Graph Data Science for Author Domain Classification and Link Prediction

Basil Ali Khan, Hamza Ansari, Hayyan Khan

Group 6
Graph Data Science, Spring 2025

I. Introduction

The source code used in this project is available on GitHub at the following link: GitHub Repository.

II. Data Preprocessing and Graph Modelling

Data source: [1]
The dataset comprises academic publications represented as a heterogeneous graph consisting of Author, Paper, and Citation entities. Preprocessing was conducted using R, where the raw data was cleaned and missing values were handled.

The graph model was designed to represent the relationships between authors, papers, topics, journals, and publishers in the academic domain. The data was modeled as a heterogeneous graph in Neo4j, with the following node types and relationships:

A. Nodes

- **Author**: Represents researchers or authors. Key properties include:
  - `id`: Unique identifier for the author.
  - `name`: Name of the author.
  - `total_citations`: Total number of citations received by the author's papers.
  - `research_domain`: The most researched topic by the author.
  - `last_collab_year`: The most recent year of collaboration with other authors.
- **Paper**: Represents academic papers. Key properties include:
  - `id`: Unique identifier for the paper.
  - `title`: Title of the paper.
  - `year`: Year of publication.
  - `citationCount`: Number of citations received by the paper.
- **Topic**: Represents research topics. Key properties include:
  - `id`: Unique identifier for the topic.
  - `name`: Name of the topic.
- **Journal**: Represents academic journals. Key properties include:
  - `name`: Name of the journal.
- **Publisher**: Represents publishers of journals. Key properties include:
  - `name`: Name of the publisher.

B. Relationships

- **WROTE**: Connects an `Author` to a `Paper` they authored.
- **CO_AUTHORED**: Connects two `Author` nodes who collaborated on the same paper. Includes:
  - `collaboration_year`: The year of collaboration.
- **HAS_TOPIC**: Connects a `Paper` to a `Topic` it addresses.
- **RESEARCHES**: Connects an `Author` to a `Topic` they have researched. Includes:
  - `paper_count`: Number of papers the author has written on the topic.
- **SHARED_TOPIC**: Connects two `Author` nodes who have researched the same topic. Includes:
  - `shared_topics`: Number of shared topics between the authors.
- **SHARED_JOURNAL**: Connects two `Author` nodes who have published in the same journal. Includes:

  - `shared_journals`: Number of shared journals between the authors.
- **PUBLISHED_IN**: Connects a `Paper` to the `Journal` it was published in.
- **PUBLISHED_BY**: Connects a `Journal` to its `Publisher`.
- **CITES**: Connects one `Paper` to another paper it cites.

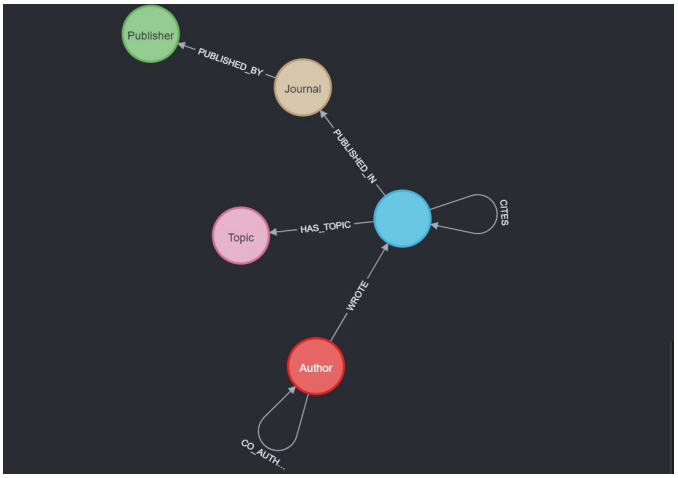

Fig. 1. Data Model for the Graph

C. Feature Engineering

To enhance the graph model, additional features were computed:

- **Total Citations**: The total number of citations received by an author's papers.
- **Research Domain**: The topic most frequently researched by an author.
- **Last Collaboration Year**: The most recent year an author collaborated with others.

D. Neo4j Script For Graph Creation

The Neo4j script to model the graph can be found in the github repository in the *graph_model_creation_script.txt* file

III. Methodology

A. Node Classification

The node classification task aims to predict the research domain (`domain_id`) of authors based on their co-authorship patterns and other features. We utilized the Neo4j Graph Data Science (GDS) library to train a machine learning model to classify authors into specific research domains. The authors were represented as nodes in a co-authorship graph, where edges between nodes (authors) represented collaborations (co-authorships) in academic papers.

