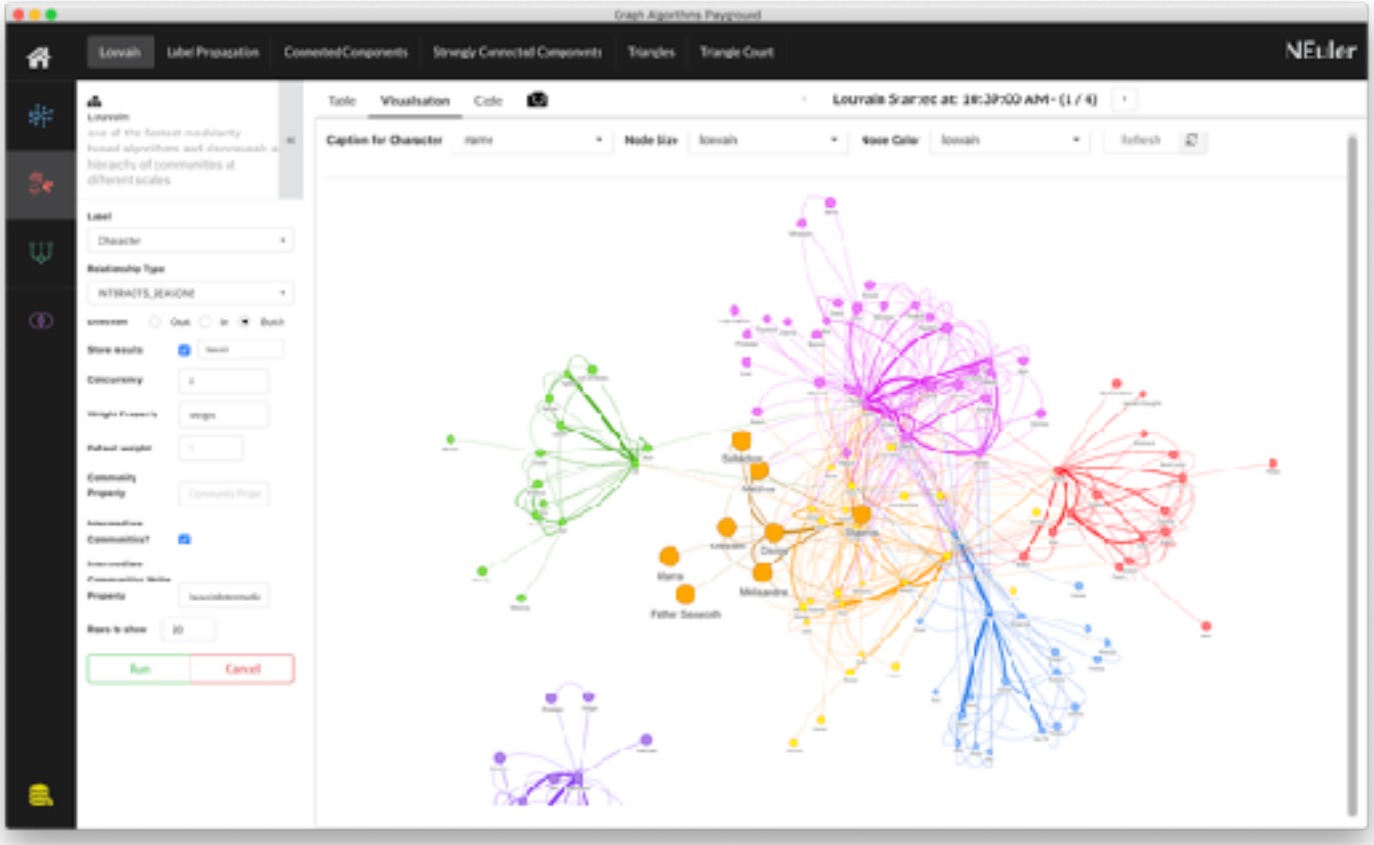
ALL Modularity algorithms:

- Merge smaller communities into larger ones
  - Review intermediates
- Can plateau with similar modularity on several partitions - forming local maxima & stalling progress
  - Treat as a guide and test/validate results

Louvain Modularity - Uses - 2

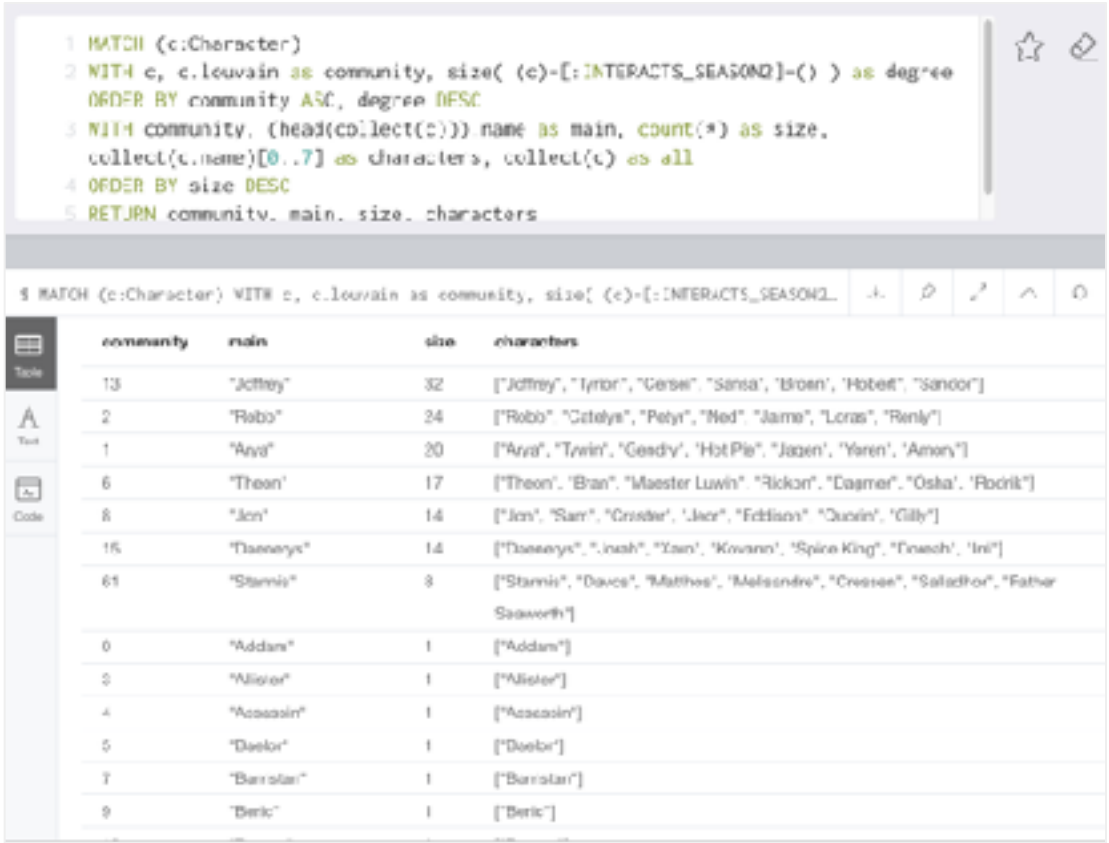
Use When
Community detection in large networks Uncover hierarchical structures in data Evaluate different grouping thresholds

Louvain Modularity: GOT



Result of Louvain Modularity: GOT

```
MATCH (c:Character)
WITH c, c.louvain as community, size( (c)-[:INTERACTS_SEASON2]-() ) as degree
ORDER BY community ASC, degree DESC
WITH community, (head(collect(c))).name as main, count(*) as size,
collect(c.name)[0..7] as characters, collect(c) as all
ORDER BY size DESC
RETURN community, main, size, characters
```

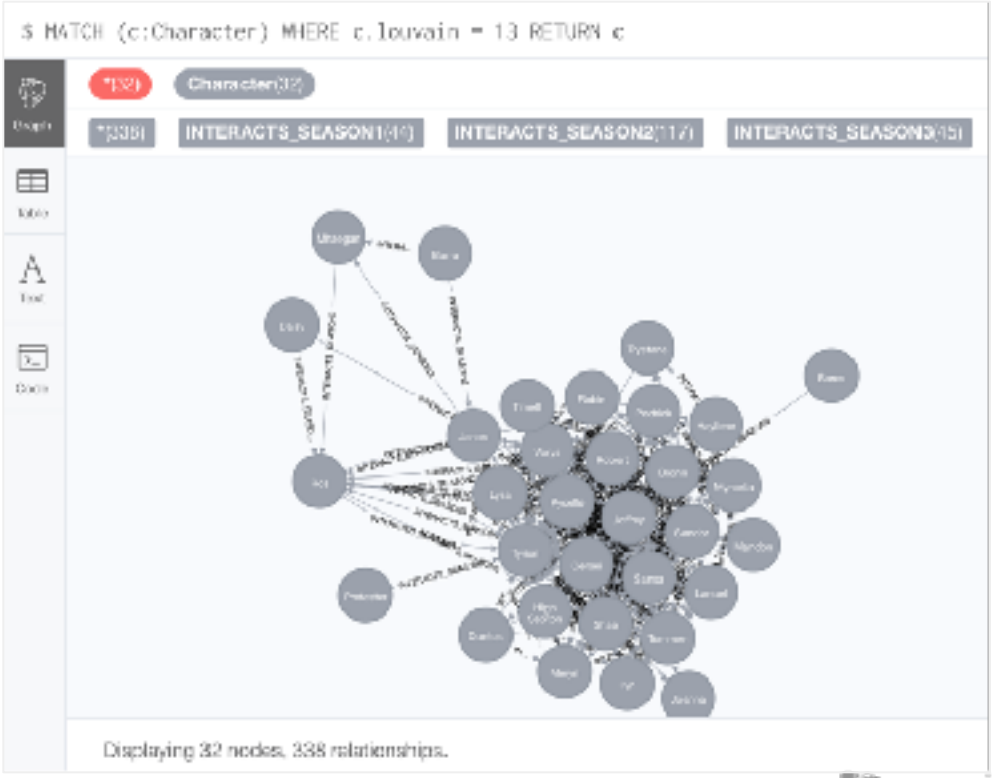


community	main	size	characters
13	"Joffrey"	32	["Joffrey", "Tyrion", "Cersei", "Sansa", "Brown", "Hobett", "Sandor"]
2	"Robb"	24	["Robb", "Catelyn", "Petyr", "Ned", "Jaime", "Loras", "Renly"]
1	"Arya"	20	["Arya", "Tywin", "Gendry", "Hot Pie", "Jagen", "Yoren", "Amory"]
6	"Theon"	17	["Theon", "Bran", "Maester Luwin", "Rickon", "Dagmer", "Osha", "Rodrik"]
8	"Jon"	14	["Jon", "Sam", "Grenn", "Jeor", "Eddison", "Ouorin", "Gilly"]
15	"Daenerys"	14	["Daenerys", "Khash", "Xaro", "Kovarro", "Spice King", "Drogo", "Ini"]
61	"Stannis"	8	["Stannis", "Davos", "Matthos", "Melisandre", "Cressen", "Salladhor", "Father Seaworth"]
0	"Addam"	1	["Addam"]
3	"Alliser"	1	["Alliser"]
4	"Assassin"	1	["Assassin"]
5	"Dacko"	1	["Dacko"]
7	"Barristan"	1	["Barristan"]
9	"Beric"	1	["Beric"]

Looking at intermediate Louvain values: GOT

```
$ MATCH (c:Character) WHERE c.louvain = 13 RETURN c.name, c.louvainIntermediate
```

"Pycelle"	[14, 13]
"Robert"	[14, 13]
"Rus"	[58, 13]
"Sansa"	[14, 13]
"Shae"	[14, 13]
"Tommen"	[14, 13]
"Tyrion"	[14, 13]
"Varys"	[14, 13]
"Barra"	[78, 13]
"Bonos"	[14, 13]
"Daisy"	[58, 13]
"Dontos"	[14, 13]
"Havlene"	[14, 13]
"Janos"	[14, 13]
"Mandon"	[14, 13]
"Meryn"	[14, 13]
"Mhaegar"	[78, 13]



Exercise 4:
Louvain Modularity

Estimated time to complete this exercise:

15 minutes

1. In NEuler
   a. Perform the Louvain Modularity algorithm on different seasons of GOT.

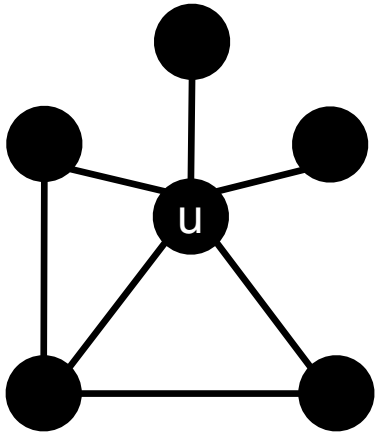
1. In Neo4j Browser:
   a. View the louvain and intermediate louvain values for GOT.
   b. :play intro-graph-algos-exercises (Louvain Modularity)

Triangle count and clustering coefficient

Triangles and Clustering Coefficient

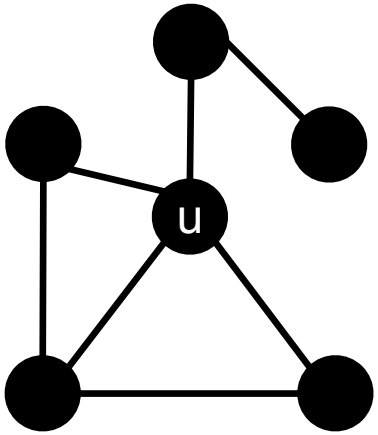
Triangle Count determines the number of triangles passing through a node in the graph.

