LPLP: LEARNING PATTERNS FOR LINK PREDICTION FROM CONTEXT

Anonymous authors
Paper under double-blind review

ABSTRACT

Link prediction is a fundamental task in graph machine learning with applications across diverse domains. It relies on structural patterns that vary substantially across graphs, making predictors learned on one graph difficult to transfer to another. Recent universal link predictors adapt to new graphs through labeled context, but still learn how to use this context from real graphs. We introduce **LPLP**, which **L**earns **P**atterns for **L**ink **P**rediction from context, following the prior-data fitted network paradigm. LPLP is trained entirely on synthetic graphs generated from a prior in which the structural patterns that predict links vary across tasks. Given an unseen graph with labeled context links, LPLP infers the structural pattern of the graph and scores query links without parameter updates. Across seven benchmarks, LPLP achieves the best average rank among methods evaluated on shared splits under both random and HeaRT hard negatives, improving average Hits@50 over the strongest baseline by 5.45 and 4.22 points, respectively. These results indicate that the ability to infer structural patterns from context can be learned from synthetic graph priors and applied to unseen graphs.

1 INTRODUCTION

Link prediction aims to infer missing links from an observed graph and is a fundamental problem in graph machine learning, with applications ranging from social networks (Liben-Nowell & Kleinberg, 2003) and recommendation systems (Zhang & Chen, 2020) to knowledge graph completion (Zhu et al., 2021). A central difficulty is that the structural patterns that predict links differ from graph to graph. Common-neighbor patterns may be highly predictive in one network, whereas longer-range paths, activity relations, or other structural configurations may be more informative in another (Kovács et al., 2019; Ghasemian et al., 2020; Mao et al., 2024). Consequently, a predictor fitted to the structural pattern of one graph can transfer poorly to another.

Recent universal link predictors address this variation by conditioning prediction on labeled links from the target graph. UniLP (Dong et al., 2025), for example, uses in-context links to adapt link representations to the target graph, while TFMLinker (Liao et al., 2026) combines topology-aware representations with in-context prediction through a tabular foundation model. These approaches demonstrate that context can provide useful evidence about the structural pattern of a new graph. However, they still learn how to use this context from collections of real graphs during pretraining. This leaves open a more fundamental question:

Can a model learn to infer from context which structural patterns predict links, without training on any real graph?

We study this question through the prior-data fitted network (PFN) paradigm (Müller et al., 2022; Hollmann et al., 2023). A PFN is trained on synthetic tasks sampled from a prior and learns to approximate the posterior predictive distribution induced by that prior. At inference time, the PFN takes the observed examples of a new task as context, implicitly infers the latent hypothesis that generated them, and makes predictions for new queries without parameter updates. For link prediction, this suggests treating each graph as a task governed by a latent link-formation rule. Pretraining on such tasks teaches a PFN to infer the structural pattern of a new graph from its labeled context links.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The central challenge lies in the design of the synthetic graph prior. Beyond generating diverse graph structures, a prior for link prediction should vary *which structural patterns predict links* across tasks. NodePFN (Choi et al., 2026) and GraphPFN (Eremeev et al., 2026b) have recently proposed synthetic graph priors for node classification. To examine whether these priors meet this requirement, we sample graphs from each prior and compute the link rates of node pairs at distances 2 and 3, as described in Appendix A. As shown in Figure 1, the NodePFN prior yields nearly equal rates at the two distances, whereas the GraphPFN prior consistently favors distance 2. In contrast, graphs from our prior differ in whether they favor distance 2 or 3, so which distance predicts links varies across tasks.

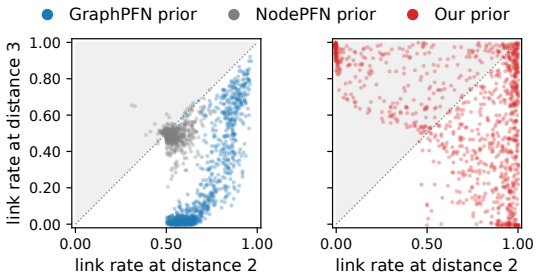


Figure 1: Diversity of structural patterns across tasks sampled from graph priors.

In this paper, we introduce **LPLP**, which **L**earns **P**atterns for **L**ink **P**rediction from context. Each synthetic graph is built from a degree-corrected stochastic block model (Karrer & Newman, 2011) with varying community structure and degree heterogeneity. On top of this graph, we sample a graph-specific link-formation rule that adds links by weighting degree-normalized path counts differently at each distance. Varying this rule produces synthetic tasks ranging from local triangle closing to regimes in which more distant node pairs are more likely to link than closer ones.

During pretraining on these tasks, the observed links of each synthetic graph and sampled unlinked pairs form the labeled context, while the held-out links and other unlinked pairs form the queries. LPLP encodes each context and query pair as a token from its enclosing subgraph, conditions each query on the labeled context through attention, and adds structural information beyond this subgraph through message passing on the observed graph. At inference, LPLP predicts the query labels of an unseen graph from its labeled context without gradient updates or graph-specific parameters.

We evaluate LPLP on seven undirected link prediction benchmarks spanning six domains. Despite never observing a real graph during pretraining, LPLP achieves the best average rank of 1.29 and an average Hits@50 of 74.57 under random negatives, improving on the strongest method evaluated on the same splits by 5.45 points. The advantage persists under the HeaRT hard-negative protocol (Li et al., 2023), where LPLP ranks first on six of the seven benchmarks and exceeds the strongest competitor by 4.22 points in average Hits@50. Together with our further analyses, these results show that a model trained only on a synthetic prior can infer from context which structural patterns predict links in unseen real graphs.

Our contributions are summarized as follows:

- We formulate universal link prediction as inferring the structural pattern of each graph and introduce LPLP, a PFN trained entirely on synthetic graphs to perform this inference.
- We design a synthetic graph prior whose link-formation rule varies across tasks, so that which structural patterns predict links must be inferred from context.
- We show that a single LPLP model trained entirely on synthetic graphs transfers to seven unseen real-world graphs, achieving the best average rank among the evaluated supervised and universal link predictors under both random-negative and hard-negative protocols, without target-graph parameter updates.

2 Preliminaries

2.1 Prior-data Fitted Networks

Prior-data fitted networks (PFNs) (Müller et al., 2022) are pretrained on synthetic learning tasks induced by a prior distribution over hypotheses. To generate a synthetic task, a latent hypothesis ϕ is sampled from a prior $p(\phi)$, and a dataset $\mathcal{D}$ is then generated from $p(\mathcal{D} \mid \phi)$. From the generated dataset, a context set $\mathcal{D}_{\text{ctx}}$ and a query set $\mathcal{D}_{\text{qry}}$ are sampled as disjoint subsets.

Conditioned on the observed context $\mathcal{D}_{\text{ctx}}$, Bayesian inference forms a posterior distribution over the possible hypotheses ϕ. For a query example $(\mathbf{x}_q, y_q) \in \mathcal{D}_{\text{qry}}$, the predictive distribution of y_q is obtained by marginalizing over this posterior.

$$p(y_q \mid \mathbf{x}_q, \mathcal{D}_{\text{ctx}}) = \int p(y_q \mid \mathbf{x}_q; \phi) p(\phi \mid \mathcal{D}_{\text{ctx}})\, d\phi. \tag{1}$$

Here, ϕ ranges over the hypotheses that could have generated the observed context. The posterior over ϕ follows Bayes' rule

$$p(\phi \mid \mathcal{D}_{\text{ctx}}) \propto p(\mathcal{D}_{\text{ctx}} \mid \phi) p(\phi). \tag{2}$$

Explicitly computing the posterior over hypotheses and evaluating the integral in equation 1 for every new task can be computationally expensive. Instead, a PFN amortizes this computation by directly producing an approximation q_θ to the posterior predictive distribution (PPD).

$$q_\theta(\cdot \mid \mathbf{x}_q, \mathcal{D}_{\text{ctx}}) \approx p(\cdot \mid \mathbf{x}_q, \mathcal{D}_{\text{ctx}}). \tag{3}$$

PFNs are pretrained by minimizing cross-entropy on the query labels of synthetic tasks. This objective is equivalent to minimizing the expected $\mathrm{KL}(p \,\|\, q_\theta)$ divergence between the prior-induced PPD and its neural approximation. We provide the derivation in Appendix B.

By pretraining on many synthetic tasks sampled from the prior, the PFN learns to approximate this Bayesian inference procedure in a single forward pass.

2.2 Link Prediction

Let $G = (\mathcal{V}, \mathcal{E})$ be a simple undirected graph with $n = |\mathcal{V}|$ nodes and adjacency matrix $\mathbf{A} \in \{0,1\}^{n \times n}$, and let $\mathcal{U} = \{(u, v) \in \mathcal{V} \times \mathcal{V} \mid u < v\}$ denote the set of unordered node pairs. We assume that $\mathcal{E}$ is an observed subset of an underlying true link set $\mathcal{E}^\star$, where $\mathcal{E} \subseteq \mathcal{E}^\star \subseteq \mathcal{U}$. The candidate set $\mathcal{P} = \mathcal{U} \setminus \mathcal{E}$ consists of unobserved positive links in $\mathcal{E}^\star \setminus \mathcal{E}$ and negative pairs in $\mathcal{U} \setminus \mathcal{E}^\star$. For each candidate pair $e \in \mathcal{P}$, a link predictor assigns a score $s(e) \in \mathbb{R}$ and aims to rank positive links above negative pairs.

3 Synthetic Graph Prior for Link Prediction

For link prediction, we treat each graph as a task governed by a latent hypothesis ϕ, which determines how links form through the base structure of the graph and a graph-specific link-formation rule. Our goal is to vary this rule across tasks, so that which structural patterns predict links must be inferred from context. We decompose the hypothesis as $\phi = (\psi, \omega)$, where the base-structure parameters ψ determine the structural environment in which links may form, and the rule ω adds further links on top of this environment. We provide implementation details and sampling distributions in Appendix C.

3.1 Base Structure

The base structure is a hierarchical degree-corrected stochastic block model (Karrer & Newman, 2011) that produces a seed graph $G_0 = (\mathcal{V}, \mathcal{E}_0)$. Its parameters ψ control community organization, degree heterogeneity, and whether nodes with similar or dissimilar activity levels are more likely to connect. Varying ψ across tasks produces graphs ranging from assortative to disassortative community structure, with heterogeneous degree profiles and hierarchical substructure. Together, these factors induce a base logit $\ell_0(u, v)$ for every pair $(u, v) \in \mathcal{U}$, from which the seed links $\mathcal{E}_0$ are sampled. The resulting G_0 defines the structural environment on which the graph-specific link-formation rule operates.

3.2 Link-Formation Rule

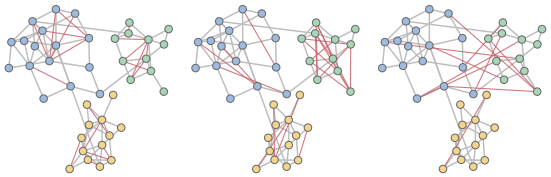
Given the seed graph G_0, we sample a graph-specific rule that determines how path counts at each distance contribute to further link formation. For a pair (u, v) at shortest-path distance k in G_0, let $\xi_k(u, v) \geq 0$ denote its path count, in which each length-k path is discounted by the degrees of its intermediate nodes and the total is normalized by the endpoint degrees. For pairs at any other

distance, $\xi_k(u, v) = 0$, so that path counts at different distances are kept separate. The rule scores each pair by

$$\ell_1(u, v) = \ell_0(u, v) + \alpha \sum_{k=2}^{K} w_k \xi_k(u, v), \tag{4}$$

where $\alpha \geq 0$ controls the strength of the rule, K is the longest distance considered, and $\mathbf{w} = (w_2, \ldots, w_K)$ distributes the rule strength across distances. Because ξ_k is placed on a common scale across distances, $\mathbf{w}$ determines which distances the rule favors. For $k = 2$, ξ_2 is a normalized resource-allocation score (Zhou et al., 2009), and larger values of k extend this weighting to longer paths, so the favored distances determine whether common-neighbor heuristics or longer-path indices such as Katz (Katz, 1953) are predictive of the links added by the rule.

