Graph-Based Machine Learning for Bibliographic Data Analysis: Node Classification and Link Prediction

Breeha Qasim, Ashbah Faisal & Namel Shahid
Department of Computer Science
Habib University
Karachi, Pakistan

***Abstract*—The application of graph-based machine learning algorithms to the analysis of bibliographic data is thoroughly examined in this research. We investigate two fundamental tasks: link prediction to predict future collaborations and node classification to determine author domains. Our approach builds and analyses a rich bibliographic network by combining machine learning frameworks with the Neo4j graph database. In addition to offering insights into research collaboration patterns and domain classification, the study shows how well graph-based techniques capture complicated linkages seen in academic literature.**

I. Introduction

Analysing bibliographic data has grown in significance for comprehending patterns of collaboration, research trends, and information sharing. In this study, node classification and link prediction tasks are especially addressed by utilising graph-based machine learning approaches to analyse bibliographic data. Using a large dataset sourced from bibliographic records, the study seeks to shed light on domain classification and academic collaboration trends.

II. Methodology

A. Data Preprocessing

The dataset consists of multiple CSV files containing information about authors, papers, journals, topics, and their relationships. The preprocessing pipeline, implemented in Python using Pandas, includes the following steps:

- **Data normalization:** Converting column names to snake case and removing special characters
- **String cleaning:** Trimming whitespace and normalizing empty strings to None values
- **Duplicate removal:** Eliminating duplicate rows and rows with null primary keys
- **ID mapping:** Creating integer IDs for authors, papers, and topics for efficient graph construction
- **Type conversion:** Converting numeric fields (year, citation count, volume) to nullable integers

The preprocessing pipeline handles seven main data files:

- `author.csv`: Author information (ID, name, URL)
- `topic.csv`: Research topic information (ID, name, URL)
- `journal.csv`: Journal details (name, publisher)
- `paper.csv`: Paper metadata (ID, DOI, title, year, citations, etc.)
- `author_paper.csv`: Author-paper relationships
- `paper_topic.csv`: Paper-topic relationships
- `paper_reference.csv`: Citation relationships between papers

Table I summarizes the number of rows and columns before and after cleaning for each file.

TABLE I
Dataset Statistics Before and After Cleaning

File	Raw Rows	Raw Cols	Cleaned Rows	Cleaned Cols
authors	38925	3	38925	4
topics	6489	3	6489	4
journals	165	2	51	2
papers	693624	9	693624	10
author_paper	56450	2	56450	2
paper_topic	41880	2	41880	2
paper_journal	32647	3	32647	3
paper_reference	1132044	2	1132044	4

B. Graph Construction

The bibliographic graph is constructed using Neo4j, with the following node types and their key attributes:

- Author: {author_id, name, url}
- Paper: {paper_id, doi, title, year, citation_count, field, volume, date, url}
- Journal: {journal_name, publisher}
- Topic: {topic_id, name, url}

The model comprises `739,089` nodes. The graph structure also includes the following relationships:

- (:Author) − [:WROTE] → (:Paper): Authorship relationships
- (:Paper) − [:PUBLISHED_IN] → (:Journal): Publication venue
- (:Paper) − [:HAS_TOPIC] → (:Topic): Research topic classification
- (:Paper) − [:CITES] → (:Paper): Citation network

The model also comprises `126,3011` relationships.
Figure 1 illustrates the overall graph schema used in this project.

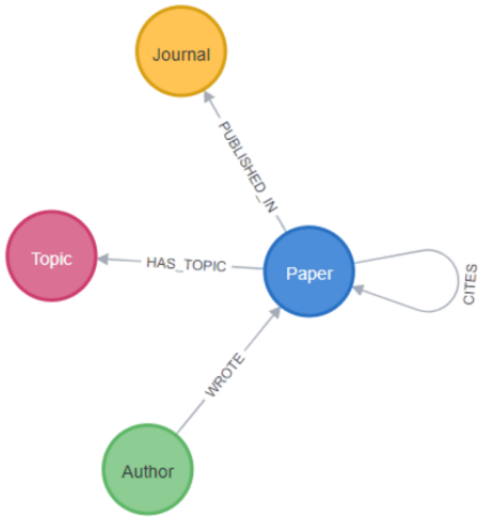

Fig. 1. Bibliographic Graph Model Schema

By handling missing values, eliminating duplicates, normalising column names, and mapping IDs for effective graph creation, the preprocessing pipeline guarantees data consistency. These Python-based procedures are crucial for creating a dependable and clean dataset for graph loading.

