// This was written for Neo4j Community 3.4.12

// Definitely compatible with 3.4.13 + 3.4.14

// Create Schema

CALL ga.nlp.createSchema();

// Set Language to English

CALL ga.nlp.config.setDefaultLanguage('en');

// Set Pipeline

CALL ga.nlp.processor.addPipeline({textProcessor: 'com.graphaware.nlp.processor.stanford.StanfordTextProcessor', name: 'Processor', processingSteps: {tokenize: true, ner: true, dependency: false}, stopWords: '+,result, all, during',

threadNumber: 20});

// Call Pipeline

CALL ga.nlp.processor.pipeline.default('Processor');

////////////--------------------------------

//Load data

// Load JSON Abstract Data:

CALL apoc.load.json("file:///finalabstracts.jsonl") YIELD value AS row

CREATE (n:Abstract {id: row.id, text:row.text});

// Load Keys and Ontology

LOAD CSV WITH HEADERS FROM "file:///ont4.csv" AS row

CREATE (n:Keys{class: row.class, value:row.value});

// Indexes Abstracts

CREATE INDEX ON :Abstract(id);

// Indexes Keys

CREATE INDEX ON :Keys(value);

// Creates /annotates

MATCH (a:Abstract)

CALL ga.nlp.annotate({text: a.text, id: id(a)})

YIELD result

MERGE (a)-[:CONTAINS\_TEXT]->(result)

RETURN count(result);

//------------------------------------------------------------------------------------------

//  **Creates NE hierarchy-Co-occrrence**

MATCH (a:AnnotatedText)-[:CONTAINS\_SENTENCE]->(s:Sentence)-[:SENTENCE\_TAG\_OCCURRENCE]-

>(to:TagOccurrence)-[:TAG\_OCCURRENCE\_TAG]->(tag)

WHERE tag:NER\_O

WITH a, to, tag

ORDER BY s.id, to.startPosition

WITH a, collect(tag) as tags

UNWIND range(0, size(tags) - 2) as i

WITH a, tags[i] as tag1, tags[i+1] as tag2 WHERE tag1 <> tag2

MERGE (tag1)-[r:OCCURS\_WITH]-(tag2)

ON CREATE SET r.freq = 1

ON MATCH SET r.freq = r.freq + 1;

**// Enrichment**

MATCH (n:Tag)

CALL ga.nlp.enrich.concept({enricher: 'conceptnet5', tag: n, depth:1, admittedRelationships:['IsA','PartOf','HasA','HasSubevent', 'ReceivesAction', 'DistinctFrom','DerivedFrom','DefinedAs','UsedFor', 'Synonym', 'CapableOf', 'Causes', 'CreatedBy', 'ObstructedBy', 'HasContext', 'MadeOf'], relationshipType:['IsA', 'PartOf','HasA','HasSubevent','ReceivesAction','DistinctFrom','DerivedFrom','DefinedAs', 'UsedFor', 'Synonym', 'CapableOf', 'Causes', 'CreatedBy', 'ObstructedBy', 'HasContext', 'MadeOf']})

YIELD result

SET n:ConceptProcessed

RETURN count(\*);

MATCH (n:Tag)

CALL ga.nlp.enrich.concept({enricher: 'microsoft', tag: n, depth:1, checkLanguage:false})

YIELD result

SET n:ConceptProcessed

RETURN count(\*);

//-----------------------------------------------------------------

**//Keyword Extraction**

MATCH (a:AnnotatedText)

CALL ga.nlp.ml.textRank({annotatedText: a, useDependencies: true})

YIELD result RETURN result;

CREATE INDEX ON :Keyword(value);

CALL ga.nlp.ml.textRank.postprocess({keywordLabel: "Keyword", method: "subgroups"})

YIELD result

RETURN result

CALL apoc.periodic.iterate(

'MATCH (n:AnnotatedText) RETURN n',

'CALL ga.nlp.ml.textRank.postprocess({annotatedText: n, method:"subgroups"}) YIELD result RETURN count(n)',

{batchSize: 100, iterateList:false}

);

//----------------------------------------------------------------

// **Connects Abstracts and Sentences**

MATCH(a:Abstract),(s:Sentence)

WHERE a.text CONTAINS s.text

MERGE (s)-[r:COMES\_FROM]->(a) RETURN count(a);

// Connects Key Occurance in Sentences

MATCH(k:Keys),(s:Sentence)

WHERE s.text CONTAINS k.value

MERGE(s)-[:HAS\_KEY]->(k);

//------------------------------------------------------------------------

// Creates first sentence property in abstract

MATCH (s:Sentence)-[r:COMES\_FROM]->(a:Abstract)