A rule is parameterized by $\omega = (\rho, \alpha, K, \mathbf{w})$, where ρ is the fraction of the link budget allocated to the rule links $\mathcal{E}_1$, which are drawn from the pairs outside $\mathcal{E}_0$ according to ℓ_1. We sample the rule independently of ψ and resample it for every task, allowing the same underlying seed structure to give rise to different link-formation regimes (Figure 2). A rule concentrated on distance 2 closes local triangles (Figure 2a), a distance-3 rule links pairs without a shared neighbor (Figure 2b), and rules emphasizing longer distances connect more distant nodes (Figure 2c).



(a) (b) (c)

Figure 2: One seed graph G_0 under different link-formation rules.

3.3 Observation and Task Construction

The seed and rule links form the true link set $\mathcal{E}^\star = \mathcal{E}_0 \cup \mathcal{E}_1$. Each link in $\mathcal{E}^\star$ is hidden independently with a graph-specific probability, as in a random split, and the remaining links form the observed graph $G = (\mathcal{V}, \mathcal{E})$. The context set $\mathcal{D}^{\text{lp}}_{\text{ctx}}$ consists of observed links from $\mathcal{E}$ and an equal number of negative pairs from $\mathcal{U} \setminus \mathcal{E}^\star$, and the disjoint query set $\mathcal{D}^{\text{lp}}_{\text{qry}}$ consists of held-out links from $\mathcal{E}^\star \setminus \mathcal{E}$ and other negative pairs from $\mathcal{U} \setminus \mathcal{E}^\star$. With probability $1/2$, we invert the labels of all context and query pairs, so that neither label index carries a fixed meaning. Because ω is resampled for every task and the labels may be inverted, which structural patterns correspond to the positive class is revealed only by the labeled context. Following equation 1, LPLP conditions on the observed graph G and the labeled context $\mathcal{D}^{\text{lp}}_{\text{ctx}}$ to predict the label y_e of each query pair $e \in \mathcal{D}^{\text{lp}}_{\text{qry}}$.

Each synthetic graph also carries a node classification task in the manner of NodePFN (Choi et al., 2026) with the communities of the base structure as classes. This task leaves the link prediction task unchanged, and LPLP is pretrained on both tasks with shared layers so that the same model also classifies nodes. Details of this task, the full pretraining objective, and the node classification results of LPLP are provided in Appendix D.

4 LPLP: Prior-data Fitted Network for Link Prediction

We introduce LPLP, a prior-data fitted network that performs link prediction on an observed graph G given a set of labeled context links, and scores every query pair. As shown in Figure 3, LPLP encodes each pair as a token from its enclosing subgraph, refines the tokens with layers that combine attention over the labeled context with message passing on G, and decodes the query tokens into link probabilities.

4.1 Encoding Subgraphs as Input Tokens

Subgraph extraction and node labeling. For a pair $e = (u, v)$, we extract the 2-hop enclosing subgraph G_e induced by all nodes within two hops of u or v, removing the link (u, v) itself if it is observed, so that the subgraph of an observed context link does not contain the link being labeled (Zhang & Chen, 2018). Each node $i \in G_e$ is then assigned a structural label through

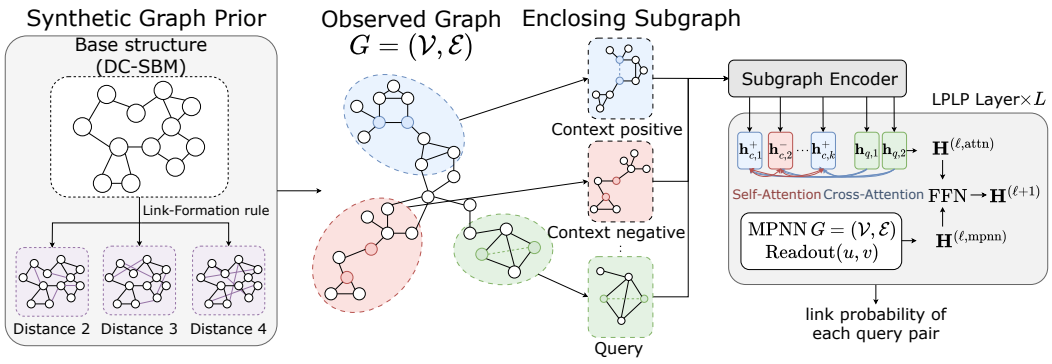


Figure 3: Overview of LPLP. Each context and query pair is encoded from its enclosing subgraph, and the LPLP layers combine attention over the labeled context with message passing on the observed graph.

DRNL+ (Dong et al., 2025), which encodes the shortest-path distances from i to u and v following double-radius node labeling (Zhang & Chen, 2018) and distance encoding (Li et al., 2020), so that nodes with the same distances to u and v, such as their common neighbors, share a label. Each label is mapped to an embedding, which exposes the role of every node relative to the pair without relying on node identities. The full labeling scheme is given in Appendix E.1.

Subgraph encoder. The label embeddings are processed by a GraphSAGE (Hamilton et al., 2017) on G_e. After the GraphSAGE layers, the states of the two endpoints u and v are read out from every layer and combined by an endpoint-symmetric MLP into the input token $\mathbf{h}_e^{(0)}$. A learned embedding of the link label is then added to each context token, allowing the subsequent layers to relate the structure of each context pair to its label, while query tokens carry no label.

4.2 Context Attention and Message Passing

We adopt the layer of NodePFN (Choi et al., 2026), which runs an attention branch and a message-passing branch (MPNN) in parallel on the same input and sums their outputs, and apply it to pair tokens. $\mathbf{H}^{(l)}$ denotes the tokens of all context and query pairs after layer l, with row $\mathbf{h}_e^{(l)}$ for pair $e = (u, v)$.

In the attention branch, following the PFN paradigm (Müller et al., 2022), context tokens attend to one another and query tokens attend only to context tokens, so that each query is predicted from its own representation and the labeled context, independently of the other queries. The message-passing branch maintains a state $\mathbf{z}_u^{(l)}$ for every node of G, initialized with quasi-orthogonal random signatures (Dong et al., 2024) drawn afresh for every graph and updated by GCN steps (Kipf & Welling, 2017). Because these signatures are random rather than learned, the branch has no node-specific parameters and transfers to unseen graphs without retraining.

Motivated by random-signature message passing that estimates link-level structural features (Dong et al., 2024), this branch reads the propagated endpoint states to capture structure beyond the enclosing subgraph, as detailed in Appendix E.3. A symmetric MLP $\mathrm{Readout}$ maps the two endpoint states of each pair into its token, and one layer computes

$$\mathbf{h}_e^{(l+1)} = \mathrm{FFN}\Big(\underbrace{\mathrm{LN}\big(\mathbf{h}_e^{(l)} + \mathrm{Attn}(\mathbf{H}^{(l)})_e\big)}_{\text{attention branch}} + \underbrace{\mathrm{LN}\big(\mathbf{h}_e^{(l)} + \mathrm{Readout}(\mathbf{z}_u^{(l+1)}, \mathbf{z}_v^{(l+1)})\big)}_{\text{message-passing branch}} \Big), \tag{5}$$

where FFN is a residual feed-forward block with layer normalization.

4.3 Training Objective and Inference

After L layers, an MLP maps each query token $\mathbf{h}_e^{(L)}$ to a predictive distribution over its link label, and the predicted probability of a link serves as the score $s(e)$. For each synthetic graph sampled

from the prior of Section 3, the link prediction loss is the cross-entropy over the query pairs,

$$\mathcal{L}_{\mathrm{lp}}(\theta) = \mathbb{E}\Big[- \frac{1}{|\mathcal{D}^{\mathrm{lp}}_{\mathrm{qry}}|} \sum_{e \in \mathcal{D}^{\mathrm{lp}}_{\mathrm{qry}}} \log q_\theta(y_e \mid e, G, \mathcal{D}^{\mathrm{lp}}_{\mathrm{ctx}}) \Big], \tag{6}$$

where the expectation is taken over synthetic graphs and their tasks. At inference, LPLP is applied to a real graph through the same forward pass without any parameter update. The observed links of the graph and sampled negative pairs form $\mathcal{D}^{\mathrm{lp}}_{\mathrm{ctx}}$, and the candidate pairs to be scored form $\mathcal{D}^{\mathrm{lp}}_{\mathrm{qry}}$. Further details of the architecture and pretraining are provided in Appendix E.

5 Experiments

5.1 Experimental Setup

5.1.1 Datasets

We evaluate on Celegans (Watts & Strogatz, 1998) (biology), USAir (Rossi & Ahmed, 2015) (transportation), PB (Ackland, 2005) (web), NS (Newman, 2006) and CS (Shchur et al., 2019) (collaboration), Cora (Yang et al., 2016) (citation), and Facebook (Rozemberczki et al., 2021) (social network), seven real-world link prediction benchmarks from six domains. Following UniLP (Dong et al., 2025) and TFMLinker (Liao et al., 2026), we use their evaluation benchmarks, which allows comparison with these universal link predictors. All graphs are treated as undirected and used without node attributes, and their statistics are provided in Appendix F.1.

5.1.2 Baselines

We compare against the structural heuristics CN (Newman, 2001), AA (Adamic & Adar, 2003), RA (Zhou et al., 2009), and Katz (Katz, 1953), the supervised GNN-based link predictors GCN (Kipf & Welling, 2017), SEAL (Zhang & Chen, 2018), Neo-GNN (Yun et al., 2021), ELPH and BUDDY (Chamberlain et al., 2023), NCNC (Wang et al., 2024), and MPLP (Dong et al., 2024), and the universal link predictors UniLP (Dong et al., 2025) and TFMLinker (Liao et al., 2026). Each supervised baseline is retrained on every split with its official implementation, with hyperparameters and checkpoints selected on the validation split, and further details are provided in Appendix F.2.

5.1.3 Evaluation Protocol

Following UniLP (Dong et al., 2025), we split the positive links of each graph into training, validation, and test sets with a 70:10:20 ratio, and only the training links form the observed graph G for every method. For in-context link predictors, a fixed context of 200 training links and 200 negative pairs sampled from the non-links of the full graph is shared by all queries. The test links serve as positive queries against an equal number of negative pairs sampled uniformly from the non-links of the full graph, and performance is measured by Hits@50 averaged over ten splits. Details of the evaluation protocol are provided in Appendix F.3.

5.2 Link Prediction under Random Negatives

Table 1 reports the results under random negatives. Despite being trained only on synthetic graphs, LPLP achieves the highest average Hits@50 of 74.57 and the best average rank of 1.29, exceeding MPLP, the strongest method on the same splits, by 5.45 points. Among the supervised GNNs, which are trained on each target graph, the best model varies across graphs, being MPLP on Celegans, USAir, and PB, SEAL on NS and Cora, and Neo-GNN on CS and Facebook. LPLP is instead first or within one point of the best on every graph, suggesting that it adapts to each graph. LPLP also exceeds the values that TFMLinker (Liao et al., 2026) reports on its own splits, on six of the seven graphs and by 3.05 points on average. The wall-clock time of LPLP and the baselines is compared in Appendix G.2, and the effect of the context size is reported in Appendix G.3.

Table 1: Link prediction performance measured by Hits@50 (%) under random negatives. Values following ± denote standard deviations. The best results are shown in **bold**, and the second-best results are underlined. †Reported by Liao et al. (2026) on different splits, since no implementation of TFMLinker is publicly available.