Clustering Coefficient is the probability that neighbors of a particular node are connected to each other.



Triangles = 2

CC= 0.2



Triangles = 2

CC= 0.33

Measures can be counted/normalized globally

Triangles and Clustering Coefficient

Use When

Basic network analysis
- small world structures

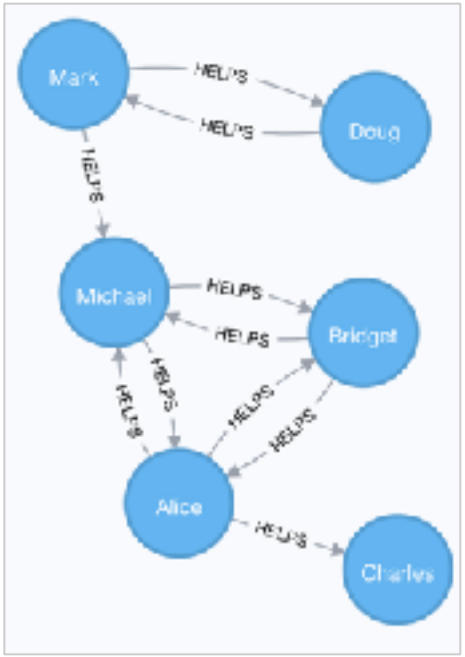
Estimating stability

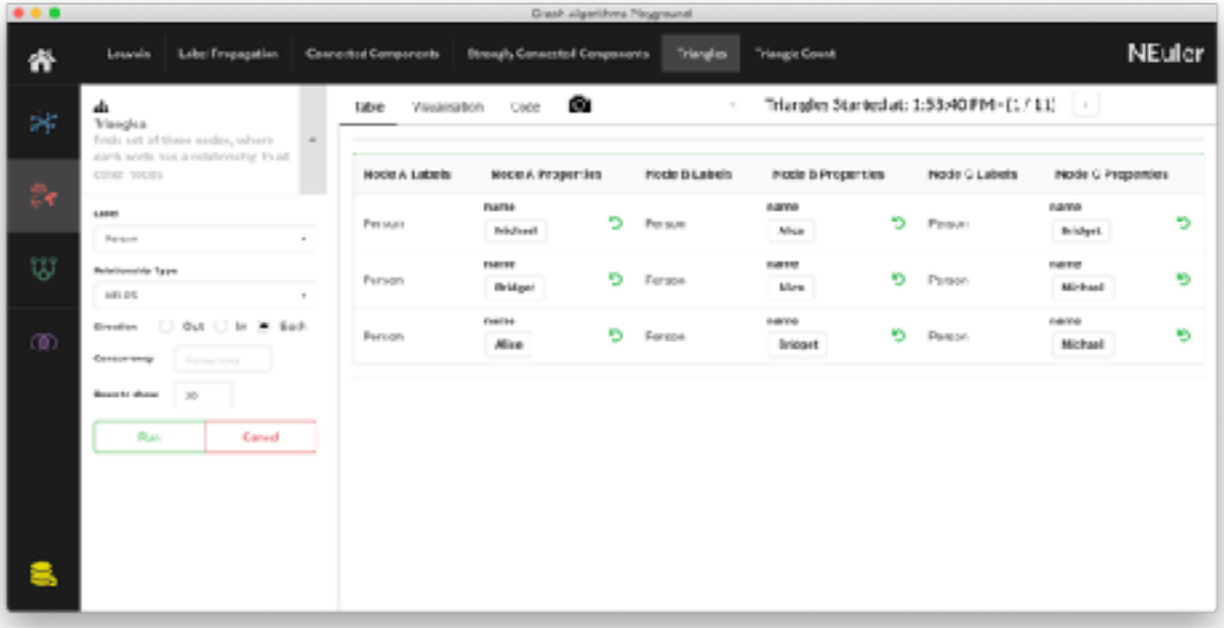
Finding structural holes

Scoring for ML

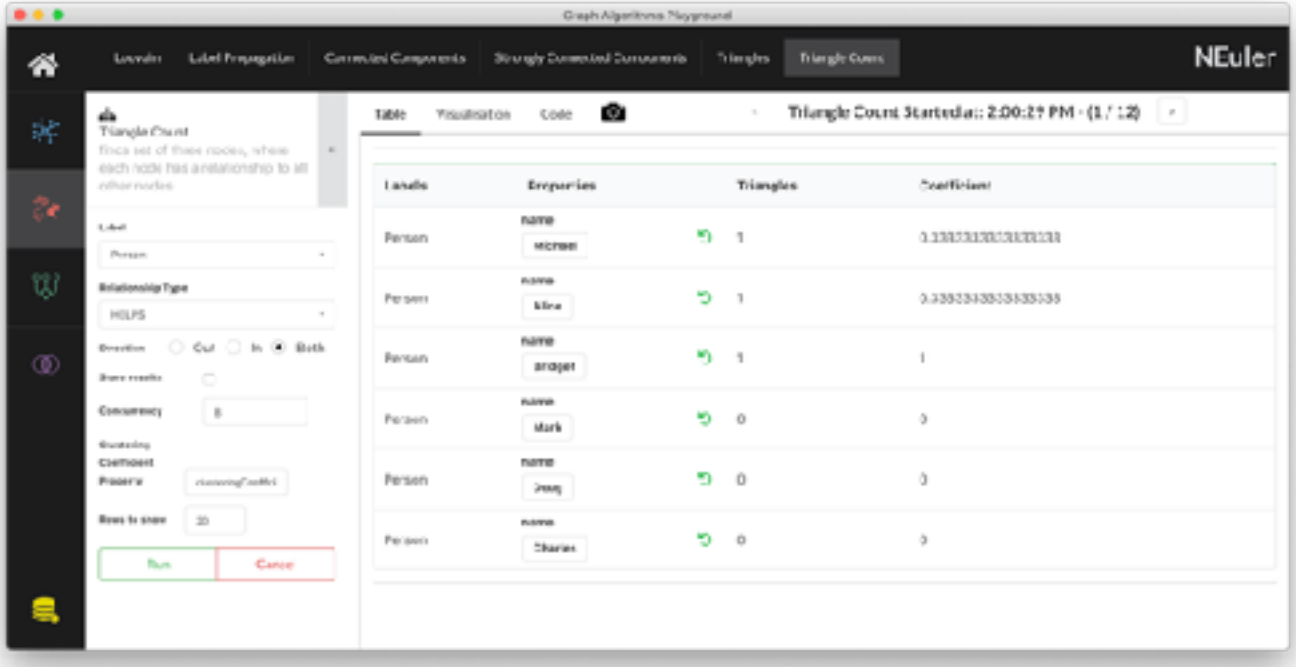
Spam Classification
Semi-streaming web page analysis (local triangle and CC)

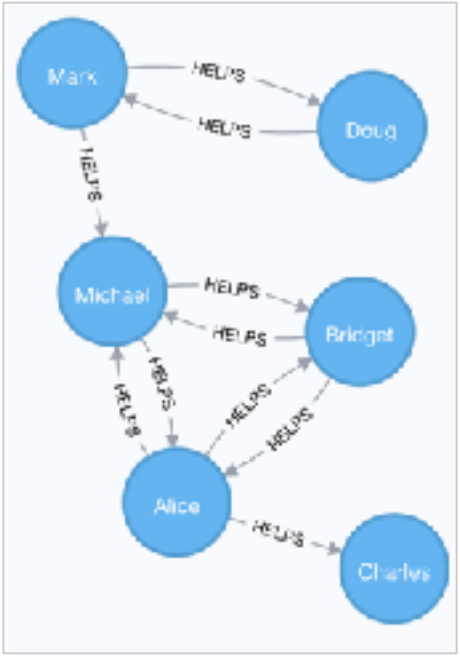
Triangles for the Person nodes



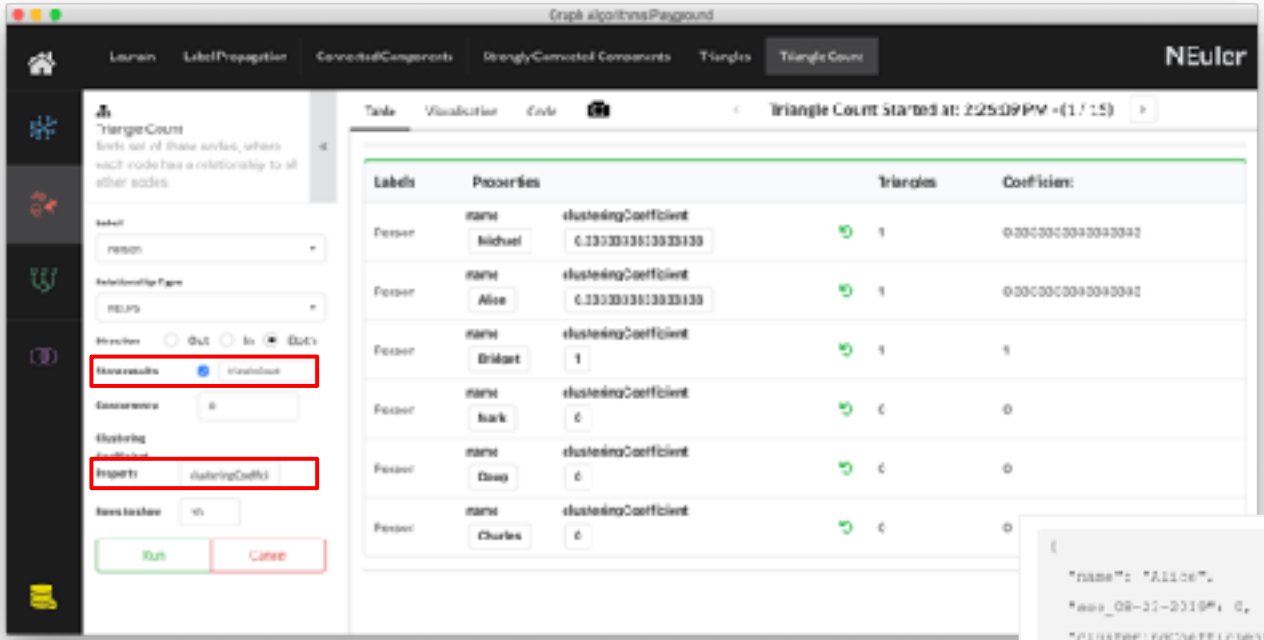


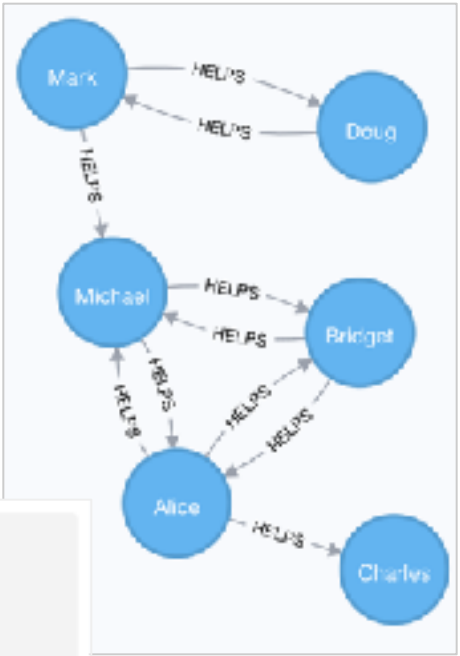
Triangle Counts and Coefficients for the Person nodes