WHERE s.sentenceNumber=0

SET r.starts='True';

MATCH (s:Sentence)-[r:COMES\_FROM]->(a:Abstract)

WHERE NOT s.sentenceNumber=0

SET r.starts='False';

// Clean up sentence "3)" and add to begining of next sentence

MATCH (s1:Sentence)-[:NEXT\_SENTENCE]->(s2:Sentence)-[:NEXT\_SENTENCE]->(s3:Sentence)

WHERE s1.id='340\_5' AND s2.id='340\_6' AND s3.id='340\_7'

MERGE(s1)-[:NEXT\_SENTENCE]->(s3)

MATCH (s1:Sentence { id: '340\_6' })

DETACH DELETE s1;

MATCH(s:Sentence{id:'340\_7'})

SET s.text='3) cocs were matured in serum-free medium containing 1 mg/ml polyvinyl alcohol and 0, 10, 100, or 1000 ng/ml leptin (l0, l10, l100, and l1000, respectively), or in medium supplemented with 10% fetal calf serum (fcs) as a positive control.';

MATCH(s:Sentence{id:'340\_7'}) SET: s.id:'340\_6',s.sentenceNumber:'6';

MATCH(s:Sentence{id:'340\_8'}) SET: {s.id:'340\_7',s.sentenceNumber:'7'}; MATCH(s:Sentence{id:'340\_9'}) SET:{s.id:'340\_8',s.sentenceNumber:'8'}; MATCH(s:Sentence{id:'340\_10'}) SET:{s.id:'340\_9',s.sentenceNumber:'9'}; MATCH(s:Sentence{id:'340\_11'}) SET: {s.id:'340\_10',s.sentenceNumber:'10'}; MATCH(s:Sentence{id:'340\_12'}) SET: {s.id:'340\_11',s.sentenceNumber:'11'}; MATCH(s:Sentence{id:'340\_13'}) SET: {s.id:'340\_12',s.sentenceNumber:'12'}; MATCH(s:Sentence{id:'340\_14'}) SET: {s.id:'340\_13',s.sentenceNumber:'13'};

//--------------------------------------------------------------------

CREATE CONSTRAINT ON (m:TagOccurence) ASSERT m.value IS UNIQUE;

// Delete Tag Occurrences

MATCH (m:TagOccurrence)

DETACH DELETE m;

// Check labels (had 1,146 blank nodes during one build

MATCH(n)

RETURN labels(n)

// If there, can run

// MATCH (n)-[r:CONTAINS\_TEXT]->()

// WHERE NOT labels(n)=’AnnotatedText’

// DETACH DELETE n

// **ReLabel to Ontology Classes**

MATCH(n),(k:Keys{class:'mdaDraAdverseEvents'})

WITH n, k.value AS ont

WHERE ont=n.value

SET n:MDAdraAdverseEvents;

MATCH(n),(k:Keys{class:'adverseEvent'})

WITH n, k.value AS ont

WHERE ont=n.value

SET n:AdverseEvent;

MATCH(n),(k:Keys{class:'drug'})

WITH n, k.value AS ont

WHERE ont=n.value

SET n:Drug;

MATCH(n),(k:Keys{class:'drugClass'})

WITH n, k.value AS ont

WHERE ont=n.value

SET n:DrugClass;

MATCH(n),(k:Keys{class:'geneOntology'})

WITH n, k.value AS ont

WHERE ont=n.value

SET n:GeneOntology;

MATCH(n),(k:Keys{class:'geneProtein'})

WITH n, k.value AS ont

WHERE n.value=ont

SET n:GeneProtein;

MATCH(n),(k:Keys{class:'humanPhenotype'})

WITH n, k.value AS ont

WHERE n.value=ont

SET n:HumanPhenotype;

MATCH(n),(k:Keys{class:'indication'})

WITH n, k.value AS ont

WHERE n.value=ont

SET n:Indication;

// Create NashNafld label

MATCH(n) WHERE ID(n) IN [“1816”,“1840”,”1841”,”1842”,”1844”,”1845”,“8732”,”8733”,”8836”,”8837”,”8879”,”9419”,”9452”,”9874”,”10031”,”10032”,”10091”,”10919”,”10959”,”13912”,”26089”,”295490”,”296137”]

SET n:NashNafld;



//--------------------------------------------------

//  **Create relationships from Conceptnet 5 relations**

// To move back to r:IS\_RELATED\_TO, swap MATCH and merge clauses

// ie-

// MATCH (n)-[r:CAUSES]-(m)