Method	Celegans	USAir	PB	NS	Cora	CS	Facebook	Avg.	Rank
Heuristic									
CN	51.98 ±8.79	82.80 ±2.34	40.76 ±3.29	74.03 ±2.42	34.19 ±0.89	62.81 ±11.98	58.46 ±3.65	57.86	10.86
AA	60.98 ±3.65	87.48 ±2.44	43.54 ±3.14	74.03 ±2.42	34.19 ±0.89	68.62 ±0.40	66.53 ±1.17	62.20	7.79
RA	63.82 ±3.79	87.91 ±2.33	43.02 ±2.20	74.03 ±2.42	34.19 ±0.89	68.62 ±0.40	69.43 ±1.00	63.00	7.07
Katz	61.45 ±3.96	85.22 ±1.71	44.72 ±3.26	79.76 ±2.14	58.17 ±1.79	68.87 ±4.42	59.69 ±2.76	65.41	7.00
GNN4LP									
GCN	39.88 ±3.24	65.88 ±4.70	19.02 ±2.28	14.73 ±2.45	24.34 ±2.15	4.22 ±0.36	4.04 ±0.66	24.59	13.00
SEAL	62.14 ±3.82	87.72 ±1.36	47.59 ±3.28	**89.03** ±1.97	62.82 ±3.36	70.85 ±2.18	61.72 ±7.59	68.84	3.86
Neo-GNN	61.84 ±6.39	86.66 ±4.37	43.43 ±3.95	81.79 ±2.21	54.14 ±3.87	71.31 ±3.33	71.08 ±2.44	67.18	5.43
ELPH	56.27 ±4.45	78.42 ±1.81	44.17 ±2.33	77.23 ±2.52	50.56 ±3.59	69.44 ±1.87	62.68 ±2.45	62.68	8.00
BUDDY	40.40 ±14.15	77.41 ±1.90	44.83 ±2.37	76.81 ±2.68	51.59 ±3.56	69.55 ±1.78	63.09 ±1.69	60.53	7.86
NCNC	56.81 ±6.37	76.35 ±16.77	40.65 ±4.85	81.50 ±2.28	59.58 ±2.33	52.38 ±3.45	59.97 ±3.26	61.03	9.14
MPLP	69.44 ±3.74	91.72 ±2.07	51.13 ±5.26	89.00 ±1.58	55.71 ±2.31	66.76 ±2.72	60.08 ±2.77	69.12	4.57
Universal Link Predictor									
UniLP	66.22 ±3.14	86.94 ±1.63	48.04 ±3.08	87.85 ±1.10	56.92 ±4.59	66.20 ±2.51	64.35 ±1.31	68.07	5.14
LPLP	**75.24** ±3.07	**92.71** ±1.70	**53.21** ±2.52	88.38 ±2.10	**65.15** ±1.77	**72.87** ±1.67	**74.40** ±1.57	**74.57**	**1.29**
Reported results									
TFMLinker†	68.62 ±5.97	91.67 ±1.37	47.75 ±2.76	90.46 ±2.43	62.63 ±1.77	70.42 ±1.91	69.06 ±1.12	71.52	–

Table 2: Link prediction performance measured by Hits@50 (%) under HeaRT hard negatives. Values following ± denote standard deviations. The best results are shown in **bold**, and the second-best results are underlined.

Method	Celegans	USAir	PB	NS	Cora	CS	Facebook	Avg.	Rank
Heuristic									
CN	38.37 ±2.35	67.76 ±2.99	22.07 ±0.52	74.03 ±2.42	28.65 ±1.06	44.67 ±0.33	52.06 ±0.28	46.80	9.14
AA	43.26 ±2.46	72.71 ±2.65	22.53 ±0.50	74.03 ±2.42	30.19 ±1.17	52.06 ±0.43	54.68 ±0.34	49.92	5.43
RA	43.08 ±2.55	73.79 ±2.21	21.67 ±0.41	74.03 ±2.42	30.17 ±1.15	51.46 ±0.42	52.92 ±0.26	49.59	7.14
Katz	40.86 ±2.74	67.65 ±3.14	22.42 ±0.49	79.25 ±2.15	41.60 ±1.15	47.11 ±0.32	53.99 ±0.29	50.41	6.14
GNN4LP									
GCN	9.09 ±0.85	26.61 ±4.20	12.69 ±0.51	6.06 ±2.79	5.73 ±1.02	4.52 ±0.13	15.67 ±0.36	11.48	13.00
SEAL	40.68 ±3.38	68.09 ±3.65	22.21 ±0.75	82.99 ±2.50	47.23 ±1.75	50.50 ±1.11	49.81 ±7.63	51.64	5.43
Neo-GNN	34.22 ±5.31	65.74 ±6.57	18.80 ±1.24	79.09 ±2.14	33.82 ±2.50	25.69 ±7.26	54.60 ±2.05	44.57	8.86
ELPH	36.78 ±2.78	58.94 ±5.01	21.39 ±0.41	77.70 ±2.46	39.62 ±2.61	48.16 ±0.65	53.88 ±0.48	48.07	8.43
BUDDY	20.89 ±13.08	58.73 ±3.57	21.40 ±0.45	76.22 ±3.78	40.45 ±2.78	48.21 ±0.48	53.55 ±0.65	45.64	8.71
NCNC	21.00 ±8.24	44.94 ±14.27	15.94 ±1.63	78.98 ±3.74	39.75 ±2.60	35.33 ±2.42	46.51 ±2.17	40.35	10.29
MPLP	47.90 ±3.69	77.62 ±2.35	27.19 ±4.09	**83.72** ±2.23	44.82 ±2.90	47.89 ±0.74	55.14 ±1.18	54.90	3.14
Universal Link Predictor									
UniLP	44.83 ±2.59	69.76 ±2.88	22.06 ±0.53	82.04 ±1.92	48.37 ±1.24	49.85 ±0.27	56.07 ±0.35	53.28	3.86
LPLP	**55.92** ±2.22	**78.59** ±2.93	**29.17** ±0.84	81.53 ±2.45	**50.13** ±1.43	**55.66** ±0.27	**62.87** ±0.28	**59.12**	**1.43**

5.3 Link Prediction under Hard Negatives

Randomly sampled negative node pairs often have little structural connection, making them relatively easy to distinguish from positive links. We therefore also evaluate under the HeaRT protocol (Li et al., 2023), which ranks each positive link against 500 hard-negative candidates selected using RA (Zhou et al., 2009) and personalized PageRank (Page et al., 1999), as described in Appendix F.3. This protocol tests whether a method can distinguish true links from structurally plausible alternatives. As shown in Table 2, LPLP achieves the highest average Hits@50 of 59.12, exceeding the strongest baseline, MPLP, by 4.22 points, and ranks first on six of the seven graphs. These results suggest that the advantage of LPLP extends beyond distinguishing positive links from randomly sampled negatives.

5.4 Does LPLP Learn Structural Patterns from Context?

To examine which structural pattern LPLP takes from its context, we keep the graph and test queries of Table 1 and replace the uniformly sampled context with two structurally biased contexts of the same size, whose labels are all correct. Context A takes training links at distances 2 and 4 as positives and non-links at distance 3 as negatives, and Context B takes training links at distances 3 and 4 as

positives and non-links at distance 2 as negatives, where the distance of a link is measured after removing it. We report Hits@50 and the AUC for ranking test pairs at one distance above those at another, regardless of their labels.

As shown in Table 3, the uniform context favors closer pairs. Context B reverses the preference between $d = 2$ and $d = 3$, lowering the AUC for ranking $d = 2$ above $d = 3$ to 0.8 on Cora and 1.3 on CS. Context A instead keeps pairs at $d = 2$ above those at $d = 3$ and moves pairs at $d = 4$ above them, raising the AUC for ranking $d = 4$ above $d = 3$ from 18.0 to 93.5 on Cora and from 8.3 to 73.8 on CS.

Table 3: Hits@50 and the AUC (%) for ranking test pairs at one distance above those at another, averaged over ten splits. Bold marks the orderings specified by each biased context, and A† applies Context A to LPLP pretrained without rule links.

			AUC		
	Context	Hits@50	2>3	4>3	4>2
Cora	Uniform	65.1	79.7	18.0	4.2
	A	39.3	**98.7**	**93.5**	8.4
	A†	22.2	**94.9**	**85.3**	23.3
	B	7.5	**0.8**	32.3	**98.6**
CS	Uniform	72.9	90.1	8.3	0.3
	A	62.7	**98.8**	**73.8**	2.2
	A†	62.0	**93.3**	**45.1**	3.8
	B	0.0	**1.3**	82.7	**99.9**

LPLP thus ranks the middle distance below both a shorter and a longer one, an ordering that no monotone preference in distance can produce. UniLP partly follows Context B but keeps preferring closer pairs under Context A, with an AUC for ranking $d = 4$ above $d = 3$ of 18.2 on Cora and 5.8 on CS, as reported in Appendix G.5. Since the context positives are observed links, they also appear in the graph on which the message-passing branch of Section 4.2 operates. To examine whether this affects the orderings above, we remove them from that graph at inference and find that both contexts are still followed, as reported in Appendix G.6.

Both biased contexts lower Hits@50, as does the label permutation in Appendix G.1, indicating that LPLP takes its pattern from the context rather than from the graph alone. These results reaffirm the premise of PFNs that the context should represent the target graph, which uniform sampling of the observed links provides under random splits.

5.5 Effect of the Link-Formation Rule

To isolate the contribution of the link-formation rule, we pretrain the same architecture on a prior without rule links, setting $\rho = 0$ while keeping the base structure of Section 3 unchanged, and evaluate it on the same splits and contexts. Removing the rule lowers the average Hits@50 slightly, from 74.57 to 73.26 under random negatives and from 59.12 to 58.31 under hard negatives.

However, the rule-free model adapts less to Context A. Under the uniform context, the two models show nearly the same preference, ranking $d = 2$ above $d = 3$ above $d = 4$ on both graphs, but under Context A the rule-free model follows the specified ordering less closely, as shown in the rows marked A† in Table 3. Its AUC for ranking $d = 4$ above $d = 3$ rises only to 85.3 on Cora and 45.1 on CS, where LPLP reaches 93.5 and 73.8, so on CS it does not move pairs at $d = 4$ above those at $d = 3$. The link-formation rule, which varies the favored distances across tasks, thus strengthens how closely LPLP follows the structural pattern of its context. Results on each graph are provided in Appendix G.4.

5.6 Ablation on Structural Information

The subgraph encoder sees only the 2-hop enclosing subgraph of each pair, which contains little structure between the endpoints when they are far apart. To examine what the message-passing branch of Section 4.2 adds beyond this local view, we pretrain LPLP without it under the same prior and training recipe. We evaluate both models on Cora and on Power (Watts & Strogatz, 1998), a sparse power-grid network with many distant test links that is used only for this analysis, and group the test links by the distance d between their endpoints in the observed graph, averaging Hits@50 over five splits. Pairs at $d \geq 4$ include disconnected pairs, and links with an isolated

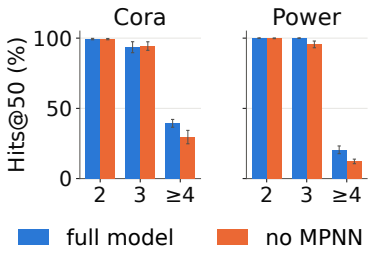


Figure 4: Hits@50 with and without the MPNN by endpoint distance d.

endpoint are omitted. As shown in Figure 4, removing the branch changes Hits@50 by at most 0.3 points at $d = 2$ but lowers it from 39.3 to 29.6 on Cora and from 20.5 to 12.2 on Power at $d \geq 4$, and over all test links it lowers Hits@50 by 4.0 and 2.2 points. These results indicate that the message-passing branch provides structural information complementary to the subgraph encoder, with larger gains for pairs whose endpoints are far apart.

6 Related Work

Link Prediction. Classical link predictors use fixed structural heuristics, such as neighborhood overlap or path-based scores (Newman, 2001; Adamic & Adar, 2003; Zhou et al., 2009; Katz, 1953; Page et al., 1999). GNN-based methods instead learn pair representations from data. SEAL (Zhang & Chen, 2018) introduced enclosing-subgraph encoding with endpoint-conditioned labels, and subsequent methods improved efficiency by combining node embeddings with dedicated pairwise structural features or readouts (Yun et al., 2021; Chamberlain et al., 2023; Wang et al., 2024; Dong et al., 2024). LPFormer (Shomer et al., 2024) further learns adaptive pairwise encodings by attending to multiple structural factors rather than relying on a fixed pairwise bias. These methods improve structural representations for the graph they are trained on, whereas LPLP infers from context which structural patterns predict links in an unseen graph.

