Final Report on Link Prediction (TOPIC 13)

DORFNER THEO, Vienna University of Technology, Austria
KOTHMAYR LORENZ, Vienna University of Technology, Austria
KONSTANTOULA ELEONORA ZOI, Vienna University of Technology, Austria
LABBE UGO, Vienna University of Technology, Austria
FADLI FIRAS, Vienna University of Technology, Austria

Link prediction is a key problem in network science, with applications in recommendation systems, biological networks, and social media analysis. This study evaluates the performance, efficiency, and applicability of various similarity-based methods in different datasets. We explore local, global, and quasi-local indices, as well as reduction techniques such as random selection and centrality-based filtering. Our results show a strong relationship between algorithm performance and data set properties, such as size and structure. Using an end-to-end pipeline for data pre-processing, graph minimization, algorithm execution, and evaluation, we provide insights into optimal algorithm selection for real-world applications. Key findings highlight the trade-offs between computational efficiency and predictive accuracy, stressing the importance of algorithm reduction strategies. Among the methods tested, the spectral perturbation method and the local path index performed best, while the local similarity indices offered computational advantages for large networks. These findings contribute to a deeper understanding of link prediction, paving the way for future scalability studies.

ACM Reference Format:
Dorfner Theo, Kothmayr Lorenz, Konstantoula Eleonora Zoi, Labbe Ugo, and Fadli Firas. 2025. Final Report on Link Prediction (TOPIC 13). 1, 1 (January 2025), 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' Contact Information: Dorfner Theo, theo.dorfner@student.tuwien.ac.at, Vienna University of Technology, Vienna, Austria; Kothmayr Lorenz, e12024018@student.tuwien.ac.at, Vienna University of Technology, Vienna, Austria; Konstantoula Eleonora Zoi, e12243772@student.tuwien.ac.at, Vienna University of Technology, Vienna, Austria; Labbe Ugo, e12409106@student.tuwien.ac.at, Vienna University of Technology, Vienna, Austria; Fadli Firas, e12402599@student.tuwien.ac.at, Vienna University of Technology, Vienna, Austria.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
Manuscript submitted to ACM

1 Introduction

Link prediction is a key problem in network science, with applications in recommendation systems, biological networks, and social networks.ks. It involves identifying missing or future links in a graph based on its current structure. This study evaluates various link prediction methods to assess their performance, efficiency, and applicability in data sets of different sizes and types.

We focus on similarity-based methods (local, global, and quasi-local) and their performance variations across datasets and network reduction techniques. Our goal is to determine if a universally superior algorithm exists and how to balance predictive accuracy with computational efficiency. Additionally, we explore how dataset properties, such as size and structure, impact predictability, and compare our findings with prior research.

Using an end-to-end pipeline, we preprocess data, minimize graphs, execute algorithms, and evaluate metrics. We analyze local similarity methods (e.g., Resource Allocation), global methods (e.g., Random Walk with Restart), and quasi-local indices (e.g., Local Path), offering insights into optimal methodology selection for practical applications.

2 Theoretical Background

2.1 Overview

Various similarity-based methods for link prediction are examined, categorized into local, global, and quasi-local methods. Local methods use direct node information (e.g., CAR, LHNL), while global methods, like Random Walk with Restart (RWR), consider the entire network but are more computationally expensive. Quasi-local methods, such as the Local Path Index (LP), balance efficiency and network structure. Alternative approaches, like the Spectral Perturbation Method (SPM), are also discussed, with each method offering different trade-offs between accuracy and computational cost.

2.2 Similarity-based methods

Different similarity-based methods are explored, which are generally easy to implement and understand. These methods aim to compute a similarity score for each node pair to predict, or in some cases, for all possible pairs within the network.

2.2.1 Local similarity.

Local similarity methods are used by using direct information from nodes, such as their neighbors, degree, or neighbor overlap with other nodes.

