

**TASK OVERVIEW:**

* Load the data from CSV
* Pre-process ground-truth labels
* Graph projection for Embedding algorithm
* Run embedding algorithm
* Graph projection for Machine learning model including the train / test graph
* Construct Node classification pipeline
* Train the Nodeclassification model and obtain predictions
* Compare predictions with test graph and obtain accuracy

**Step 1 - ensure no timeout:**

CALL dbms.setConfigValue('dbms.transaction.timeout','0');

**Step 2 - Load nodes first**

LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/Kristof-Neys/Neo4j-Cora/main/node\_list.csv" AS row

WITH toInteger(row.id) AS paperId, row.subject AS subject, row.features AS features

MERGE (p:Paper {paper\_Id: paperId})

SET p.subject = subject, p.features = apoc.convert.fromJsonList(features)

RETURN count(p)

Time: 16496 ms.

**Step 3 - Load edges:**

LOAD CSV WITH HEADERS FROM "https://raw.githubusercontent.com/Kristof-Neys/Neo4j-Cora/main/edge\_list.csv" AS row

MATCH(source: Paper {paper\_Id: toInteger(row.source)})

MATCH(target: Paper {paper\_Id: toInteger(row.target)})

MERGE (source)-[r:CITES]->(target)

Time: 34786 ms.

**Step 4 - have a check what we have in the database**

MATCH (n) WHERE EXISTS(n.features) RETURN DISTINCT n.paper\_Id as PaperId, n.subject AS Paper\_Subject, n.features AS features LIMIT 5

**Step 5 - Convert subjects (strings) to numerical**

MATCH (p:Paper)

WITH collect(DISTINCT p.subject) as listSubjects

WITH listSubjects, size(listSubjects) AS sizeListSubjects

WITH listSubjects, range(1, sizeListSubjects) AS rangeSubjects

WITH apoc.map.fromLists(listSubjects, rangeSubjects) AS mapSubjects

MATCH (p:Paper)

SET p.subjectClass = mapSubjects[p.subject];

Time: 1002 ms.

**Time to start the GDS Engines**

We will be doing two different projections:

Projection to run Fast RP on

Projection with the embeddings included

**Step 6 – projection to run embedding**

CALL gds.graph.project(

'cora-graph',

{

Paper: {

label: 'Paper',

properties: {

subjectClass:{property:'subjectClass',defaultValue:0},

features:{property:'features',defaultValue: [x in range(1,1433) | 0]}

}

}

}, {

CITES: {

type: 'CITES',

orientation: 'UNDIRECTED',

aggregation: 'SINGLE'

}

})

Time: 6703 ms.

**Step 7 - run embedding algorithm and obtain vector representations of the nodes**

CALL gds.fastRP.write('cora-graph',{

relationshipTypes:['CITES'],

featureProperties: ['features'],

embeddingDimension: 128,

iterationWeights: [0, 0, 1.0, 1.0],

normalizationStrength:0.05,

writeProperty: 'fastRP\_Extended\_Embedding'

})

Time: 4677 ms.

**Computing the Test / Train split**

**Step 8 - Create Train graph**

MATCH (p:Paper) WITH collect(ID(p)) as papers

WITH apoc.coll.randomItems(papers, toInteger(0.8 \* size(papers))) as trainPapers

UNWIND trainPapers as trainPaper

MATCH (p:Paper) WHERE id(p) = trainPaper

SET p.is\_train\_data = 1

Time: 312 ms.

**Step 9 - create Test graph**

MATCH (p:Paper)

WHERE p.is\_train\_data IS NULL

SET p.is\_train\_data = 0

Time: 68 ms.

**Step 10 - Second projection - projection to run ML model on**

CALL gds.graph.project(

'cora-graph-rp2',

{

Paper: {

label: 'Paper',

properties: {

subjectClass:{property:'subjectClass',defaultValue:0},

is\_train\_data:{property:'is\_train\_data',defaultValue:0},

fastRP\_Extended\_Embedding:{property:'fastRP\_Extended\_Embedding'}

}

}

}, {

CITES: {

type: 'CITES',

orientation: 'UNDIRECTED',

aggregation: 'SINGLE'

}

})

Time: 820 ms.

**Step 11 - Creating two subgraphs from the main graph in memory: a train and test graph - train:**

CALL gds.beta.graph.project.subgraph('cora\_graph\_rp-train', 'cora-graph-rp2', 'n:Paper AND n.is\_train\_data = 1', '\*')

YIELD graphName, fromGraphName, nodeCount, relationshipCount

Time: 1124 ms.

**Step 11B - create Test graph for ML model prediction**

CALL gds.beta.graph.project.subgraph('cora\_graph\_rp-test', 'cora-graph-rp2', 'n:Paper AND n.is\_train\_data = 0', '\*')

YIELD graphName, fromGraphName, nodeCount, relationshipCount

Time: 493 ms.

**Let’s check what we have projected in memory**

CALL gds.graph.list()

**Step 12 - Create pipeline for ML training and prediction**

CALL gds.beta.pipeline.nodeClassification.create('pipe')

Time: 9ms

**Step 13 - Add features**

CALL gds.beta.pipeline.nodeClassification.selectFeatures('pipe', 'fastRP\_Extended\_Embedding')

Time: 69 ms.

**Step 14 - add which model you want to use**

CALL gds.beta.pipeline.nodeClassification.addLogisticRegression('pipe', {maxEpochs: 500, penalty: 0.001})

YIELD parameterSpace

RETURN parameterSpace.LogisticRegression AS logisticRegressionSpace

Time: 14ms

**Step 15 - Train the model:**

CALL gds.beta.pipeline.nodeClassification.train('cora\_graph\_rp-train', {

pipeline: 'pipe',

nodeLabels: ['Paper'],

modelName: 'Cora\_model\_FRP',

targetProperty: 'subjectClass',

randomSeed: 42,

metrics: ['F1\_WEIGHTED','ACCURACY']

})

Train: 133948 ms = 2min15

**Step 16 - make predictions on the Test graph:**

CALL gds.beta.pipeline.nodeClassification.predict.mutate('cora\_graph\_rp-test', {

nodeLabels: ['Paper'],

modelName: 'Cora\_model\_FRP',

mutateProperty: 'predicted\_Cora\_FRP'

})

Time: 54ms

**Step 17: write predictions from in memory to disk**

CALL gds.graph.writeNodeProperties(

'cora\_graph\_rp-test',

['predicted\_Cora\_FRP'],

['Paper']

)

Time: 343 ms.

**Step 18 - Check the accuracy of the prediction:**

MATCH (p:Paper)

WHERE p.is\_train\_data = 0

WITH count(p) AS nbPapers

MATCH (p:Paper)

WHERE p.is\_train\_data = 0

AND p.subjectClass = p.predicted\_Cora\_FRP

RETURN toFloat(count(p)) / nbPapers AS ratio

Time: 75 ms