Property Graph DB

Onur BACAKSIZ & Nils VAN ES OSTOS

March 2024

Introduction

Cypher is a declarative graph query language that allows for expressive and efficient data querying in a property graph. In the following article, a graph database on papers related to Data Science will be created in Neo4j making use of Cypher. In addition, all the data used has been requested from Semantic Scholar (https://www.semanticscholar.org/).

The entire code of the laboratory can be accessed at https://github.com/Nilsvanesostos/SDM-P1.

1 Modeling, Loading, Evolving

1.1 Modeling

In Fig.(1), the schema for the dataset is shown. The schema is a representation of the schema state in [1] with the following modifications:

- The keywords of the papers are represented as a node in order to differentiate better the paper on the same field.

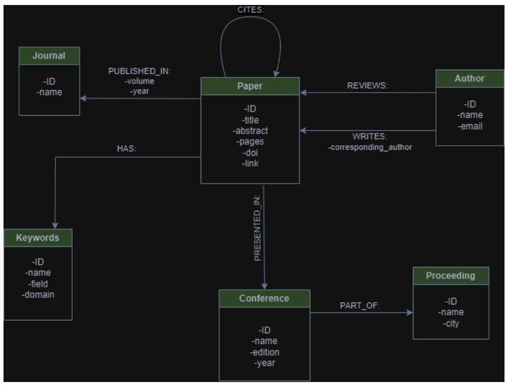


Figure 1: Representation of the schema of the graph.

- We separated the proceeding and the conference in two nodes in order to easily group the conference on the proceedings they are part of.

In addition, an example of how the different nodes and edges relate to each other is shown in the Fig.(2).

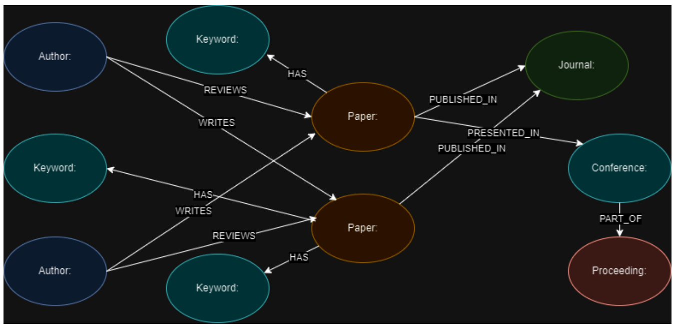


Figure 2: Representation of the nodes and edges of the graph.

1.2 Instantiating/Loading

Once the schema is designed and discussed, it is time for the pre-processing of the data. As mentioned before, the data corresponds to the Semantic Scholar website. Nevertheless, multiple properties can be requested from this data, so some selection and steps must be followed. Details about the pre-process and all the supositions can be found in the file *Preprocess.py*. In addition, the loading of the schema to *Neo4j* can be found in the file *Loading.py*.

1.3 Evolving the graph

For the evolving part, the node *Organization* and the relationship *Affiliated_to* were added in order to connect authors with a certain university or company. The new schema of the graph is represented in Fig.(3). In addition, the attributes *comment* and *acceptanceProbability* were added to the *REVIEWS*'s relationship. More details about the evolving can be found in the file *Evolve.py*.

2 Querying

This section describes using Cypher queries to extract insightful data from the graph, focusing on identifying top papers, core community authors, journal impact factors, and authors' h-indices.

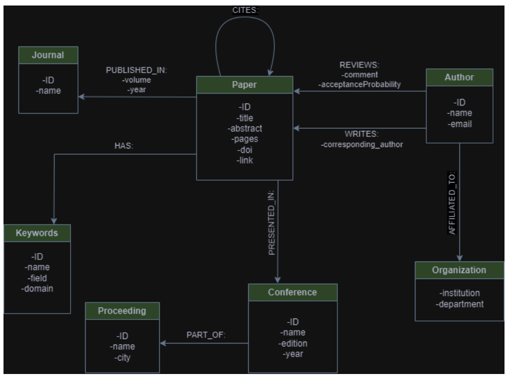


Figure 3: Modification of the schema of the graph.

