PaGE-Link: Path-based Graph Neural Network Explanation for Heterogeneous Link Prediction

Shichang Zhang[1*], Jiani Zhang[2], Xiang Song[2], Soji Adeshina[2], Da Zheng[2], Christos Faloutsos[23], Yizhou Sun[12]
[1]University of California, Los Angeles, [2]Amazon, [3]Carnegie Mellon University
[1]{shichang, yzsun}@cs.ucla.edu, [2]{zhajiani, xiangsx, adesojia, dzzhen}@amazon.com, [3]christos@cs.cmu.edu

ABSTRACT

Transparency and accountability have become major concerns for black-box machine learning (ML) models. Proper explanations for the model behavior increase model transparency and help researchers develop more accountable models. Graph neural networks (GNN) have recently shown superior performance in many graph ML problems than traditional methods, and explaining them has attracted increased interest. However, GNN explanation for link prediction (LP) is lacking in the literature. LP is an essential GNN task and corresponds to web applications like recommendation and sponsored search on web. Given existing GNN explanation methods only address node/graph-level tasks, we propose Path-based GNN Explanation for heterogeneous Link prediction (*PaGE-Link*) that generates explanations with *connection interpretability*, enjoys model *scalability*, and handles graph *heterogeneity*. Qualitatively, PaGE-Link can generate explanations as paths connecting a node pair, which naturally captures connections between the two nodes and easily transfer to human-interpretable explanations. Quantitatively, explanations generated by PaGE-Link improve AUC for recommendation on citation and user-item graphs by *9 - 35%* and are chosen as better by *78.79%* of responses in human evaluation.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; • **Mathematics of computing → Graph algorithms**.

KEYWORDS

Model Transparency, Model Explanation, Graph Neural Networks, Link Prediction

ACM Reference Format:
Shichang Zhang, Jiani Zhang, Xiang Song, Soji Adeshina, Da Zheng, Christos Faloutsos, and Yizhou Sun. 2023. PaGE-Link: Path-based Graph Neural Network Explanation for Heterogeneous Link Prediction. In *Proceedings of the ACM Web Conference 2023 (WWW '23), May 1–5, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583511

*Work done while being an intern at Amazon Web Services

WWW '23, May 1–5, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04.
https://doi.org/10.1145/3543507.3583511

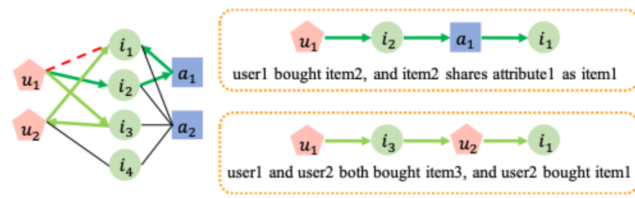


Figure 1: Given a GNN model and a predicted link (u_1, i_1) (dashed red) on a heterogeneous graph of user u, item i, and attribute a (left). PaGE-Link **generates two path explanations (green arrows). We illustrate the interpretations on the right.**

1 INTRODUCTION

Transparency and accountability are significant concerns when researchers advance black-box machine learning (ML) models [19, 33]. Good explanations of model behavior improve model transparency. For end users, explanations make them trust the predictions and increase their engagement and satisfaction [1, 10]. For researchers and developers, explanations enable them to understand the decision-making process and create accountable ML models. Graph Neural Networks (GNNs) [41, 53] have recently achieved state-of-the-art performance on many graph ML tasks and attracted increased interest in studying their explainability [25, 43, 45, 50]. However, to our knowledge, GNN explanation for link prediction (LP) is missing in the literature. LP is an essential task of many vital Web applications like recommendation [26, 40, 47] and sponsored search [9, 20]. GNNs are widely used to solve LP problems [48, 54], and generating good GNN explanations for LP will benefit these applications, e.g., increasing user satisfaction with recommended items.

Existing GNN explanation methods have addressed node/graph-level tasks on homogeneous graphs. Given a data instance, most methods generate an explanation by learning a mask to select an edge-induced subgraph [25, 43] or searching over the space of subgraphs [46]. However, explaining GNNs for LP is a new and more challenging task. Existing node/graph-level explanation methods do not generalize well to LP for three challenges. 1) *Connection Interpretability*: LP involves a pair of the source node and the target node rather than a single node or graph. Desired interpretable explanations for a predicted link should reveal connections between the node pair. Existing methods generate subgraphs with no format constraints, so they are likely to output subgraphs disconnected from the source, the target, or both. Without revealing connections between the source and target, these subgraph explanations are hard for humans to interpret and investigate. 2) *Scalability*: For LP, the number of edges involved in GNN computation almost grows from m to $\sim 2m$ compared to the node prediction task because neighbors

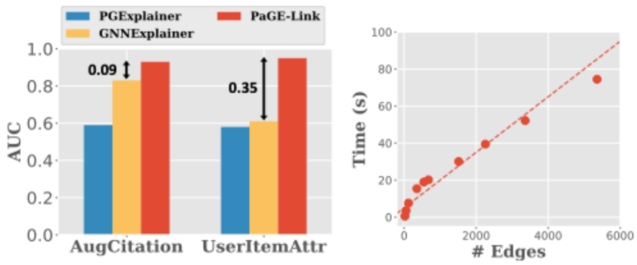


Figure 2: (a) PaGE-Link **outperforms GNNExplainer and PG-Explainer in terms of explanation AUC on the citation graph and the user-item graph. (b) The running time of** PaGE-Link **scales linearly in the number of graph edges.**

of both the source and the target are involved. Since most existing methods consider all (edge-induced) subgraphs, the increased edges will scale the number of subgraph candidates by a factor of $O(2^m)$, which makes finding the optimal subgraph explanation much harder. 3) *Heterogeneity*: Practical LP is often on heterogeneous graphs with rich node and edge types, e.g., a recommendation graph can have user->buys->item edges and item->has->attribute edges, but existing methods only work for homogeneous graphs.

In light of the importance and challenges of GNN explanation for LP, we formulate it as a post hoc and instance-level explanation problem and generate explanations for it in the form of important paths connecting the source and target nodes. Paths have played substantial roles in graph ML and are the core of many non-GNN LP methods [15, 16, 21, 34]. Paths as explanations can solve the connection interpretability and scalability challenges. Firstly, paths connecting two nodes naturally explain connections between them. Figure 1 shows an example on a recommendation graph. Given a GNN and a predicted link between user u_1 and item i_1, human-interpretable explanations may be based on user's preference of attributes (e.g., user u_1 bought item $i2$ that shared the same attribute a_1 as item i_1) or collaborative filtering (e.g, user u_1 had similar preference as user u_2 because they both bought item i_3 and user u_2 also bought item i_1, so that user u_1 would like item i_1). Both explanations boil down to paths. Secondly, paths have a considerably smaller search space than general subgraphs. As we will see in Proposition 4.1, compared to the expected number of edge-induced subgraphs, the expected number of paths grows strictly slower and becomes negligible. Therefore, path explanations exclude many less-meaningful subgraph candidates, making the explanation generation much more straightforward and accurate.

To this end, we propose Path-based GNN Explanation for heterogeneous Link prediction (PaGE-Link), which achieves a better explanation AUC and scales linearly in the number of edges (see Figure 2). We first perform *k-core pruning* [2] to help find paths and improve scalability. Then we do *heterogeneous path-enforcing* mask learning to determine important paths, which handles heterogeneity and enforces the explanation edges to form paths connecting source to target. In summary, the contributions of our method are:

- **Connection Interpretability:** PaGE-Link produces more interpretable explanations in path forms and quantitatively improves explanation AUC over baselines.

Table 1: Methods and desired explanation properties. A question mark (?) means "unclear", or "maybe, after non-trivial extensions". "Rec. Exp." stands for the general recommendation explanation methods.