Unique constraints and indexes are built into Neo4j to ensure data integrity and speed up queries on important properties like names and IDs. When importing millions of records, the use of `CALL IN TRANSACTIONS` allows for the effective bulk loading of huge CSV files, reducing memory utilisation and preventing transaction timeouts.

This method guarantees the accuracy and performance of the final graph database, facilitating operations related to machine learning and downstream analysis.

Since we only focused on the author-paper-co-authorship topology and did not model "year" or other temporal fields as distinct nodes, considering year as a straightforward relationship attribute proved adequate for the author/domain and future-collaboration use cases. The graph construction process includes:

- Creating unique constraints for primary keys
- Establishing indexes for frequently queried fields (paper year, author name)
- Batch loading of nodes and relationships in transactions
- Handling of missing and null values during node creation

C. Node Classification (Author Classification)

The node classification task focuses on *Author Classification:* predicting the research domain or expertise of an author based on their co-authorship network.

Preprocessing and Feature Engineering:

A co-authorship network is constructed by creating `CO_AUTHOR_WITH` relationships between authors who have co-written papers. Around `44520` relationships were created. The label `author_domain` was applied to each author around `12744` properties, indicating the dominating research topic (the most common topic among their works). Since Neo4j GDS does not support string labels, each unique domain is mapped to a numeric `domain_id`. The training set (marked as `:TrainAuthor`) only contains authors with a known domain (`domain_id IS NOT NULL`).

Modeling Approach:

We implemented three RandomForest variants:

1) *fastRP* + *domain_id* only.
2) *fastRP* + *communityId* + *domain_id*.
3) *fastRP* + *communityId* + *pageRank* + *domain_id*.

Node features are generated via fastRP graph embeddings (capture local and global structure), Louvain community IDs (meso-scale clusters), and PageRank scores (centrality). Due to long training times on the full dataset, we reduced:

- `embeddingDimension` from 128 $\rightarrow$ 64
- `numberOfDecisionTrees` from 50 $\rightarrow$ 25

We train with 4-fold cross-validation and a 25% test split. Evaluation metrics: `accuracy, weighted F1-score, and out-of-bag error`.

Assumptions and Rationale:

The position and relationships of an author's co-author graph might be used to determine their research area. The Louvain community identifies authors who share structural similarities and frequently discuss comparable subjects. Node centrality serves as an extra predictive signal when PageRank is included.

D. Link Prediction (Future Collaboration)

By identifying trends in their current co-authorship network, the link prediction challenge seeks to anticipate which pairs of researchers will work together in the future.

Preprocessing and Feature Engineering:

We begin by constructing an undirected co-author graph in Neo4j, for every pair of authors who co-wrote a paper, we merge a `CO_AUTHOR_WITH` relationship, carrying two edge properties:

- `year`: the publication year of at least one shared paper,
- `sharedTopics`: the list of overlapping topics between their papers.

After loading all such relationships, the in-memory projection `predictionGraph` contains `38 925` author nodes and `46 116` edges. We then compute the following *node-level* features:

- **fastRP embeddings** (capture both local neighborhood and global position):
  - *128 dimensions* for Random Forest pipelines,

  - *56 dimensions* for the Logistic Regression baseline,
  - `randomSeed`=42.
- **degree** (number of collaborators) and its Min–Max scaled version, to encode each author's collaboration activity.

In addition, for some variants we compute:

- `Louvain communityId` to capture meso-scale clusters of closely-connected authors;
- `PageRank` centrality and `triangleCount` (and scaled versions) to measure author prominence and local group cohesion.