Prior-Data Fitted Networks. PFNs (Müller et al., 2022) approximate the posterior predictive distribution of a prior by pretraining a transformer on tasks sampled from it. TabPFN (Hollmann et al., 2023; 2025) established this approach for tabular prediction, and PFNs have been extended to Bayesian optimization (Müller et al., 2023) and time-series forecasting (Dooley et al., 2023). For node-level prediction on graphs, recent studies have explored tabular foundation models with graph-derived features (Choi et al., 2025; Eremeev et al., 2026a), while GraphPFN (Eremeev et al., 2026b) and NodePFN (Choi et al., 2026) pretrain on synthetic graph priors. LPLP brings the PFN approach to link prediction by placing the prior over graph-specific link-formation rules.

Graph Foundation Models. Graph foundation models aim to transfer across datasets with a single pretrained model. Language-based approaches rely on textual semantics (Tang et al., 2024; Chai et al., 2023; Chen et al., 2024; Liu et al., 2024; Kong et al., 2025), whereas structure-based approaches transfer without text. OpenGraph (Xia et al., 2024) pretrains a graph transformer on LLM-generated synthetic graphs for zero-shot prediction, and PRODIGY (Huang et al., 2023) formulates in-context learning on graphs with prompt graphs that connect labeled examples to queries. For universal link prediction, UniLP (Dong et al., 2025) conditions link representations on labeled in-context links from the target graph, while TFMLinker (Liao et al., 2026) combines topology-aware link representations with a tabular foundation model. Both methods establish the value of context for adapting to unseen graphs but learn how to use it from real graphs, whereas LPLP learns this from synthetic graphs alone.

7 Conclusion

We introduced LPLP, a prior-data fitted network that infers the structural pattern of a graph from its observed links and labeled context. Pretrained only on a synthetic graph prior that exhibits a different link-formation rule in every graph, LPLP achieved the best average rank among the methods evaluated on shared splits across seven real-world benchmarks under both random and HeaRT hard negatives, without updating parameters on the target graph. Our analyses further indicate that LPLP follows the structural pattern of its context and that the link-formation rule strengthens this behavior. Together, these results suggest that synthetic graph priors can serve as a foundation for context-adaptive link prediction.

Like other PFNs, LPLP assumes that its labeled context represents the target graph. Random splits satisfy this assumption, but under temporal splits a context drawn from earlier links may not reflect the pattern of later links. LPLP also handles only undirected graphs without node attributes, and extracting an enclosing subgraph for every pair makes its inference more costly than scoring pairs from shared node representations. We leave extending the prior to directed, attributed, and temporal link formation and reducing this per-pair cost to future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist in drafting and polishing portions of the manuscript and to provide support during code implementation. All AI-assisted text was reviewed, revised, and verified by the authors. Any AI-generated or AI-suggested code was independently inspected, integrated, and tested by the authors, and all experiments and reported results were validated by the authors. Generative AI tools were not used to determine the research questions, scientific claims, or conclusions of this work. The authors take full responsibility for the final content of the paper, including all text, code, claims, and experimental results produced with the aid of generative AI.

ETHICS STATEMENT

This work uses only synthetic graphs and publicly available benchmark datasets and does not involve collecting new data from human subjects. As with other link predictors, LPLP could be applied to infer undisclosed relationships in social networks, and such applications should respect privacy and applicable regulations. We do not anticipate other significant negative societal impacts arising directly from this work.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide detailed descriptions of the synthetic graph prior in Appendix C, the LPLP architecture in Appendices E.2 and E.3, the training procedure in Appendix E.4, and the experimental setup in Appendix F. All experiments were conducted on a single NVIDIA RTX A6000 GPU. The source code and pretrained weights will be made publicly available upon acceptance.

REFERENCES

Robert Ackland. Mapping the u.s. political blogosphere: Are conservative bloggers more prominent? In *BlogTalk Downunder 2005 Conference*, Sydney, Australia, 2005.

Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003. ISSN 0378-8733. doi: https://doi.org/10.1016/S0378-8733(03)00009-1. URL https://www.sciencedirect.com/science/article/pii/S0378873303000091.

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. Graphllm: Boosting graph reasoning ability of large language model, 2023. URL https://arxiv.org/abs/2310.05845.

Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Yannick Hammerla, Michael M. Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=m1oqEOAozQU.

Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. Llaga: large language and graph assistant. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Jeongwhan Choi, Woosung Kang, Minseo Kim, Jongwoo Kim, and Noseong Park. Can tabpfn compete with gnns for node classification via graph tabularization?, 2025. URL https://arxiv.org/abs/2512.08798.

Jeongwhan Choi, Jongwoo Kim, Woosung Kang, and Noseong Park. Learning posterior predictive distributions for node classification from synthetic graph priors. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=FmxRzlu0rT.

Kaiwen Dong, Zhichun Guo, and Nitesh V Chawla. Pure message passing can estimate common neighbor for link prediction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Xa3dVaolKo.

Kaiwen Dong, Haitao Mao, Zhichun Guo, and Nitesh V Chawla. Universal link predictor by in-context learning on graphs. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=EYpqmoejB8.

Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha Venkat Naidu, and Colin White. ForecastPFN: Synthetically-trained zero-shot forecasting. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=tScBQRNgjk.

Dmitry Eremeev, Gleb Bazhenov, Oleg Platonov, Artem Babenko, and Liudmila Prokhorenkova. Turning tabular foundation models into graph foundation models, 2026a. URL https://arxiv.org/abs/2508.20906.

Dmitry Eremeev, Oleg Platonov, Gleb Bazhenov, Artem Babenko, and Liudmila Prokhorenkova. GraphPFN: A prior-data fitted graph foundation model. In *International Conference on Machine Learning*, 2026b.

Amir Ghasemian, Homa Hosseinmardi, Aram Galstyan, Edoardo M. Airoldi, and Aaron Clauset. Stacking models for nearly optimal link prediction in complex networks. *Proceedings of the National Academy of Sciences*, 117(38):23393–23400, 2020. doi: 10.1073/pnas.1914950117. URL https://www.pnas.org/doi/abs/10.1073/pnas.1914950117.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cp5PvcI6w8_.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, Jan 2025. ISSN 1476-4687. doi: 10.1038/s41586-024-08328-6. URL https://doi.org/10.1038/s41586-024-08328-6.

Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy Liang, and Jure Leskovec. PRODIGY: Enabling in-context learning over graphs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=pLwYhNNnoR.

Brian Karrer and M. E. J. Newman. Stochastic blockmodels and community structure in networks. *Phys. Rev. E*, 83:016107, Jan 2011. doi: 10.1103/PhysRevE.83.016107. URL https://link.aps.org/doi/10.1103/PhysRevE.83.016107.

Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, Mar 1953. ISSN 1860-0980. doi: 10.1007/BF02289026. URL https://doi.org/10.1007/BF02289026.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Lecheng Kong, Jiarui Feng, Hao Liu, Chengsong Huang, Jiaxin Huang, Yixin Chen, and Muhan Zhang. GOFA: A generative one-for-all model for joint graph language modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mIjblC9hfm.

István A. Kovács, Katja Luck, Kerstin Spirohn, Yang Wang, Carl Pollis, Sadie Schlabach, Wenting Bian, Dae-Kyum Kim, Nishka Kishore, Tong Hao, Michael A. Calderwood, Marc Vidal, and Albert-László Barabási. Network-based prediction of protein interactions. *Nature Communications*, 10(1):1240, Mar 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-09177-y. URL https://doi.org/10.1038/s41467-019-09177-y.

Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. Evaluating graph neural networks for link prediction: current pitfalls and new benchmarking. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: design provably more powerful neural networks for graph representation learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Tianyin Liao, Chunyu Hu, Yicheng Sui, Xingxuan Zhang, Peng Cui, Jianxin Li, and Ziwei Zhang. TFMLinker: Universal link predictor by graph in-context learning with tabular foundation models. In *Workshop on Graph Foundation Models: A New Era for Graph Machine Learning*, 2026. URL https://openreview.net/forum?id=wukGf8u9Um.

David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pp. 556–559, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581137230. doi: 10.1145/956863.956972. URL https://doi.org/10.1145/956863.956972.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4IT2pgc9v6.

Haitao Mao, Juanhui Li, Harry Shomer, Bingheng Li, Wenqi Fan, Yao Ma, Tong Zhao, Neil Shah, and Jiliang Tang. Revisiting link prediction: a data perspective. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8Ur2xmuw7w.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. Pfns4bo: in-context learning for bayesian optimization. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

M. E. J. Newman. Clustering and preferential attachment in growing networks. *Phys. Rev. E*, 64: 025102(R), Jul 2001. doi: 10.1103/PhysRevE.64.025102. URL https://link.aps.org/doi/10.1103/PhysRevE.64.025102.

M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104, Sep 2006. doi: 10.1103/PhysRevE.74.036104. URL https://link.aps.org/doi/10.1103/PhysRevE.74.036104.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. In *The Web Conference*, 1999. URL https://api.semanticscholar.org/CorpusID:1508503.

Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL https://networkrepository.com.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 04 2021. ISSN 2051-1329. doi: 10.1093/comnet/cnab014. URL https://doi.org/10.1093/comnet/cnab014.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation, 2019. URL https://arxiv.org/abs/1811.05868.

Harry Shomer, Yao Ma, Haitao Mao, Juanhui Li, Bo Wu, and Jiliang Tang. Lpformer: An adaptive graph transformer for link prediction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 2686–2698, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3672025. URL https://doi.org/10.1145/3637528.3672025.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pp. 491–500, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657775. URL https://doi.org/10.1145/3626772.3657775.

Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sNFLN3itAd.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, Jun 1998. ISSN 1476-4687. doi: 10.1038/30918. URL https://doi.org/10.1038/30918.

Lianghao Xia, Ben Kao, and Chao Huang. OpenGraph: Towards open graph foundation models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 2365–2379, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.132. URL https://aclanthology.org/2024.findings-emnlp.132/.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on Machine Learning, Volume 48*, ICML'16, pp. 40–48. JMLR.org, 2016.

Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J. Kim. Neo-gnns: neighborhood overlap-aware graph neural networks for link prediction. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 5171–5181, Red Hook, NY, USA, 2018. Curran Associates Inc.

Muhan Zhang and Yixin Chen. Inductive matrix completion based on graph neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ByxxgCEYDS.

Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, Oct 2009. ISSN 1434-6036. doi: 10.1140/epjb/e2009-00335-8. URL https://doi.org/10.1140/epjb/e2009-00335-8.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal A. C. Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=DEsIX_D_vR.

A Diversity of Link Formation in Graph Priors

Figure 1 compares the diversity of structural patterns among graphs sampled from three synthetic graph priors. We sample graphs from each prior with the configuration used to pretrain the corresponding model and summarize each graph by its class-balanced link rates at distances 2 and 3.

A.1 Graph Priors

The NodePFN prior (Choi et al., 2026) generates each graph either as an Erdős–Rényi graph or as a stochastic block model whose blocks are the classes of synthetic node labels. We sample it with the official NodePFN pretraining pipeline.[1] The GraphPFN prior (Eremeev et al., 2026b) generates each graph with a multi-level degree-corrected stochastic block model followed by preferential attachment. We sample it with the pretraining configuration released in the official GraphPFN repository.[2] Our prior is the prior of Section 3, sampled with the distributions used to train LPLP (Table 5). We draw 1000 graphs each from the GraphPFN prior and our prior. Since the NodePFN prior mixes two graph models, we draw 2000 graphs from it, about half from each model. All statistics are computed on the complete sampled graph, before any links are hidden for a link prediction task, and thus reflect only how each prior forms links.

A.2 Class-Balanced Link Rate

Let $G = (V, E)$ be a sampled graph and $\bar{E}$ the set of unlinked node pairs. For a linked pair $(u, v) \in E$, we measure the shortest-path distance $d(u, v)$ in G with only that link removed. For an unlinked pair, we measure the distance in G. Distance 2 therefore means that the pair has at least one common neighbor, and distance 3 means that the pair has no common neighbor but is connected by a path of length 3. Let

$$\pi^+(k) = \frac{|\{(u,v) \in E : d(u,v) = k\}|}{|E|}, \qquad \pi^-(k) = \frac{|\{(u,v) \in \bar{E} : d(u,v) = k\}|}{|\bar{E}|} \tag{7}$$

be the fractions of linked and unlinked pairs at distance k. The class-balanced link rate at distance k is

$$q(k) = \frac{\pi^+(k)}{\pi^+(k) + \pi^-(k)}, \tag{8}$$

which is the fraction of links among distance-k pairs when linked and unlinked pairs are drawn in equal numbers. We leave $q(k)$ undefined when fewer than 50 node pairs lie at distance k and plot a graph only if both $q(2)$ and $q(3)$ are defined.