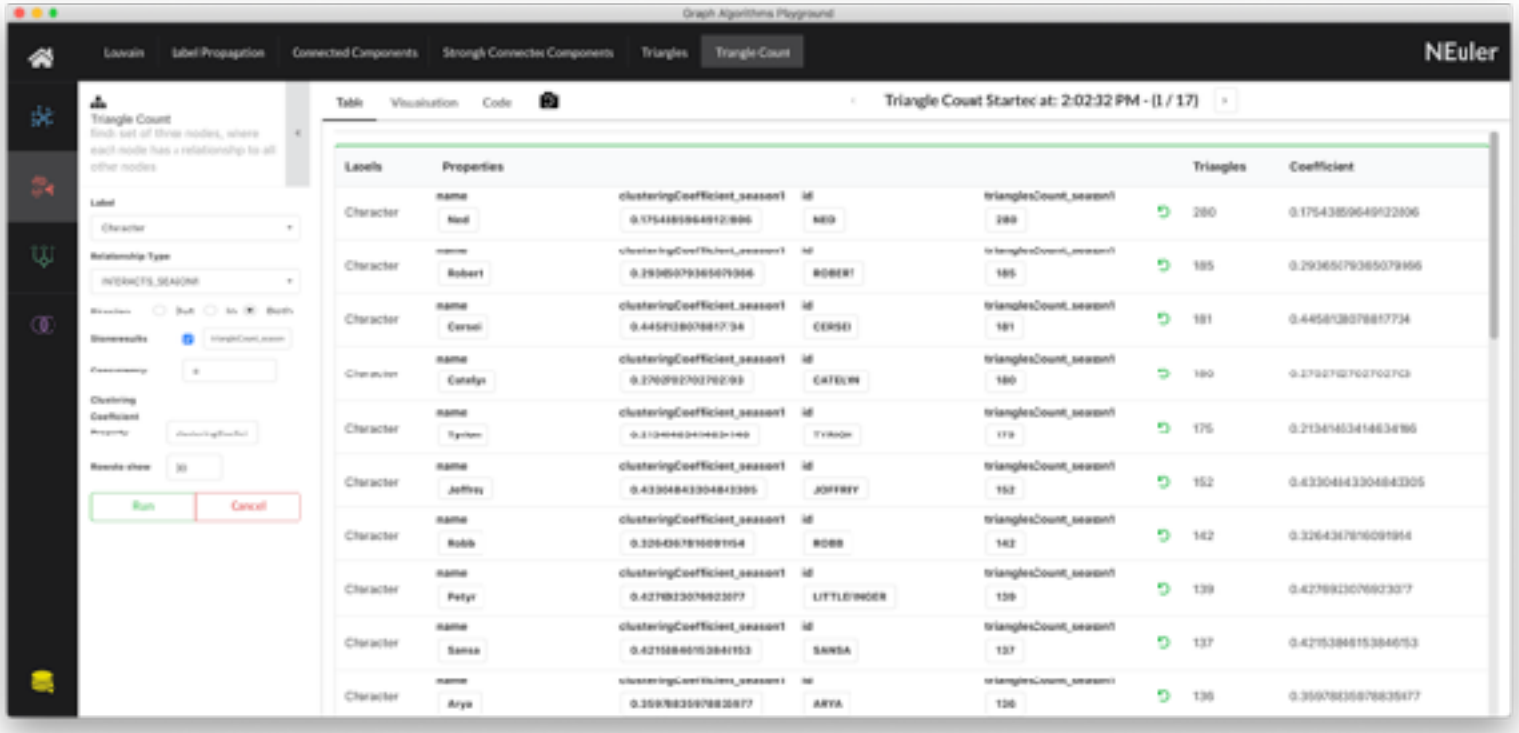


NEuler: Setting Triangle Counts and Coefficients for the Person nodes

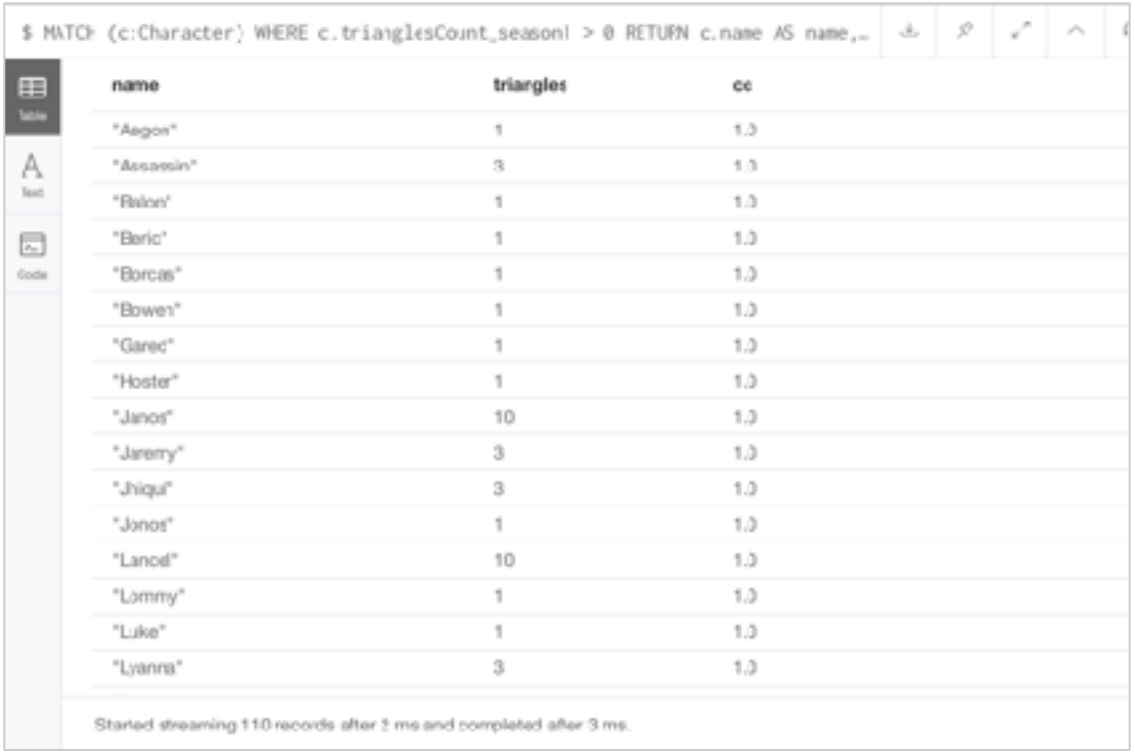




Triangle counts and coefficients for GOT Characters



Ordered clustering coefficients for GOT Characters



name	triangles	cc
"Aegon"	1	1.0
"Assassin"	3	1.0
"Balon"	1	1.0
"Beric"	1	1.0
"Borcas"	1	1.0
"Bowen"	1	1.0
"Garec"	1	1.0
"Hoster"	1	1.0
"Janos"	10	1.0
"Jaremy"	3	1.0
"Jhiqui"	3	1.0
"Jonos"	1	1.0
"Lancel"	10	1.0
"Lommy"	1	1.0
"Luke"	1	1.0
"Lyanna"	3	1.0

Started streaming 110 records after 1 ms and completed after 3 ms.

Exercise 6:
Triangle Count

Estimated time to complete this exercise:

10 minutes

1. In NEuler:
   a. Perform some Triangle and Triangle Count algorithms on the Person data, noting what, if anything is written to the graph.
   b. Perform some Triangle and Triangle Count algorithms on different seasons of GOT.

1. In Neo4j Browser:

```
:play intro-graph-algos-exercises  (Triangle Count)
```

Relationship Directions? Weights?

Community Detection algorithms

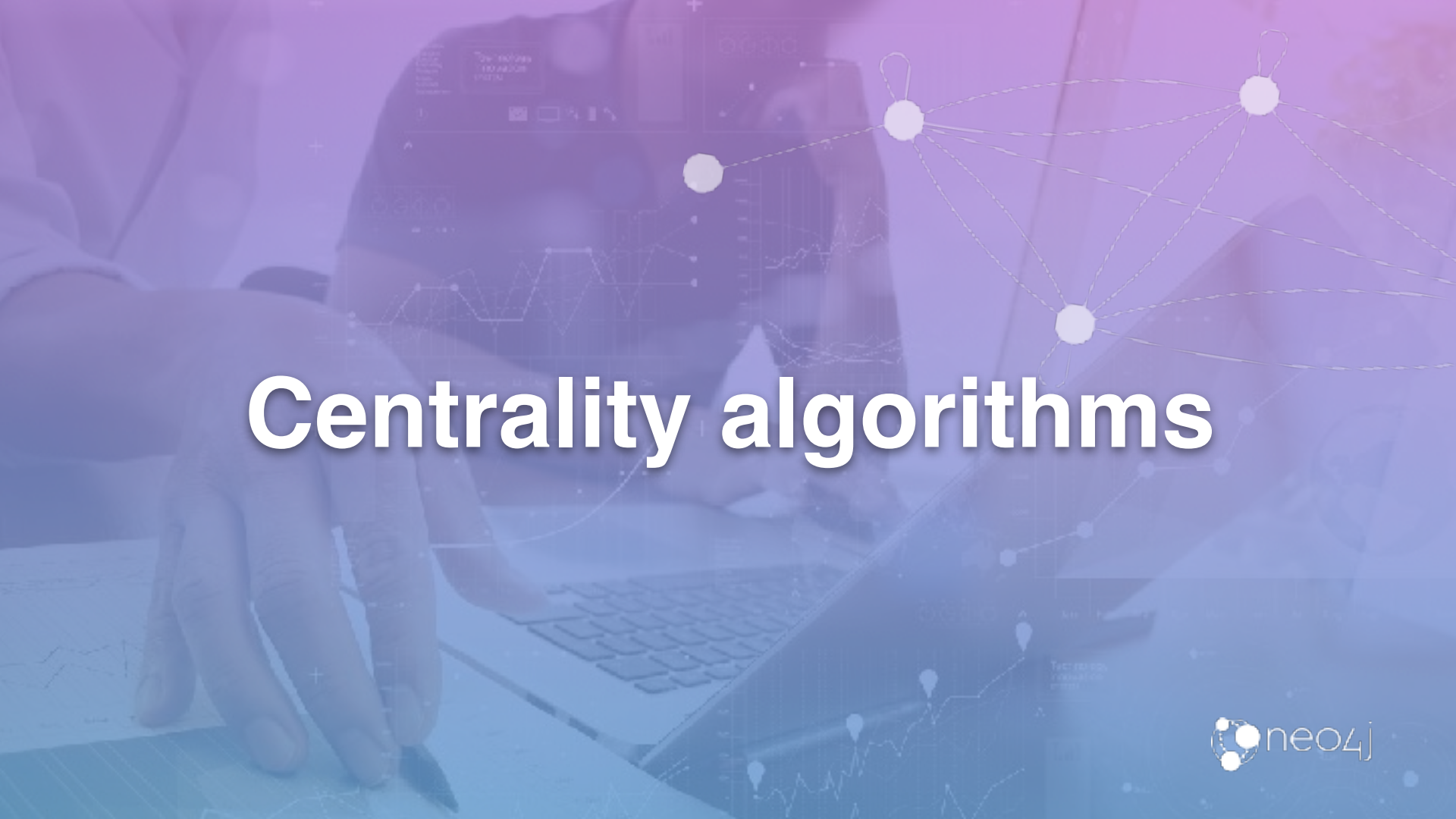
Algorithm	Directed	Undirected	Weighted	Unweighted
Louvain		✓	✓	✓
Label Propagation	✓	✓	✓	✓
Connected Components		✓	✓	✓
Strongly Connected Components	✓			✓
Triangle Counting		✓		✓
Balanced Triads		✓	✓	

weight used as threshold

relationship must exist in both directions

weight indicates a positive or negative relationship (friend or enemy)

BREAK


Centrality algorithms
neo4j

Centrality algorithms

Determines the importance of distinct nodes in the network.

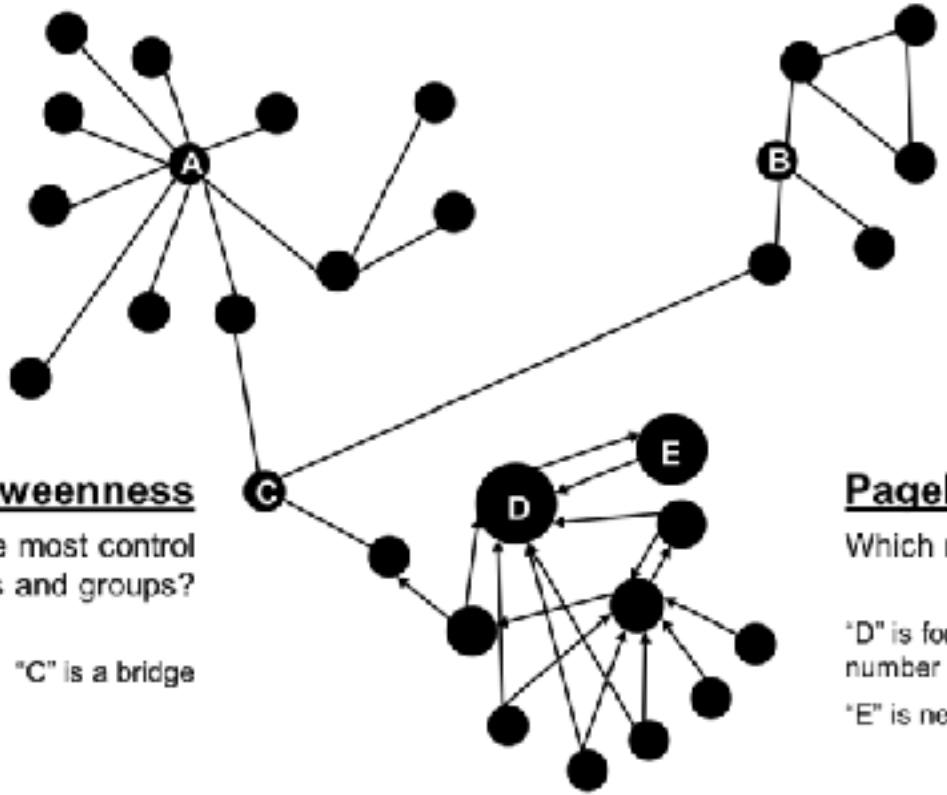
Developed for distinct uses or types of importance.