The main *edge feature* in all pipelines is the `HADAMARD` product of node embeddings; the "RF + PageRank" variant additionally adds `COSINE` and `L2` similarities of embeddings, while the "Time-based Hadamard" variant limits edges to those with valid `year` and `sharedTopics`. We divide 70% of the current edges for training, 20% for testing, and reserve 10% for validation (using five-fold cross-validation). In order to balance positive cases, a 1:1 negative sampling ratio creates non-edges.

Modeling Approach:
We compare five distinct pipelines in the GDS link-prediction framework:

1. Logistic Regression (LR) Baseline
Makes use of raw and scaled degrees + Hadamard product + fastRP embeddings (dimension 56) + split by 70 % for train and 20 % for test.

2. RandomForest (RF) Baseline
The same features as LR,embedding dimension increased to 128, but with a RandomForest ensemble of 25 trees. Shows how switching to a non-linear, tree-based classifier using embedding-derived link characteristics improves performance.

3. RF + CommunityID
Adds each node's participation in the Louvain community to the RF baseline features. The hypothesis posits that there is a much higher likelihood of collaboration among writers who belong to the same community, such as departmental or subject clusters.

4. RF + PageRank
Time-based splitting when all years are counted in train set except for 2020. Adds PageRank centrality and (scaled) triangle counts, as well as Cosine and L2 embedding similarity, to the baseline. Rationale: Authors in tightly-clustered neighbourhoods and central, well-connected authors may be more likely to form new partnerships; combining these orthogonal signals should enhance the quality of the ranking.

5. RF + Time-based Hadamard
The feature set is changed to Hadamard embedding + scaled degree. The RF baseline is used after filtering to only those edges with valid topical (`sharedTopics`) and temporal (`year`) annotations. Motivation: By limiting the model to well-annotated edges, the model is focused on the most informative signals(same as for page rank).
All RandomForest variants use 25–50 trees (down from 100 in initial experiments to reduce training time), while LR remains a fast baseline.

Evaluation Metrics:
We assess each model on

- `AUPR/AUCPRs` (area under the precision–recall curve) on held-out test edges,
- `Out-Of-Bag (OOB) error` for RandomForest variants,
- `Validation AUPRs` from 5-fold cross-validation.

Assumptions and Rationale:

- Structural embeddings (fastRP) summarize the graph neighborhood patterns that underlie collaboration.
- Community detection groups authors by latent topical or institutional affiliation.
- Centrality measures (PageRank) and local clustering (triangle counts) capture different aspects of author influence and cohesion.
- Temporal and topical annotations on edges allow the model to learn recency and subject-matching effects.
- Comparing five variants quantifies the incremental benefit of each feature class, guiding future feature engineering in bibliographic link prediction.

III. Results

A. Node Classification Performance

We compare three RandomForest variants on the author-domain task:

Model	Test Accuracy	Test F1	Test OOB	Validation
fastRP + domain_id	0.5399	0.5032	0.5464	0.4995
fastRP + communityId + domain_id	0.5588	0.5295	0.5367	0.5142
fastRP + communityId + pageRank + domain_id	0.5599	0.5254	0.5287	0.5135

TABLE II
PERFORMANCE OF RF CLASSIFIERS WITH DIFFERENT FEATURE SETS (EMBEDDINGDIMENSION=64, #TREES=25).

B. Link Prediction Performance

Model	Test AUPR	Test OOB	Validation
RF + CommunityId	0.6872	0.4981	0.6849

TABLE III
TABLE III: RF + COMMUNITYID PERFORMANCE

Model	Avg Train	Outer Train	Test Score	Validation Scores
RF + Time-Hadamard	0.6966	0.6984	0.6860	0.6866
RF + PageRank	0.9559	0.9436	0.9411	0.9107

TABLE IV
TABLE IV(A): GRAPH FEATURE MODEL VARIANTS PERFORMANCE

Model	Avg Train	Outer Train	Test Score	Validation Scores
Random Forest	0.7319	0.7292	0.7254	0.7091
Logistic Regression	0.8283	0.8283	0.8301	0.8283

TABLE V
TABLE IV(B): BASELINE MODEL PERFORMANCE

IV. DISCUSSION

A. Key Findings

1) ***Community Detection (Louvain)****:* With nearly perfect modularity (0.9914), the Louvain partitioning results show a hyper-fragmented cooperation network with remarkably robust, segregated communities. While the greatest outlier (446 authors) suggests a rare, large-scale research hub, almost 17,000 little clusters (mean size: 2.23 authors) predominate, indicating that most collaborations are temporary or restricted to pairs. This excessive localisation suggests highly specialised, insular teamwork with little interaction between communities, which may be a reflection of biases particular to a dataset or disciplinary barriers. The 95th percentile cap of five writers emphasises even more how rare long-term group collaborations outside of tiny teams are.