2.1 Finding top 3 most cited papers of each conference.

Objective: The goal was to pinpoint the three most cited papers within every conference in the dataset, shedding light on influential research and identifying prevailing topics within specific academic circles.
Method: Count citations per paper, sort them within each conference, and select the top three.

```
MATCH (p:paper)-[:PRESENTED_IN]->(c:conference)
OPTIONAL MATCH (p)<-[:CITES]-(citingPaper)
WITH p, c, COUNT(citingPaper) as citations
ORDER BY c.name, citations DESC
WITH c.name as conferenceName, COLLECT(p)[0..3] as topCitedPapers
RETURN conferenceName, [paper IN topCitedPapers | paper.title] AS paperTitles
```

Listing 1: First query

Results Summary: Showcased pivotal papers across conferences, such as a notable paper in the "ACM-SIAM Symposium on Discrete Algorithms" related to structural graph theory. Details are in the Python notebook *PartB.1_BacaksizOstos.ipynb*.

2.2 Finding Each Conference's Community

Objective: Find authors consistently contributing to a conference across editions.

```
MATCH (a:author)-[:WRITES]->(:paper)-[:PRESENTED_IN]->(c:conference)
WITH a.name AS authorName, c.name AS conferenceName, COUNT(DISTINCT c.edition)
    AS editionCount
WHERE editionCount >= 4
RETURN conferenceName, COLLECT(authorName) AS communityAuthors
```

```
ORDER BY conferenceName
```

Listing 2: Second query

Results Summary: Uncovered core author communities, like notable contributors to the "IEEE International Conference on Computer Vision." For more, see *PartB.2_BacaksizOstos.ipynb.*

2.3 Calculating Journal Impact Factors

Objective: Determine the impact factor for journals based on citations to recent articles.

```
WITH 2022 AS currentYear
MATCH (j:journal)<-[:PUBLISHED_IN]-(p:paper)
WHERE p.year IN [currentYear - 1, currentYear - 2]
WITH j, p, currentYear
OPTIONAL MATCH (p)<-[:CITES]-(citingPaper:paper)
WHERE citingPaper.year = currentYear
WITH j, currentYear, COUNT(citingPaper) AS citations, COLLECT(p) AS
    papersPublished
RETURN j.name AS journalName,
     citations,
     SIZE(papersPublished) AS articlesPublished,
     CASE WHEN SIZE(papersPublished) > 0 THEN citations * 1.0 / SIZE(
    papersPublished) ELSE 0 END AS impactFactor
ORDER BY impactFactor DESC
```

Listing 3: Third query

Results Summary: Identified varying influence levels across journals, with "Journal of Open Source Software" leading. Details are in *PartB.3_BacaksizOstos.ipynb.*

2.4 Calculating H-Indexes of Authors

Objective: Calculate each author's h-index to gauge their research impact.

```
MATCH (a:author)-[:WRITES]->(p:paper)
OPTIONAL MATCH (p)<-[:CITES]-(citing:paper)
WITH a, p, COUNT(citing) AS citations
ORDER BY citations DESC
WITH a, COLLECT(citations) AS citationCounts
WITH a, citationCounts, RANGE(0, SIZE(citationCounts)-1) AS indices
UNWIND indices AS idx
WITH a, citationCounts, idx
WHERE citationCounts[idx] >= idx + 1
```

```
WITH a.name AS authorName, MAX(idx + 1) AS hIndex
RETURN authorName, hIndex
ORDER BY hIndex DESC
```

Listing 4: Fourth query

Result Summary: Efficiently computed h-indices, highlighting authors like Madeleine Ernst with an h-index of 19. For an in-depth analysis, refer to *PartB.4_BacaksizOstos.ipynb*.

3 Recommender

This section outlines the construction of a recommender system to identify potential reviewers within the database community, utilizing a series of interconnected steps to refine the selection process efficiently.