Methods	GNNExp [43]	PGExp [25]	SubgraphX [46]	J-RECS [28]	PRINCE [6]	Rec. Exp. [51]	PaGE-Link
On Graphs	✓	✓	✓	✓	✓	?	✓
Explains GNN	✓	✓	✓				✓
Explains LP	?	?	?	✓	✓	✓	✓
Connection				?	?	?	✓
Scalability	✓	✓		✓	?	?	✓
Heterogeneity			✓	✓	✓	?	✓

- **Scalability**: PaGE-Link reduces the explanation search space by magnitudes from subgraph finding to path finding and scales linearly in the number of graph edges.
- **Heterogeneity:** PaGE-Link works heterogeneous graphs and leverages edge-type information to generate better explanations.

As an additional contribution, we create augmented-real and synthetic graphs to benchmark GNN explanation for LP. We also plan to open-source our code with the data generator.

2 RELATED WORK

We review relevant research on (a) GNNs (b) GNN explanation (c) recommendation explanation and (d) paths for LP. We summarize the properties of PaGE-Link vs. representative methods in Table 1.

GNNs. GNNs are a family of ML models on graphs [17, 36, 42]. They take graph structure and node/edge features as input and output node representations by transforming and aggregating features of nodes' (multi-hop) neighbors. The node representations can be used for LP and achieved great results on LP applications [7, 26, 40, 47–49, 52]. We review GNN-based LP models in Section 3.

GNN explanation. GNN explanation was studied for node and graph classification, where the explanation is defined as an important subgraph. Existing methods majorly differ in their definition of importance and subgraph selection method. Mutual information (MI) and edge mask learning are popular ways proposed by GNNExplainer [43]. Fully parameterized masks on graph edges and node features are learned to maximize the MI between the masked graph and the prediction made with the original graph. PGExplainer [25] adopts the same MI importance and mask-learning idea, but it trains a mask predictor to generate a discrete mask. Another popular importance is game theory values. SubgraphX [46] uses the Shapley value [32] and performs Monte Carlo Tree Search (MCTS) on subgraphs. GStarX [50] uses a structure-aware HN value [8] to measure the importance of nodes and generates the important-node-induced subgraph. There are more studies from other perspectives but are less related to this work, i.e., surrogate models [12, 37], counterfactual explanations [24], and causality [22, 23]. [44] provides a good review. While all these methods produce subgraphs as explanations, what makes a good explanation is a complex topic, especially about how to meet "stakeholders' desiderata" [18]. Our work is different from all above because we focus on a new task of explaining heterogeneous LP, and we generate paths instead of unrestricted subgraphs as explanations for better interpretability.

Recommendation explanation. This line of works explains why a recommendation is made [51]. J-RECS [28] generates recommendation explanations on product graphs using a justification score that balances item relevance and diversity. PRINCE [6] produces end-user explanations as a set of minimal actions performed by the user on graphs with users, items, reviews, and categories. The set of actions is selected using counterfactual evidence. Typically, recommendations on graphs can be formalized as an LP task. However, the recommendation explanation problem differs from explaining GNNs for LP because the recommendation data may not be graphs, and the models to be explained are primarily not GNN-based [38]. GNNs have their unique message passing step compared to general recommendation systems, and LP for GNNs corresponds to more general applications beyond recommendation, e.g., drug repurposing [13], and knowledge graph completion [3, 27]. Since GNN is the core model we try to explain, the recommendation explanation is related but not directly comparable to our method.

Paths. Many LP methods are path-based, such as graph distance [21], Katz index [16], SimRank [15], PathSim [34] and so on. The "connection subgraphs" algorithm [5] finds good paths between the source and target nodes, based on electricity analogs, but works only on homogeneous graphs. Although GNN methods have better accuracy, we embrace paths for explainability.

3 NOTATIONS AND PRELIMINARY

In this section, we define necessary notations, summarize them in Table 2, and review the GNN-based LP model.

Definition 3.1. A heterogeneous graph is defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and an edge type mapping function $\tau : \mathcal{E} \rightarrow \mathcal{R}$. Each node $v \in \mathcal{V}$ belongs to one node type $\phi(v) \in \mathcal{A}$ and each edge $e \in \mathcal{E}$ belongs to one edge type $\tau(e) \in \mathcal{R}$.

Let $\Phi(\cdot,\cdot)$ denote a trained GNN-based model for predicting the missing links in $\mathcal{G}$, where a prediction $Y = \Phi(\mathcal{G}, (s,t))$ denotes the predicted link between a source node s and a target node t. The model Φ learns a conditional distribution $P_\Phi(Y|\mathcal{G}, (s,t))$ of the binary random variable Y. The commonly used GNN-based LP models [48, 52, 54] involve two steps. The first step is to generate node representations $(\boldsymbol{h}_s, \boldsymbol{h}_t)$ of (s,t) with an L-hop GNN encoder. The second step is to apply a prediction head on $(\boldsymbol{h}_s, \boldsymbol{h}_t)$ to get Y. An example prediction head is the inner product operation.

To explain $\Phi(\mathcal{G}, (s,t))$ with an L-Layer GNN encoder, we restrict to the *computation graph* $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$. $\mathcal{G}_c$ is the L-hop ego-graph of the predicted pair (s,t), i.e., the subgraph with node set $\mathcal{V}_c = \{v \in V | dist(v,s) \leq L \text{ or } dist(v,t) \leq L\}$. It is called computation graph because the L-layer GNN only collects messages from the L-hop neighbors of s and t to compute $\boldsymbol{h}_s$ and $\boldsymbol{h}_t$. The LP result is thus fully determined by $\mathcal{G}_c$, i.e., $\Phi(\mathcal{G}, (s,t)) \equiv \Phi(\mathcal{G}_c, (s,t))$. Figure 3b shows a 2-hop ego-graph of u_1 and i_1. u_3 and a_3^1 are excluded since they are more than 2 hops away from either u_1 or i_1.

Table 2: Notation table

Notation	Definition and description
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	a heterogeneous graph $\mathcal{G}$, node set $\mathcal{V}$, and edge set $\mathcal{E}$
$\phi : \mathcal{V} \rightarrow \mathcal{A}$	a node type mapping function
$\tau : \mathcal{E} \rightarrow \mathcal{R}$	an edge type mapping function
D_v	the degree of node $v \in \mathcal{V}$
$\mathcal{E}^r$	edges with type $r \in \mathcal{R}$, i.e., $\mathcal{E}^r = \{e \in \mathcal{E} | \tau(e) = r\}$
$\Phi(\cdot,\cdot)$	the GNN-based LP model to explain
(s,t)	the source and target node for the predicted link
$\boldsymbol{h}_s$ & $\boldsymbol{h}_t$	the node representations for s & t
$Y = \Phi(\mathcal{G}, (s,t))$	the link prediction of the node pair (s,t)
$\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$	the computation graph, i.e., L-hop ego-graph of (s,t)

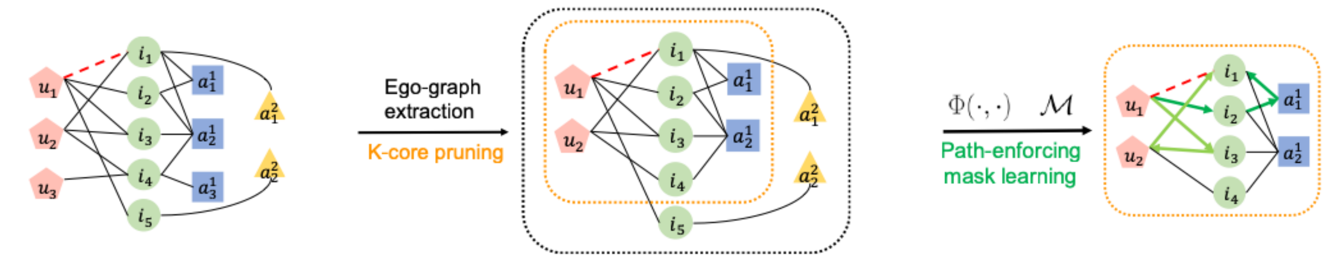
4 PROPOSED PROBLEM FORMULATION: LINK-PREDICTION EXPLANATION

In this work, we address a *post hoc* and *instance-level* GNN explanation problem. The post hoc means the model $\Phi(\cdot,\cdot)$ has been trained. To generate explanations, we won't change its architecture nor parameters. The instance level means we generate an explanation for each instance (an (s,t) pair and their $\mathcal{G}_c$) and explain why the model makes such a prediction. Specifically, the explanation method answers the question of why a missing link is predicted by $\Phi(\cdot,\cdot)$. In a practical web recommendation system, this question can be "*why an item is recommended to a user by the model*".