Metric	Value
Modularity	0.9914
Number of communities	17,432
Mean community size	2.23 authors
Maximum community size	446 authors
Minimum community size	1 author
95th percentile community size	5 authors

TABLE VI
SUMMARY STATISTICS OF LOUVAIN COMMUNITIES IN THE CO-AUTHOR GRAPH.

2) ***PageRank Centrality****:* With the majority of writers scoring close to the median (0.9741) and a small elite (top $1\% > 2.2703$) controlling network centrality, the PageRank study shows a highly skewed influence distribution. A weakly linked core, with a few high-impact hubs bridging disparate communities, is suggested by the algorithm's inability to converge after 50 iterations. This structure shows the "rich-get-richer" dynamics that are common in academic networks, where peripheral contributors (min: 0.15) stay isolated while prominent authors disproportionately control the flow of information. These core nodes are probably essential to the network's resiliency and knowledge spread.

Metric	Value
Minimum PageRank	0.1500
Mean PageRank	0.7525
Median (50th percentile)	0.9741
95th percentile	1.4590
99th percentile	2.2703
Maximum PageRank	10.7348
Iterations run	50

TABLE VII
DISTRIBUTION OF PAGERANK SCORES ON THE CO-AUTHOR GRAPH.

B. Comparison

1) ***Comparison of all three Random Forest variants (Node Classification)****:* Our model tests at 53.99% accuracy in the baseline `(fastRP + domain_id)` with `F1` = 0.5032, `OOB` = 0.5464, and `validation` = 0.4995, indicating that embeddings by themselves capture a large portion of domain structure but not all of it. Since community labels directly represent subfield clusters that raw embeddings must otherwise infer, adding Louvain `communityId` results in the biggest jump: accuracy increases by 1.69 points to 55.68 percent, `F1` to 0.5295, `OOB` drops to 0.5367, and `validation` to 0.5142. When PageRank is added to embeddings and communityIDs, accuracy increases by an additional +0.22 points to 55.90% and `OOB` decreases to 0.5287, however `F1` marginally decreases to 0.5254 and `validation` to 0.5135.
Centrality adds depth by emphasising hub versus expert authors, but it also overlaps with community structure, so it refines rather than completely overhauls the decision boundaries, which is why this smaller uplift happens. Collectively, these jumps demonstrate that meso-scale clusters provide the largest performance benefit, while macro-scale prominence through PageRank offers further, if smaller, improvements.

2) ***Comparison of Random Forest vs Linear Forest (Link Prediction)****:* We compare two distinct supervised link prediction pipelines one based on Logistic Regression and the other on Random Forest. The Logistic Regression pipeline achieves strong and consistent performance with an average training `AUCPR` of 0.8283, an `outer training score` of 0.8283, `test score` of 0.8301 and a `validation score` of 0.8283. These tightly clustered scores suggest high generalizability and low variance. In contrast, while the Random Forest pipeline uses a deeper `FastRP embedding` (dimension 128) and the same feature set, yet showing worse performance an `average training AUCPR` of 0.7319, `outer training score` of 0.7292, `test score` of 0.7254, and `validation score` of 0.7091. The decline in `AUCPR` scores, drastically despite increasing dimensions, for the Random Forest model suggests that it may be more prone to overfitting or that the added complexity did not translate into improved predictive power. However, the Logistic Regression model generalizes better in this setting and yields

more stable results in training, testing, and validation. This implies that for this co-authorship link prediction task, logistic regression is more effective, possibly due to the nature of the underlying graph features which may be linearly separable in the embedding space. Additionally, this finding reinforces the models, simplicity, which, when aligned with well-engineered features, can outperform more complex ensembles.