// WITH n,m,r.source AS sour, r.weight AS wei

// WHERE sour='CONCEPT\_NET\_5'

// MERGE (n)-[r2:IS\_RELATED\_TO{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

// RETURN count(r);

MATCH(n)-[r:IS\_RELATED\_TO{type:'Causes'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:CAUSES{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'PartOf'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:PART\_OF{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'HasA'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:HAS\_A{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'HasSubevent'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:HAS\_SUBEVENT{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'DistinctFrom'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:DISTINCT\_FROM{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'DerivedFrom'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:DERIVED\_FROM{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'DefinedAs'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:DEFINED\_AS{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'Synonym'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:SYNONYM{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'IsA'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:IS\_A{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'CapableOf'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:CAPABLE\_OF{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'CreatedBy'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:CREATED\_BY{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'MadeOf'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:MADE\_OF{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'UsedFor'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:USED\_FOR{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'ReceivesAction'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:RECEIVES\_ACTION{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'HasContext'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:HAS\_CONTEXT{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

MATCH(n)-[r:IS\_RELATED\_TO{type:'ObstructedBy'}]->(m)

WITH n,m,r.source AS sour, r.weight AS wei

WHERE sour='CONCEPT\_NET\_5'

MERGE (n)-[r2:OBSTRUCTED\_BY{source:'CONCEPT\_NET\_5', weight:wei}]-(m)

RETURN count(r2);

**// Delete the CONCEPTNET5 is related to**

MATCH (a)-[rel:IS\_RELATED\_TO{source:'CONCEPT\_NET\_5'}]->(m)

WHERE NOT rel.source='MICROSOFT\_CONCEPT'

DELETE rel;

**// See where we are after all of this**

MATCH(n)

RETURN count(n), labels(n);

MATCH()-[r]-()

RETURN count(r), type(r);

// Begin moving properties from annotated text to abstracts

MATCH (a:Abstract)<-[:COMES\_FROM]-(s:Sentence),(w:AnnotatedText)-[:CONTAINS\_SENTENCE]->(s)

WHERE s.sentenceNumber=0

SET a.anntextid=w.id;

MATCH (a:Abstract),(n:AnnotatedText)

WHERE a.anntextid=n.id

SET a.numTerms=n.numTerms;

// recreate keyword extraction to abstracts

MATCH (a:Abstract),(k:Keyword)-[r:DESCRIBES]->(an:AnnotatedText)

WHERE a.anntextid=an.id

MERGE(k)-[d:EXTRACTED\_FROM{relevance:r.relevance}]-(a)

RETURN count(d);

// Delete Annotated text

MATCH (n:AnnotatedText)

DETACH DELETE n;

// Gained another rogue node

//MATCH(n) REMOVE n.startPosition

//CREATE(n:TagOccurence {value:'l'})

//MATCH(n) WHERE n.value='l' DETACH DELETE n

DROP INDEX ON :Keys(value)

DROP CONSTRAINT ON ( tagoccurence:TagOccurence ) ASSERT tagoccurence.value IS UNIQUE

// REMOVE KEYS label

MATCH(k:Keys:AdverseEvent)

REMOVE k:Keys

RETURN labels(k);

MATCH(k:Keys:GeneProtein)

REMOVE k:Keys

RETURN labels(k);

MATCH(k:Keys:Drug)

REMOVE k:Keys

RETURN labels(k);

MATCH(k:Keys:DrugClass)

REMOVE k:Keys

RETURN labels(k);

MATCH(k:Keys:GeneOntology)

REMOVE k:Keys

RETURN labels(k);

MATCH(k:Keys:DrugClass)

REMOVE k:Keys

RETURN labels(k);

MATCH(k:Keys:HumanPhenotype)

REMOVE k:Keys

RETURN labels(k);

MATCH(k:Keys:Indication)

REMOVE k:Keys

RETURN labels(k);

MATCH(k:Keys:MDAdraAdverseEvents)

REMOVE k:Keys

RETURN labels(k);

// Can delete all nodes with keys label-should get no matches

MATCH(n:Keys)

DETACH DELETE n;

// Can now see Keys under ontology class with HAS\_KEY relationship we set

MATCH p=()-[r:HAS\_KEY]->()

RETURN p LIMIT 25;