An explanation for a GNN prediction should be some substructure in $\mathcal{G}_c$, which should also be concise, i.e., limited by a size budget B. This is because an explanation with a large size is often neither informative nor interpretable, for example, an extreme case is that $\mathcal{G}_c$ could be a non-informative explanation for itself. Also, a fair comparison between different explanations should consume the same budget. In the following, we define budget B as the maximum number of edges included in the explanation.

We list three desirable properties for a GNN explanation method on heterogeneous LP: capturing the connection between the source node and the target node, scalable to large graphs, and addressing graph heterogeneity. Using a path-based method inherently possesses all the properties. Paths capture the connection between a pair of nodes and can be transferred to human-interpretable explanations. Besides, the search space of paths with the fixed source node and the target node is greatly reduced compared to edge-induced subgraphs. Given the ego-graph $\mathcal{G}_c$ of s and t, the number of paths between s and t and the number of edge-induced subgraphs in $\mathcal{G}_c$ both rely on the structure of $\mathcal{G}_c$. However, they can be estimated using random graph approximations. The next proposition on random graphs shows that the expected number of paths grows strictly slower than the expected number of edge-induced subgraphs as the random graph grows. Also, the expected number of paths becomes insignificant for large graphs.

Proposition 4.1. *Let $\mathcal{G}(n,d)$ be a random graph with n nodes and density d, i.e., there are $m = d\binom{n}{2}$ edges chosen uniformly randomly from all node pairs. Let $Z_{n,d}$ be the expected number of paths between any pair of nodes. Let $S_{n,d}$ be the expected number of edge-induced subgraphs. Then $Z_{n,d} = o(S_{n,d})$, i.e., $\lim_{n\to\infty} \frac{Z_{n,d}}{S_{n,d}} = 0$.*



(a) A GNN predicted link (u_1, i_1) (dashed red) that needs explanation.

(b) Extract 2-hop ego-graph of (u_1, i_1) excluding u_3 and a^1_3 (black box). Then prune it to get the k-core excluding i_5, a^2_1, and a^2_2 (orange box).

(c) Human-interpretable path explanations (u_1, i_2, a^1_1, i_1) and (u_1, i_3, u_2, i_1) (green arrows) that capture the connection between u_1 and i_1.

Figure 3: PaGE-Link **on a graph with user nodes u, item nodes i, and two attribute types a^1 and a^2. (Best viewed in color.)**

Proof. In Appendix A. ■

Paths are also a natural choice for explanations on heterogeneous graphs. On homogeneous graphs, features are important for prediction and explanation. A s-t link may be predicted because s and t their node feature similarity. However, the heterogeneous graphs we focus on, as defined in Definition 3.1, often do not store feature information but explicitly model it using new node and edge types. For example, for the heterogeneous graph in Figure 3a, instead of making it a user-item graph and assigning each item node a two-dimensional feature with attributes a^1 and a^2, the attribute nodes are explicitly created and connected to the item nodes. Then an explanation like "i_1 and i_2 share node feature a^1_1" on homogeneous graphs should be transferred to "i_1 and i_2 are connected through the attribute node a^1_1" on such heterogeneous graphs.

Given the advantages of paths over general subgraphs on connection interpretability, scalability, and their capability to capture feature similarity on heterogeneous graphs, we use paths to explain GNNs for heterogeneous LP. Our design principle is that a good explanation should be concise and informative, so we define the explanation to contain only *short* paths *without high-degree* nodes. Long paths are less desirable since they could correspond to unnecessarily complicated connections, making the explanation neither concise nor convincing. For example, in Figure 3c, the long path $(u_1, i_3, a^1_2, i_2, a^1_1, i_1)$ is not ideal since it takes four hops to go from item i_3 to item i_1, making it less persuasive to be interpreted as "item1 and item3 are similar so item1 should be recommended". Paths containing high-degree nodes are also less desirable because high-degree nodes are often generic, and a path going through them is not as informative. In the same figure, all paths containing node a^1_2 are less desirable because a^1_2 has a high degree and connects to all the items in the graph. A real example of a generic attribute is the attribute "grocery" connecting to both "vanilla ice cream" and "vanilla cookie". When "vanilla ice cream" is recommended to a person who bought "vanilla cookie", explaining this recommendation with a path going through "grocery" is not very informative since "grocery" connects many items. In contrast, a good informative path explanation should go through the attribute "vanilla", which only connects to vanilla-flavored items and has a much lower degree.

We formalize the GNN explanation for heterogeneous LP as:

Problem 4.2. Generating path-based explanations for a predicted link between node s and t:

- **Given**
  - a trained GNN-based LP model $\Phi(\cdot, \cdot)$,
  - a heterogeneous computation graph $\mathcal{G}_c$ of s and t,
  - a budget B of the maximum number of edges in the explanation,
- **Find** an explanation $\mathcal{P} = \{p | p$ is a s-t path with maximum length l_{max} and degree of each node less than $D_{max}\}$,
- **By optimizing** $p \in \mathcal{P}$ to be influential to the prediction, concise, and informative.

5 PROPOSED METHOD: PAGE-LINK

This section details PaGE-Link. PaGE-Link has two innovations: (i) a k-core pruning module to eliminate spurious neighbors and improve speed, and (ii) a heterogeneous path-enforcing mask learning module to identify important paths. An illustration is in Figure 3.

5.1 The k-core Pruning

The *k-core pruning* algorithm of PaGE-Link reduces the complexity of $\mathcal{G}_c$. The k-core of a graph is defined as the unique maximal subgraph with a minimum node degree k. We use the superscript k to denote the k-core, i.e., $\mathcal{G}^k_c = (\mathcal{E}^k_c, \mathcal{V}^k_c)$ for the k-core of $\mathcal{G}_c$. The k-core pruning is a recursive algorithm that removes nodes $v \in \mathcal{V}$ such that their degrees $D_v < k$, until the remaining subgraph only has nodes with $D_v \geq k$, which gives the k-core. The difference in nodes between a $(k+1)$-core and a k-core is called the k-shell. The nodes in the orange box of Figure 3b is an example of a 2-core pruned from the 2-hop ego-graph, where node a^2_1 and a^2_2 are pruned in the first iteration because they are degree one. Node i_5 is recursively pruned because it becomes degree one after node a^2_2 is pruned. All three nodes belong to the 1-shell. We perform k-core pruning to help path finding because the pruned k-shell nodes are unlikely to be part of meaningful paths when k is small. For example, the 1-shell nodes are either leaf nodes or will become leaf nodes during the recursive pruning, which will never be part of a path unless s or t are one of these 1-shell nodes. The k-core pruning module in PaGE-Link is modified from above by adding a condition of never pruning s and t.

The following theorem shows that for a random graph $\mathcal{G}(n, d)$, k-core will reduce the expected number of nodes by a factor of $\delta_{\mathcal{V}}(n, d, k)$ and reduce the expected number of edges by a factor of

$\delta_{\mathcal{E}}(n, d, k)$. Both factors are functions of n, d, and k. We defer the exact expressions of these two factors in Appendix B, since they are only implicitly defined based on Poisson distribution. Numerically, for a random $\mathcal{G}(n, d)$ with average node degree $d(n-1) = 7$, its 5-core has $\delta_{\mathcal{V}}(n, d, 5)$ and $\delta_{\mathcal{E}}(n, d, 5)$ both ≈ 0.69.