A.3 Plot and Summary Statistics

Table 4: Distance preferences of the graphs in Figure 1. A preference requires a margin of 0.1 between $q(2)$ and $q(3)$.

Prior	Distance 2 (%)	No preference (%)	Distance 3 (%)	Median $\lvert q(2) - q(3) \rvert$
GraphPFN	96.3	3.7	0.0	0.540
NodePFN	5.4	94.3	0.3	0.009
Ours	50.6	17.6	31.9	0.417

Each point in Figure 1 places one graph at $(q(2), q(3))$. The dotted line is the diagonal $q(2) = q(3)$, and the shaded region above it contains graphs in which distance-3 pairs are more likely to be linked than distance-2 pairs. Points below the diagonal are graphs in which pairs sharing a common neighbor are more likely to be linked, and the vertical distance from the diagonal shows how strongly one distance is preferred over the other. A prior whose graphs spread to both sides of the diagonal therefore generates tasks in which this preference reverses, whereas a prior whose graphs stay on one

[1] https://github.com/jeongwhanchoi/NodePFN

[2] https://github.com/yandex-research/graphpfn

side or near the diagonal does not. Table 4 reports the fractions of graphs below, near, and above the diagonal with a margin of 0.1.

B Posterior Predictive Approximation with PFNs

We follow the notation of Section 2.1. Let the context set, query input, and query label be sampled jointly from the synthetic task prior. For link prediction, the observed graph G is part of every input, and we omit it from the notation. The training objective of a PFN is the following.

$$\mathcal{L}_{\mathrm{PFN}}(\theta) = \mathbb{E}_{(\mathcal{D}_{\mathrm{ctx}}, \mathbf{x}_q, y_q)} \left[-\log q_\theta \left(y_q \mid \mathbf{x}_q, \mathcal{D}_{\mathrm{ctx}} \right) \right]. \tag{9}$$

Conditioning on $\mathbf{x}_q$ and $\mathcal{D}_{\mathrm{ctx}}$ gives

$$\begin{aligned} \mathcal{L}_{\mathrm{PFN}}(\theta) &= \mathbb{E}_{\mathbf{x}_q, \mathcal{D}_{\mathrm{ctx}}} \left[-\sum_{y \in \mathcal{Y}} p\left(y \mid \mathbf{x}_q, \mathcal{D}_{\mathrm{ctx}} \right) \log q_\theta \left(y \mid \mathbf{x}_q, \mathcal{D}_{\mathrm{ctx}} \right) \right] \\ &= \mathbb{E}_{\mathbf{x}_q, \mathcal{D}_{\mathrm{ctx}}} \left[H\left(p(\cdot \mid \mathbf{x}_q, \mathcal{D}_{\mathrm{ctx}}), q_\theta(\cdot \mid \mathbf{x}_q, \mathcal{D}_{\mathrm{ctx}}) \right) \right] \\ &= \mathbb{E}_{\mathbf{x}_q, \mathcal{D}_{\mathrm{ctx}}} \left[\mathrm{KL}\left(p(\cdot \mid \mathbf{x}_q, \mathcal{D}_{\mathrm{ctx}}) \,\|\, q_\theta(\cdot \mid \mathbf{x}_q, \mathcal{D}_{\mathrm{ctx}}) \right) \right] + C, \end{aligned} \tag{10}$$

where $C = \mathbb{E}_{\mathbf{x}_q, \mathcal{D}_{\mathrm{ctx}}}[H(p(\cdot \mid \mathbf{x}_q, \mathcal{D}_{\mathrm{ctx}}))]$ does not depend on θ.

Minimizing equation 9 is therefore equivalent to minimizing the expected KL divergence between the posterior predictive distribution induced by the prior and its approximation q_θ.

C Details of the Synthetic Graph Prior

C.1 Sampling Distributions

Table 5 lists the sampling distributions of the prior in Section 3, all of which are drawn anew for every synthetic graph. $\mathrm{U}[a, b]$ denotes the continuous uniform distribution, $\mathrm{U}\{a, \dots, b\}$ the discrete uniform distribution, and $\mathrm{LogU}[a, b]$ the log-uniform distribution.

Table 5: Per-graph sampling distributions of the synthetic prior.

Quantity	Distribution
Base structure ψ	
Number of nodes n	$\mathrm{U}\{256, \dots, 4096\}$
Average degree $\bar{d}$	$\mathrm{LogU}[1.5, 32]$
Number of communities C	$\mathrm{U}\{2, \dots, 20\}$
Community proportions	$\mathrm{Dirichlet}(\mathbf{1}_C)$
Number of macro-groups	$\mathrm{U}\{1, \dots, \mathrm{round}(\sqrt{C})\}$
Within-vs-between contrast a	$\mathrm{U}[-2.5, 2.5]$
Macro-group bonus a_{mac}	$\mathrm{U}[0, 1.5]$
Affinity jitter scale σ_B	$\mathrm{U}[0, 1]$
Cell size	$\mathrm{LogU}[3, 20]$, rounded
Cell bonus b	$\mathrm{U}[2, 6]$
Activity scale σ_θ	$\mathrm{U}[0.3, 0.7]$
Activity relation γ	$\mathrm{U}[-3, 3]$
Link-formation rule ω	
Rule fraction ρ	$\mathrm{U}[0, 0.5]$
Rule strength α	$\mathrm{U}[0, 10]$
Longest distance K	$\mathrm{U}\{2, \dots, 8\}$
Weight concentration η	$\mathrm{LogU}[0.1, 10]$
Distance weights $(w_2, \dots, w_K)$	$\mathrm{Dirichlet}(\eta \mathbf{1}_{K-1})$
Observation and task	
Hiding probability r	$\mathrm{U}[0.1, 0.5]$
Label inversion	with probability $1/2$

C.2 Graph Generation

Base logit. Each node u is assigned to a community c_u according to the sampled community proportions, and each community c is assigned uniformly at random to a macro-group g_c. The block affinity between communities c and c' is

$$B_{cc'} = a\left(2\,\mathbf{1}[c = c'] - 1\right) + a_{\mathrm{mac}}\,\mathbf{1}[c \neq c',\ g_c = g_{c'}] + \epsilon_{cc'}, \tag{11}$$

where $\boldsymbol{\epsilon} = \frac{1}{2}(\mathbf{E} + \mathbf{E}^\top)$ with $E_{cc'} \sim \mathcal{N}(0, \sigma_B^2)$. Within each community, the nodes are shuffled and grouped into consecutive cells of the sampled cell size, and κ_u denotes the cell of node u. Each node receives a latent activity $\log\theta_u \sim \mathcal{N}(0, \sigma_\theta^2)$ clipped to $[-3, 3]$, and $\tilde{\theta}_u$ denotes $\log\theta_u$ standardized over the nodes of the graph and clipped to $[-3, 3]$. The base logit is

$$\ell_0(u, v) = B_{c_u c_v} + b\,\mathbf{1}[\kappa_u = \kappa_v] + \log\theta_u + \log\theta_v + \gamma\,\tilde{\theta}_u\tilde{\theta}_v. \tag{12}$$

The last term favors links between nodes of similar activity when $\gamma > 0$ and between nodes of dissimilar activity when $\gamma < 0$.

Path counts. Write $\mathbf{A}_0$ for the adjacency matrix of the seed graph G_0, $d_{0,u} = \sum_v (\mathbf{A}_0)_{uv}$ for its node degrees, and $\mathbf{D}_0 = \mathrm{diag}(\max\{d_{0,u}, 1\})$. The matrix $\mathbf{S}_k = (\mathbf{A}_0\mathbf{D}_0^{-1})^{k-1}\mathbf{A}_0$ counts the walks of length k between two nodes, each walk weighted by the product of $1/d_{0,j}$ over its intermediate nodes j, so that $\mathbf{S}_2$ is the resource-allocation matrix. The unnormalized path count is

$$h_k(u, v) = \log\Big(1 + \frac{n\,(\mathbf{S}_k)_{uv}}{\sqrt{\max\{d_{0,u}, 1\}\max\{d_{0,v}, 1\}}}\Big). \tag{13}$$

For a pair at distance exactly k, every walk of length k between the two nodes is a shortest path.

Per-distance normalization. Let $\mathrm{dist}_0(u, v)$ be the shortest-path distance between u and v in G_0 and μ_k the mean of h_k over the pairs with $\mathrm{dist}_0(u, v) = k$. The path count used in equation 4 is

$$\xi_k(u, v) = \tau_k\,\mathbf{1}[\mathrm{dist}_0(u, v) = k]\,h_k(u, v), \tag{14}$$

with $\tau_k = \mu_2/\mu_k$ when both means are positive and $\tau_k = 1$ otherwise. When both means are positive, ξ_k therefore has the same mean over the pairs at distance k as ξ_2 has over the pairs at distance 2, which places the distances on the common scale assumed in Section 3. If no pair lies at distance k, the distance-k term contributes zero, and the distance-2 term is unchanged by this normalization. Without the rescaling, walk mass would decay with k and $\mathbf{w}$ would have little effect at long range.

Sampling with a fixed budget. The sampled average degree $\bar{d}$ sets a total link budget $m = \mathrm{round}(n\bar{d}/2)$. Of this budget, $\mathrm{round}(\rho m)$ links are rule links and the rest are seed links, so $\rho = 0$ gives a graph without rule links. The seed links $\mathcal{E}_0$ are drawn without replacement with probabilities proportional to $\exp\ell_0$ over $\mathcal{U}$, and the rule links $\mathcal{E}_1$ are then drawn in the same way from $\exp\ell_1$ over $\mathcal{U} \setminus \mathcal{E}_0$, with ℓ_1 computed once on G_0.

D Node Classification Ability of LPLP

D.1 Joint Prior for Node Classification and Link Prediction

Every graph sampled from the prior in Section 3 can also be used for node classification, so one pretraining task includes both link prediction and node classification on the same graph.

Node features. The node features are generated by the MLP-based structural causal model (SCM) prior of TabPFN (Hollmann et al., 2023), as used in NodePFN (Choi et al., 2026). For every node u, a randomly initialized SCM produces a feature vector $\mathbf{x}_u$ and a scalar latent target t_u from the same causal process. The number of used features is drawn from $\mathrm{U}\{3, \ldots, 100\}$, and, as in NodePFN, the features are rescaled by 100 divided by this number and zero-padded to 100 dimensions.

Class labels. The classes are the communities c_u of the base structure described in Appendix C.2. To couple the features with the classes, the n pairs $(\mathbf{x}_i, t_i)$ are sorted by t_i, the communities are put in a random order, and the sorted sequence is cut into consecutive blocks whose sizes equal the community sizes in that order. The pairs of each block are assigned to the nodes of the corresponding community in random order, and each community receives a randomly permuted class index, which we write as y_u for its nodes u. Each feature vector therefore stays with its own latent target, and each class covers a contiguous range of targets, which makes the class predictable from the features. The graph carries the same classes through its community structure, which the block affinity makes assortative or disassortative. NodePFN instead derives the classes from the targets and then draws a stochastic block model on these classes. Because our graph is already generated by the link prediction prior, we condition the partition of the targets on the existing communities, which leaves the link prediction prior unchanged. Graphs that contain a community with fewer than two nodes are discarded.

Context and query nodes. All n nodes form the token sequence. The nodes are split at random into context nodes $\mathcal{V}_{\mathrm{ctx}}$ and query nodes $\mathcal{V}_{\mathrm{qry}} = \mathcal{V} \setminus \mathcal{V}_{\mathrm{ctx}}$, where $|\mathcal{V}_{\mathrm{ctx}}|$ is drawn from $\mathcal{U}\{64, \ldots, n - C\}$, with C the number of communities, and every class has at least one node in both sets. The context set $\mathcal{D}^{\mathrm{nc}}_{\mathrm{ctx}} = \{(\mathbf{x}_u, y_u) : u \in \mathcal{V}_{\mathrm{ctx}}\}$ contains the features and classes of the context nodes, and the query set $\mathcal{D}^{\mathrm{nc}}_{\mathrm{qry}} = \{(\mathbf{x}_u, y_u) : u \in \mathcal{V}_{\mathrm{qry}}\}$ contains the remaining nodes. The model predicts the class y_u of each query node from $\mathbf{x}_u$, G, and $\mathcal{D}^{\mathrm{nc}}_{\mathrm{ctx}}$ with a softmax over 20 classes, the largest possible number of communities. Message passing runs on the observed graph G of the link prediction task, so the hidden links are invisible to both tasks.