- **CAR-based Common Neighbor Index (CAR)**:
  The CAR index algorithm predicts link strength in a graph $G = (V, E)$ by combining two structural properties: Common Neighbors (CN) and Local Clustering Level (LCL). For an edge (u, v), the number of common neighbors is computed as
  $$\mathrm{CN}(u, v) = |\Gamma(u) \cap \Gamma(v)|,$$
  where $\Gamma(u)$ represents the neighbors of node u. The LCL is defined as
  $$\mathrm{LCL}(u, v) = \frac{1}{2} \sum_{z \in \Gamma(u) \cap \Gamma(v)} |\Gamma(z) \cap \Gamma(u) \cap \Gamma(v)|,$$

quantifying the local connectivity among common neighbors. The CAR score is calculated as

$$\mathrm{CAR}(u, v) = \mathrm{CN}(u, v) \times \mathrm{LCL}(u, v),$$

providing a measure of the structural similarity between u and v.

- **Leicht–Holme–Newman Local Index (LHNL)**:
  The LHNL (Leicht-Holme-Newman for Link prediction) algorithm computes a score to predict links on a graph $G = (V, E)$ based on the number of Common Neighbors (CN) and the degrees of the source and target nodes. For an edge (u, v), the degrees of the nodes are denoted k_u and k_v, and the number of common neighbors is calculated as

  $$\mathrm{CN}(u, v) = |\Gamma(u) \cap \Gamma(v)|,$$

  where $\Gamma(u)$ represents the neighbors of node u. The LHNL score is then defined as

  $$\mathrm{LHNL}(u, v) = \frac{\mathrm{CN}(u, v)}{k_u \cdot k_v},$$

  where $k_u \cdot k_v$ represents the product of the degrees of u and v. If $k_u \cdot k_v = 0$, the score is set to 0 to handle edge cases. This score evaluates the likelihood of a link by balancing the number of shared neighbors with the node degree, normalizing the score for high-degree nodes.
- **Resource Allocation Index (RA)**:
  The Resource Allocation Index (RA) measures similarity between two non-adjacent vertices, x and y, based on resource transfer through their common neighbors. The similarity is calculated as:

  $$S(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{k_z},$$

  where $\Gamma(x)$ and $\Gamma(y)$ are the neighbor sets of nodes x and y, and k_z is the degree of common neighbor z. The RA index is similar to the Adamic/Adar index but applies a stronger penalty to high-degree nodes. It outperforms the Adamic/Adar index in heterogeneous networks with high clustering, making it particularly effective for link prediction in complex networks like transportation systems.
- **Hub Depressed Index (HDI)**:
  The Hub Depressed Index (HDI) discourages links between high-degree (hub) nodes and low-degree nodes. Unlike traditional methods, it promotes links between nodes with similar degrees, making it useful for balancing connectivity in networks.The HDI is calculated using the following mathematical expression:

  $$S(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{\max(k_x, k_y)}$$

  Here, $\Gamma(x)$ and $\Gamma(y)$ represent the sets of neighbors for nodes x and y, respectively, while k_x and k_y denote their degrees. The numerator $|\Gamma(x) \cap \Gamma(y)|$ counts the shared neighbors between x and y, while the denominator normalizes this count using the maximum degree between the two nodes.

2.2.2 Global similarity. Global methods rely on the entire network's structure to compute node similarity, often analyzing paths or eigenvalues. They offer a comprehensive view but are more computationally expensive than local and quasi-local methods.

- **Random Walk with Restart (RWR)**:
  The Random Walk with Restart (RWR) algorithm models a random walker that moves iteratively through a

network. At each step, the walker transitions to a random neighbor with probability α, or returns to the starting vertex with probability $1 - \alpha$. Let q_{xy} represent the probability that a walker starting from vertex x reaches vertex y in the steady-state. This probability is mathematically expressed as:

$$\vec{q}_x = \alpha P^T \vec{q}_x + (1 - \alpha)\vec{e}_x,$$

where $\vec{e}_x$ is the seed vector of length $|V|$ (i.e., the total number of vertices in the graph). The vector $\vec{e}_x$ has all components set to zero except for the x-th element, which is set to 1. The transition matrix P is defined as:

$$P_{xy} = \begin{cases} \frac{1}{k_x}, & \text{if } x \text{ and } y \text{ are connected,} \\ 0, & \text{otherwise,} \end{cases}$$

where k_x denotes the degree of vertex x. Simplifying the above equation, we can rewrite $\vec{q}_x$ as:

$$\vec{q}_x = (1 - \alpha)(I - \alpha P^T)^{-1}\vec{e}_x.$$

Since this similarity measure is not symmetric, the final similarity score between a pair of nodes (x, y) is calculated as:

$$S(x, y) = q_{xy} + q_{yx}.$$

It is evident from the equation that solving $\vec{q}_x$ involves matrix inversion, which can be computationally expensive and prohibitive for large networks.

2.2.3 Quasi-local similarity. Quasi-local methods are trying to be a trade-off between the 2 previous type of methods, they get topological information from the network while being less expensive than global methods.

- **Local Path Index (LP)**:
  The Local Path algorithm predicts the strength of links in a graph $G = (V, E)$ using a combination of the graph's adjacency matrix powers. Let A represent the adjacency matrix of G. The algorithm computes the LP matrix as

$$LP = A^2 + \epsilon A^3,$$

  where A^2 captures two-hop connections, A^3 captures three-hop connections, and ϵ is a weighting factor (e.g., $\epsilon = 0.5$) that controls the contribution of higher-order paths. For a given edge (u, v), the LP score is defined as

$$\text{LP}(u, v) = LP[i, j],$$

  where i and j are the indices of nodes u and v in A.

2.2.4 Other approach.

- **Spectral Perturbation Method (SPM)**
  The Spectral Perturbation Method (SPM) algorithm predicts the likelihood of links in a graph $G = (V, E)$ by leveraging spectral properties of the graph's adjacency matrix. Let A denote the adjacency matrix of G, with N nodes and L edges. The algorithm performs the following key steps:

(1) Edge Perturbation: A fraction p_H of the edges is randomly removed from G to create a perturbed graph G', with the corresponding perturbed adjacency matrix A_R. This step is repeated over epochs iterations to account for different perturbation scenarios.

(2) Spectral Decomposition: For each A_R, the eigenvalues λ_i and eigenvectors v_i of the matrix are computed. If k eigenvalues are specified, the computation is restricted to the largest k eigenvalues.

(3) Perturbation Correction: The difference matrix $\Delta A = A - A_R$ is used to adjust the eigenvalues. The correction term for each eigenvalue is calculated as:

$$\Delta\lambda_i = \frac{\sum_j \left(v_i^T \Delta A v_j\right) v_j}{\sum_j v_j^2}.$$

The adjusted eigenvalues are given by $\lambda_i + \Delta\lambda_i$, while the eigenvectors remain unchanged.

(4) Reconstruction: The perturbed adjacency matrix is updated as:

$$A_{\text{perturbed}} = \frac{1}{\text{epochs}} \sum_{t=1}^{\text{epochs}} V \Lambda_t V^T,$$

where V is the eigenvector matrix and Λ_t is the diagonal matrix of corrected eigenvalues for the t-th iteration.

(5) Edge Scoring: For a test edge (u, v), the SPM score is obtained as $\text{SPM}(u, v) = A_{\text{perturbed}}[i, j]$, where i and j are the indices of nodes u and v in A. If u or v are not in the graph, the score is set to 0.

(6) **Perturbation-based Prediction using NMF (NMF)**
We also tried to implement a method from a different paper [4]. The goal here was to implement a method that goes beyond the provided paper and see if we could beat SPM (as this was the claim of the paper). Specifically, we implemented the NMF-A2 version from the paper which combines network perturbation with non-negative matrix factorization (NMF) to predict links. The mathematics are quite detailed and high-level, but qualitatively the method behaves in the following way:

(a) Multiple times random "fake" connections are added to the existing network, leading to different perturbed versions

(b) For each version the structure is then broken down and analysed using NMF

(c) The results from this structural analysis for all altered networks is then used to create a similarity matrix for all pairs of nodes.