Some Centrality algorithms

Degree

Number of connections?

"A" has a high degree



Closeness

Which node can most easily reach all other nodes in a graph or subgraph?

"B" is closest with the fewest hops in it's subgraph

Betweenness

Which node has the most control over flow between nodes and groups?

"C" is a bridge

PageRank

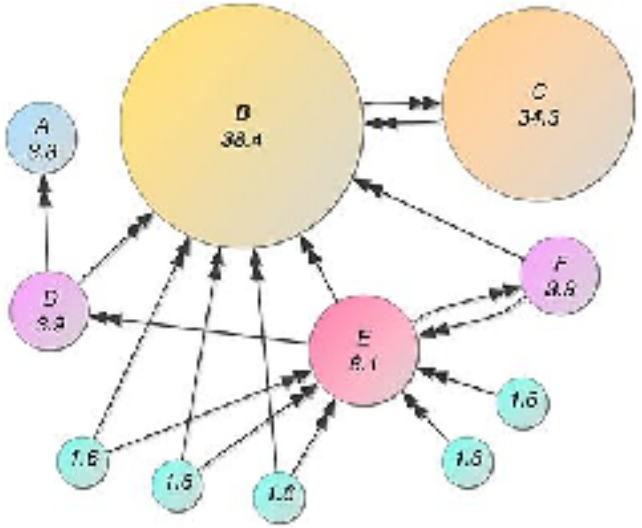
Which node is the most important?

"D" is foremost based on number & weighting of in-links

"E" is next, due to the influence of D's link

PageRank

PageRank



Measures the transitive (directional) influence of nodes and considers the influence of neighbors and their neighbors.

- **Dampening factor**: Probability that end user will click through a link. Default value on Neo4j algo is .85.

Tip / Caution

Test your dampening factor as it will change the outcome.

Note that Spark uses a inverse dampening (0.15)

Careful with mixing node types

PageRank - Uses

Use When
Looking for broad influence over a network. Many domain-specific variations for differing analysis (e.g. Personalized Pagerank for personalized recommendations)

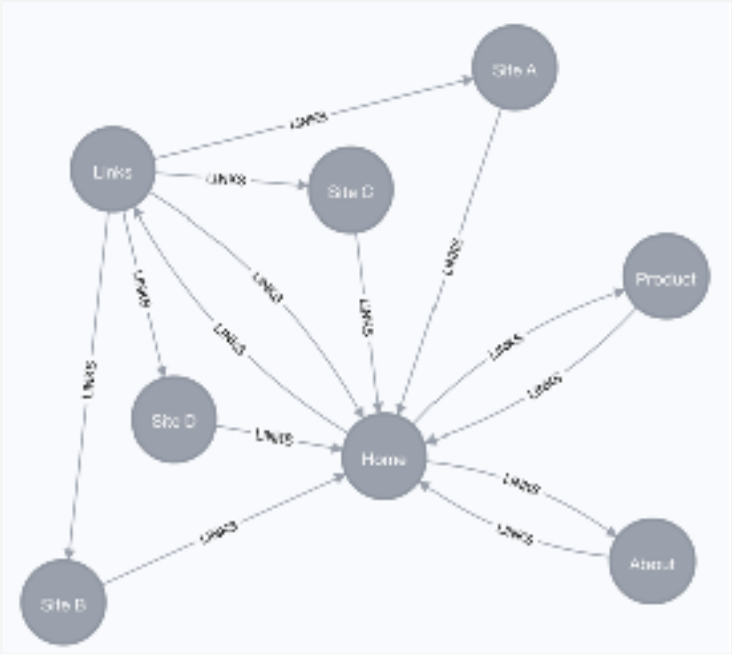
Recommendations

Who To follow with personalized PR

Fraud Detection

Feature engineering for machine learning

PageRank example: Using the Page graph



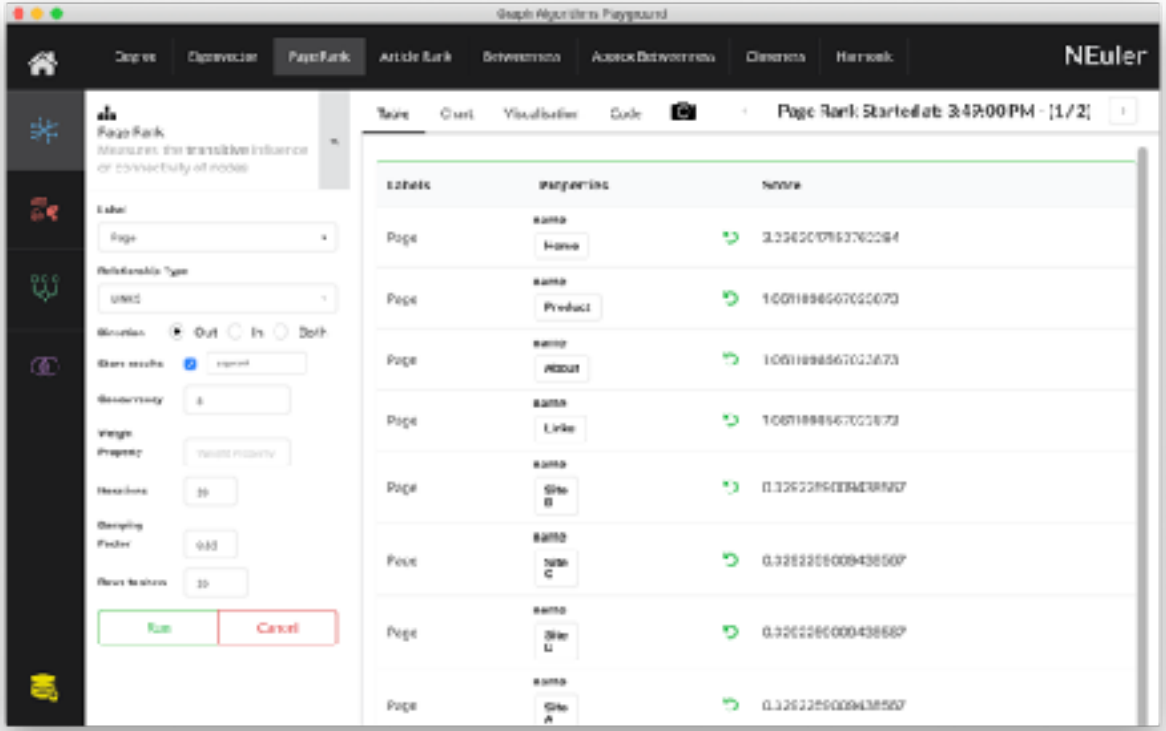

```
CALL algo.pageRank.stream("Page", "LINKS",
{iterations:20})
YIELD nodeId, score
MATCH (node) WHERE id(node) = nodeId
RETURN node.name AS page,score
ORDER BY score DESC
```

```
$ CALL algo.pageRank.stream("Page", "LIN.
```

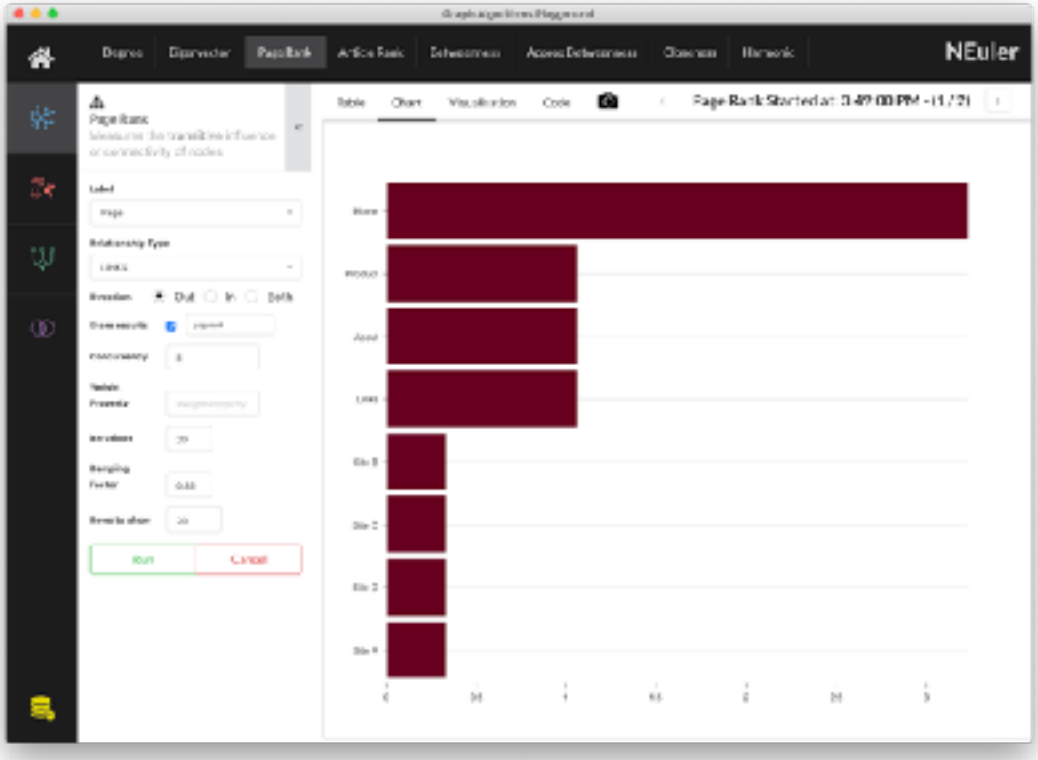
page	score
"Home"	3.2362017153782284
"About"	1.0511098567023973
"Product"	1.0511098567023873
"Links"	1.0511098567023973
"Site A"	0.3292259009438567
"Site B"	0.3292259009438567
"Site C"	0.3292259009438567
"Site D"	0.3292259009438567

Started streaming 8 records after 48 ms and completed after 49 ms.