D.2 Joint Training Objective

LPLP is pretrained on the sum of the link prediction loss in equation 6 and the node classification loss,

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathrm{lp}}(\theta) + \mathbb{E}\Big[- \frac{1}{|\mathcal{V}_{\mathrm{qry}}|} \sum_{u \in \mathcal{V}_{\mathrm{qry}}} \log q_\theta(y_u \mid \mathbf{x}_u, G, \mathcal{D}^{\mathrm{nc}}_{\mathrm{ctx}}) \Big]. \tag{15}$$

D.3 Node Classification Results

Models. We evaluate LPLP, jointly pretrained on link prediction and node classification, against two pretrained node-level PFNs, NodePFN (Choi et al., 2026) and GraphPFN-1.3 (Eremeev et al., 2026b). All model weights remain frozen, and no model is fine-tuned on the target graphs.

Benchmarks. Following NodePFN, we evaluate on its 23 graph benchmarks, 13 homophilous and 10 heterophilous, and use its data splits. All models receive the same graphs, node features, labels, and splits.

Inference. Only the labels of training nodes are given as context, while all node features and the full graph are available under a transductive protocol. LPLP and NodePFN use the released NodePFN evaluation presets for feature smoothing, truncated SVD, and ensembling, without an additional hyperparameter search. GraphPFN-1.3 uses its own in-context inference pipeline with feature standardization, PCA to at most 64 dimensions except on Pubmed, eight random features, and ten ensemble members. Each dataset is evaluated with five inference seeds, and we report the mean and standard deviation of test accuracy over these runs. All experiments are conducted on NVIDIA RTX A6000 GPUs.

Results. Beyond link prediction, the same jointly pretrained checkpoint achieves competitive node classification performance against established node-level PFNs, without target-graph fine-tuning. As shown in Table 6, across 23 datasets, LPLP reaches a mean accuracy of 71.94%, compared with 70.10% for NodePFN and 72.71% for GraphPFN-1.3. This suggests that a single set of shared layers can support in-context prediction at both the link and the node level.

Table 6: Node classification on the NodePFN benchmark. Test accuracy (%) is reported as mean ± standard deviation over five runs with shared splits and frozen checkpoints.

Dataset	NodePFN	GraphPFN-1.3	LPLP
Homophily			
AirBrazil	69.23 ± 5.44	82.31 ± 1.88	76.92 ± 2.43
AirEU	54.25 ± 1.65	62.88 ± 0.85	62.62 ± 1.21
AirUS	60.36 ± 0.70	66.41 ± 0.81	59.06 ± 0.84
Cora	82.06 ± 0.29	74.28 ± 0.78	81.44 ± 0.23
Citeseer	67.90 ± 0.25	69.60 ± 0.42	68.42 ± 0.30
Pubmed	78.64 ± 0.29	78.00 ± 1.63	78.50 ± 0.19
WikiCS	75.98 ± 0.80	78.99 ± 0.79	79.52 ± 0.65
Amazon-Photo	89.58 ± 0.37	91.11 ± 0.14	90.69 ± 0.18
Amazon-Comp	73.65 ± 0.88	81.13 ± 0.36	77.45 ± 0.17
DBLP	72.24 ± 1.04	68.84 ± 1.54	74.97 ± 0.64
Coauthor-CS	90.50 ± 0.45	91.24 ± 0.22	89.73 ± 0.14
Coauthor-Physics	91.79 ± 0.41	90.56 ± 0.18	91.81 ± 0.47
Deezer	55.87 ± 0.96	57.50 ± 0.19	51.67 ± 0.50
Heterophily			
Cornell	71.89 ± 2.76	70.81 ± 2.65	75.68 ± 5.13
Texas	74.59 ± 6.53	78.92 ± 6.92	83.24 ± 3.97
Wisconsin	80.00 ± 6.13	80.78 ± 3.37	85.10 ± 5.35
Chameleon	46.46 ± 3.06	50.54 ± 3.47	47.73 ± 1.95
Actor	33.20 ± 0.34	37.34 ± 1.51	34.29 ± 0.75
Minesweeper	80.66 ± 0.25	82.93 ± 0.57	82.50 ± 0.37
Tolokers	78.62 ± 0.19	83.91 ± 0.42	78.84 ± 0.11
Amazon-Ratings	44.68 ± 0.48	48.23 ± 0.34	43.78 ± 0.49
Questions	97.01 ± 0.00	97.27 ± 0.03	97.03 ± 0.02
Squirrel	43.18 ± 1.05	48.70 ± 1.79	43.67 ± 1.44
Homophily mean (13)	74.00	76.37	75.60
Heterophily mean (10)	65.03	67.94	67.19
Mean (23)	70.10	72.71	71.94

E Implementation Details of LPLP

E.1 Structural Node Labeling

For a pair $e = (u, v)$ and its enclosing subgraph G_e, let $d_u(i)$ and $d_v(i)$ be the shortest-path distances from node i to u and v in G_e, where $d_u(i)$ is computed with v removed from G_e and $d_v(i)$ with u removed. Removing the other endpoint prevents a path through one endpoint from shortening the distance to the other. Double-radius node labeling (DRNL) (Zhang & Chen, 2018) maps the two distances to a single integer,

$$\mathrm{DRNL}(i) = 1 + \min(d_u, d_v) + \lfloor s/2 \rfloor \big(\lfloor s/2 \rfloor + s \bmod 2 - 1 \big), \qquad s = d_u + d_v, \tag{16}$$

with $\mathrm{DRNL}(u) = \mathrm{DRNL}(v) = 1$. The code is a perfect hash of the unordered distance pair (d_u, d_v), so nodes at the same distances from the two endpoints share a label, and every common neighbor of u and v receives the label 2.

DRNL assigns a single label to every node that cannot reach one of the two endpoints within G_e, which discards its distance to the endpoint it does reach. DRNL+ (Dong et al., 2025) keeps the DRNL code for nodes that reach both endpoints and records the single finite distance otherwise,

$$\mathrm{DRNL}^{+}(i) = \begin{cases} (\mathrm{DRNL}(i),\, 0), & d_u, d_v < \infty, \\ (0,\, d_u), & d_v = \infty, \\ (0,\, d_v), & d_u = \infty. \end{cases} \tag{17}$$

Each distinct value of $\mathrm{DRNL}^{+}(i)$ is mapped to a learned embedding, which serves as the input feature of node i to the subgraph encoder. Because the labels depend only on distances within G_e, they are invariant to node identities and require no node features.

E.2 LPLP Architecture

Table 7 lists the configuration of the components described in Section 4.1 and Section 4.2. The subgraph encoder produces a 128-dimensional pair representation, which a linear layer with layer normalization maps to the d-dimensional input token. Context tokens additionally receive a learned embedding of their label, and query tokens receive none. The node signatures are drawn from $\{\pm 1/\sqrt{d}\}^d$ independently for every graph. The link prediction decoder outputs two logits, and their softmax gives $q_\theta(y_e \mid e, G, \mathcal{D}^{\mathrm{lp}}_{\mathrm{ctx}})$.

For node classification, every node of G is a token, as in NodePFN (Choi et al., 2026). A linear feature encoder maps the node features to d dimensions, and a linear label encoder adds the class of each context node. The node tokens pass through the same L layers, whose attention, message-passing, and feed-forward weights are shared with link prediction, and a separate decoder outputs the class logits. A learned embedding of each task is added to the attention inputs so that the shared layers can distinguish the two tasks.

Table 7: Architecture of LPLP.

Component	Setting
Subgraph encoder	
Enclosing subgraph	2-hop, link (u, v) removed if observed
Node labeling	DRNL+
GNN	GraphSAGE, 3 layers, hidden dimension 128
Activation, dropout	ReLU, 0.5
Endpoint readout	symmetric MLP, hidden dimension 256
Shared layers	
Number of layers L	12
Token dimension d	512
Attention heads	4
FFN hidden dimension	1024
Dropout	0
Node signature dimension	512
Pair readout	symmetric MLP, hidden dimension 256, shared across layers
Link prediction head	
Decoder	MLP, hidden dimension 1024
Node classification	
Feature encoder	linear, input dimension 100
Label encoder	linear map of the class index
Decoder	MLP, hidden dimension 1024, up to 20 classes
Number of parameters	30.5M

E.3 Layer Computation and Parameter Sharing

This subsection details the computation of the layer in equation 5. Let $\tilde{\mathbf{A}} = \mathbf{A}+\mathbf{I}$ be the adjacency matrix of the observed graph G with self-loops, $\tilde{\mathbf{D}}$ its diagonal degree matrix, and $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$. Layer normalization LN is applied to each row, and every layer normalization below has its own parameters.

Node states. The node states start from the random signatures, $\mathbf{Z}^{(0)} = \tilde{\mathbf{Z}}^{(0)} = \mathbf{R}$, whose rows are drawn uniformly from $\{\pm 1/\sqrt{d}\}^d$. Each layer applies one GCN step with a residual connection,

$$\mathbf{Z}^{(l+1)} = \tilde{\mathbf{Z}}^{(l)} + \hat{\mathbf{A}}\tilde{\mathbf{Z}}^{(l)}\mathbf{W}^{(l)} + \mathbf{1}\mathbf{b}^{(l)\top}, \qquad \tilde{\mathbf{Z}}^{(l+1)} = \mathrm{LN}^{(l)}_{z}\big(\mathbf{Z}^{(l+1)}\big). \tag{18}$$

The pair readout below uses $\mathbf{Z}^{(l+1)}$, the state before layer normalization, which keeps the magnitude of the propagated signatures.

Pair tokens. For a pair $e = (u, v)$, the message-passing branch reads the two endpoint states with a symmetric MLP and adds the result to the pair token with a learned scalar $s^{(l)}$,

$$\mathbf{m}_e^{(l)} = \mathrm{LN}_m^{(l)}\Big(\mathbf{h}_e^{(l)} + \tfrac{s^{(l)}}{2}\big(f([\mathbf{z}_u^{(l+1)}; \mathbf{z}_v^{(l+1)}]) + f([\mathbf{z}_v^{(l+1)}; \mathbf{z}_u^{(l+1)}])\big)\Big), \tag{19}$$

where $f(\mathbf{x}) = \mathbf{W}_2\,\mathrm{GELU}(\mathbf{W}_1\mathbf{x} + \mathbf{c}_1) + \mathbf{c}_2$ is a two-layer MLP. The layer in equation 5 abbreviates this term as $\mathrm{LN}\big(\mathbf{h}_e^{(l)} + \mathrm{Readout}(\mathbf{z}_u^{(l+1)}, \mathbf{z}_v^{(l+1)})\big)$. The attention branch uses multi-head attention $\mathrm{MHA}^{(l)}(\mathbf{q}, \mathcal{S})$ with 4 heads, query $\mathbf{q}$, and keys and values $\mathcal{S}$,

$$\mathbf{g}_e^{(l)} = \mathrm{LN}_a^{(l)}\Big(\mathbf{h}_e^{(l)} + \mathrm{MHA}^{(l)}\big(\mathbf{h}_e^{(l)} + \mathbf{t}_{\mathrm{lp}},\ \{\mathbf{h}_c^{(l)} + \mathbf{t}_{\mathrm{lp}} : c \in \mathcal{D}_{\mathrm{ctx}}^{\mathrm{lp}}\}\big)\Big), \tag{20}$$

where $\mathbf{t}_{\mathrm{lp}}$ is a learned task embedding. The keys and values always come from the context tokens, so every token, whether a context or a query token, attends to all context tokens and to no query token. The two branches are summed and passed through a residual feed-forward block,

$$\mathbf{r}_e^{(l)} = \mathbf{m}_e^{(l)} + \mathbf{g}_e^{(l)}, \qquad \mathbf{h}_e^{(l+1)} = \mathrm{LN}_{f,\mathrm{lp}}^{(l)}\big(\mathbf{r}_e^{(l)} + \mathbf{W}_4^{(l)}\,\mathrm{GELU}(\mathbf{W}_3^{(l)}\mathbf{r}_e^{(l)} + \mathbf{c}_3^{(l)}) + \mathbf{c}_4^{(l)}\big). \tag{21}$$

Node tokens. For node classification, the input token of node u is $\mathbf{W}_x\mathbf{x}_u + \mathbf{c}_x$, to which context nodes add $\mathbf{w}_y y_u + \mathbf{c}_y$. The same layer is applied to the node tokens $\mathbf{X}^{(l)}$ with three changes. The message-passing branch applies the GCN step of equation 18 to $\mathbf{X}^{(l)}$ itself in place of $\mathbf{m}_e^{(l)}$, the attention of equation 20 uses the task embedding $\mathbf{t}_{\mathrm{nc}}$ with the context nodes as keys and values, and the feed-forward block of equation 21 ends with its own layer normalization $\mathrm{LN}_{f,\mathrm{nc}}^{(l)}$.