Unfortunately, the code provided was very messy and not commented/documented, so it's difficult to tell if we have implemented the algorithm correctly. Additionally, we also had to run NMF with low repetition cycles because of large runtime. Still, we only received results in very few cases and the method did not terminate most of the time. That's why it's not included in the results.

3 Approach

In this section the approach for this project will be described.

3.1 Pipeline Design

Basically, the task of the pipeline is to perform the different prediction algorithms. There are also other framework conditions that need to be taken into account. The pipeline has to process different data sets that are made available. Before the data can be processed, it must be prepared accordingly. Once the data has been prepared accordingly, you can proceed to the next step.

This involves the reduction of the data set. The data sets that were provided are of a size that cannot be easily processed on a normal laptop. Therefore, it was decided that a reduction of the provided data would be carried out in order to obtain a size of the data set that can be processed with a normal laptop. Once the size of the data set has been reduced, the next step can be carried out. This involves splitting the edges into training and test data. It was important not only to subdivide existing edges, but also to subdivide negative edges so that it can be assessed whether the prediction also performs negative edges correctly.

This means that the necessary preparations have been made to carry out the algorithms for link prediction. To do this, certain information is passed to the algorithm, and the result of the algorithm is an evaluation of the passed edges. Once the prediction algorithm has been successfully executed, the next step is to evaluate the quality of the prediction. To do this, the results of the prediction algorithm are passed to the metrics algorithms so that an assessment can be made. In order to save the generated data, it is stored in a CSV file for further analyses.

3.2 Dataset Description

3.2.1 Selected "Raw" Datasets. In our study, we focused on undirected and static networks for the link prediction task, similar to the approach taken by Kumar et al. This decision limited our dataset selection to five undirected networks from the original provided datasets. Due to our intention to use global algorithms and based on initial prediction test performance, we excluded the undirected combined dataset[1] from our analysis, as it would have required excessive computational resources. Despite the provision of additional datasets, we maintained our focus on the original four edge lists and did not incorporate any of the new data. We made this decision because we believed that the additional data would not have provided significant further insights or information for our study. As a result, our final dataset comprised four networks from "derStandard" and one random control graph. The descriptions of these networks were sourced from Bernhard Steindl's thesis.

Postings to same Articles. This dataset contains information about users who have posted in the comment section of the same article. The weight of the links reflects the number of news articles under which both users have posted. The dataset does not create links between users who have not yet posted under the same article and each article is counted maximally once. This results in an inherently symmetric dataset with regards to the links, since the connection to an article is the same for both users. This might not be relevant for the actual prediction, but as a result the graph provides a better description of the actual real-world network.

Votes to same Postings. Similar to the previous dataset, this one also contains identical actions by users to the same entity (and thus is symmetric with regards to links too). In this instance however, it's votes on the same postings (by other users). The weight is determined by whether or not the users vote in the same way (e.g. both voting positively). The reactions are then accumulated by subtracting the times the users voted differently from the number of times the users voted the same.

Votes to Postings. This dataset covers votes of users towards postings of other users. The edge weight results from the number of positive votes minus the number of negative votes, with a link only being created for strictly positive weights. This inherently represents and thus creates a directed network; for the undirected network edges (where there exist two edges in opposite directions) are consolidated by adding them together. This effectively means that there is (significant) information lost through the transformation from directed to undirected graph. One could thus expect the prediction algorithms to perform worse for this dataset.