// Queries

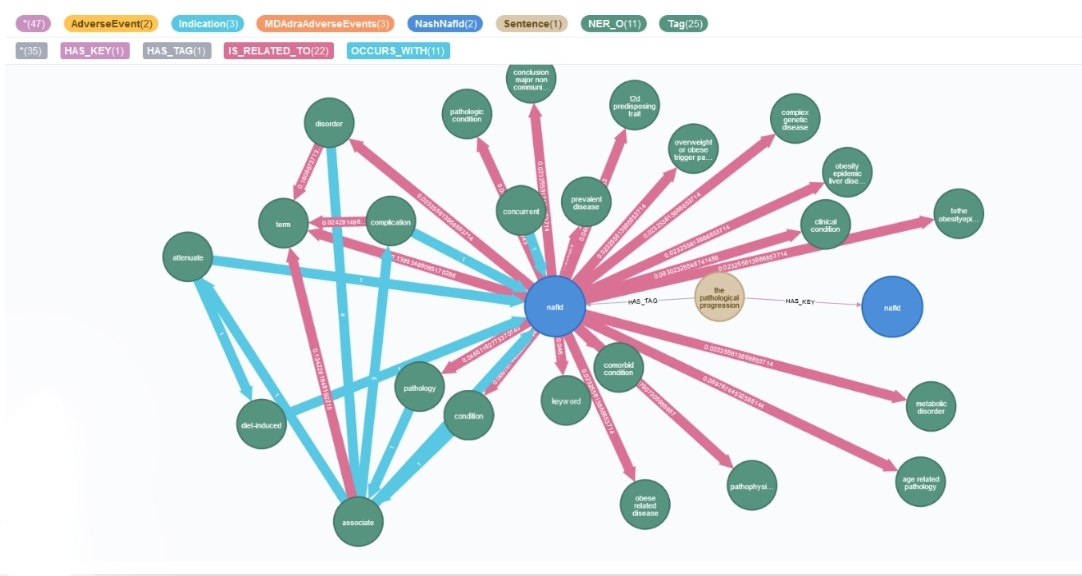
// Creates path from NAFLD nodes => 2 hops => “nonalcoholic steatohepatitis”

//Hidden hop are sentence nodes, and sentence => keys is a concurrent query statement

//Information returned includes tags that co-occur, and a related to, NAFLD and nonalcoholic steatohepatitis, with respective frequencies and weights

MATCH p=({value:'nafld'})-[]-(m)-[]-({value:'nonalcoholic steatohepatitis'}),(m)-[r:HAS\_KEY]-(j)