Parameter sharing. Each of the $L = 12$ layers has its own attention weights, GCN weights $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$, feed-forward weights, and layer normalizations. Within a layer, the two tasks share the attention weights, $\mathrm{LN}_a^{(l)}$, the GCN weights, $\mathrm{LN}_z^{(l)}$, and the feed-forward weights $\mathbf{W}_3^{(l)}$, $\mathbf{c}_3^{(l)}$, $\mathbf{W}_4^{(l)}$, and $\mathbf{c}_4^{(l)}$, and each task has its own final layer normalization of the feed-forward block. The readout MLP f, shared by all layers, the per-layer scalar $s^{(l)}$, initialized to 0.1, and $\mathrm{LN}_m^{(l)}$ are used only for link prediction. The subgraph encoder, the token projection, the label embedding, $\mathbf{t}_{\mathrm{lp}}$, and the link prediction decoder are also used only for link prediction, and $\mathbf{W}_x$, the label encoder, $\mathbf{t}_{\mathrm{nc}}$, and the node classification decoder only for node classification.

E.4 Pretraining

LPLP is pretrained on synthetic graphs sampled from the prior in Section 3, with the distributions in Table 5. Each sampled graph carries a link prediction task and a node classification task, and we optimize the joint objective in equation 15. The link prediction task contains at most 1024 context and query pairs with equal numbers of positive and negative pairs. Training runs for 30 epochs of 1024 steps, each on a batch of 8 graphs, and gradients are accumulated over 2 steps, so the parameters are updated 512 times per epoch with 16 graphs per update and the model sees 245,760 graphs in total. Pretraining takes about 37 hours on a single NVIDIA RTX A6000 GPU, with the synthetic graphs generated on the fly by 24 CPU worker processes. Evaluation uses the exponential moving average (EMA) of the weights after the final epoch. Table 8 lists the remaining hyperparameters.

F Experimental Details

F.1 Datasets and Data Splits

Following UniLP (Dong et al., 2025) and TFMLinker (Liao et al., 2026), we evaluate on seven real-world graphs from six domains for the main comparison. Celegans, USAir, NS, and PB, as well as Power used in the ablation of Section 5.6, are taken from the UniLP repository[3], Cora from Planetoid[4], CS from the GNN benchmark of Shchur et al. (2019)[5], and Facebook from SNAP[6]. Each

[3] https://github.com/Barcavin/context_LP/tree/master/data

[4] https://github.com/kimiyoung/planetoid/raw/master/data

[5] https://github.com/shchur/gnn-benchmark/raw/master/data/npz

[6] https://snap.stanford.edu/data/facebook_large.zip

Table 8: Pretraining hyperparameters of LPLP.

Hyperparameter	Value
Link prediction tasks	
Pairs per task	at most 1024
Context size	$\mathrm{U}\{2, \ldots, \min(1022, 2\|\mathcal{E}\|)\}$
Positive-to-negative ratio	1:1 in context and query
Node classification tasks	
Feature dimension	100
Used features	$\mathrm{U}\{3, \ldots, 100\}$, remaining dimensions zero-padded
Classes	communities of the base structure
Context size	$\mathrm{U}\{64, \ldots, n - C\}$, all other nodes as queries
Optimization	
Optimizer	AdamW, weight decay 0
Learning rate	2×10^{-4}
Schedule	linear warmup for 2 epochs, cosine decay
Epochs	30, each with 1024 steps and 512 parameter updates
Batch size	8 graphs per step, 16 graphs per update (2 accumulation steps)
Gradient clipping	norm 1.0
EMA decay	0.999

Table 9: Statistics of the real-world graphs used for evaluation. Edges are unique undirected pairs, the degree standard deviation is the population standard deviation, and density is $2|E|/(|V|(|V|-1))$.

Dataset	Domain	#Nodes	#Edges	Avg. Deg.	Std. Deg.	Max. Deg.	Density (%)
Celegans	Biology	297	2,148	14.46	12.94	134	4.8867
USAir	Transportation	332	2,126	12.81	20.10	139	3.8693
PB	Web	1,222	16,714	27.36	38.40	351	2.2404
NS	Collaboration	1,589	2,742	3.45	3.47	34	0.2173
CS	Collaboration	18,333	81,894	8.93	9.11	136	0.0487
Cora	Citation	2,708	5,278	3.90	5.23	168	0.1440
Facebook	Social	22,470	170,823	15.20	26.41	709	0.0677
Power	Infrastructure	4,941	6,594	2.67	1.79	19	0.0540

graph is converted to a simple undirected graph without self-loops. Table 9 summarizes the resulting statistics.

For every graph we use ten fixed splits, indexed by seeds 123–132 and shared by all locally evaluated methods, that partition the positive links into training, validation, and test sets with proportions of 70%, 10%, and 20%. Each split also contains fixed, mutually disjoint negative sets for training, validation, and test, with a one-to-one positive-to-negative ratio. The negatives are sampled uniformly from node pairs that are not links of the full graph. The random-negative results of LPLP and of all locally evaluated baselines use exactly these splits, and the hard-negative candidates are built on the same splits.

F.2 Baselines

We compare against structural heuristics, graph neural networks for link prediction (GNN4LP), and universal link predictors. The heuristic baselines are Common Neighbors (CN), Adamic–Adar (AA), Resource Allocation (RA), and Katz. Katz is computed in closed form as $(I - \beta A)^{-1} - I$ on the training graph with $\beta = c/\lambda_{\max}(\mathbf{A})$, where $\lambda_{\max}(\mathbf{A})$ is the largest eigenvalue of the adjacency matrix. For each dataset, c is selected from $\{0.01, 0.03, 0.05, 0.1, 0.2, 0.3, 0.5, 0.8\}$ by the mean validation Hits@50 over seeds 123–125, and the same value is used for the hard-negative evaluation. The GNN4LP baselines are GCN, SEAL, Neo-GNN, ELPH, BUDDY, NCNC, and MPLP. The GCN baseline uses the `cn0` configuration of the NCNC repository, which combines a GCN encoder with a multilayer perceptron applied to the Hadamard product of the endpoint representations. The universal link predictors are UniLP and TFMLinker. UniLP and TFMLinker are pretrained on collections of

Table 10: Implementations of the baselines.

Method	Implementation
CN, AA, RA, Katz	Our implementation
GCN, NCNC	`https://github.com/GraphPKU/NeuralCommonNeighbor`
SEAL	`https://github.com/facebookresearch/SEAL_OGB`
ELPH, BUDDY	`https://github.com/melifluos/subgraph-sketching`
MPLP	`https://github.com/Barcavin/efficient-node-labelling`
Neo-GNN	`https://github.com/seongjunyun/Neo-GNNs`
UniLP	`https://github.com/Barcavin/context_LP`
TFMLinker	Not publicly available (reported results)

Table 11: Hyperparameter candidates of the GNN4LP baselines. Each model is evaluated with eight configurations on seeds 123–125, and the configuration with the highest mean validation Hits@50 is used for all ten splits.

Method	Candidates
GCN, NCNC	hidden dimension $\{64, 256\}$ $\times$ learning rates of the GNN and the predictor $\{(10^{-3}, 10^{-3}), (4.3\times10^{-3}, 2.4\times10^{-3})\}$ $\times$ predictor dropout $\{0.05, 0.3\}$
SEAL	learning rate $\{10^{-4}, 5\times10^{-4}\}$ $\times$ hidden dimension $\{32, 64\}$ $\times$ number of hops $\{1, 2\}$
Neo-GNN	hidden dimension $\{64, 256\}$ $\times$ learning rate $\{5\times10^{-4}, 10^{-3}\}$ $\times$ β $\{0.01, 0.1\}$
ELPH, BUDDY	hash hops $\{1, 2\}$ $\times$ learning rate $\{10^{-4}, 10^{-3}\}$ $\times$ label dropout $\{0.1, 0.5\}$
MPLP	a base configuration following the MPLP repository, and seven variants, each changing the learning rate to one of $\{5\times10^{-4}, 5\times10^{-3}, 2\times10^{-2}\}$, the label dropout to one of $\{0.1, 0.4, 0.8\}$, or the signature dimension to 512

real-world graphs, whereas LPLP is pretrained exclusively on graphs sampled from synthetic priors and uses no real-world graph before evaluation. Table 10 lists the implementation used for each method.

Training of GNN4LP baselines. We retrain every GNN4LP baseline on each split with its official implementation and the training recipe of its repository. Original node attributes are not used, so the feature branches of SEAL, ELPH, BUDDY, and MPLP are disabled and GCN, NCNC, and Neo-GNN receive constant node inputs.

For each model and dataset, we evaluate the eight configurations in Table 11 on seeds 123–125 and select the one with the highest mean validation Hits@50, which is then trained on all ten splits.

F.3 Evaluation Protocols

Input graph. For our local evaluations, the input graph contains only training positive edges, represented in both directions. Validation and test positive edges are not added to the input graph. Training links used as context positives remain in this graph.

Random negatives. We compute global-pool Hits@50 by comparing each positive test score with the 50th-largest score in the fixed test-negative pool. A positive contributes a hit only when its score is strictly greater than this threshold. Thus, the one-to-one ratio specifies the total positive and negative pool sizes, rather than a separate one-negative ranking task for each positive. In both result tables, Avg. is the unweighted mean over the seven benchmarks, and Rank is the mean per-benchmark rank among the ranked methods, with tied methods sharing the average of their ranks.

Hard negatives. We use fixed HeaRT (Li et al., 2023) candidates generated from RA and personalized PageRank scores on the training graph. Each test positive is ranked against 500 negative candidates, 250 from each endpoint-conditioned candidate set. We retain the generated candidate

order and duplicates. With positive score s_i^+ and negative scores s_{ij}^-, the average-tie rank is

$$r_i = 1 + \frac{1}{2}\sum_{j=1}^{500}\left(\mathbf{1}[s_{ij}^- > s_i^+] + \mathbf{1}[s_{ij}^- \geq s_i^+]\right). \tag{22}$$

We report Hits@50, the fraction of test positives with $r_i \leq 50$.

G Additional Experiments

G.1 Does LPLP Use the Context Labels?

To test whether LPLP actually uses the context labels to infer the structural pattern, we keep the graph, context pairs, and query pairs fixed and perturb only the context labels at inference. *Scrambled* randomly permutes the labels across context pairs, preserving the class balance while breaking the association between pair structure and label, whereas *Flipped* replaces every label y with $1 - y$. As shown in Table 12, scrambling the labels reduces Hits@50 from 53.2–92.7 to below 14 across all three datasets. Flipping the labels drives Hits@50 to zero and yields AUC below 5, indicating that the query ranking is nearly reversed. Because the prior inverts all labels with probability $1/2$, a flipped context corresponds to a task in which non-links form the positive class, and LPLP reverses its ranking accordingly. These results indicate that LPLP relies on the context labels rather than on the pair structure alone.

Table 12: Hits@50 when context labels are perturbed at inference, averaged over the ten splits of Table 1.