[1]`"df_edge_list_undirected_users_combined_postings_replies_and_votes_to_postings_net"`

Posting Replies. Users can also reply to postings, which is covered in this dataset. The edge weight corresponds to the number of replies from one user to the other. Similar to the previous dataset, this initially leads to a directed graph and needs to be consolidated by adding the edge weights (in instances where two edges in opposite directions exist). Once again, this mean that information is lost in generating predictions which could reduce the quality of prediction

Random ER Network for Control. Lastly, in order to provide a reference and control network we added a random network, specifically an Erdős–Rényi (ER) graph. In order to keep the graphs as similar as possible (while still being random), we used the *votes to postings* dataset as reference for the number of nodes (12377) and edge density (0.0705). We selected this network since it's reasonably small and contains the full network information (symmetry).

3.3 Reduction Methods

In order to be able to use the given datasets, we had to (significantly) reduce them since we wanted to also test global methods which usually have strong-scaling complexity. The reduction methods we use effectively all work in the same way: either nodes or edges are sorted by some metric and afterwards we cut parts of the sorted list. Specifically, we sorted edges by their weight and nodes both by their degree and their node centrality. For each of those three sortings, we then performed the following four selections

(1) Select every nth edge/nodes (n is determined by the reduction percentage)
(2) Select the top, bottom or middle/centered n edges/nodes (again, n is determined by the amount of reduction)

Lastly, we also implemented a control reduction method, that simply cut edges at random. For comparisons of the reduction methods with the original networks, see the Appendix

Comparison with original Datasets. Due to computational constraints of computing network statistics (especially global ones) on large networks and the already extensive nature of our report, we only performed a detailed comparison of network properties for one original dataset with most of our reduction mechanisms. Despite these limitations, we can still gain valuable insights from the analysis. The results suggest that our reduction methods generally preserve key structural properties of the original network, albeit with some notable differences. Top-based methods, particularly those using centrality and degree, appear to maintain higher levels of clustering and connectivity compared to other approaches. While the average degree in all reduced networks is singificantly lower than the original (largely understandable since we significantly reduce the network size), top-based methods can still retain about 55% of the original connectivity. Interestingly, the modularity values remain consistently low across all versions, indicating a preservation of community structure. The degree distribution shapes are largely maintained, though with expected lower frequencies in the reduced networks. Obviously, these findings are limited as they we only looked at one "raw" dataset and a few network metrics, but given that the focus of this project lies on link prediction we think this provides enough insight and justification for our reduction mechanisms. Additionally, a comparison of the different reduction mechanisms with regards to prediction performance is also included in the results section.

3.4 Operationalization

In order for the pipeline to work, the different steps had to be take the same input and output. Each prediction methods was given a graph, a set of existing (positive) links and a set of non-existing (negative) links. All the methods then output 2 lists, a list of scores for the positive edges and a list for the negative edges. In order to measure performances, all the different metrics were given the 2 sets of edges (positive and negative), as well as the 2 corresponding lists from the prediction algorithm.

3.5 Metrics

To evaluate the performance of various link prediction algorithms, several metrics were used. These metrics provide a comprehensive understanding of the quality of the predictions and enable comparisons between methods. Below is a detailed explanation of the metrics used:

Evaluation Metrics

- **Area Under Precision-Recall Curve (AUPR):** AUPR evaluates the balance between precision and recall across various thresholds. It highlights how well the algorithm identifies true positive links among all predicted positives.
- **Area Under Receiver Operating Characteristic Curve (AUROC):** AUROC measures the trade-off between the true positive rate and false positive rate. It provides an overall assessment of the algorithm's ability to distinguish between linked and unlinked nodes in the network.
- **Mean Reciprocal Rank (MRR):** MRR calculates the average reciprocal rank of the first correctly predicted link for each test case. It reflects how early a correct prediction appears in a ranked list, emphasizing the algorithm's ranking performance.
- **Precision@5 and Recall@5:** Precision@5 evaluates the proportion of true positives in the top 5 predictions, while Recall@5 measures the fraction of actual positives captured in the top 5. These metrics assess the quality of predictions for the most critical edges.