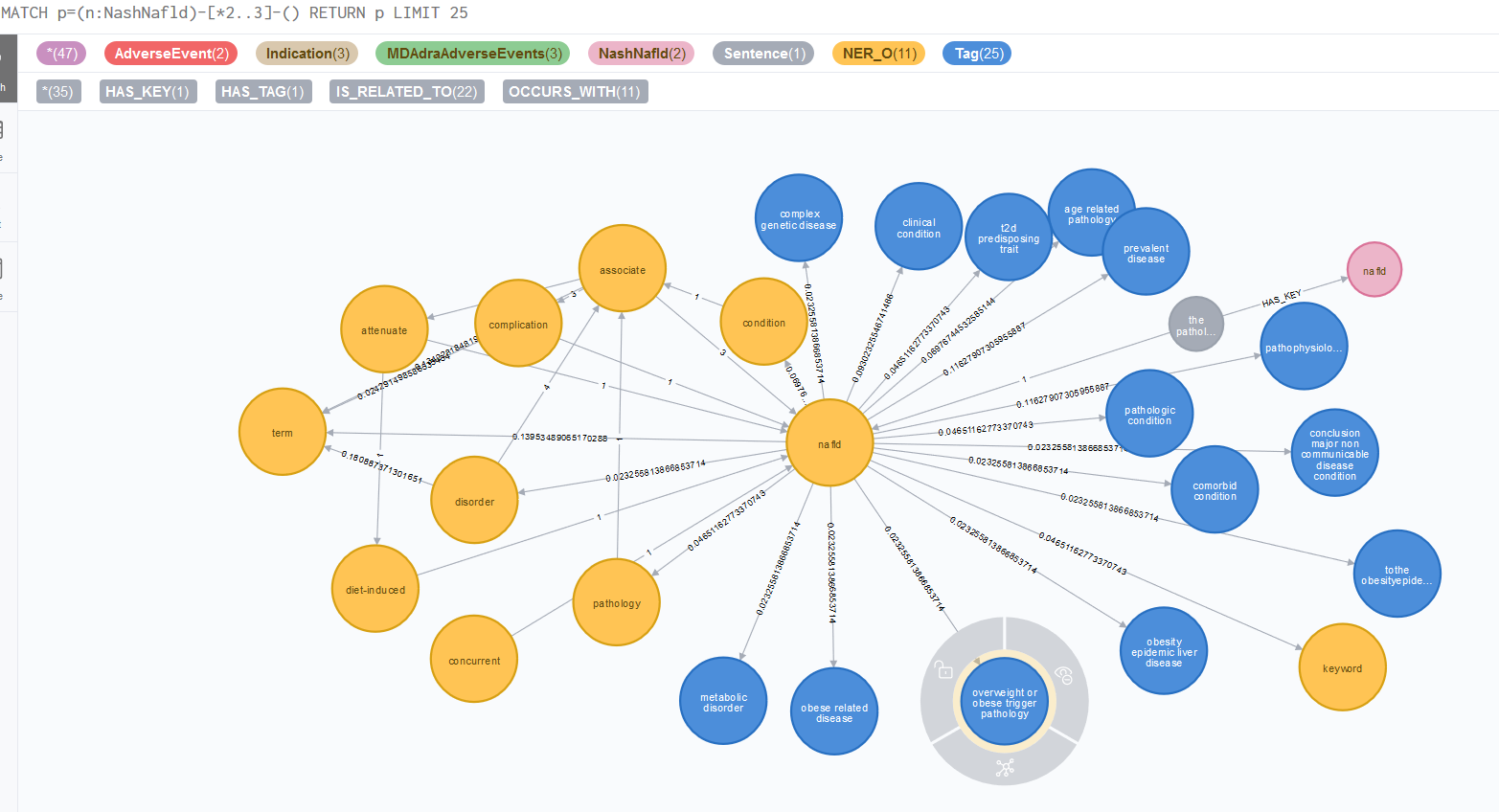
RETURN p,m,r,j LIMIT 25



//

MATCH p=(n:NashNafld)-[\*2..3]-()

RETURN p LIMIT 25



//

// Keywords Extracted From Abstracts, Along With The Sentences

// In That Abstract….AND THE Relationships of Those Sentences to ‘nash

MATCH(n)-[r:EXTRACTED\_FROM->(a)<-[w:COMES\_FROM]-(s)-[q]->(x)

WHERE x.value= ’nash’ AND r.relevance > .01

RETURN n,r,a,w,s,q,x LIMIT 200;

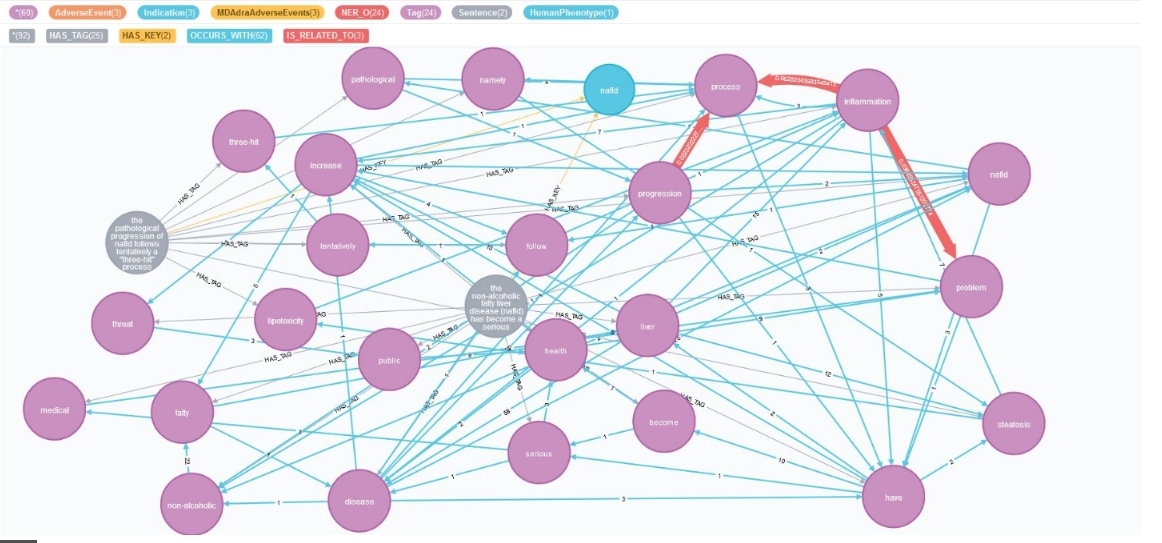


//

// Observe Community Members of community 12005 created earlier that are two hops(of any relationships) away from NAFLD nodes

MATCH p=(n)-{\*2]-(value:'nafld’})