STEP 1: Establishing Community
Created a 'Database Community' node to anchor subsequent queries.

```
MERGE (:Community \{name: 'Database Community'\})
```

STEP 2: Associating Keywords
Linked relevant keywords to the community to define its scope

```
MERGE (kw:Keyword {name: $keyword, domain: 'Computer Science'})
WITH kw
MATCH (dbComm:Community {name: 'Database Community'})
MERGE (kw)-[:DEFINES]->(dbComm)
```

STEP 3: Strengthening Relationships
Enhanced connections between the community and its defining keywords.

```
MATCH (kw:keywords {domain: 'Computer Science'}),
(dbComm:Community {name: 'Database Community'})
MERGE (kw)-[r:RELATED_TO]->(dbComm)
RETURN kw.name AS keyword, dbComm.name AS community
```

STEP 4&5: Identifying Relevant Publications
Determined publications closely aligned with the community through keyword analysis.

```
MATCH (p:Paper)-[:HAS]->(kw:Keyword)
WHERE kw.name IN ['data management', 'indexing', 'data modeling',
'big data', 'data processing', 'data storage', 'data querying']
WITH p, COLLECT(kw.name) AS keywords
MATCH (p)-[:PUBLISHED_IN]->(pub)
WITH pub, COUNT(p) AS totalPapers, COUNT(keywords) AS keywordPapers
WHERE keywordPapers >= 0.9 * totalPapers
SET pub:RelatedToDatabaseCommunity
RETURN pub
```

```
MATCH (c:Community {name: "Database Community"})<-[:RELATED_TO]-(kw:keywords)
WITH c, kw
MATCH (kw)<-[:HAS]-(p:paper)
WITH kw, p
MATCH (p)-[:PUBLISHED_IN]->(publication)
WITH publication, COUNT(DISTINCT p) AS papersWithKeywords
MATCH (publication)<-[:PUBLISHED_IN]-(p2:paper)
WITH publication, papersWithKeywords, COUNT(DISTINCT p2) AS totalPapers
WHERE papersWithKeywords >= 0.9 * totalPapers
SET publication:RelatedToDatabaseCommunity
RETURN publication, papersWithKeywords, totalPapers
```

STEP 6: Highlighting Influential Papers
Identified top papers within the community based on citation counts.

```
MATCH (journal:RelatedToDatabaseCommunity)<-[:PUBLISHED_IN]-(paper:paper)
WITH paper, SIZE([(paper)<-[:CITES]-(citing:paper) | citing]) AS citations
ORDER BY citations DESC
LIMIT 100
SET paper:TopPaper
RETURN paper AS TopPaper, citations
```

STEP 7: Finding Potential Reviewers
Selected authors of top papers as potential reviewers.

```
MATCH (a:author)-[:WRITES]->(p:TopPaper)
WITH a, COUNT(p) AS contributions
SET a:PotentialReviewer
RETURN a AS Author, contributions
```

STEP 8: Distinguishing Gurus
Recognized authors with substantial contributions as community gurus.

```
MATCH (a:author)-[:WRITES]->(p:TopPaper)
WITH a, COUNT(p) AS contributions
WHERE contributions >= 2
SET a:Guru
RETURN a AS Author, contributions
```

Efficiency and Structure: Each step was designed to build upon the previous one, ensuring an efficient and focused approach to developing the recommendation engine. By progressively refining the selection criteria and utilizing efficient Cypher queries, the system efficiently narrows down to the most suitable reviewer candidates, enhancing the accuracy and relevance of its recommendations.
Conclusion: The stepwise approach ensures a methodical narrowing down of potential reviewers, starting from broad community definitions to pinpointing individual contributors. This structured process, coupled with efficient Cypher queries, not only enhances the recommendation engine's performance but also ensures the relevance and accuracy

of the recommendations provided.