*3) **Comparison of Random Forest PageRank & similarity metrics vs Hadamard & embeddings (Link Prediction)**:* Despite all these features, Hadamard achieves only modest performance, with an average training `AUCPR` of 0.6966 and a `test score` of 0.6860, suggesting limited predictive power. In contrast, the second model with PageRank centrality scores (which capture global network influence),along with usage of similarity metrics like `cosine similarity` and `L2`. This modification yields the best results, achieving a training `AUCPR` of 0.9559 and a `test score` of 0.9411, demonstrating better discriminative ability. The PageRank model also generalizes more effectively, as evidenced by its higher validation score **(0.9107 vs. 0.6866)**, indicating robustness against overfitting. These findings highlight that global structural features (PageRank) along with similarity metrics can outperform localized embedding interactions (Hadamard) for link prediction in co-authorship networks. The performance gap suggests that future work should prioritize feature selection potentially combining centrality measures with similarity metrics rather than relying solely on embedding arithmetic.

C. Success Cases

- **Handling Missing Labels:** We eliminated infinite runs and limited training to labelled nodes by labelling only authors with `domain_id IS NOT NULL` as `:TrainAuthor`.
- **Feature and Hyperparameter Tuning:** Reasonable runtimes and several experiment iterations were made possible by reducing `embeddingDimension` (128 $\rightarrow$ 64), decision trees (50 $\rightarrow$ 25), validation folds (5 $\rightarrow$ 4), and test fraction.
- **Link Prediction Iterations:** To identify the best-performing alternative, we constructed and analysed five different future-collaboration pipelines: degree, PageRank, triangleCount, time-based Hadamard, and modifying embeddings.

D. Failure Cases

- **Node Classification Complexity:** Our first pipeline attempted to train and predict on all authors, including those with `domain_id = NULL`, by projecting the whole `CO_AUTHOR_WITH` graph. This resulted in jobs running for more than a day without convergence.
- **Incorrect Prediction Targets:** The model was "predicting" on the incorrect set and generating meaningless outputs in the early runs since predictions were streamed against unlabelled or incorrectly filtered nodes.
- **Train vs. Predict Graph Mismatch:** Results were further delayed when we unintentionally used `predict.stream` on the same in-memory graph that was used for training, skewing evaluation.
- **Version and Procedural Limitations:** We attempted node-regression pipelines, but Neo4j GDS 1.6.1 does not support `gds.beta.pipeline.nodeRegression.*`, forcing us to fall back on classification.
- **General Engineering and Debugging:** We struggled against datatype mismatches (String vs. Long) that broke projections, sporadic out-of-heap issues on huge graphs, and Cypher syntax deprecations when updating GDS.

E. Improvements and Extensions

We kept with fastRP, RandomForest, and a limited set of graph metrics in our present tests, and we restricted the embedding dimensions to 64–128. By increasing the embedding dimension or substituting alternative node-embedding methods like node2vec or DeepWalk, one could expand the feature space in the future. It would also be possible to include other structural measurements, such as betweenness centrality, clustering coefficient, and edge-weight characteristics based on topic overlap or co-publication counts. On the modelling side, richer train/test splits and more thorough hyperparameter sweeps (such as larger forests and various negative-sampling ratios) might aid in stabilising performance. Lastly, a workable way to further improve link-prediction accuracy is to investigate alternative unsupervised graph-embedding techniques.

V. Conclusion

This study shows that graph-based machine learning on a co-authorship network may reasonably predict future collaborations and identify author domains. When community and PageRank characteristics are added to fastRP embeddings for node classification, the max test accuracy is 55.99% (F1 = 0.5254). Even the most basic RF + communityId variation obtains 0.6872 AUPR for link prediction, whereas a RandomForest employing PageRank and triangle-count similarities achieves a test AUPR of 0.9411. These findings highlight the importance of integrating centrality measurements, meso-scale communities, and structural embeddings. To further improve performance, future expansions may investigate deeper temporal modelling, more graph metrics, and richer embeddings (such as node2vec).