Theorem 5.1 (Pittel, Spencer and Wormald [29]). *Let $\mathcal{G}(n, d)$ be a random graph with m edges as in Proposition 4.1. Let $\mathcal{G}^k(n, d) = (\mathcal{V}^k(n, d), \mathcal{E}^k(n, d))$ be the nonempty k-core of $\mathcal{G}(n, d)$. Then $\mathcal{G}^k(n, d)$ contain $\delta_{\mathcal{V}}(n, d, k)n$ nodes and $\delta_{\mathcal{E}}(n, d, k)m$ edges with high probability for large n, i.e., $|\mathcal{V}^k(n, d)|/n \xrightarrow{p} \delta_{\mathcal{V}}(n, d, k)$ and $|\mathcal{E}^k(n, d)|/m \xrightarrow{p} \delta_{\mathcal{E}}(n, d, k)$ ($\xrightarrow{p}$ stands for convergence in probability).*

Proof. Please refer to Appendix B and [29]. ∎

The k-core pruning helps reduce the graph complexity and accelerates path finding. One concern is whether they prune too much and disconnect s and t. We found this is very unlikely to happen for two reasons. First, practical heterogeneous link predictions are rarely made between nodes in separate components. $\mathcal{G}$ will usually be connected, and a relatively short path connects any pair of nodes [39]. Moreover, the model is unlikely to predict that a link should appear between the disconnected (s, t). Empirically, we observe that there are usually too many paths connecting a predicted (s, t) instead of no paths, even in the k-core.

5.2 Heterogeneous Path-Enforcing Mask Learning

The second module of PaGE-Link learns heterogeneous masks to find important path-forming edges. We perform mask learning to select edges from the k-core-pruned computation graph. For notation simplicity in this section, we use $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to denote the graph for mask learning to save superscripts and subscripts, and $\mathcal{G}_c^k$ is the actual graph in the complete version of our algorithm.

The idea is to learn a mask over all edges of all edge types to select the important edges. Let $\mathcal{E}^r = \{e \in \mathcal{E} | \tau(e) = r\}$ be edges with type $r \in \mathcal{R}$. Let $\mathcal{M} = \{\mathcal{M}^r\}_{r=1}^{|\mathcal{R}|}$ be learnable masks of all edge types, with $\mathcal{M}^r \in \mathbb{R}^{|\mathcal{E}^r|}$ corresponds type r. We denote applying $\mathcal{M}^r$ on its corresponding edge type by $\mathcal{E}^r \odot \sigma(\mathcal{M}^r)$, where σ is the sigmoid function, and $\odot$ is the element-wise product. Similarly, we also overload the notation $\odot$ to indicate applying the set of masks on all types of edges, i.e., $\mathcal{E} \odot \sigma(\mathcal{M}) = \cup_{r\in\mathcal{R}}\{\mathcal{E}^r \odot \sigma(\mathcal{M}^r)\}$. We call the graph with the edge set $\mathcal{E} \odot \sigma(\mathcal{M})$ a *masked graph*. Applying a mask on graph edges will change the edge weights, which makes GNNs pass more information between nodes connected by highly-weighted edges and less on others. The general idea of mask learning is to learn an $\mathcal{M}$ that produces high weights for important edges and low weights for others. To learn an $\mathcal{M}$ that better fits the LP explanation, we measure edge importance from two perspectives: important edges should be both influential for the model prediction and form meaningful paths. Below, we introduce two loss terms $\mathcal{L}_{pred}$ and $\mathcal{L}_{path}$ for achieving these two measurements.

$\mathcal{L}_{pred}$ is to learn to select influential edges for model prediction. The idea is to do a perturbation-based explanation, where parts of the input are considered important if perturbing them changes the model prediction significantly. In the graph sense, if removing an edge e significantly influences the prediction, then e is a critical counterfactual edge that should be part of the explanation. This idea can be formalized as maximizing the mutual information between the masked graph and the original graph prediction Y, which is equivalent to minimizing the prediction loss $\mathcal{L}_{pred}$ as

$$\mathcal{L}_{pred}(\mathcal{M}) = -\log P_{\Phi}(Y = 1 | \mathcal{G} = (\mathcal{V}, \mathcal{E} \odot \sigma(\mathcal{M})), (s, t)). \quad (1)$$

$\mathcal{L}_{pred}(\mathcal{M})$ has a straightforward meaning, which says the masked subgraph should provide enough information for predicting the missing link (s, t) as the whole graph. Since the original prediction is a constant, $\mathcal{L}_{pred}(\mathcal{M})$ can also be interpreted as the performance drop after the mask is applied to the graph. A good masked graph should give a minimum performance drop. Oftentimes, regularizations of the mask entropy and mask norm are also included in $\mathcal{L}_{pred}(\mathcal{M})$ to encourage the mask to be discrete and sparse.

$\mathcal{L}_{path}$ is the loss term for $\mathcal{M}$ to learn to select path-forming edges, and its formula is given in Equation (2). The idea is first to identify a set of candidate edges denoted by $\mathcal{E}_{path}$ (specified below), where these edges can form concise and informative paths. Then $\mathcal{L}_{path}(\mathcal{M})$ will enforce the mask weights for $e \in \mathcal{E}_{path}$ to increase and mask weights for other edges to decrease during learning,

$$\mathcal{L}_{path}(\mathcal{M}) = -\sum_{r\in\mathcal{R}}\Big(\sum_{\substack{e\in\mathcal{E}_{path}\\ \tau(e)=r}} \mathcal{M}_e^r - \sum_{\substack{e\in\mathcal{E}, e\notin\mathcal{E}_{path}\\ \tau(e)=r}} \mathcal{M}_e^r\Big). \quad (2)$$

The key question for computing $\mathcal{L}_{path}(\mathcal{M})$ is to find a good $\mathcal{E}_{path}$ containing edges of concise and informative paths. As in Section 4, paths with these two desired properties should be short and without high-degree generic nodes. We thus define a score function of a path p reflecting these two properties as below

$$Score(p) = \log \prod_{\substack{e\in p\\ e=(u,v)}} \frac{P(e)}{D_v} = \sum_{\substack{e\in p\\ e=(u,v)}} Score(e), \quad (3)$$

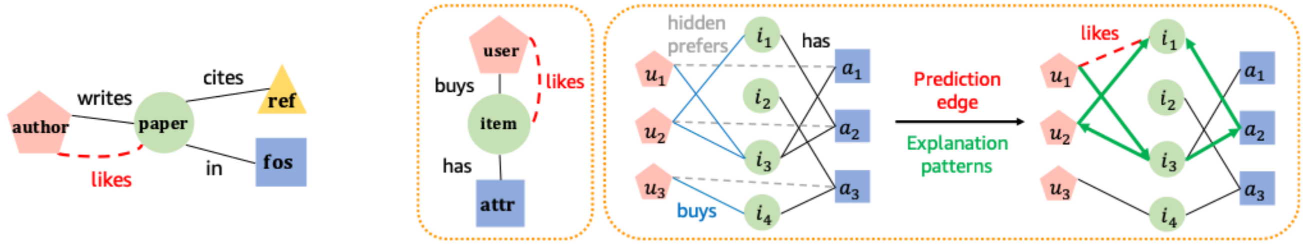
$$Score(e) = \log \sigma(\mathcal{M}_e^{\tau(e)}) - \log(D_v). \quad (4)$$

In this score function, $\mathcal{M}$ gives the probability of e to be included in the explanation, i.e., $P(e) = \sigma(\mathcal{M}_e^{\tau(e)})$. To get the importance of a path, we first use a mean-field approximation for the joint probability by multiplying $P(e)$ together, and we normalize each $P(e)$ for edge $e = (u, v)$ by its target node degree D_v. Then, we perform log transformation, which improves numerical stability for multiplying many edges with small $P(e)$ or large D_v and break down a path score to a summation of edge scores $Score(e)$ that are easier to work with. This path score function captures both desired properties mentioned above. A path score will be high if the edges on it have high probabilities, and the path score will be low if there are more edges or edges with high target degrees are included.