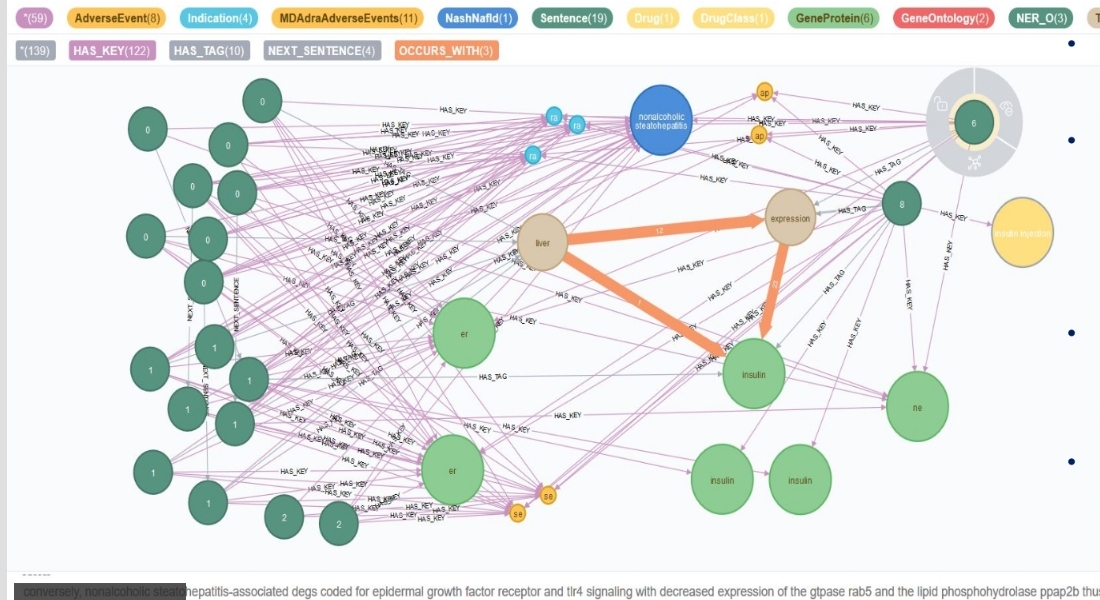
WHERE n.community=12005

RETURN p LIMIT 25

// All shortest Paths

MATCH(n:NashNafld),(m:DrugClass) WHERE ID(n)=1841 AND ID(m)=2543

RETURN allShortestPaths((n)-[\*]-(m)) LIMIT 30



// Label Propagation-Graph Algorithms Plugin. Community Detection

CALL algo.labelPropagation.stream("Drug", "OCCURS\_WITH",

{ iterations: 10 })

YIELD nodeId, label

RETURN label,

collect(algo.getNodeById(nodeId).id) AS lp

ORDER BY size(lp) DESC;

// End of initial project

**// Another way to import data, with same results, and also offers a // // suggested methodology for adding new data to database via batch**

**// Green same code; blue new code**

// Create Schema

CALL ga.nlp.createSchema();

// Set Language to English

CALL ga.nlp.config.setDefaultLanguage('en');

// Set Pipeline

CALL ga.nlp.processor.addPipeline({textProcessor: 'com.graphaware.nlp.processor.stanford.StanfordTextProcessor', name: 'Processor', processingSteps: {tokenize: true, ner: true, dependency: false}, stopWords: '+,result, all, during',

threadNumber: 20});

//Call Pipeline

CALL ga.nlp.processor.pipeline.default('Processor');

// Load data-- do not need to delete start and end on entities

// Load JSON Abstract Data

CALL apoc.load.json("file:///abstracts.jsonl") YIELD value AS row

CREATE (n:Abstract {id: toInteger(row.id), text:row.text});

// Use APOC procedure to makes all property values lower case for all nodes abstract text is including

MATCH (n)

WITH n, [x IN keys(n)

WHERE n[x] =~ '.\*'

] as props

UNWIND props as p

CALL apoc.create.setProperty(n, p, toLower(n[p])) YIELD node

RETURN node

// Load Keys and Ontology ‘definitions.csv’ can be substituted

LOAD CSV WITH HEADERS FROM "file:///fullontdef.csv" AS row

CREATE (n:Definitions{code: row.code, value:row.value});

// Load Keys and Ontology -we change from varchar/text to Integers during import with

// Cypher here instead of in Excel/Python

LOAD CSV WITH HEADERS FROM "file:///fullkeyont.csv" AS row

CREATE (n:Keys{abstractId: toInteger(row.abstractId), code: row.code, value: row.value, start: toInteger(row.start), end: toInteger(row.end) });

// check annotation/tag occurrences vs termite start and end

(MATCH(a:Keys),(n:TagOccurrences)