1) Graph Projection: The first step in our methodology was to project the graph as follows:

- **Nodes:**
  - `Author`: Represents an academic author. Each author node has a property `domain_id` (the target classification label)
- **Relationships:**
  - `CO_AUTHOR`: Represents a collaboration between two authors on a paper.
- **Label:** Authors with known research domains were tagged with the label `AuthorWithDomain` for the classification task. Authors without a known `domain_id` were excluded from the node classification task. This left us with 12738 authors out of 38925, approximately 32.7% of the total count.

2) Feature Engineering: To predict the `domain_id`, we used the following features in the classification model:

- **Louvain Community ID:** This feature was generated using the Louvain algorithm, which detects communities within the graph based on co-authorship patterns. Authors within the same community tend to have similar research topics, which makes this feature useful for domain classification. This was done in the node classification pipeline using the addNodeProperty function.
- **Graph Embedding:** We generated graph embeddings using the Fast Random Projection (FastRP) algorithm. This technique transforms the graph structure into a vector space, capturing the local and global structure of the co-authorship network. Each author node was assigned an embedding vector representing its position within the network. These embeddings were created in the node classification pipeline using the addNodeProperty function. Models were trained over two separate embeddings: 64

and 128 while keeping the no of trees in the random forest constant.

- **Domain ID (Target Property):** The target property for node classification was the `domain_id`, which denotes the author's research domain. This was used to train the classification model.

These features were chosen because they provide both structural (co-authorship network) and domain-related information (community structure, embeddings) that can help in predicting the author's research domain.

3) Machine Learning Model: For node classification, we used the Random Forest classifier, which is robust to overfitting and effective for high-dimensional datasets like those encountered in graph-based tasks. The classifier was trained using the Neo4j Graph Data Science library, which provides built-in support for training machine learning models on graph data. The training process was configured as follows:

- **Random Forest Classifier:** This ensemble method was chosen because it handles complex datasets well by combining multiple decision trees and averaging their outputs to improve accuracy and reduce overfitting. 25 and 40 trees were used in two separate models to compare results.
- **5-Fold Cross-Validation:** To ensure generalization and prevent overfitting, we employed 5-fold cross-validation, splitting the data into 5 subsets. The model was trained on 4 subsets and validated on the remaining 1, rotating through all subsets.
- **Evaluation Metrics:** We evaluated the model's performance using several metrics, including:
  - **F1-Weighted:** A harmonic mean of precision and recall, useful for imbalanced datasets.
  - **Accuracy:** The proportion of correctly classified instances.
  - **Out-of-Bag (OOB) Error:** This error metric provides an estimate of the classifier's performance on unseen data.

The model was trained on the graph, where the features (Louvain community ID, graph embeddings) were used to predict the `domain_id` of each author.

4) Model Training and Evaluation: The training process was carried out in the Neo4j Graph Data Science environment using the following steps:

- **Graph Projection:** A native projection was created for the `AuthorWithDomain` label using Cypher queries, ensuring that only the relevant nodes and relationships were included in the machine learning task.
- **Pipeline Construction:** The training pipeline was built using the Neo4j GDS library, including steps for feature extraction (Louvain, FastRP embedding), training the Random Forest model, and evaluating performance using cross-validation.
- **Model Training:** After defining the pipeline, the Random Forest model was trained using the features and the target `domain_id`. The model parameters were tuned for optimal performance.
- **Model Evaluation:** The trained model's performance was evaluated using the metrics mentioned earlier, and the results were analyzed to identify areas of strength and potential improvement.

Once the model was trained, we used it to predict the `domain_id` of authors who were not part of the training set. There were a total of 3 iterations. First one was used as a base model for comparison, the second one had a different embedding but same number of trees as the first one, while the third one had different number of trees while keeping the same embedding as the first one.

B. Link Prediction

IV. Results

A. Node Classification Results

The first node classification model was trained to predict the `domain_id` of authors in the co-authorship graph using a Random Forest classifier. After training and cross-validation, the following evaluation metrics were obtained from the Neo4j GDS pipeline:

- **Test Accuracy:** 0.60
- **Test F1 Score:** 0.54
- **Out-of-Bag (OOB) Error:** 0.49

Fig. 2. Output after training pipeline

These initial metrics reflect the model's performance on a held-out validation set during training.