Finding paths with the highest $Score(p)$ can be implemented using Dijkstra's shortest path algorithm [4]. We set the distance represented by each edge to be the negative score of the edge, i.e., $-Score(e)$. We let $\mathcal{E}_{path}$ be the set of edges in the top five shortest paths found by Dijkstra's algorithm.

5.3 Mask Optimization and Path Generation

We optimize $\mathcal{M}$ alternatively with $\mathcal{L}_{pred}$ and $\mathcal{L}_{path}$. Each $\mathcal{M}$ update with $\mathcal{L}_{pred}$ will upweight the prediction-influential edges. Then the update with $\mathcal{L}_{path}$ will upweight the path-forming edges that are also highly-weighted by the current $\mathcal{M}$. Finally, after the



(a) Schema of AugCitation. "writes", "cites", and "in" edges are original. The "likes" edges (dashed red) are augmented for prediction.

(b) Schema of UserItemAttr (the left box) and its generation process (the right box) . Three types of base edges are generated first, i.e., "has" (black), "hidden prefers" (dashed gray), and "buys" (blue). The solid "has" and "buys" edges are then used to generate "likes" edges (dashed red) for prediction and the ground truth explanation patterns (green arrows).

Figure 4: The proposed augmented graph AugCitation and synthetic graph UserItemAttr.

Table 3: Time complexity of PaGE-Link and existing methods.

GNNExp [43]	PGExp [25]	SubgraphX [46]	PaGE-Link (ours)
$O(\|\mathcal{E}_c\|T)$	$O(\|\mathcal{E}\|T)$ / $O(\|\mathcal{E}_c\|)$	$\Theta(\|\mathcal{V}_c\|\hat{D}^{2B_{node}-2})$	$O(\|\mathcal{E}_c\| + \|\mathcal{E}_c^k\|T)$

mask learning converges, we run one more shortest-path step to generate paths from the final $\mathcal{M}$ and select the top paths according to budget B. These top paths give the explanation $\mathcal{P}$ as defined in Section 4. A pseudo-code of PaGE-Link is shown in Appendix C.

5.4 Complexity Analysis

In Table 3, we summarize the time complexity of PaGE-Link and representative existing methods for explaining a prediction with computation graph $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$ on a full graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Let T be the mask learning epochs. GNNExplainer has complexity $|\mathcal{E}_c|T$ as it learns a mask on $\mathcal{E}_c$. PGExplainer has a training stage and an inference stage (separated by / in the table). The inference stage is linear in $|\mathcal{E}_c|$, but the training stage covers edges in the entire graph and thus scales in $O(|\mathcal{E}|T)$. SubgraphX has much higher time complexity exponential in $|\mathcal{V}_c|$, so a size budget of B_{node} nodes is forced to replace $|\mathcal{V}_c|$, and $\hat{D} = \max_{v \in \mathcal{V}} D_v$ denotes the maximum degree (derivation in Appendix D). For PaGE-Link, the k-core pruning step is linear in $|\mathcal{E}_c|$. The mask learning with the Dijkstra's algorithm has complexity $|\mathcal{E}_c^k|T$. PaGE-Link has a better complexity than existing methods since $|\mathcal{E}_c^k|$ is usually smaller than $|\mathcal{E}_c|$ by an important factor (see Theorem 5.1), and PaGE-Link often converges faster (i.e., has a smaller T) given noisy nodes are pruned.

6 EXPERIMENTS

In this section, we conduct extensive empirical studies to evaluate explanations generated by PaGE-Link. Benchmark evaluation is a general challenge when studying model explainability, since standard datasets do not have ground truth explanations. Many works [25, 43] use synthetic data as benchmarks, but no benchmark datasets are available for evaluating GNN explanations for heterogeneous LP. Therefore, we created a synthetic graph and an augmented graph with a new type of edge to evaluate explanations. The augmented and synthetic graphs allow us to generate ground truth explanation patterns and evaluate explainers quantitatively.

6.1 Datasets

We describe the datasets used for evaluation below. More details including the graph statistics and the hyperparameters for graph creation are shown in Appendix E.

The augmented graph. AugCitation is constructed by augmenting the AMiner citation network [35]. A graph schema is shown in Figure 4a. The original AMiner graph contains four node types: author, paper, reference (ref), and field of study (fos), and edge types "cites", "writes", and "in". We construct AugCitation by augmenting the original graph with new (author, paper) edges typed "likes" and define a paper recommendation task on AugCitation for predicting the "like" edges. A new edge (s, t) is augmented if there is at least one concise and informative path p between them. In our augmentation process, we require the path p to have length shorter than a hyperparameter l_{max} and with degrees of nodes on p (excluding s & t) all bounded by a hyperparameter D_{max}. We highlight these two hyperparameters because of the conciseness and informativeness principles discussed in Section 4. The augmented edge (s, t) is used for prediction. The ground truth explanation is the set of paths satisfying the two hyperparameter requirements. We only take the top P_{max} paths with the smallest degree sums if there are many qualified paths. We train a GNN LP model to predict these new "likes" edges and evaluate explainers by comparing their output explanations with these path patterns as ground truth.

The synthetic graph. UserItemAttr is generated to mimic user-item-attribute recommendation graphs. Figure 4a shows the graph schema and Figure 4b is an illustration of the generation process. We include three node types: "user", "item", and item attributes ("attr"), and we build different types of edges step by step. Firstly, the "has" edges are created by randomly connecting items to attrs, and the "hidden prefers" edges are created by randomly connecting users to attrs. These edges represent items having attributes and user preferences for these attributes. Next, we randomly sample a set of items for each user, and we connect a (user, item) pair by a "buys" edge, if the user "hidden prefers" any attr the item "has". The "hidden prefers" edges correspond to an intermediate step for generating the observable "buys" edges. We remove the "hidden prefers" edges after "buys" edges are generated since we cannot observe 'hidden prefers" information in reality. An example of the rationale behind the generation process is that items have certain attributes,

Table 4: Path hit rate (HR). PaGE-Link **has high HR with a small budget** B**. Baselines achieve nonzero HR for large** B**.**

	B	GNNExp-Link	PGExp-Link	PaGE-Link (ours)
AugCitation	3	0.000	0.000	**0.713**
	10	0.000	0.000	**0.782**
	50	0.024	0.000	**0.974**
	200	0.364	0.020	**1.000**
UserItemAttr	3	0.000	0.000	**0.432**
	10	0.000	0.000	**0.662**
	50	0.054	0.000	**0.986**
	200	0.162	0.014	**1.000**

Table 5: ROC-AUC scores on learned masks. PaGE-Link **outperforms baselines.**

	GNNExp-Link	PGExp-Link	PaGE-Link (ours)
AugCitation	0.829	0.586	**0.928**
UserItemAttr	0.608	0.578	**0.954**

like the item "ice cream" with the attribute "vanilla". Then given that a user likes the attribute "vanilla" as hidden information, we observe that the user buys "vanilla ice cream". The final step is generating edges for prediction and their corresponding ground truth explanations, which follows the same augmentation process described above for `AugCitation`. For `UserItemAttr`, we have "has" and "buys" as base edges to construct the ground truth, and we augment "likes" edges between users and items for prediction.

6.2 Experiment settings

The GNN-based LP model. As we described in Section 3, the model involves a GNN encoder and a prediction head. We use RGCN [31] as the encoder to learn node representations on heterogeneous graphs, and we use the inner product as the prediction head. We train the model using the cross-entropy loss. On each dataset, our prediction task covers one edge type r. We randomly split the observed edges with type r into train:validation:test = 7:1:2 as positive samples and draw negative samples from the unobserved edges of type r. Edges with other types are used for GNN message passing.

