Codes

* Similarity Algorithms

1. Jaccard Similarity

:param limit => 50;

:param weightProperty => null;

:param config => {

concurrency: 8,

writeProperty: 'score',

writeRelationshipType: 'SIMILAR\_JACCARD',

similarityCutoff: 0.1,

degreeCutoff: 0,

write: true

};

MATCH (item)-->(category)

WITH {item:id(item), categories: collect(distinct id(category))} as userData

WITH collect(userData) as data

CALL algo.similarity.jaccard(data, $config)

YIELD nodes, similarityPairs, write, writeRelationshipType, writeProperty, min, max, mean, stdDev, p25, p50, p75, p90, p95, p99, p999, p100

RETURN nodes, similarityPairs, write, writeRelationshipType, writeProperty, min, max, mean, p95

MATCH (from)-[rel:`SIMILAR\_JACCARD`]-(to)

WHERE not (rel[$config.writeProperty] is null)

RETURN from, to, rel[$config.writeProperty] AS similarity

ORDER BY similarity DESC

LIMIT $limit

1. Overlap Similarity

:param limit => 50;

:param weightProperty => null;

:param config => {

concurrency: 8,

writeProperty: 'score',

writeRelationshipType: 'SIMILAR\_OVERLAP',

similarityCutoff: 0.1,

degreeCutoff: 0,

write: true

};

MATCH (item)-->(category)

WITH {item:id(item), categories: collect(distinct id(category))} as userData

WITH collect(userData) as data

CALL algo.similarity.overlap(data, $config)

YIELD nodes, similarityPairs, write, writeRelationshipType, writeProperty, min, max, mean, stdDev, p25, p50, p75, p90, p95, p99, p999, p100

RETURN nodes, similarityPairs, write, writeRelationshipType, writeProperty, min, max, mean, p95

MATCH (from)-[rel:`SIMILAR\_OVERLAP`]-(to)

WHERE not (rel[$config.writeProperty] is null)

RETURN from, to, rel[$config.writeProperty] AS similarity

ORDER BY similarity DESC

LIMIT $limit

* Community Detection Algorithm

1. Triangle Count Algorithm

:param label => null;

:param relationshipType => null;

:param limit => 50;

:param config => {

concurrency: 8,

direction: 'Both',

writeProperty: 'trianglesCount',

clusteringCoefficientProperty: 'clusteringCoefficient'

};

CALL algo.triangleCount($label, $relationshipType, $config)

MATCH (node)

WHERE not(node[$config.writeProperty] is null) AND not(node[$config.clusteringCoefficientProperty] is null)

RETURN node, node[$config.writeProperty] AS triangles, node[$config.clusteringCoefficientProperty] AS coefficient

ORDER BY triangles DESC

LIMIT $limit

* Centrality Algorithm

1. Degree Centrality Algorithm

:param label => null;

:param relationshipType => null;

:param limit => 50;

:param config => {

concurrency: 8,

direction: 'Both',

weightProperty: null,

defaultValue: 1,

writeProperty: 'degree'

};

CALL algo.degree($label, $relationshipType, $config)

MATCH (node)

WHERE not(node[$config.writeProperty] is null)

RETURN node, node[$config.writeProperty] AS score

ORDER BY score DESC

LIMIT $limit

1. BETWEENNESS CENTRALITY

:param label => null;

:param relationshipType => null;

:param limit => 50;

:param config => {

concurrency: 8,

direction: 'Both',

writeProperty: 'betweenness'

};

CALL algo.betweenness($label, $relationshipType, $config)

MATCH (node)

WHERE not(node[$config.writeProperty] is null)

RETURN node, node[$config.writeProperty] AS score

ORDER BY score DESC

LIMIT $limit

1. PageRank Centrality

:param label => null;

:param relationshipType => null;

:param limit => 50;

:param config => {

concurrency: 8,

direction: 'Both',

weightProperty: null,

defaultValue: 1,

dampingFactor: 0.85,

iterations: 20,

writeProperty: 'pagerank'

};

CALL algo.pageRank($label, $relationshipType, $config)

MATCH (node)

WHERE not(node[$config.writeProperty] is null)

RETURN node, node[$config.writeProperty] AS score

ORDER BY score DESC

LIMIT $limit

1. Approx. Betweenness

:param label => null;

:param relationshipType => null;

:param limit => 50;

:param config => {

concurrency: 8,

direction: 'Both',

maxDepth: null,

probability: null,

strategy: 'random',

writeProperty: 'approxBetweenness'

};

CALL algo.betweenness.sampled($label, $relationshipType, $config)

MATCH (node)

WHERE not(node[$config.writeProperty] is null)

RETURN node, node[$config.writeProperty] AS score

ORDER BY score DESC

LIMIT $limit

1. Closeness Centrality

:param label => null;

:param relationshipType => null;

:param limit => 50;

:param config => {

concurrency: 8,

direction: 'Both',

writeProperty: 'closeness'

};

CALL algo.closeness($label, $relationshipType, $config)

MATCH (node)

WHERE not(node[$config.writeProperty] is null)

RETURN node, node[$config.writeProperty] AS score

ORDER BY score DESC

LIMIT $limit

1. Harmonic

:param label => null;

:param relationshipType => null;

:param limit => 50;

:param config => {

concurrency: 8,

direction: 'Both',

writeProperty: 'harmonic'

};

CALL algo.closeness.harmonic($label, $relationshipType, $config)

MATCH (node)

WHERE not(node[$config.writeProperty] is null)

RETURN node, node[$config.writeProperty] AS score

ORDER BY score DESC

LIMIT $limit

* Code for Obtaining Scores

MATCH (c:Company)

RETURN c.name,c.betweenness,c.approxBetweenness,c.closeness,c.pagerank,c.clusteringCoefficient,c.harmonic,c.degree,c.trianglesCount