4 Graph algorithms

4.1 PageRank

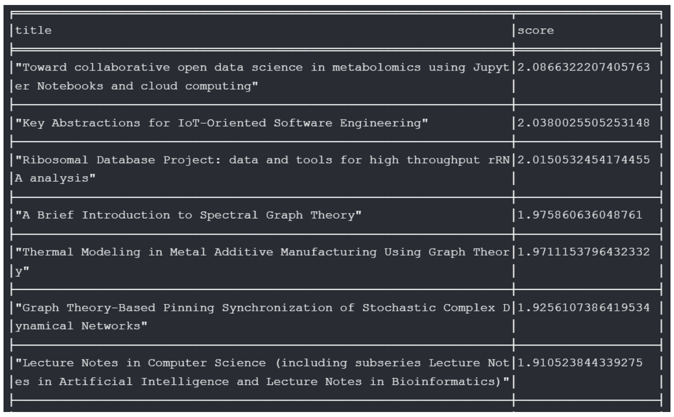
The RageRank algorithm measures the importance of each node within a graph. The importance of each node is based on the number incoming relationships and the importance of the corresponding source nodes[2]. For example, the algorithm will assign a higher importance to a person with few friends that have many friends than to a person that has many friends with no friends at all. In the graph proposed, it would be interesting to analyze the influence of every paper, according to the chain of citation it is involved in.

```
CALL gds.graph.project('myGraph1', 'paper', 'CITES');

CALL gds.pageRank.stream('myGraph1')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).title AS title, score
ORDER BY score DESC, title;
```

Listing 5: Query for the PageRank algorithm

In Fig.(4), the most important papers are shown. Notice that most relevant paper according to this algorithm has *score* ≈ 2.08, meaning by this that the dataset used is highly uncorrelated.



```
╒════════════════════════════════════════════════════════════════════════╤═══════════════════╕
│title                                                                   │score              │
╞════════════════════════════════════════════════════════════════════════╪═══════════════════╡
│"Toward collaborative open data science in metabolomics using Jupyt│2.0866322207405763 │
│er Notebooks and cloud computing"                                       │                   │
├────────────────────────────────────────────────────────────────────────┼───────────────────┤
│"Key Abstractions for IoT-Oriented Software Engineering"                │2.0380025505253148 │
├────────────────────────────────────────────────────────────────────────┼───────────────────┤
│"Ribosomal Database Project: data and tools for high throughput rRN│2.0150532454174455 │
│A analysis"                                                             │                   │
├────────────────────────────────────────────────────────────────────────┼───────────────────┤
│"A Brief Introduction to Spectral Graph Theory"                         │1.975860636048761  │
├────────────────────────────────────────────────────────────────────────┼───────────────────┤
│"Thermal Modeling in Metal Additive Manufacturing Using Graph Theor│1.9711153796432332 │
│y"                                                                      │                   │
├────────────────────────────────────────────────────────────────────────┼───────────────────┤
│"Graph Theory-Based Pinning Synchronization of Stochastic Complex D│1.9256107386419534 │
│ynamical Networks"                                                      │                   │
├────────────────────────────────────────────────────────────────────────┼───────────────────┤
│"Lecture Notes in Computer Science (including subseries Lecture Not│1.910523844339275  │
│es in Artificial Intelligence and Lecture Notes in Bioinformatics)"│                   │
├────────────────────────────────────────────────────────────────────────┼───────────────────┤
```

Figure 4: Results for the PageRank algorithm.

4.2 Node Similarity

The Node Similarity algorithm compares a set of nodes based on the nodes they are connected to. Two nodes are considered similar if they share many of the same neighbors. In the graph given, it is interesting to analyze the similarity between different papers by relating common keywords.