Explainer baselines. Existing GNN explanation methods cannot be directly applied to heterogeneous LP. Therefore we extend the widespread GNNExplainer [43] and PGExplainer [25] as our baseline models. We re-implement a heterogeneous version of their mask matrix and mask predictor similarly to the heterogeneous mask learning module in PaGE-Link. For these two explainers, we perform mask learning using their original objectives, and then we generate edge-induced subgraph explanations from the learned mask. We refer to these two adapted explainers as GNNExp-Link and PGExp-Link below. We do not compare to other search-based explainers like SubgraphX [46] because they have high computational complexity, as discussed in Section 5.4. They work well on small molecule graphs as in the original papers, but they are hard to scale to large and dense LP graphs.

6.3 Evaluation Results

Quantitative evaluation. We evaluate explanations against the ground truth using the path hit rate (HR) and the ROC-AUC score. HR is the evaluation metric that best fits our problem formalization. Specifically, we fix the budget of B edges and evaluate whether an explanation can hit a path in the ground truth. Note that the ground truth only has the top P_{max} paths with the smallest degree sums, so hitting a less informative path with a high-degree generic node will not count. We show results with different budget B in Table 4. Explanations generated by PaGE-Link have hit rates 0.713 and 0.432 on `AugCitation` and `UserItemAttr`, respectively, with B equals only 3, and achieve a hit rate close to one for large B. In contrast, GNNExp-Link and PGExp-Link generate explanations from their mask by taking the top B edges, but they can barely hit any path in the ground truth for B less than 50.

Since PaGE-Link and both baselines can generate masks $\mathcal{M}$, we also follow [25] to compare explainers by the masks they generated using the ROC-AUC score. We treat the edges in the ground truth as positive, other edges as negative, and weights in $\mathcal{M}$ as the prediction scores for edges. The result is shown in Table 5, where PaGE-Link outperforms both methods.

Qualitative evaluation. A critical advantage of PaGE-Link is that it generates path explanations, which can capture the connections between node pairs and enjoy better interpretability. In contrast, the top important edges found by baseline methods are often disconnected from the source, the target, or both, which makes their explanations hard for humans to interpret and investigate. We conduct case studies to visualize explanations generated by PaGE-Link on the paper recommendation task for `AugCitation`.

Figure 5 shows a case in which the model recommends the source author "Vipin Kumar" a target paper titled "Fast and exact network trajectory similarity computation: a case-study on bicycle corridor planning". The top path explanation generated by PaGE-Link goes through the coauthor "Shashi Shekhar", which explains the recommendation as Vipin Kumar and Shashi Shekhar coauthored the paper "Correlation analysis of spatial time series datasets: a filter-and-refine approach", and Shashi Shekhar wrote the recommended paper. Given the same budget of three edges, explanations generated by baselines are less interpretable.

Figure 6 shows another example with the source author called "Huan Liu" and the recommended target paper titled "Using association rules to solve the cold-start problem in recommender systems". PaGE-Link generates paths going through the common fos of the recommended paper and three other papers written by Huan Liu: $p22646$, $p25160$, and $p35294$. We show the top three selected paths in green. We also show other unselected fos shared by the $p22646$, $p25160$, and $p35294$ and the target paper. Note that the top paths selected by PaGE-Link all have length three, even though there are many paths with length five or longer, e.g., $(a328, p22646, f4, p25260, f4134, p5670)$. Also, the top paths go through the fos "Redundancy (engineering)" and "User profile" instead of generic fos like "Artificial intelligence" and "Computer science". This case demonstrates that the paths selected by PaGE-Link are more concise and informative.

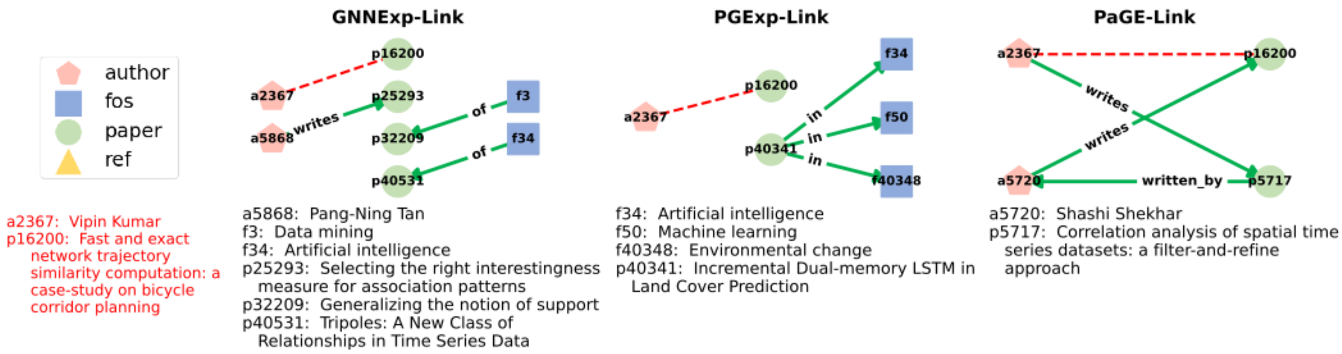


Figure 5: Explanations (green arrows) by different explainers for the predicted link $(a2367, p16200)$ (dashed red). The explanation generated by PaGE-Link **explains the recommendation by co-authorship, whereas baseline explanations are less interpretable.**

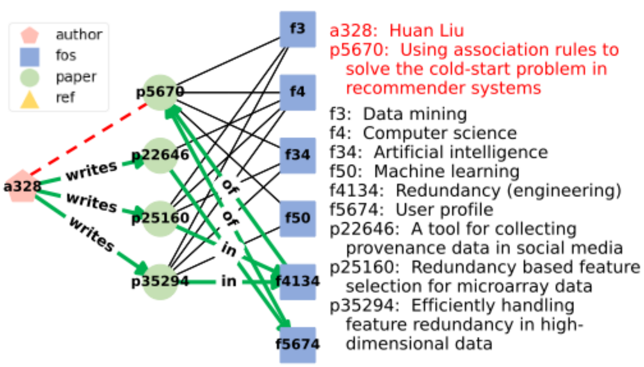


Figure 6: Top three paths (green arrows) selected by PaGE-Link **for explaining the predicted link $(a328, p5670)$ (dashed red). The selected paths are short and do not go through a generic field of study like "Computer Science".**

6.4 Ablation and Scalability Study

We conduct an ablation study to show the effectiveness of the k-core pruning module and the heterogeneous path-enforcing mask learning module in PaGE-Link. We compare three PaGE-Link variants, where we exclude the k-core pruning, the $\mathcal{L}_{path}$ loss, or both. Table 6 shows the path hit rate with budget B equals ten. Removing either module causes the HR to drop, especially on `UserItemAttr`. The second study is on the choice of different k for the k-core. We show the path HR results with $B = 10$ with different k in Table 7. We notice that the HR is the highest when $k = 3$ and $k = 2$ on `AugCitation` and `UserItemAttr` respectively. This is because `AugCitation` has larger ego-graphs and needs more pruning. Larger k values cause performance drops because of over-pruning. A relatively small k delivers the best result.

PaGE-Link is proved to scale linearly in $O(|\mathcal{E}_c^k|)$ in Section 5.4. Here we study its scalability empirically by generating ten different synthetic graphs with various sizes from 20 to 5,500 edges in $\mathcal{G}_c$. The results as shown in Figure 2b is indeed linear.

Table 6: All components are needed: Path hit rate on different PaGE-Link **variants. 'w/o X':** PaGE-Link **without X module.**

	w/o k-core & $\mathcal{L}_{path}$	w/o k-core	w/o $\mathcal{L}_{path}$	PaGE-Link
AugCitation	0.747	0.759	0.771	**0.782**
UserItemAttr	0.581	0.622	0.554	**0.662**

Table 7: Smaller k are better: Path hit rate on PaGE-Link **models with different k for the k-core algorithm.**

k	2	3	5	8
AugCitation	0.758	**0.782**	0.701	0.577
UserItemAttr	**0.662**	0.605	0.608	0.473

7 HUMAN EVALUATION

We conduct a human evaluation by randomly picking 100 predicted links from the test set of `AugCitation` and generate explanations for each link using GNNExp-Link, PGExp-Link, and PaGE-Link. We design a survey with single-choice questions. In each question, we show respondents the predicted link and three explanations with both the graph structure and the node/edge type information, similarly as in Figure 5. We ask respondents "please select the best explanation of *'why the model predicts this author will like the recommended paper?'* ". We collect at least three answers from different people for each question. In total, 340 evaluations are collected and 78.79% select the PaGE-Link explanation as the best.