WHERE a.start=n.StartPosition AND a.end=a.EndPosition AND a.value=m.value

RETURN a,n

//Create Indices

CREATE INDEX ON :Abstract(id);

// Indexes Keys

CREATE INDEX ON :Keys(value);

// Creates /annotates Abstracts-we take properties from Created AnnotatedText nodes

// and set as abstract properties immediately here

MATCH (a:Abstract)

CALL ga.nlp.annotate({text: a.text, id: id(a)})

YIELD result

SET a.atid=result.id, a.numTerms=result.numTerms;

// Create length of characters property for each abstract to go along with

MATCH(a:Abstract) SET a.lenChar=length(a.text);

// This apoc procedure creates label for a node class from properties of node in another label class //in one procedure, eliminating 8(would be 40 here) previous statements . Deleting that property could //be set at end like we did in code for other model but we cannot confirm that no information will be //lost using the apoc procedure. Simpler and faster-property transfer needs to be fixed

//MATCH (n:Keys)

//CALL apoc.create.addLabels([ id(n) ], [ n.code ])

//YIELD node

//RETURN node

//  **Creates NE hierarchy-co-occurences**

MATCH (a:AnnotatedText)-[:CONTAINS\_SENTENCE]->(s:Sentence)-[:SENTENCE\_TAG\_OCCURRENCE]-

>(to:TagOccurrence)-[:TAG\_OCCURRENCE\_TAG]->(tag)

WHERE tag:NER\_O

WITH a, to, tag

ORDER BY s.id, to.startPosition

WITH a, collect(tag) as tags

UNWIND range(0, size(tags) - 2) as i

WITH a, tags[i] as tag1, tags[i+1] as tag2 WHERE tag1 <> tag2

MERGE (tag1)-[r:OCCURS\_WITH]-(tag2)

ON CREATE SET r.freq = 1

ON MATCH SET r.freq = r.freq + 1;

**// Enrichment- ConceptProcessed nodes are not created, only Tags //enriched.**

MATCH (n:Tag)

CALL ga.nlp.enrich.concept({enricher: 'conceptnet5', tag: n, depth:1, admittedRelationships:['IsA','PartOf','HasA','HasSubevent', 'ReceivesAction', 'DistinctFrom','DerivedFrom','DefinedAs','UsedFor', 'Synonym', 'CapableOf', 'Causes', 'CreatedBy', 'ObstructedBy', 'HasContext', 'MadeOf'], relationshipType:['IsA', 'PartOf','HasA','HasSubevent','ReceivesAction','DistinctFrom','DerivedFrom','DefinedAs', 'UsedFor', 'Synonym', 'CapableOf', 'Causes', 'CreatedBy', 'ObstructedBy', 'HasContext', 'MadeOf']})

RETURN count(\*);

MATCH(n:Tag) CALL ga.nlp.enrich.concept({enricher: 'microsoft', tag: n, depth:1, checkLanguage:false}) YIELD result

RETURN count(\*);

**// proceed as before**

**// Update graph with new abstracts**

**// The idea here is that since annotated text nodes were deleted after being moved to abstracts, label**

**// annotated text will now include all new information. Since [:CONTAINS\_TEXT] relationships were**

**// deleted to, these relationships will also be new and easily identifiable.**

//New Batch

MATCH (a:Abstract{id:12345678})

CALL ga.nlp.annotate({text: a.text, id: id(a)})

YIELD result

MERGE (a)-[:CONTAINS\_TEXT]->(result)

RETURN count(result);

// Creates /annotates

MATCH (a:Abstract)

CALL ga.nlp.annotate({text: a.text, id: id(a)})

YIELD result

SET a.atid=result.id, a.numTerms=result.numTerms;

//  **Creates NE hierarchy-co-occurences**

**// Since tag occurrences were deleted, only new co-occurences will be created**

MATCH (a:AnnotatedText)-[:CONTAINS\_SENTENCE]->(s:Sentence)-[:SENTENCE\_TAG\_OCCURRENCE]-

>(to:TagOccurrence)-[:TAG\_OCCURRENCE\_TAG]->(tag)

WHERE tag:NER\_O

WITH a, to, tag

ORDER BY s.id, to.startPosition

WITH a, collect(tag) as tags

UNWIND range(0, size(tags) - 2) as i

WITH a, tags[i] as tag1, tags[i+1] as tag2 WHERE tag1 <> tag2

MERGE (tag1)-[r:OCCURS\_WITH]-(tag2)

ON CREATE SET r.freq = 1

ON MATCH SET r.freq = r.freq + 1;