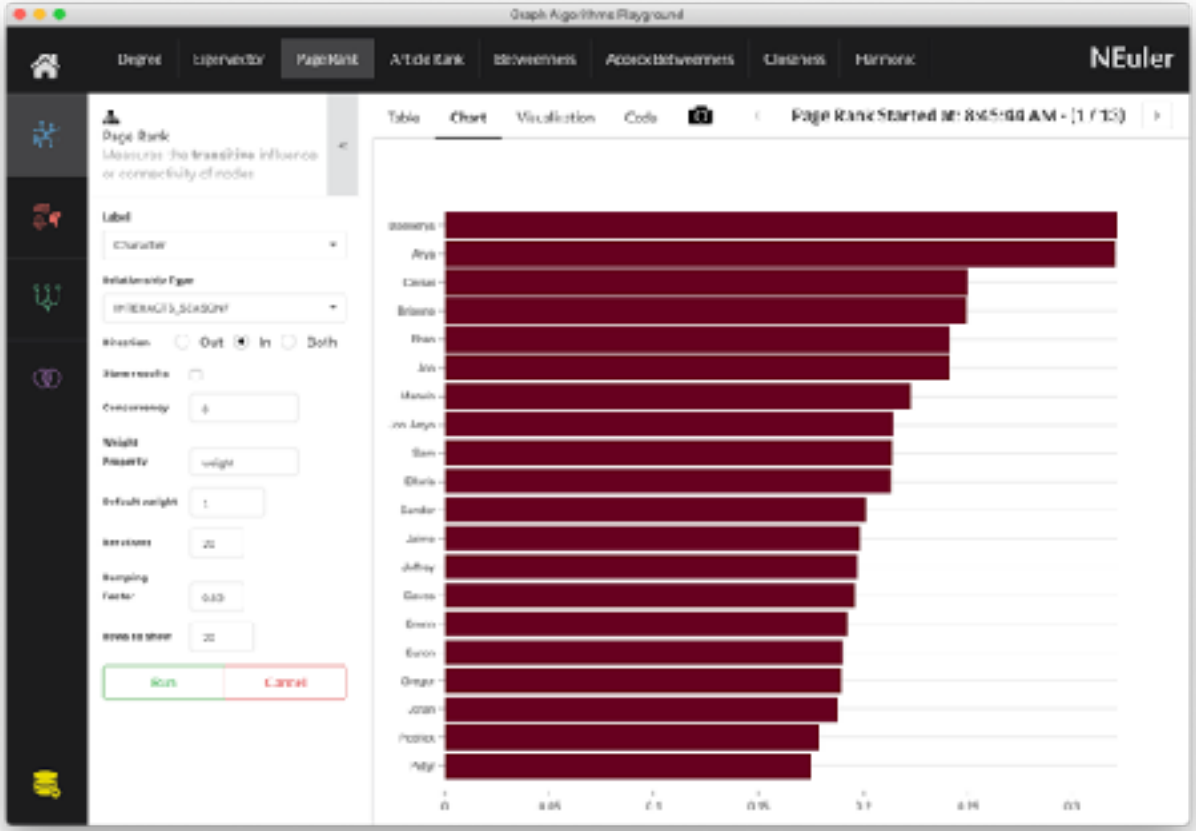
PageRank example in NEuler - 1



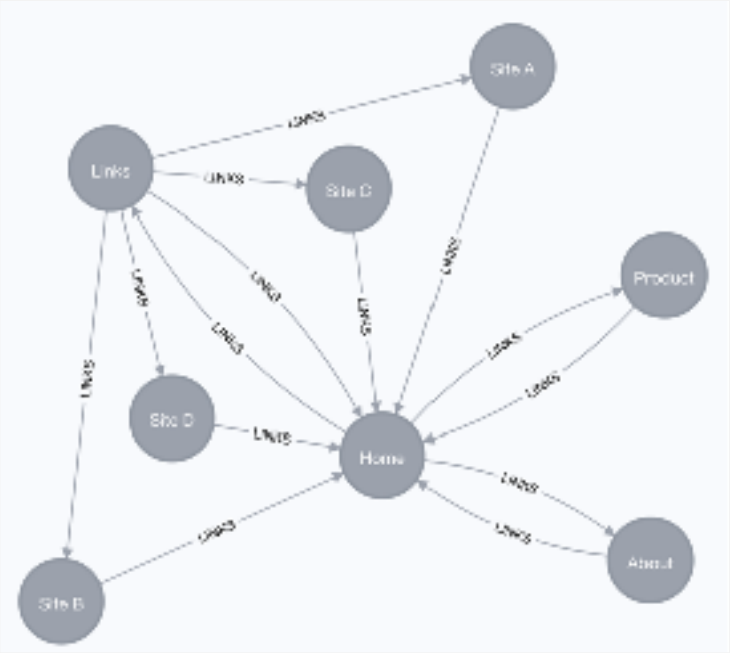
PageRank example in NEuler - 2



PageRank: GOT



Personalized PageRank example: Page graph




```
MATCH (p:Page)
WHERE p.name = 'Site A' OR p.name = 'Site C'
with collect(p) AS sites
CALL algo.pageRank.stream("Page", "LINKS",
{iterations:20, sourceNodes: sites})
YIELD nodeId, score
MATCH (node) WHERE id(node) = nodeId
RETURN node.name AS page,score
ORDER BY score DESC
```

Exercise 7
PageRank

Estimated time to complete this exercise:

10 minutes

1. **In NEuler:**

   Perform the PageRank analysis on different seasons of GOT.

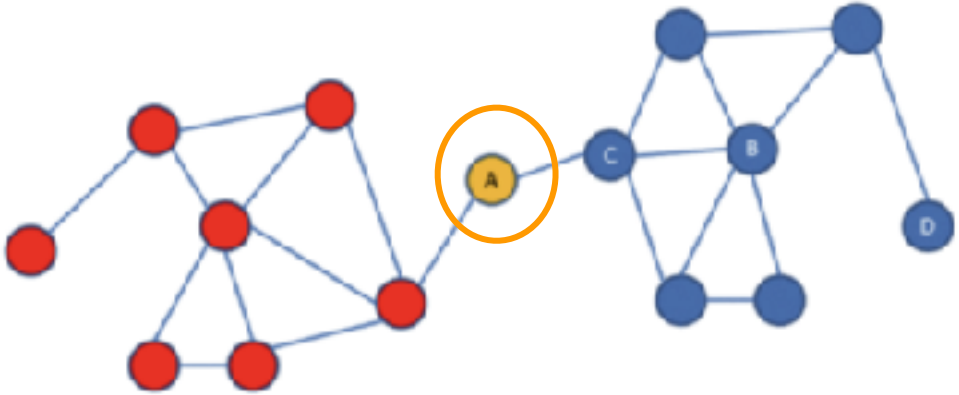
1. **In Neo4j Browser:**

   :play intro-graph-algos-exercises (PageRank)

Betweenness Centrality

Betweenness Centrality

The sum of the % shortest paths that pass through a node, calculated by pairs.



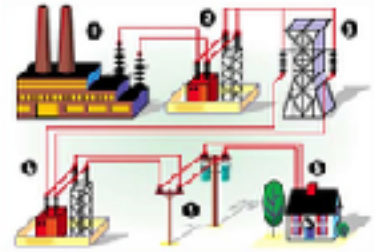
Tip / Caution

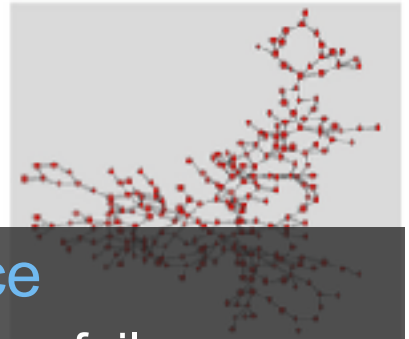
Computationally intensive: use RA Brandes approximation on large graphs.

Assumes all communication between nodes happens along the shortest path and with the same frequency (not always the case in real life)

Betweenness Centrality - Uses

Use When
Identify bridges
Uncover control points
Find bottlenecks and vulnerabilities





Network Resilience

Key points of cascading failure

neo4j

Relationship Directions? Weights?

Centrality algorithms

Algorithm	Directed	Undirected	Weighted	Unweighted
PageRank	✓	✓	✓	✓
ArticleRank	✓	✓		✓
Betweenness Centrality	✓	✓		✓
Closeness Centrality	✓	✓		✓
Harmonic Centrality		✓		✓
Eigenvector Centrality	✓	✓		✓
Degree Centrality	✓	✓	✓	✓

Exercise 8
Centrality

Estimated time to complete this exercise:

20 minutes

1. In NEuler
   a. View the closeness and betweenness centrality scores for other GOT seasons and examine the paths in Neo4j Browser for these nodes.
   b. Try some of the other centrality algorithms with the GOT data.
2. In Neo4j Browser:

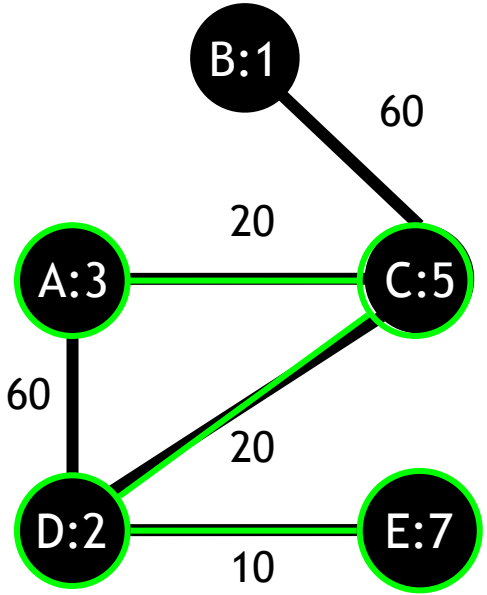
```
:play intro-graph-algos-exercises (Centrality)
```


Pathfinding algorithms
neo4j

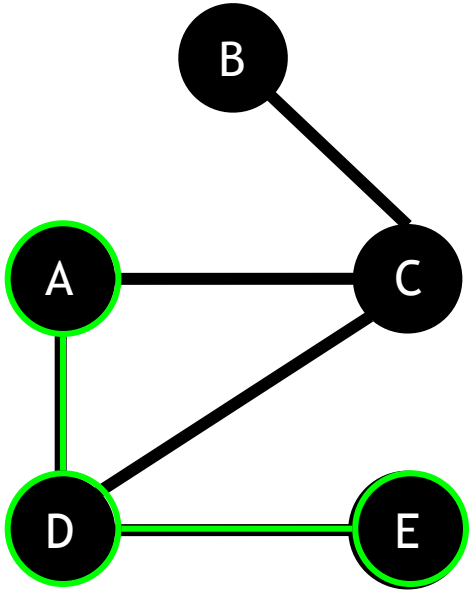
Pathfinding and Graph Search algorithms

Pathfinding and Graph Search algorithms are used to identify optimal routes, and they are often a required first step for many other types of analysis.

Shortest Path

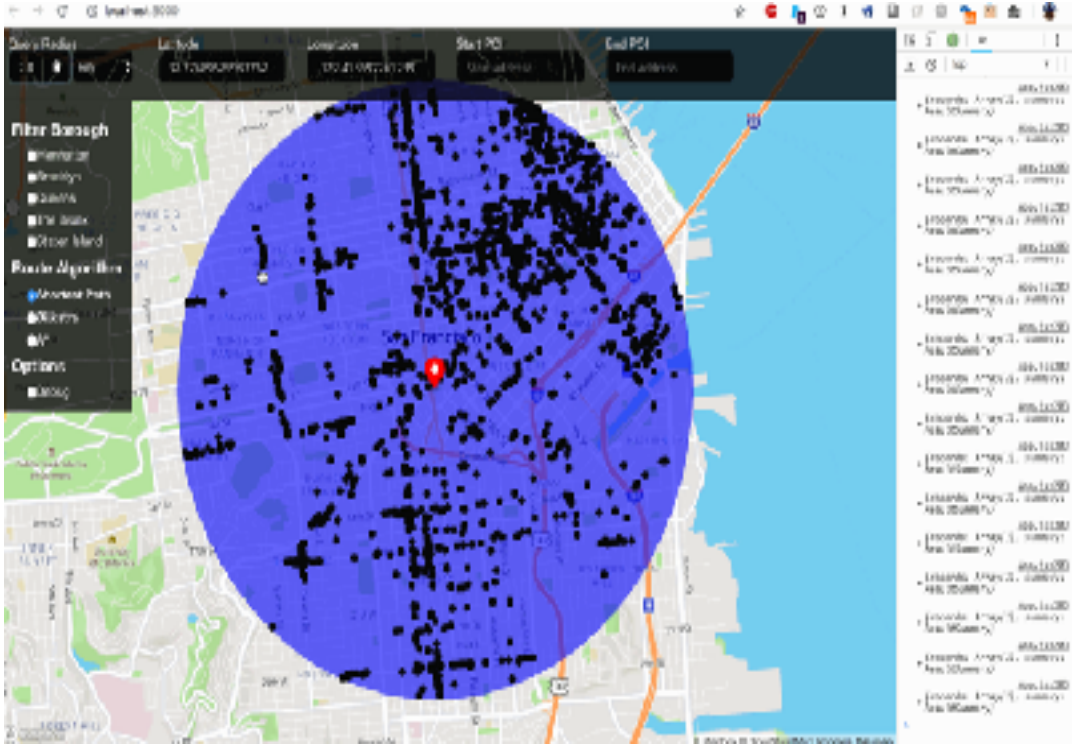


Weighted



Unweighted

Shortest Path example



https://github.com/johnymontana/osm-routing-app

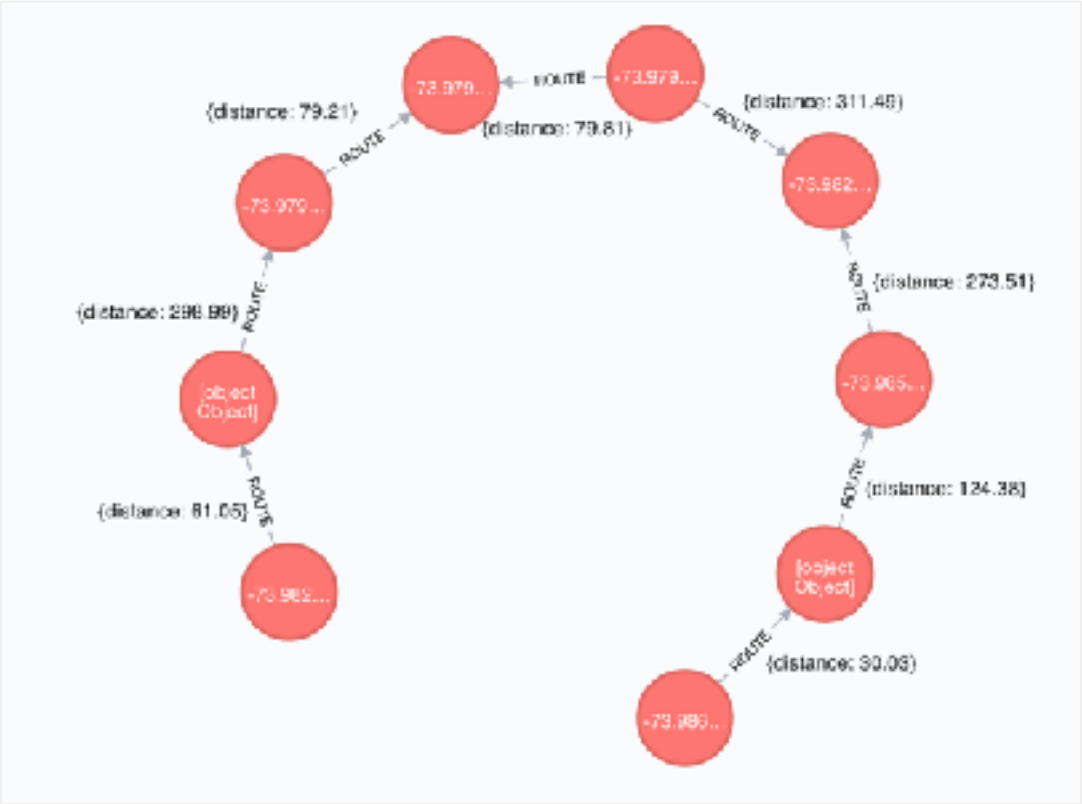
Routing with Cypher's shortestPath() function

```
MATCH (a:PointOfInterest)
 WHERE a.name = 'The View Restaurant & Lounge'
MATCH (b:PointOfInterest)
 WHERE b.name = 'Gregory Coffee'
MATCH p=shortestPath((a)-[:ROUTE*..100]-(b))
RETURN p
```

Cypher's shortestPath() not always best?

```
MATCH (a:PointOfInterest)
 WHERE a.name = 'The View Restaurant & Lounge'
MATCH (b:PointOfInterest)
 WHERE b.name = 'Gregory Coffee'
MATCH p=shortestPath((a)-[:ROUTE*..100]-(b))
RETURN p
```

Weighted Paths



Total weight: 1258.48m

Shortest Weighted Path algorithms

Dijkstra

- Similar to BFS.
- Uses priority queue to **explore shorter paths first**.
- No concept of direction.
- No intermediate evaluation.