8 CONCLUSION

In this work, we study model transparency and accountability on graph-structured web data. We investigate a new task: GNN explanation for heterogeneous LP. We identify three challenges for the task and propose a new path-based *scalable* method that produces explanations with *interpretable connections* and handles graph *heterogeneity*. Explanations generated by our method quantitatively improve AUC by 9 - 35% over baselines and are chosen by 78.79% responses as qualitatively more interpretable in human evaluation.

ACKNOWLEDGMENTS

We thank Ziniu Hu for helpful discussions on this work. This work is partially supported by NSF (2211557, 1937599, 2119643), NASA, SRC, Okawa Foundation Grant, Amazon Research Awards, Cisco Research Grant, Picsart Gifts, and Snapchat Gifts.

REFERENCES

[1] Mustafa Bilgic and Raymond J Mooney. 2005. Explaining recommendations: Satisfaction vs. promotion. In *Beyond personalization workshop, IUI*, Vol. 5. 153.

[2] Béla Bollobás. 1984. The evolution of sparse graphs, Graph theory and combinatorics (Cambridge, 1983).

[3] Kewei Cheng, Ziqing Yang, Ming Zhang, and Yizhou Sun. 2021. UniKER: A Unified Framework for Combining Embedding and Definite Horn Rule Reasoning for Knowledge Graph Inference. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 9753–9771. https://doi.org/10.18653/v1/2021.emnlp-main.769

[4] Edsger W Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271.

[5] Christos Faloutsos, Kevin S McCurley, and Andrew Tomkins. 2004. Fast discovery of connection subgraphs. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 118–127.

[6] Azin Ghazimatin, Oana Balalau, Rishiraj Saha Roy, and Gerhard Weikum. 2020. PRINCE: Provider-side interpretability with counterfactual explanations in recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 196–204.

[7] Zhichun Guo, William Shiao, Shichang Zhang, Yozen Liu, Nitesh Chawla, Neil Shah, and Tong Zhao. 2022. Linkless Link Prediction via Relational Distillation. *arXiv preprint arXiv:2210.05801* (2022).

[8] Gérard Hamiache and Florian Navarro. 2020. Associated consistency, value and graphs. *International Journal of Game Theory* 49, 1 (2020), 227–249.

[9] Yu Hao, Xin Cao, Yufan Sheng, Yixiang Fang, and Wei Wang. 2021. Ks-gnn: Keywords search over incomplete graphs via graphs neural network. *Advances in Neural Information Processing Systems* 34 (2021), 1700–1712.

[10] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. 241–250.

[11] Yuval Filmus (https://cs.stackexchange.com/users/683/yuval filmus). 2018. number of connected subgraphs of G with at most k > 0 vertices. (2018). https://cs.stackexchange.com/q/87434

[12] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. 2020. GraphLIME: Local Interpretable Model Explanations for Graph Neural Networks. arXiv:2001.06216 [cs.LG]

[13] Vassilis N Ioannidis, Da Zheng, and George Karypis. 2020. Few-shot link prediction via graph neural networks for covid-19 drug-repurposing. *arXiv preprint arXiv:2007.10261* (2020).

[14] Svante Janson and Malwina J Luczak. 2008. Asymptotic normality of the k-core in random graphs. *The annals of applied probability* 18, 3 (2008), 1085–1137.

[15] Glen Jeh and Jennifer Widom. 2002. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 538–543.

[16] Leo Katz. 1953. A new status index derived from sociometric analysis. *Psychometrika* 18, 1 (1953), 39–43.

[17] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[18] Markus Langer, Daniel Oster, Timo Speith, Holger Hermanns, Lena Kästner, Eva Schmidt, Andreas Sesing, and Kevin Baum. 2021. What do we want from Explainable Artificial Intelligence (XAI)?–A stakeholder perspective on XAI and a conceptual model guiding interdisciplinary XAI research. *Artificial Intelligence* 296 (2021), 103473.

[19] Bruno Lepri, Nuria Oliver, Emmanuel Letouzé, Alex Pentland, and Patrick Vinck. 2018. Fair, transparent, and accountable algorithmic decision-making processes. *Philosophy & Technology* 31, 4 (2018), 611–627.

[20] Chaozhuo Li, Bochen Pang, Yuming Liu, Hao Sun, Zheng Liu, Xing Xie, Tianqi Yang, Yanling Cui, Liangjie Zhang, and Qi Zhang. 2021. Adsgnn: Behavior-graph augmented relevance modeling in sponsored search. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 223–232.

[21] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology* 58, 7 (2007), 1019–1031.

[22] Wanyu Lin, Hao Lan, and Baochun Li. 2021. Generative causal explanations for graph neural networks. In *International Conference on Machine Learning*. PMLR, 6666–6679.

[23] Wanyu Lin, Hao Lan, Hao Wang, and Baochun Li. 2022. OrphicX: A Causality-Inspired Latent Variable Model for Interpreting Graph Neural Networks. *arXiv preprint arXiv:2203.15209* (2022).

[24] Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. 2022. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 4499–4511.

[25] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. 2020. Parameterized Explainer for Graph Neural Network. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 19620–19631.

[26] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: ultra simplification of graph convolutional networks for recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1253–1262.

[27] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *Proc. IEEE* 104, 1 (2015), 11–33.

[28] Namyong Park, Andrey Kan, Christos Faloutsos, and Xin Luna Dong. 2020. J-Recs: Principled and Scalable Recommendation Justification. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1208–1213.

[29] Boris Pittel, Joel Spencer, and Nicholas Wormald. 1996. Sudden emergence of a giantk-core in a random graph. *Journal of Combinatorial Theory, Series B* 67, 1 (1996), 111–151.

[30] Ben Roberts and Dirk P Kroese. 2007. Estimating the Number of st Paths in a Graph. *J. Graph Algorithms Appl.* 11, 1 (2007), 195–214.

[31] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer, 593–607.

[32] Lloyd Shapley. 1953. A value fo n-person Games. *Ann. Math. Study28, Contributions to the Theory of Games, ed. by HW Kuhn, and AW Tucker* (1953), 307–317.

[33] Donghee Shin and Yong Jin Park. 2019. Role of fairness, accountability, and transparency in algorithmic affordance. *Computers in Human Behavior* 98 (2019), 277–284. https://doi.org/10.1016/j.chb.2019.04.019

[34] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment* 4, 11 (2011), 992–1003.

[35] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 990–998.

[36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[37] Minh Vu and My T. Thai. 2020. PGM-Explainer: Probabilistic Graphical Model Explanations for Graph Neural Networks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 12225–12235.

[38] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5329–5336.

[39] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *nature* 393, 6684 (1998), 440–442.

[40] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2020. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)* (2020).

[41] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[42] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[43] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems* 32 (2019), 9240.

[44] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2020. Explainability in graph neural networks: A taxonomic survey. *arXiv preprint arXiv:2012.15445* (2020).

[45] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2022. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[46] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021. On Explainability of Graph Neural Networks via Subgraph Explorations. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 12241–12252.