4 Results

4.1 Performance of Algorithms

The performance of the algorithms varies depending on the characteristics of the dataset, reduction methods, and computational trade-offs. In general, the evaluation reveals consistent trends in prediction quality and resource usage across datasets. For example, some algorithms maintain a balance between efficiency and accuracy, while others achieve high prediction quality under specific parameter configurations, at the expense of computational cost. These findings highlight the importance of selecting algorithms and reduction strategies based on the specific constraints and goals of the task.

Figure 1 demonstrates the interplay between computational efficiency and prediction quality. Certain algorithms exhibit consistent performance across diverse configurations, while others achieve high accuracy but demand greater computational resources. This trade-off underlines the need to balance performance and efficiency depending on application requirements.

4.2 Trade-Offs Between Prediction Accuracy and Efficiency

Reduction methods significantly influence algorithm performance. By analyzing their effects, we observe patterns of improvement or trade-offs that vary across datasets. Reduction strategies can improve predictive quality while reducing resource consumption, though these benefits depend on the compatibility between algorithms and datasets.

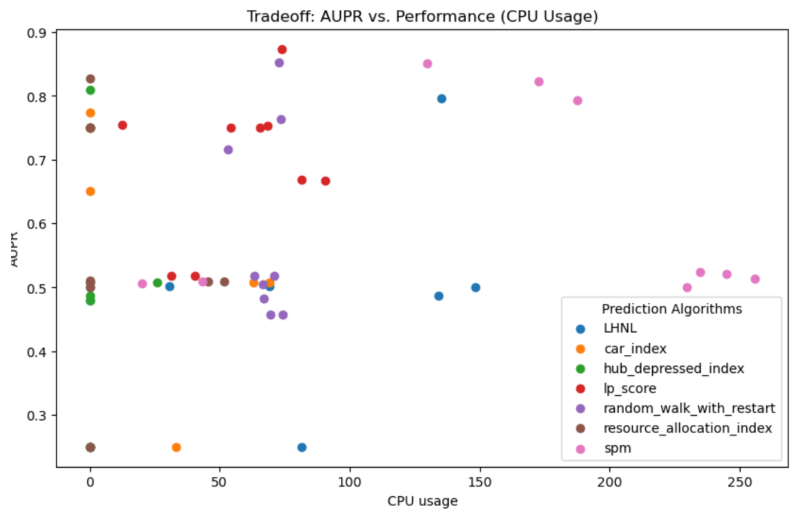

Fig. 1. Scatterplot showing the trade-off between prediction accuracy (AUPR) and CPU usage across datasets, algorithms, and reduction methods. Each point represents a unique dataset-reduction combination.

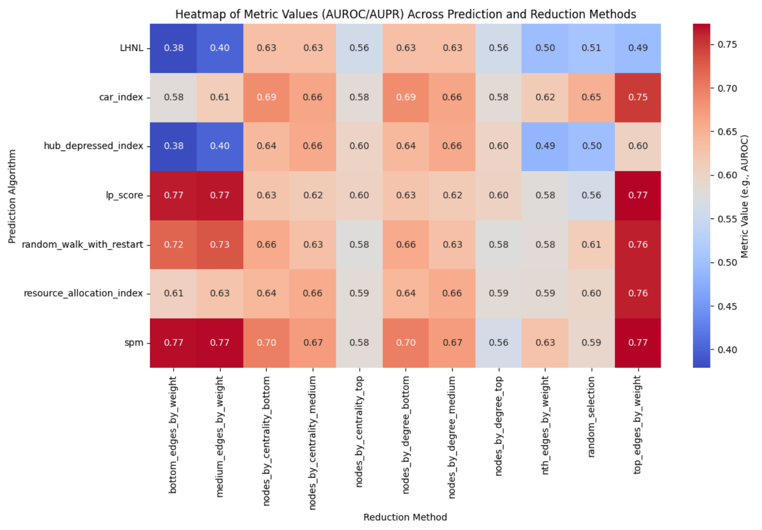

Fig. 2. Heatmap of AUROC values across prediction algorithms, reduction methods.