Context	USAir	PB	Facebook
Normal	92.7	53.2	74.4
Scrambled	7.2	13.7	10.4
Flipped	0.0	0.0	0.0

G.2 Inference Time

We measure the wall-clock time needed to obtain test predictions on one graph under the random-negative protocol of Table 1, using seeds 123–125 and a single NVIDIA RTX A6000 GPU per run. We use Cora, CS, and Facebook, which range from a small graph to the two largest graphs of our benchmarks in Table 9. For LPLP and UniLP, the time covers loading the pretrained model, building the context, and scoring the test pairs. For each GNN4LP baseline, it covers preprocessing such as subgraph extraction and hashing, training with the selected hyperparameters of Appendix F.2, evaluating the validation set after every epoch, and scoring the test pairs. The time is measured over the whole process, from loading the data to writing the test predictions.

Table 13 reports the results. LPLP obtains the test predictions for Facebook in 194.7 seconds without any training on the graph, and UniLP, which also encodes an enclosing subgraph for every pair, takes 156.3 seconds. On Cora, LPLP takes 5.3 seconds and is the fastest among the methods reported in Table 13. On Facebook, a single training run with the selected configuration takes 50.4 seconds for GCN, 51.8 for BUDDY, 110.8 for MPLP, and 144.2 for Neo-GNN, whereas NCNC, ELPH, and SEAL take 380.2, 1,452.7, and 11,021.5 seconds. These times exclude the hyperparameter search, which in our protocol trains 24 additional models on every graph, and every GNN4LP baseline must be trained again for each new graph, whereas LPLP needs only forward passes.

G.3 Effect of Context Size

We vary the number of labeled context links per class from 1 to 1,600 on NS, PB, and CS while keeping the splits and evaluation of Table 1. As shown in Figure 5, a single labeled link per class already recovers most of the performance, and increasing the context from 1 to 200 links per class improves Hits@50 by 3.3, 2.3, and 1.6 points on NS, PB, and CS. Beyond 200 links per class, Hits@50 changes by less than one point, so a small labeled context suffices on these graphs.

G.4 Effect of the Link-Formation Rule

Tables 14 and 15 report the ablation discussed in Section 5.5. The model without rule links shares the architecture and training recipe of LPLP and differs only in $\rho = 0$, so the whole link budget of each

Table 13: Wall-clock time in seconds to obtain test predictions on one graph under random negatives, averaged over seeds 123–125. For the GNN4LP baselines, the time includes training with the selected hyperparameters, validation after every epoch, and checkpoint selection, but not the hyperparameter search.

Method	Cora	CS	Facebook
No training on the target graph			
UniLP	5.9 ±0.1	36.2 ±0.8	156.3 ±0.3
LPLP	5.3 ±0.1	58.9 ±0.2	194.7 ±1.4
Training on the target graph			
GCN	6.3 ±0.1	25.5 ±0.2	50.4 ±0.7
SEAL	174.0 ±5.3	2,501.2 ±23.0	11,021.5 ±65.5
Neo-GNN	6.9 ±0.3	72.3 ±0.2	144.2 ±0.5
ELPH	54.3 ±0.7	656.7 ±2.5	1,452.7 ±0.6
BUDDY	7.2 ±0.1	26.8 ±0.2	51.8 ±0.0
NCNC	10.5 ±0.4	91.1 ±0.7	380.2 ±0.7
MPLP	15.9 ±0.6	94.7 ±27.9	110.8 ±1.3

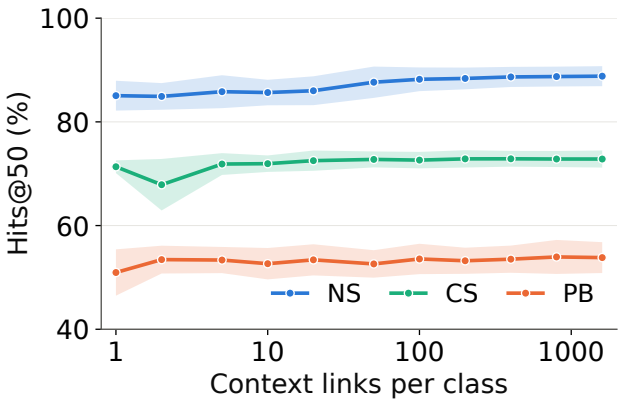


Figure 5: Hits@50 against the number of labeled context links per class, averaged over ten splits. Shaded bands show one standard deviation.

synthetic graph consists of seed links. Under random negatives, removing the rule lowers Hits@50 by 3.36 points on CS, 2.35 on Celegans, 1.70 on USAir, 1.30 on Facebook, and 0.55 on PB, whereas NS and Cora change by less than 0.2 points. Under hard negatives, the largest decreases occur on PB and Celegans, with 2.65 and 1.96 points, and NS and Cora again change by less than 0.4 points.

Under the uniform context, both models rank $d = 2$ above $d = 3$ above $d = 4$ on both graphs. Under Context A, the model without rule links separates the specified distances less clearly, and on CS its AUC for ranking $d = 4$ above $d = 3$ stays below 50, so the pairs at $d = 4$ are not moved above those at $d = 3$. On Cora, its Hits@50 under Context A falls to 22.2, compared with 39.3 for LPLP, and varies widely across splits.

G.5 Structural Preference of UniLP under Biased Contexts

We repeat the experiment of Section 5.4 with UniLP, the universal link predictor that we evaluate on the same splits. For every split, UniLP receives the same 200 positive and 200 negative context pairs as LPLP, and the graph, test queries, and metrics are unchanged.

As shown in Table 16, UniLP keeps preferring closer pairs under Context A. Its AUC for ranking $d = 4$ above $d = 3$ rises only from 2.1 to 18.2 on Cora and from 1.4 to 5.8 on CS, whereas LPLP reaches 93.5 and 73.8, and its AUC for ranking $d = 2$ above $d = 3$ stays near its value under the uniform context. Under Context B, UniLP moves toward the specified ordering, lowering the AUC for ranking $d = 2$ above $d = 3$ to 26.5 and 31.3, whereas LPLP lowers it to 0.8 and 1.3. UniLP also

Table 14: Hits@50 of LPLP pretrained with and without rule links, averaged over the ten splits of Tables 1 and 2. Bold marks the better of the two models.

Negatives	Model	Celegans	USAir	PB	NS	Cora	CS	Facebook	Avg.
Random	LPLP	**75.24** ±3.07	**92.71** ±1.70	**53.21** ±2.52	**88.38** ±2.10	65.15 ±1.77	**72.87** ±1.67	**74.40** ±1.57	**74.57**
	w/o rule	72.89 ±3.36	91.01 ±1.48	52.66 ±2.32	88.36 ±1.99	**65.32** ±2.12	69.51 ±1.61	73.10 ±1.69	73.26
HeaRT	LPLP	**55.92** ±2.22	**78.59** ±2.93	**29.17** ±0.84	81.53 ±2.45	50.13 ±1.43	**55.66** ±0.27	**62.87** ±0.28	**59.12**
	w/o rule	53.96 ±2.03	78.47 ±3.02	26.52 ±0.99	**81.81** ±1.94	**50.48** ±1.05	54.51 ±0.40	62.40 ±0.25	58.31

Table 15: Hits@50 and the AUC (%) for ranking test pairs at one distance above those at another, for LPLP pretrained with and without rule links under the uniform context and Context A of Section 5.4, averaged over ten splits.

				AUC		
Dataset	Context	Model	Hits@50	2>3	4>3	4>2
Cora	Uniform	LPLP	65.1 ±1.8	79.7 ±3.9	18.0 ±2.7	4.2 ±0.7
		w/o rule	65.3 ±2.1	84.7 ±2.8	13.8 ±2.0	2.3 ±0.5
	A	LPLP	39.3 ±3.5	98.7 ±0.4	93.5 ±3.4	8.4 ±1.5
		w/o rule	22.2 ±8.0	94.9 ±1.8	85.3 ±3.4	23.3 ±7.6
CS	Uniform	LPLP	72.9 ±1.7	90.1 ±1.2	8.3 ±0.5	0.3 ±0.1
		w/o rule	69.5 ±1.6	88.4 ±0.5	8.4 ±0.5	0.7 ±0.1
	A	LPLP	62.7 ±1.7	98.8 ±0.2	73.8 ±3.4	2.2 ±0.5
		w/o rule	62.0 ±3.1	93.3 ±0.6	45.1 ±8.2	3.8 ±1.4

ranks pairs at $d = 4$ above those at $d = 3$, with AUCs of 92.1 and 98.4, so its preference becomes a monotone preference for more distant pairs. UniLP thus partly follows the reversal specified by Context B but does not reproduce the non-monotone ordering specified by Context A.

G.6 Context Positives in the Message-Passing Graph

Context positives are observed links and therefore remain in the graph on which the message-passing branch operates, whereas held-out query positives do not. Pretraining follows the same construction. We examine the effect of this difference by removing the 200 context-positive links from the message-passing graph at inference, so that context positives, like query positives, are absent from that graph, and repeat the experiment of Section 5.4. As shown in Table 17, the uniform context gives nearly the same results with and without these links, with Hits@50 of 65.1 and 65.1 on Cora and 72.9 and 72.8 on CS. Both biased contexts are still followed, and the ordering of Context A becomes stronger, with the AUC for ranking $d = 4$ above $d = 3$ rising from 93.5 to 98.5 on Cora and from 73.8 to 83.6 on CS.

Table 16: Hits@50 and the AUC (%) for ranking test pairs at one distance above those at another for LPLP and UniLP under the uniform context and the biased contexts of Section 5.4, averaged over ten splits. Both models receive the same context pairs. Bold marks the orderings specified by each biased context.

				AUC		
Dataset	Context	Model	Hits@50	2>3	4>3	4>2
Cora	Uniform	LPLP	65.1 ±1.8	79.7 ±3.9	18.0 ±2.7	4.2 ±0.7
		UniLP	56.9 ±4.6	83.2 ±1.8	2.1 ±0.8	0.2 ±0.1
	A	LPLP	39.3 ±3.5	**98.7** ±0.4	**93.5** ±3.4	8.4 ±1.5
		UniLP	36.0 ±5.8	**87.1** ±1.4	**18.2** ±2.4	2.6 ±0.5
	B	LPLP	7.5 ±2.5	**0.8** ±0.3	32.3 ±5.9	**98.6** ±0.3
		UniLP	4.5 ±1.1	**26.5** ±3.1	92.1 ±2.2	**92.9** ±2.2
CS	Uniform	LPLP	72.9 ±1.7	90.1 ±1.2	8.3 ±0.5	0.3 ±0.1
		UniLP	66.2 ±2.5	89.5 ±0.3	1.4 ±0.1	0.1 ±0.0
	A	LPLP	62.7 ±1.7	**98.8** ±0.2	**73.8** ±3.4	2.2 ±0.5
		UniLP	42.8 ±4.3	**88.0** ±0.4	**5.8** ±2.0	0.3 ±0.1
	B	LPLP	0.0 ±0.0	**1.3** ±0.4	82.7 ±1.6	**99.9** ±0.0
		UniLP	1.1 ±0.4	**31.3** ±8.0	98.4 ±0.5	**96.2** ±1.5

Table 17: Hits@50 and the AUC (%) for ranking test pairs at one distance above those at another when the context-positive links are kept in or removed from the message-passing graph, averaged over ten splits. Bold marks the orderings specified by each biased context.

				AUC		
Dataset	Context	Context positives	Hits@50	2>3	4>3	4>2
Cora	Uniform	kept	65.1	79.7	18.0	4.2
		removed	65.1	79.8	18.3	3.9
	A	kept	39.3	**98.7**	**93.5**	8.4
		removed	34.3	**98.6**	**98.5**	15.4
	B	kept	7.5	**0.8**	32.3	**98.6**
		removed	2.5	**1.3**	54.1	**98.9**
CS	Uniform	kept	72.9	90.1	8.3	0.3
		removed	72.8	89.9	8.4	0.4
	A	kept	62.7	**98.8**	**73.8**	2.2
		removed	62.1	**98.7**	**83.6**	3.8
	B	kept	0.0	**1.3**	82.7	**99.9**
		removed	0.0	**1.5**	84.5	**99.9**