[47] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. 2019. Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems. *arXiv*

preprint arXiv:1905.13129 (2019).
[48] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems* 31 (2018).
[49] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. 2020. Revisiting graph neural networks for link prediction. (2020).
[50] Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. 2022. GStarX: Explaining Graph Neural Networks with Structure-Aware Cooperative Games. In *Advances in Neural Information Processing Systems.*
[51] Yongfeng Zhang and Xu Chen. 2020. Explainable Recommendation: A Survey and New Perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101. https://doi.org/10.1561/1500000066
[52] Tong Zhao, Gang Liu, Daheng Wang, Wenhao Yu, and Meng Jiang. 2022. Learning from Counterfactual Links for Link Prediction. In *International Conference on Machine Learning*. PMLR, 26911–26926.
[53] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.
[54] Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems* 34 (2021), 29476–29490.

A PROOF OF PROPOSITION 4.1

Proof. We prove $Z_{n,d} = o(S_{n,d})$ by definition, where we show $\lim_{n\to\infty} \frac{Z_{n,d}}{S_{n,d}} = 0$. As we can permute the indices of nodes in $\mathcal{G}(n,d)$, without loss of generality, we assume $Z_{n,d}$ is the expected number of paths between nodes indexed 1 and n. Our proof is mainly based on the result in [30], which computes the expected number of all 1-n paths, i.e., $Z_{n,d} = (n-2)!d^{n-1}e(1+o(1))$. On the other hand, the number of edge-induced subgraphs considered in [25, 43] equals the size of the power set of all edges, i.e., $S_{n,d} = 2^{d\binom{n}{2}}$. We thus have

$$\log Z_{n,d} = \log\left[(n-2)!d^{n-1}e(1+o(1))\right] \tag{1}$$

$$< \log\left[\sqrt{2\pi(n-2)}(\frac{n-2}{e})^{(n-2)} e^{\frac{1}{12(n-2)}} d^{n-1} e(1+o(1))\right] \tag{2}$$

$$= \frac{1}{2}\log(2\pi(n-2)) + (n-2)\log(\frac{n-2}{e}) + \log\frac{1}{12(n-2)} + (n-1)\log d + 1 + \log(1+o(1)) \tag{3}$$

$$= O(\log n) + O(n\log n) + O(\log\frac{1}{n}) + O(n\log d) \tag{4}$$

$$+ \log(1+o(1)) \tag{5}$$

$$= O(n\log n) + \log(1+o(1)) \tag{6}$$

$$\log S_{n,d} = \log 2^{d\binom{n}{2}} = d\binom{n}{2}\log 2 = O(n^2) \tag{7}$$

$$\lim_{n\to\infty}\frac{Z_{n,d}}{S_{n,d}} = \lim_{n\to\infty}\exp(\log\frac{Z_{n,d}}{S_{n,d}}) \tag{8}$$

$$= \exp(\lim_{n\to\infty}\log\frac{Z_{n,d}}{S_{n,d}}) \tag{9}$$

$$= \exp(\lim_{n\to\infty}\log Z_{n,d} - \log S_{n,d}) \tag{10}$$

$$= \exp(\lim_{n\to\infty} O(n\log n) + \log(1+o(1)) - O(n^2)) \tag{11}$$

$$= 0 \tag{12}$$

Step (1) to (2) is Stirling's formula. Step (8) to (9) is because exp is continuous. ■

Algorithm 1 PaGE-Link

Input: heterogeneous graph $\mathcal{G}$, trained GNN-based LP model $\Phi(\cdot,\cdot)$, predicted link (s,t), size budget B, k for k-core.
Extract the computation graph $\mathcal{G}_c$
Prune $\mathcal{G}_c$ for the k-core $\mathcal{G}_c^k = (\mathcal{E}_c^k, \mathcal{V}_c^k)$.
Initialize $\mathcal{M}$
while $\mathcal{M}$ not converge **do**
  Update $\mathcal{M}$ to minimize $\mathcal{L}_{pred}(\mathcal{M})$ { Eq.(1)}
  Update $\mathcal{M}$ to minimize $\mathcal{L}_{path}(\mathcal{M})$ { Eq.(2)}
end while
Generate shortest paths on $\mathcal{M}$ with edge distance $-Score(e)$
Pick the top shortest paths $\mathcal{P}$ under budget B
Return: A set of paths $\mathcal{P}$

Table 8: Statistics of the graph datasets.

	#Nodes($\lvert\mathcal{V}\rvert$)	#Edges($\lvert\mathcal{E}\rvert$)	#Predict Edges ($\lvert\mathcal{E}^{likes}\rvert$)
AugCitation	45,961	238,771	4,545
UserItemAttr	250	1,800	116

Table 9: Hyperparameters for the augmentation of AugCitation and the generation of UserItemAttr

	l_{max}	D_{max}	P_{max}
AugCitation	3	30	5
UserItemAttr	3	15	5

B DETAILED THEOREM 5.1

We now state a more detailed version of Theorem 5.1 below. This theorem gives the exact formula of $\delta_{\mathcal{V}}(n,d,k)$ and $\delta_{\mathcal{E}}(n,d,k)$, which are built upon a Poisson random variable. The argument is adapted from [14, 29]. Readers can refer to [14, 29] for the proof.

For $\mu > 0$, let $Po(\mu)$ denote a Poisson distribution with mean μ. Let $\psi_k(dn) = P(Po(dn) \geq k)$ be the tail probability of $Po(dn)$. Let $c_k = \inf_{\mu>0} \mu/\phi_{k-1}(\mu)$. When $dn > c_k$, the equation $\mu/\psi_{k-1}(\mu) = dn$ will have two roots for μ. Let $\mu(dn,k)$ be the larger root. Then we have the following more detailed version of Theorem 5.1 with $\delta_{\mathcal{V}}(n,d,k)$ and $\delta_{\mathcal{E}}(n,d,k)$ as functions of $\mu(dn,k)$.

Theorem B.1 (Pittel, Spencer and Wormald). *Let $\mathcal{G}(n,d)$ be a random graph with m edges as in Proposition 4.1. Let $\mathcal{G}^k(n,d) = (\mathcal{V}^k(n,d), \mathcal{E}^k(n,d))$ be the k-core of $\mathcal{G}(n,d)$. When $dn > c_k$, $\mathcal{G}^k(n,d)$ will be nonempty with high probability (w.h.p.) for large n. Also, $\mathcal{G}^k(n,d)$ will contain $\psi_k(\mu(dn,k))n$ nodes and $[\mu(dn,k)^2/(d^2n(n-1))]m$ edges w.h.p. for large n, i.e., $|\mathcal{V}^k(n,d)|/n \xrightarrow{p} \psi_k(\mu(dn,k))$ and $|\mathcal{E}^k(n,d)|/m \xrightarrow{p} \mu(dn,k)^2/(d^2n(n-1))$ ($\xrightarrow{p}$ stands for convergence in probability).*

C PSEUDO-CODE OF PAGE-LINK

A pseudo-code for PaGE-Link is shown in Algorithm 1.

D COMPLEXITY OF SUBGRAPHX

The search-based methods often have much higher time complexity exponential in the number of nodes or edges. Thus, a budget is forced instead of searching subgraphs with all sizes. For example, SubgraphX finds all connected subgraphs with at most B_{node} nodes, which has complexity $\Theta(|\mathcal{V}_c|\hat{D}^{2B_{node}-2})$ for a graph with maximum degree $\hat{D} = \max_{v \in \mathcal{V}} D_v$. This complexity can be shown using the following two lemmas.

Lemma D.1. *For a graph $\mathcal{G}$ with n vertices, the number of the connected subgraph of $\mathcal{G}$ having B_{node} nodes is bounded below by the number of trees in $\mathcal{G}$ having B_{node} nodes.*

Proof. Each connected subgraph has a spanning tree. ■

Lemma D.2. *For a graph $\mathcal{G}$ with node set $\mathcal{V}$, the number of trees in $\mathcal{G}$ having B_{node} tree nodes is $\Theta(|\mathcal{V}|\hat{D}^{2B_{node}-2})$.*

Proof. See [11] for proof using an encoding procedure. ■

E DATASET DETAILS

We show detailed graph statistics in Table 8 and hyperparameters for augmenting and generating the graphs in Table 9.