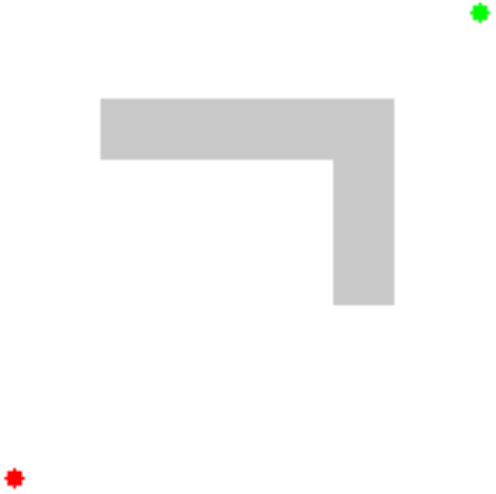
A*

- Extends Dijkstra with heuristic.
- Priority queue is ordered by sum of weights plus **heuristic** (how much further).
- Explore paths in the right direction first!

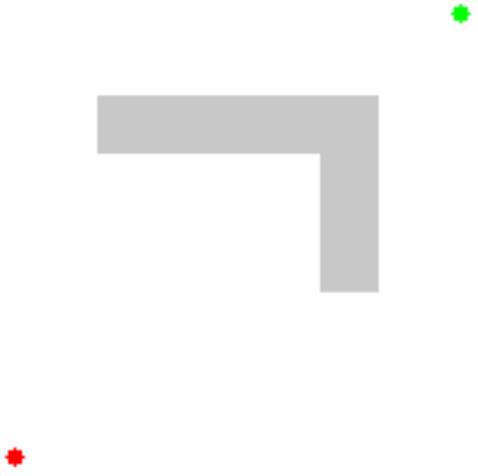
neo4j

Shortest Weighted Path algorithms

Dijkstra



*A**



neo4j

Example: Routing with Dijkstra algorithm

```
MATCH (a:PointOfInterest) WHERE a.name = 'The View Restaurant & Lounge'
MATCH (b:PointOfInterest) WHERE b.name = 'Gregory Coffee'

CALL algo.shortestPath.stream(a, b, 'distance',
{relationshipQuery: 'ROUTE', nodeQuery:'Routable', direction:'BOTH', defaultValue:1.0})
YIELD nodeId, cost

MATCH (n) WHERE id(n) = nodeId
RETURN COLLECT({lat: n.location.latitude, lon: n.location.longitude}) AS route
```


neo4j

Example: Routing with A*

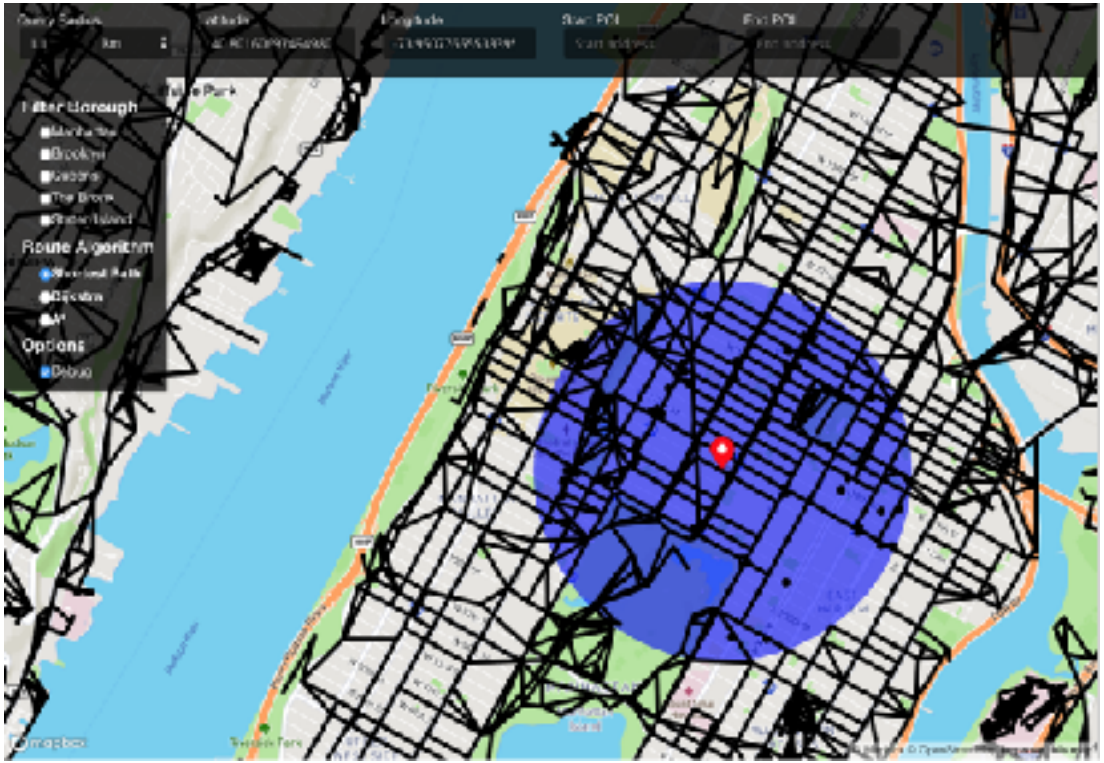
```
MATCH (a:PointOfInterest) WHERE a.name = 'The View Restaurant & Lounge'
MATCH (b:PointOfInterest) WHERE b.name = 'Gregory Coffee'

CALL algo.shortestPath.astar.stream(a, b, distance, 'lat', 'lon',
{relationshipQuery: 'ROUTE', nodeQuery:'Routable', direction:'BOTH', defaultValue:1.0})

YIELD nodeId, cost
MATCH (route) WHERE id(route) = nodeId
RETURN COLLECT({lat: route.location.latitude, lon: route.location.longitude}) AS route
```


neo4j

Limit search space with Cypher projections



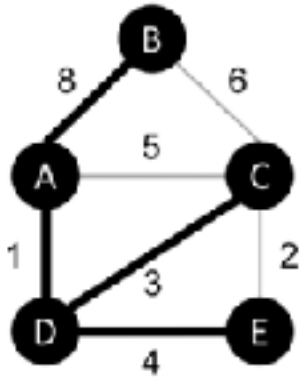
Example: Limit search space with Cypher projections

```
MATCH (a:PointOfInterest) WHERE a.poi_id = $startPOI
MATCH (b:PointOfInterest) WHERE b.poi_id = $endPOI
CALL algo.shortestPath.astar.stream(a, b, 'weight', 'lat', 'lon',
{
        relationshipQuery: "MATCH (a1:Routable)-[r:ROUTE]-(a2:Routable) WHERE distance(a1.location,$center) < $radius
AND distance(a2.location, $center) < $radius RETURN id(a1) as source, id(a2) as target,r.distance as weight",
        nodeQuery:"MATCH (a1:Routable) WHERE distance(a1.location, $center) < $radius RETURN id(a1) AS id",

        direction:'both', defaultValue:1.0, graph:'cypher',
        params: {center: point({latitude: $routeCenterLat, longitude: $routeCenterLon}), radius: $routeRadius}
})
YIELD nodeId, cost
MATCH (route) WHERE id(route) = nodeId
RETURN COLLECT({lat: route.location.latitude, lon: route.location.longitude}) AS route
```


neo4j

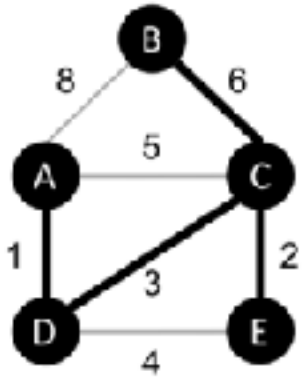
Other types of Pathfinding algorithms



Single Source Shortest Path

Shortest path from **a root** node (A shown) to **all** other nodes

Traverses to the next unvisited node via the lowest cumulative weight **from the root**



Minimum Weight Spanning Tree

Shortest path **connecting all nodes** (A start shown)

Traverses to the next unvisited node via the lowest weight **from any visited node**

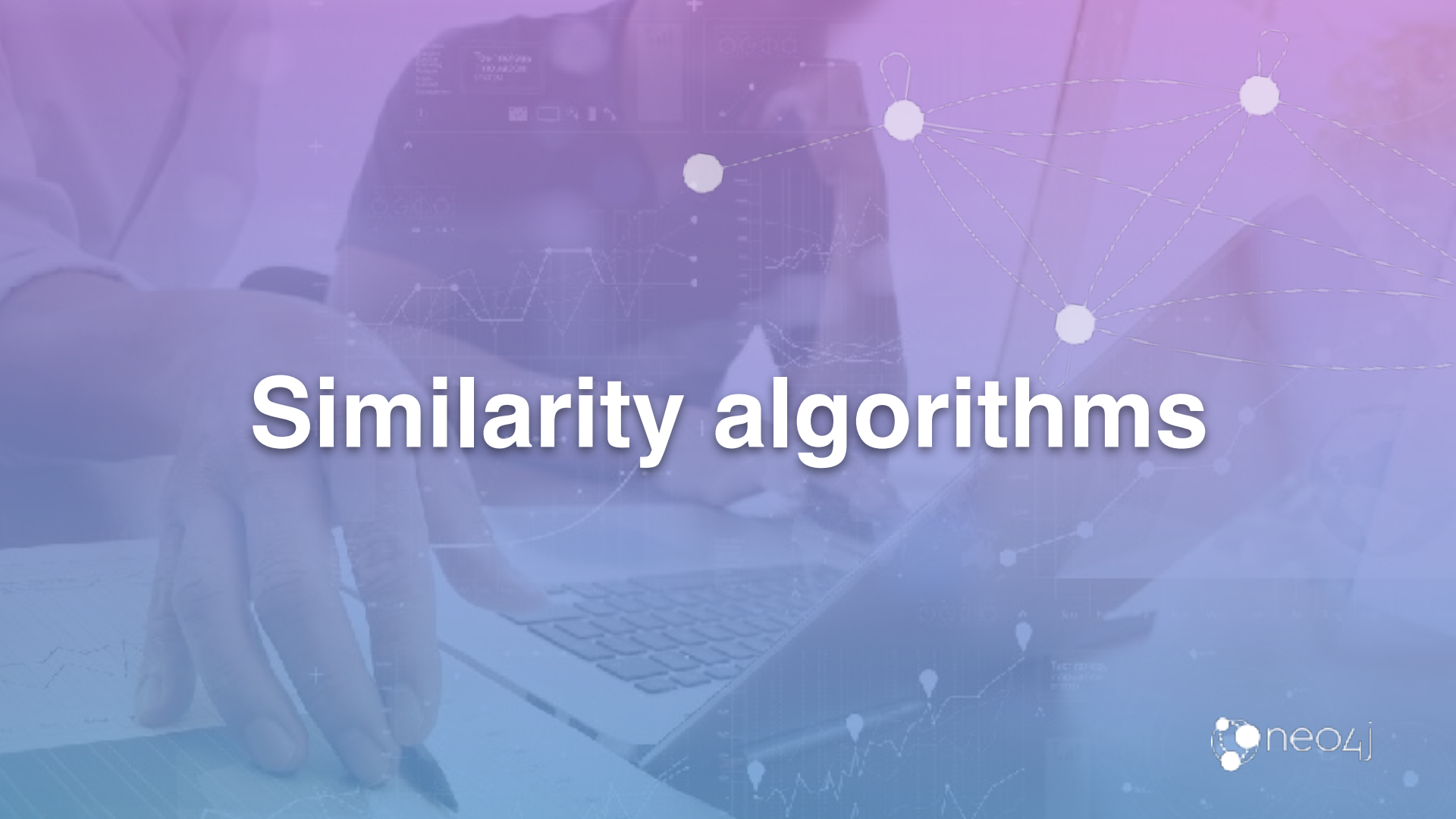
neo4j

Relationship Directions? Weights?