1) Prediction Performance: To further validate the model, we applied it to unseen nodes using the 'predict.stream' procedure in GDS and compared the predicted domain labels with the ground truth labels. The following evaluation metrics were calculated based on the predictions:

- **Accuracy:** 0.77
- **Precision:** 0.77
- **Recall:** 1.00
- **F1 Score:** 0.87

total	correct	accuracy
12738	9858	0.7739048516250588

Fig. 3. Accuracy

precision	recall	f1_score
0.7739048516250588	1.0	0.8725438130642591

Fig. 4. Precision Recall and F1 Score

The high recall value indicates that the classifier successfully predicted nearly all actual instances of each domain class. However, the slightly lower precision suggests that there were some false positives, indicating the need for improved feature representation or model tuning.

2) Iteration 2 – Embedding = 128, Number of Trees = 25:

- **Test Accuracy:** 0.59
- **Test F1 Score:** 0.53
- **Out-of-Bag (OOB) Error:** 0.49
- **Accuracy:** 0.78

testAccuracy	testF1	testOutOfBagError
0.59392988	0.5305171211445211	0.4965231045311799

Fig. 5. Output after training pipeline with embedding = 128 and trees = 25

- **Precision:** 0.78
- **Recall:** 1.00
- **F1 Score:** 0.88

total	correct	accuracy
12738	9932	0.7797142408541372

Fig. 6. Accuracy for embedding = 128 and trees = 25

precision	recall	f1_score
0.7801067671533993	1.0	0.8764718853362734

Fig. 7. Precision Recall and F1 Score for embedding = 128 and trees = 25

The change in embedding does not show any significant change in the model output.

3) Iteration 3 – Embedding = 64, Number of Trees = 40:

- **Test Accuracy:** 0.63
- **Test F1 Score:** 0.56
- **Out-of-Bag (OOB) Error:** 0.44

testAccuracy	testF1	testOutOfBagError
0.62663527	0.5592646420575051	0.44010767160161507

Fig. 8. Output after training pipeline with embedding = 64 and trees = 40

- **Accuracy:** 0.79
- **Precision:** 0.80
- **Recall:** 1.00
- **F1 Score:** 0.88

The increase in the number of trees for the random forest model slows a slight betterment in the results of accuracy and precision.

total	correct	accuracy
12738	10125	0.7948657560056523

Fig. 9. Accuracy for embedding = 64 and trees = 40

precision	recall	f1_score
0.7969853980216675	1.0	0.8870249017038008

Fig. 10. Precision Recall and F1 Score for embedding = 64 and trees = 40

B. Link Prediction Results

V. Discussion

The node classification task yielded promising results, demonstrating that structural graph features like community membership and FastRP embeddings can reasonably predict an author's research domain. However, several limitations and areas for improvement were identified.

A. Limitations and Challenges

A key limitation was the quality of node embeddings. Several authors—particularly those with few or no co-authors—received zero or low-informative embedding vectors, which hindered classification accuracy. Additionally, the presence of interdisciplinary authors, who collaborate across multiple research domains, introduced ambiguity in both graph structure and labels. These factors contributed to misclassifications and highlight the need for more robust features beyond network structure. Also, the training of the model was a resource heavy task, forcing us to limit our iterations for this task to only 3

B. Alternative Approaches

Alternative graph embedding techniques, such as Node2Vec, could be explored to better capture node similarity in sparse or heterogeneous graphs. Additionally, incorporating non-structural node features—such as publication keywords, journal categories, or citation counts—could enrich the feature set and provide context where structural signals are weak. These features could be extracted using NLP techniques applied to paper titles or abstracts.

C. Possible Extensions

A natural extension would involve deploying Graph Neural Networks (GNNs), which can learn more expressive representations by aggregating features from neighborhood nodes. Another promising direction is to use semi-supervised learning to infer domain labels for authors not included in the initial labeled set. This would expand the model's applicability and improve generalization across the graph.

Overall, while the Random Forest classifier with FastRP embeddings performed well under certain conditions, integrating richer node features and leveraging advanced GNN-based methods could significantly enhance model robustness and accuracy in future work.

References

[1] L. Rothenberger, M. Q. Pasta, and D. Mayerhoffer, "Mapping and impact assessment of phenomenon-oriented research fields: The example of migration research," *Quantitative Science Studies*, vol. 2, no. 4, pp. 1466–1485, 12 2021. [Online]. Available: https://doi.org/10.1162/qss_a_00163