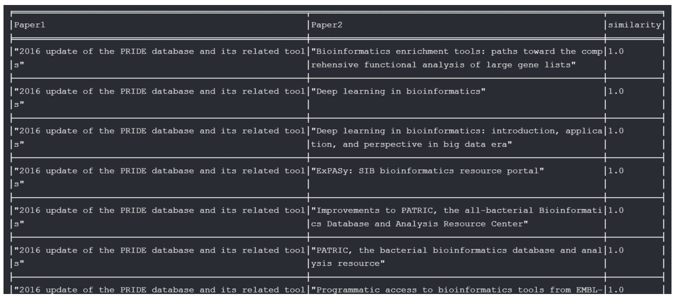
```
CALL gds.graph.project('myGraph2', ['paper', 'keywords'], 'HAS');

CALL gds.nodeSimilarity.write.estimate('myGraph2', {
  writeRelationshipType: 'SIMILAR', writeProperty: 'score'});

CALL gds.nodeSimilarity.stream('myGraph2')
YIELD node1, node2, similarity
RETURN gds.util.asNode(node1).title AS Paper1, gds.util.asNode(node2).title AS
    Paper2, similarity
ORDER BY similarity DESC, Paper1, Paper2;
```

Listing 6: Query for the Node Similarity algorithm

In Fig.(5), pairs of similar papers with their similarity rate are presented. Notice that several papers will be considered has the same for the algorithm because the graph only considers one keyword per paper. In order to have more interesting result, a more detailed dataset would be necessary.



```
╒═══════════════════════════════════════════════════════╤═══════════════════════════════════════════════════════╤══════════╕
│Paper1                                                 │Paper2                                                 │similarity│
╞═══════════════════════════════════════════════════════╪═══════════════════════════════════════════════════════╪══════════╡
│"2016 update of the PRIDE database and its related tool│"Bioinformatics enrichment tools: paths toward the comp│1.0       │
│s"                                                     │rehensive functional analysis of large gene lists"     │          │
├───────────────────────────────────────────────────────┼───────────────────────────────────────────────────────┼──────────┤
│"2016 update of the PRIDE database and its related tool│"Deep learning in bioinformatics"                      │1.0       │
│s"                                                     │                                                       │          │
├───────────────────────────────────────────────────────┼───────────────────────────────────────────────────────┼──────────┤
│"2016 update of the PRIDE database and its related tool│"Deep learning in bioinformatics: introduction, applica│1.0       │
│s"                                                     │tion, and perspective in big data era"                 │          │
├───────────────────────────────────────────────────────┼───────────────────────────────────────────────────────┼──────────┤
│"2016 update of the PRIDE database and its related tool│"ExPASy: SIB bioinformatics resource portal"           │1.0       │
│s"                                                     │                                                       │          │
├───────────────────────────────────────────────────────┼───────────────────────────────────────────────────────┼──────────┤
│"2016 update of the PRIDE database and its related tool│"Improvements to PATRIC, the all-bacterial Bioinformati│1.0       │
│s"                                                     │cs Database and Analysis Resource Center"              │          │
├───────────────────────────────────────────────────────┼───────────────────────────────────────────────────────┼──────────┤
│"2016 update of the PRIDE database and its related tool│"PATRIC, the bacterial bioinformatics database and anal│1.0       │
│s"                                                     │ysis resource"                                         │          │
├───────────────────────────────────────────────────────┼───────────────────────────────────────────────────────┼──────────┤
│"2016 update of the PRIDE database and its related tool│"Programmatic access to bioinformatics tools from EMBL-│1.0       │
```

Figure 5: Results for the Node Similarity algorithm.

References

[1] Learn-SQL v2: Learning Environment for Automatic Rating Notions of SQL: Inicia sessió en aquest lloc. (s. f.-b). https://learnsql2.fib.upc.edu/moodle/pluginfile.php/3424/mod_resource/content/8/property-graphs.pdf

[2] PageRank - NeO4J Graph Data Science. (s. f.). Neo4j Graph Data Platform. https://neo4j.com/docs/graph-data-science/current/algorithms/page-rank/

[3] Node similarity - NEO4J Graph Data Science. (s. f.). Neo4j Graph Data Platform. https://neo4j.com/docs/graph-data-science/current/algorithms/node-similarity/