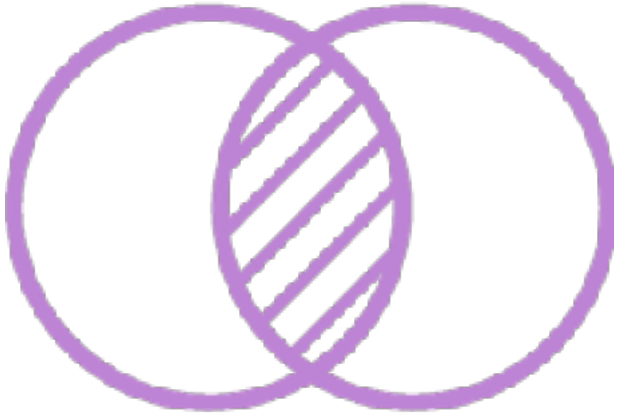
Pathfinding algorithms

Algorithm	Directed	Undirected	Weighted	Unweighted
Minimum Weight Spanning Tree		✓	✓	
Shortest Path	✓	✓	✓	✓
Single Source Shortest Path	✓	✓	✓	✓
All Pairs Shortest Path		✓	✓	✓
A*	✓	✓	✓	✓
Ken's K-shortest Paths	✓	✓	✓	✓
Random Walk		✓		✓

BREAK


Similarity algorithms
neo4j

Similarity Algorithms



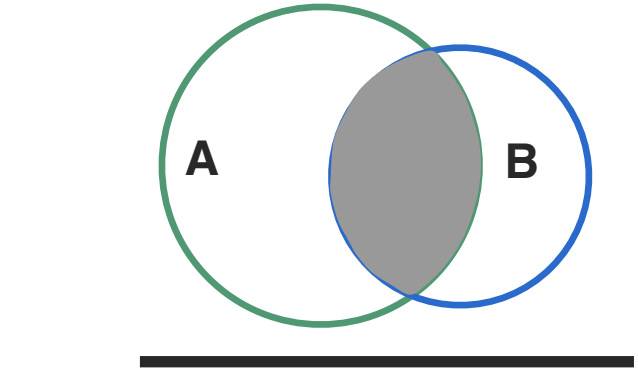
Evaluates how alike nodes are at an individual level either based on node attributes, neighboring nodes, or relationship properties

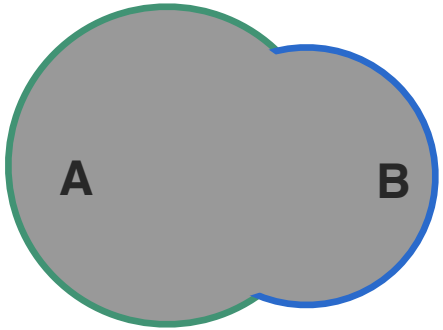
neo4j

Jaccard Similarity coefficient

Often used to find recommendations of similar items as well as part of link prediction.

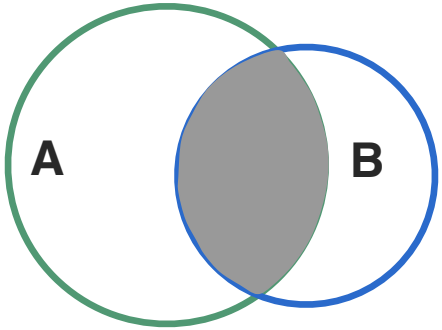
Jaccard Similarity measures the similarity between sets.

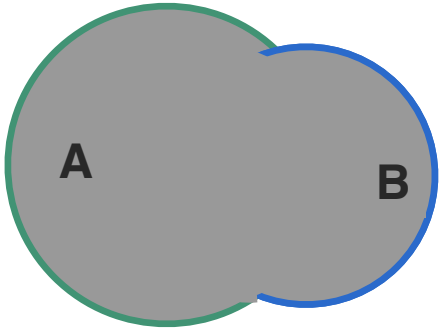




$$\frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Jaccard Similarity coefficient





A = {Orange, Banana, Cherry, Pineapple}
B = {Orange, Banana, Apple}

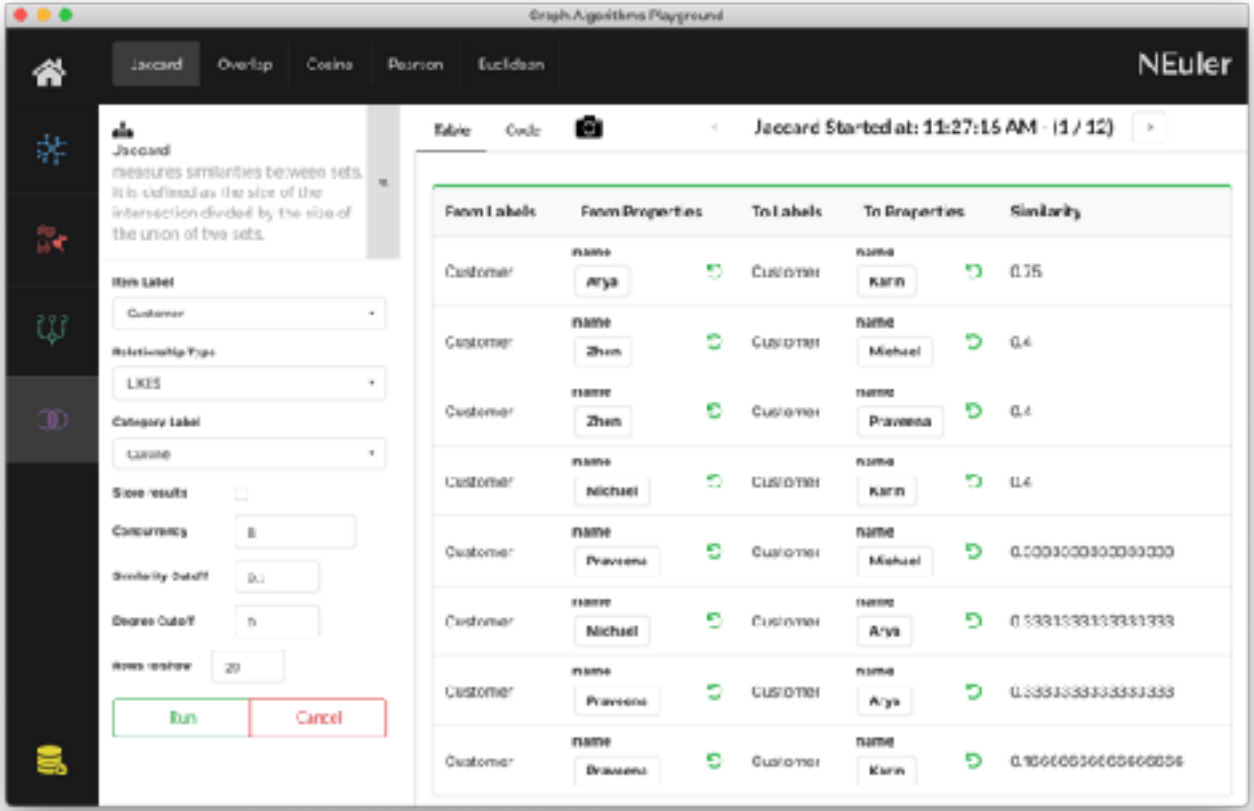
$|A \cap B| = |\{\text{Orange, Banana}\}| = 2$

$|A \cup B| = |A| + |B| - |A \cap B|$

$|A \cup B| = |\{\text{Orange, Banana, Cherry, Pineapple Apple}\}| = 5$

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = 2/5 = 0.4$$

Jaccard Similarity in NEuler



Example: Jaccard Similarity procedure

```
MATCH (c:Customer)-[:LIKES]->(cuis)
WITH {item:id(c), categories:
collect(id(cuis))} as userData
WITH collect(userData) as data
CALL
algo.similarity.jaccard.stream(data)
YIELD item1, item2, count1, count2,
intersection, similarity
RETURN algo.asNode(item1).name AS
from,
      algo.asNode(item2).name AS to,
intersection,
      similarity
ORDER BY similarity DESC
```

from	to	intersection	similarity
"Arya"	"Karin"	3	0.75
"Zhen"	"Praveena"	2	0.4
"Zhen"	"Michael"	2	0.4
"Michael"	"Karin"	2	0.4
"Praveena"	"Michael"	2	0.3333333333333333
"Praveena"	"Arya"	2	0.3333333333333333
"Michael"	"Arya"	2	0.3333333333333333
"Praveena"	"Karin"	1	0.16666666666666666
"Zhen"	"Arya"	0	0.0
"Zhen"	"Karin"	0	0.0

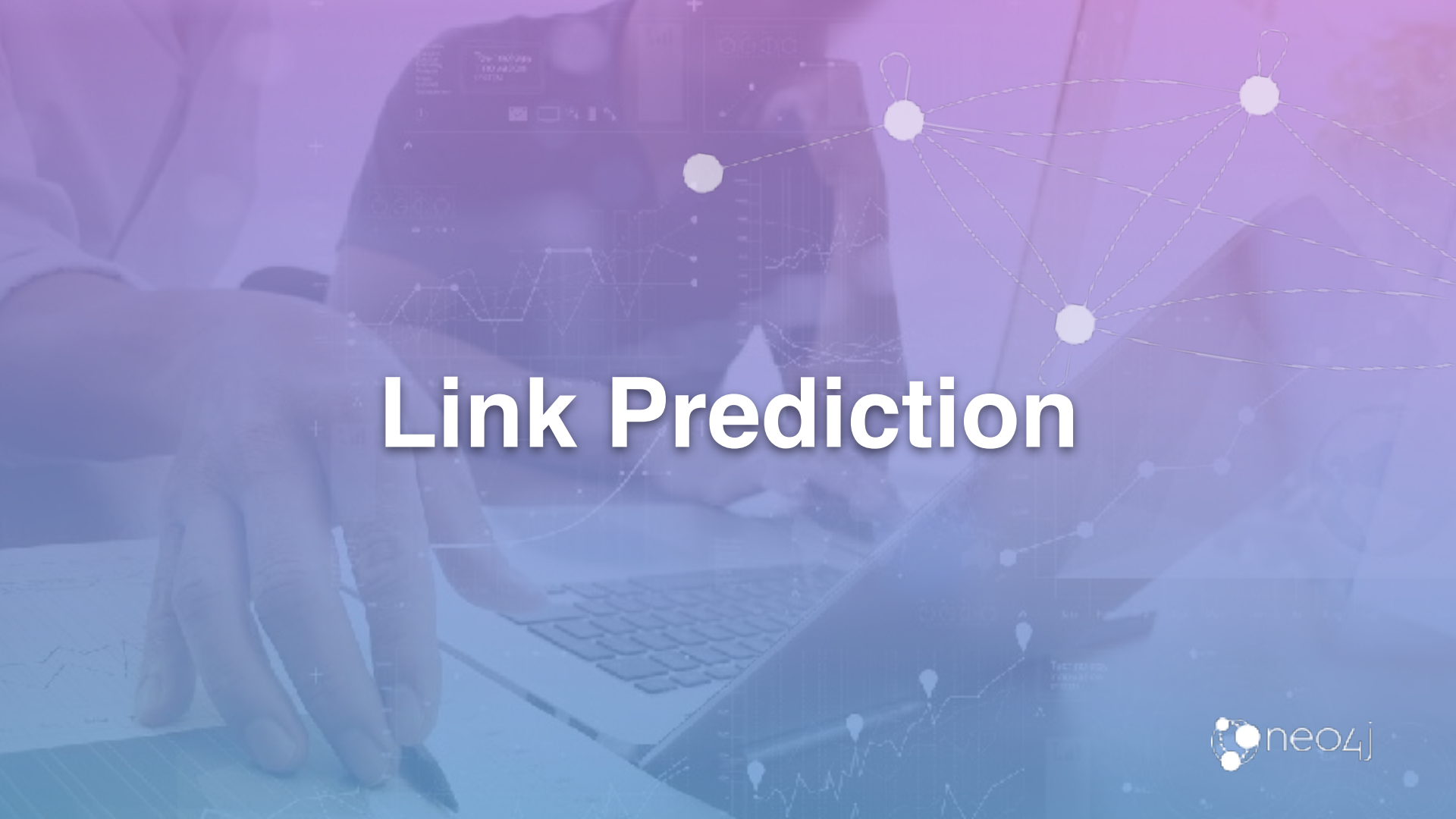
Warning: Similarity algorithms are computationally intensive

Similarity calculations can be slow:

- create vectors for every node to calculate metrics
- Pair comparison between all nodes in the graph
- Writing back many relationships

What can you do instead?

- **Set your similarity threshold high** and only writeback relationships above threshold
- **Pre-process your data** (instead of `unwind`) to create vectors ahead of the algorithm call
- **Subset your graph** using and algorithm like UnionFind or LabelPropagation


Link Prediction
neo4j

Link Prediction

Can we infer which new interactions are likely to occur in the future?