As shown in Figure 2, the heat map highlights the variability in performance in combinations of reduction methods. Some algorithms show robustness to different reductions, achieving stable performance, while others are more sensitive, with results fluctuating significantly based on the reduction method applied. These observations underscore the critical role of reduction strategies in influencing outcomes.

4.3 Analysis of Dataset-Specific Behaviors

Algorithms also exhibit distinct behaviors depending on the datasets they are applied to, with some showing consistent performance across all datasets, while others excel in specific cases. Figure 3 illustrates this variability.

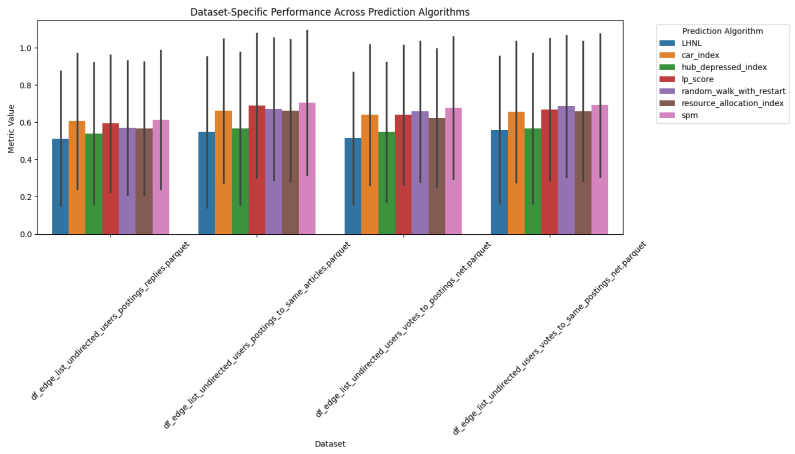

Fig. 3. Performance across datasets, evaluated using AUROC and AUPR metrics. Each dataset-algorithm combination highlights the interaction between dataset properties and algorithm behavior.

As seen in Figure 3, certain datasets align well with specific algorithms, leading to higher prediction quality. In contrast, other algorithms demonstrate broad applicability, delivering consistent results across datasets regardless of their characteristics. This variability points to the importance of understanding dataset-specific properties when choosing prediction algorithms.

The results demonstrate the complex interplay between algorithms, reduction methods, and datasets. These findings provide a foundation for further exploration of correlations between dataset properties and algorithm performance, as well as comparisons with prior research. In the following section, we discuss the broader implications of these results, outline limitations, and suggest directions for future work.

5 Discussion

5.1 Key Findings and Insights

Our results demonstrate that algorithm performance varies significantly depending on dataset properties and reduction techniques. Although the absolute performance values differ across datasets, the relative ranking of the algorithms remains consistent when compared to the result of the paper. The dataset properties influence their absolute effectiveness. This finding aligns with existing literature, which emphasizes the sensitivity of link prediction models to data structure and distribution.

Moreover, reduction techniques play a crucial role in shaping algorithm outcomes. Notably, reductions such as *Top Edges by Weight*, *Medium Edges by Weight*, and *Bottom Edges by Weight* consistently yield better results compared to other reduction methods. These techniques likely preserve key structural information that is critical for predicting links effectively.

5.2 Correlation Between Dataset Properties and Algorithm Performance

As shown in our experiments, the structural properties of datasets heavily influence algorithm performance. This observation underscores the need to better understand the relationship between dataset characteristics and algorithm performance. Heatmaps illustrating AUROC and AUPR values across different datasets (**Figure 2**) highlight these performance discrepancies, showing that some algorithms perform consistently well, while others are more specific to certain type of data.

These findings are supported by previous studies, which suggest that structural features like sparsity, clustering coefficients, and degree distributions are strong determinants of link prediction effectiveness [2, 3]. Future work could further analyze these relationships to create a framework for matching algorithms to datasets based on their structural properties.

Interestingly, some algorithms manage to maintain a balance between accuracy and efficiency, making them suitable for large-scale applications. This observation suggests that such algorithms may be ideal for scenarios where computational resources are constrained, while higher-cost methods may be reserved for tasks that require maximum predictive accuracy.