// For enrichment, only new tags will be related to annotated text, so create a relationship including annotated text for tags to enrich…so don’t have cost of enriching entire db again

MATCH (n:Tag)-[\*2]-(a:AnnotatedText)

CALL ga.nlp.enrich.concept({enricher: 'conceptnet5', tag: n, depth:1, admittedRelationships:['IsA','PartOf','HasA','HasSubevent', 'ReceivesAction', 'DistinctFrom','DerivedFrom','DefinedAs','UsedFor', 'Synonym', 'CapableOf', 'Causes', 'CreatedBy', 'ObstructedBy', 'HasContext', 'MadeOf'], relationshipType:['IsA', 'PartOf','HasA','HasSubevent','ReceivesAction','DistinctFrom','DerivedFrom','DefinedAs', 'UsedFor', 'Synonym', 'CapableOf', 'Causes', 'CreatedBy', 'ObstructedBy', 'HasContext', 'MadeOf']})

YIELD result

SET n:ConceptProcessed

RETURN count(\*);

// same

MATCH (n:Tag)-[\*2]-(a:AnnotatedText)

CALL ga.nlp.enrich.concept({enricher: 'microsoft', tag: n, depth:1, checkLanguage:false})

YIELD result

SET n:ConceptProcessed

RETURN count(\*);

**// Can do the same throughout the keyword extraction procedure-I ran out of time to send this**

**//Keyword Extraction**

MATCH (a:AnnotatedText)

CALL ga.nlp.ml.textRank({annotatedText: a, useDependencies: true})

YIELD result RETURN result;

CREATE INDEX ON :Keyword(value);

CALL ga.nlp.ml.textRank.postprocess({keywordLabel: "Keyword", method: "subgroups"})

YIELD result

RETURN result

CALL apoc.periodic.iterate(

'MATCH (n:AnnotatedText) RETURN n',

'CALL ga.nlp.ml.textRank.postprocess({annotatedText: n, method:"subgroups"}) YIELD result RETURN count(n)',

{batchSize: 100, iterateList:false}

);

//----------------------------------------------------------------

// **Connects Abstracts and Sentences**

MATCH(a:Abstract),(s:Sentence)

WHERE a.text CONTAINS s.text

MERGE (s)-[r:COMES\_FROM]->(a) RETURN count(a);

// Connects Key Occurrences in Sentences

MATCH(k:Keys),(s:Sentence)

WHERE s.text CONTAINS k.value

MERGE(s)-[:HAS\_KEY]->(k);

//------------------------------------------------------------------------

// Creates first sentence property in abstract

MATCH (s:Sentence)-[r:COMES\_FROM]->(a:Abstract)

WHERE s.sentenceNumber=0

SET r.starts='True';

MATCH (s:Sentence)-[r:COMES\_FROM]->(a:Abstract)

WHERE NOT s.sentenceNumber=0

SET r.starts='False';

CREATE CONSTRAINT ON (m:TagOccurence) ASSERT m.value IS UNIQUE;

// Delete Tag Occurrences

MATCH (m:TagOccurrence)

DETACH DELETE m;

// Check labels (had 1,146 blank nodes during one build

MATCH(n)

RETURN labels(n)

// If there, can run

// MATCH (n)-[r:CONTAINS\_TEXT]->()

// WHERE NOT labels(n)=’AnnotatedText’

// DETACH DELETE n

// Proceed as before

// Begin moving properties from annotated text to abstracts

MATCH (a:Abstract)<-[:COMES\_FROM]-(s:Sentence),(w:AnnotatedText)-[:CONTAINS\_SENTENCE]->(s)

WHERE s.sentenceNumber=0

SET a.anntextid=w.id;

MATCH (a:Abstract),(n:AnnotatedText)

WHERE a.anntextid=n.id

SET a.numTerms=n.numTerms;

// recreate keyword extraction to abstracts

MATCH (a:Abstract),(k:Keyword)-[r:DESCRIBES]->(an:AnnotatedText)

WHERE a.anntextid=an.id

MERGE(k)-[d:EXTRACTED\_FROM{relevance:r.relevance}]-(a)

RETURN count(d);

// Delete Annotated text

MATCH (n:AnnotatedText)

DETACH DELETE n;

**// Proceed as before**

**// New Batch**

/MATCH (a:Abstract{id:ABCDEFG})

/CALL ga.nlp.annotate({text: a.text, id: id(a)})

//YIELD result

//MERGE (a)-[:CONTAINS\_TEXT]->(result)

//RETURN count(result);

**// I would also like to clean up tags and keys from duplification too**