*"We formalize this question as the **link prediction problem**, and develop approaches to link prediction based on measures for **analyzing the "proximity" of nodes** in a network."*

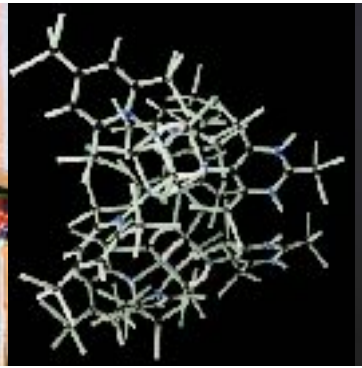
Jon Kleinberg and David Liben-Nowell

A Goal, an Approach
&
an Algorithm Category

What's common across all these use cases?

- future associations in a terrorist network
- co-authorships in a citation network
- associations between molecules in a biology network
- interest in an artist or artwork





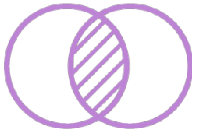

Graph Algorithms used with Link Prediction

Link Prediction

These methods compute a score for a pair of nodes, where the score could be considered a **measure of proximity** or "similarity" between those nodes **based on the graph topology**.

Other Algorithms types

It's common when our goal is link prediction to use a variety of algorithm types to extract features and use them together in a machine learning model



Similarity

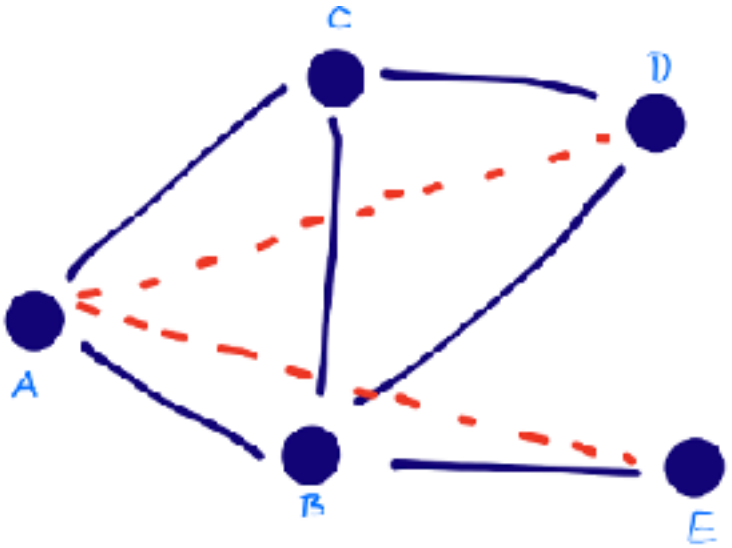
Community Detection

Centrality

Common Neighbors

Based on number of potential triangles / closing triangles.

Concept is that if 2 strangers have a friend/colleague in common, they are more likely to be introduced.



http://be.amazd.com/link-prediction/

Example: Common Neighbors


```
MATCH (e1:Employee {name: 'Larry'})
MATCH (e2:Employee {name: 'Sophia'})
RETURN algo.linkprediction.commonNeighbors(e1, e2) AS score
```

score
2.0

```
MATCH (e1:Employee {name: 'Robert'})
MATCH (e2:Employee {name: 'Joe'})
RETURN algo.linkprediction.commonNeighbors(e1, e2) AS score
```

score
1.0

```
MATCH (e1:Employee {name: 'Sophia'})
MATCH (e2:Employee {name: 'Joe'})
RETURN algo.linkprediction.commonNeighbors(e1, e2,
 {relationshipQuery: 'WORKS_WITH'}) AS score
```

score
0.0

Relationship Directions?

Link Prediction algorithms

Algorithm	Directed	Undirected	Additional Parameter	Unweighted
Adamic Adar	✓	✓		
Common Neighbors	✓	✓		
Preferential Attachment	✓	✓		
Resource Allocation	✓	✓		
Same Community			community	
Total Neighbors	✓	✓		

Exercise 11
Link Prediction

Estimated time to complete this exercise:

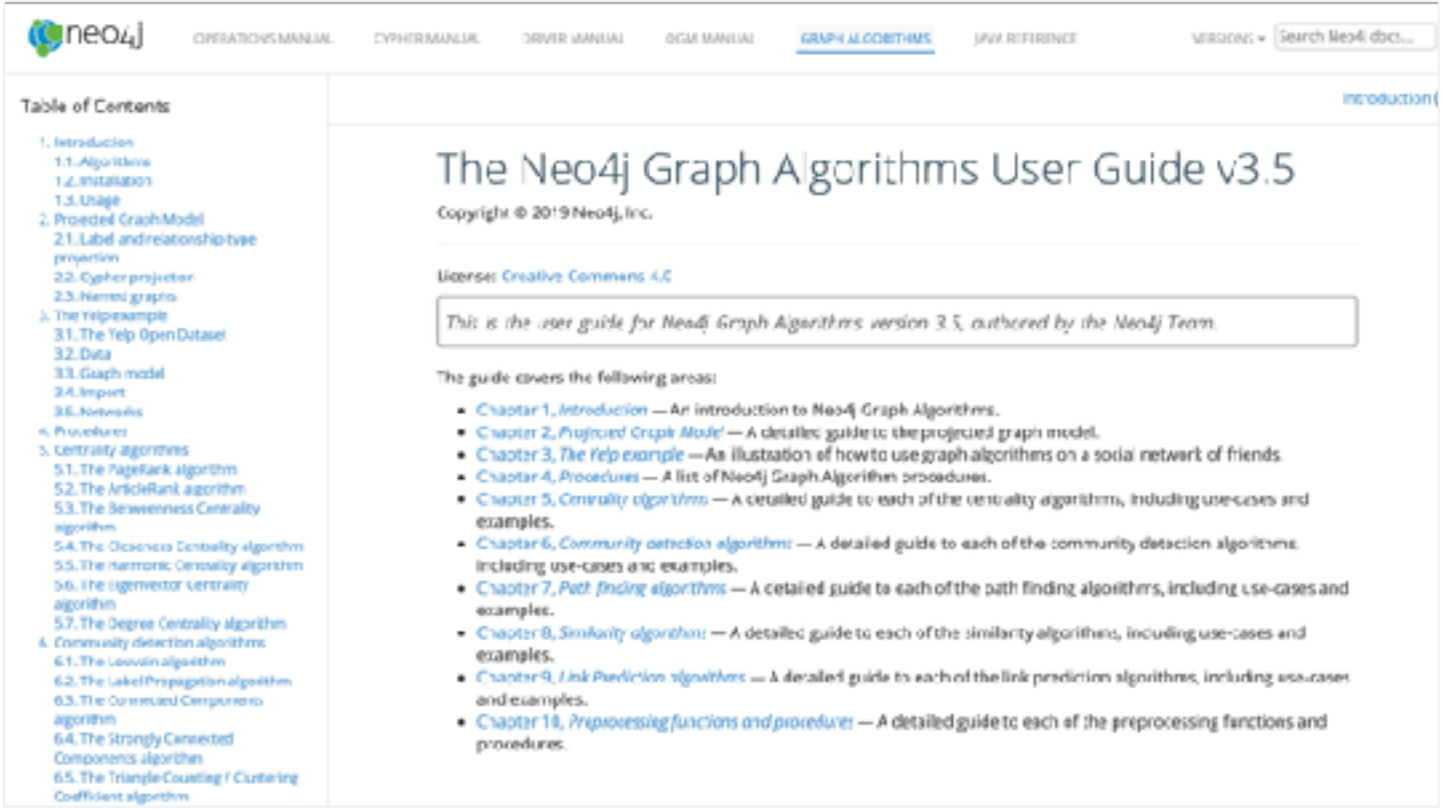
15 minutes

In Neo4j Browser:

:play intro-graph-algos-exercises (Link Prediction)

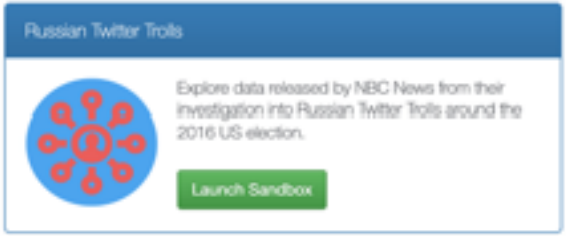

Resources

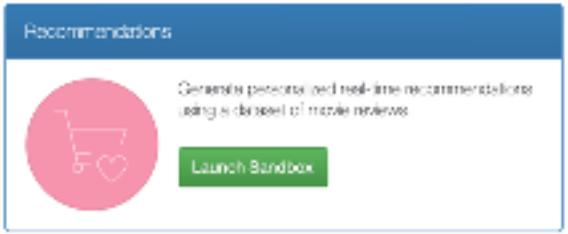
neo4j.com/docs

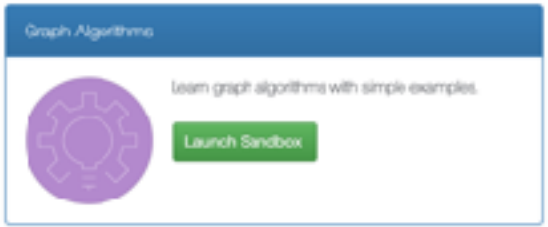


https://neo4j.com/docs/graph-algorithms/current/

neo4jsandbox.com

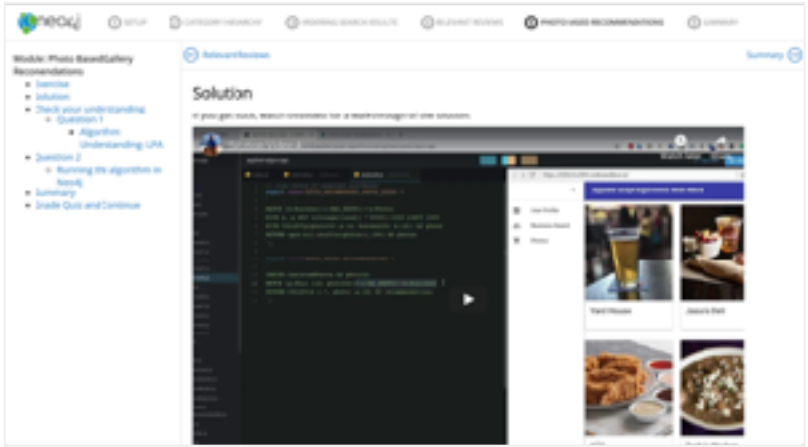






Online Training!

Applied Graph Algorithms



https://neo4j.com/graphacademy/online-training/applied-graph-algorithms

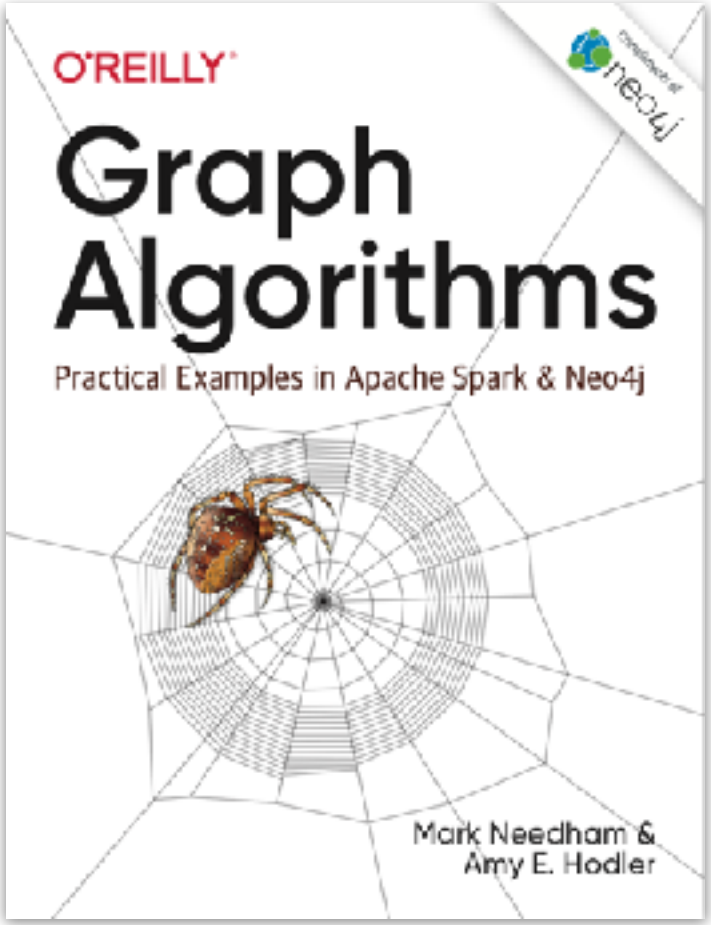
Data Science With Neo4j

https://neo4j.com/graphacademy/online-training/data-science

Free O'Reilly Book

neo4j.com/graph-algorithms-book

- Spark & Neo4j Examples
- Machine Learning Chapter



neo4j

Questions?

Justin Fine
Justin.Fine@neo4j.com

Thank you!

neo4j