5.3 Comparison with Prior Work

Our findings complement and extend prior research by providing a detailed empirical evaluation of link prediction algorithms across diverse datasets and reduction methods. Previous studies have highlighted that local and quasi-local approaches generally perform well, while global approaches, which rely on exploring paths, tend to be computationally complex and introduce additional noise into the networks.[1].

Furthermore, identifying consistent trends in algorithm ranking, despite variations in absolute performance, provides valuable insight into the generalizability of these methods.

5.4 Future Directions

Future research could build on these findings by exploring new reduction methods, incorporating more diverse datasets, and developing theoretical frameworks to explain the observed trends. Additionally, investigating the scalability of these approaches for real-world applications, such as recommendation systems. Lastly, integrating these insights into automated systems for algorithm and reduction selection represents an exciting avenue for further exploration.

6 Conclusion

In this paper, we evaluated various dataset reduction techniques and link prediction methods, ranging from local to global similarity measures, as well as exploring more unconventional approaches. Our findings reveal that the performance of these algorithms varies significantly depending on the network structure and the reduction technique employed. Among the tested methods, the Spectral Perturbation Method and the Local Path Index emerged as the most effective link prediction techniques, while local similarity methods, such as the Leicht–Holme–Newman Local Index and the Hub Depressed Index, generally performed worse. However, despite their lower predictive performance, local methods offer a distinct advantage in terms of computational efficiency, making them suitable for large-scale networks. Overall, our results align closely with the findings of the initial reference paper, as the ranking of methods across the various metrics used was consistent, reinforcing the validity of our analysis.

7 Organization of Group Work

In this paragraph, the goal is to briefly outline the distribution of the group's work:

- **Lorenz:** Lorenz was responsible for the initial testing of link prediction methods, the implementation of the pipeline, and support for generating results. Lorenz also designed and provided the interface for various functions, enabling seamless integration of different components.
- **Theo:** Theo contributed to both the theoretical background and data reduction strategies. He summarized the theoretical framework from the provided paper, discussed baseline similarity-based methods, and analyzed dataset properties. For data reduction, Theo worked on implementing and evaluating strategies such as random row selection and subsetting based on properties like node degree and centrality. Additionally, he implemented the NMF-A2 algorithm
- **Ugo:** Ugo focused on implementing algorithms such as CAR, LHNL, SPM and Local Path while contributing to operationalizing the pipeline and producing descriptive statistics of the datasets. Ugo also collaborated on designing and testing dataset reduction strategies, ensuring they aligned with the project goals.
- **Eleonora:** Eleonora focused on implementing and analyzing local similarity-based algorithms, such as Resource Allocation (RA), Hub Depressed Index (HDI), and Random Walk with Restart (RWR). Additionally, Eleonora worked on the abstract and introduction sections of the report and ensured that results were well-documented for these algorithms.
- **Firas:** Firas handled the design and calculation of metrics to evaluate the performance of the algorithms. They also worked extensively on analyzing and visualizing the results, comparing prediction quality, runtime, and efficiency. Firas summarized key insights for the discussion and conclusion sections of the report.

References

[1] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. 2020. Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications* 553 (2020), 124289. doi:10.1016/j.physa.2020.124289

[2] Y. Rong, H. Zhi, et al. 2022. Clustering Coefficient as a Criterion for Link Prediction. *Applied Network Science* 7, 12 (2022). doi:10.1007/s41109-022-00471-1

[3] K. Smith and A. Johnson. 2025. Graph Density and Algorithm Selection in Sparse Networks. *ArXiv preprint arXiv:2501.06721* (2025). https://arxiv.org/html/2501.06721v1

[4] Wenjun Wang, Fei Cai, Pengfei Jiao, and Lin Pan. 2016. A perturbation-based framework for link prediction via non-negative matrix factorization. *Scientific Reports* 6 (2016). https://api.semanticscholar.org/CorpusID:20296452