LINE GRAPH NEURAL NETWORKS FOR LINK PREDICTION

BY: LEI CAI, JUNDONG LI, JIE WANG, SHUIWANG JI

3/10/25

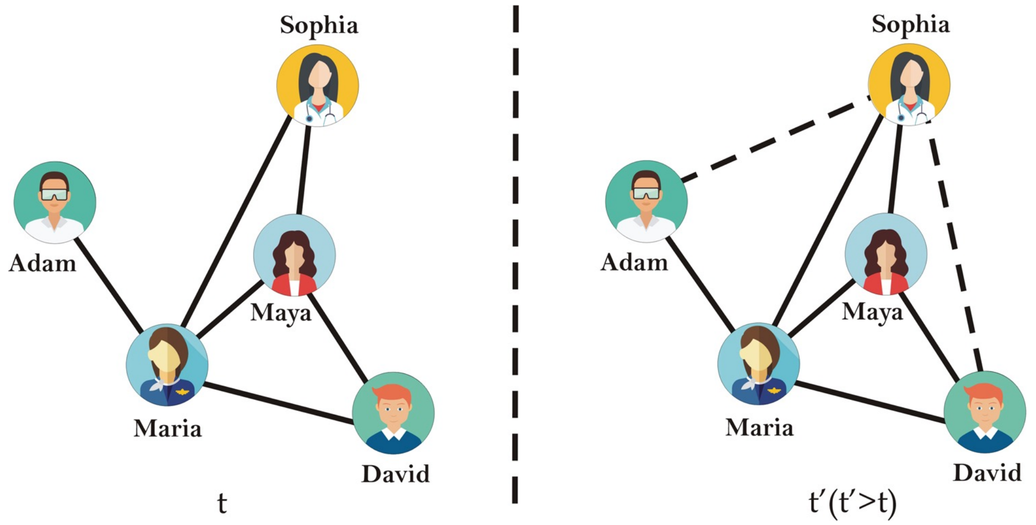
Presenter: Gurjot Singh

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

Motivation and Background

Link Prediction Task

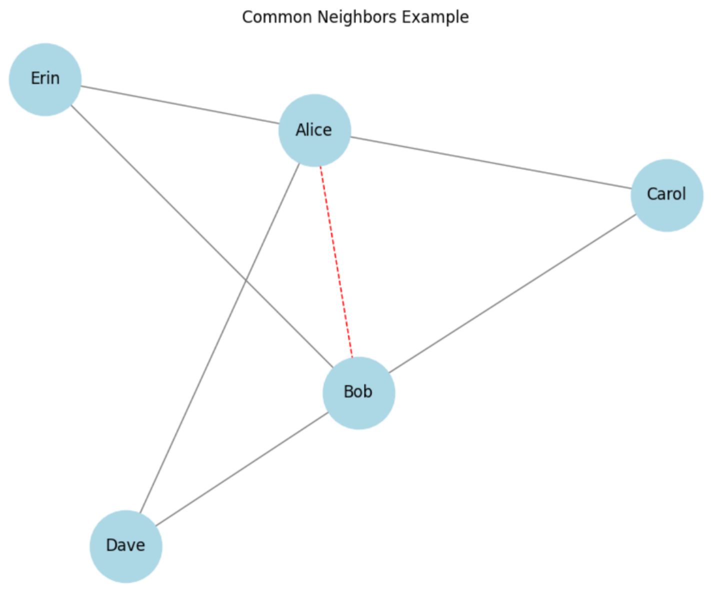
- **Goal**: Given a graph G=(V,E) with nodes V and edges E, the link prediction task aims to determine whether a link (edge) exists or may appear in the future between a pair of nodes.
- **Applications**: Social networks (friend recommendations), e-commerce (product recommendation), knowledge graphs, protein–protein interactions, etc.

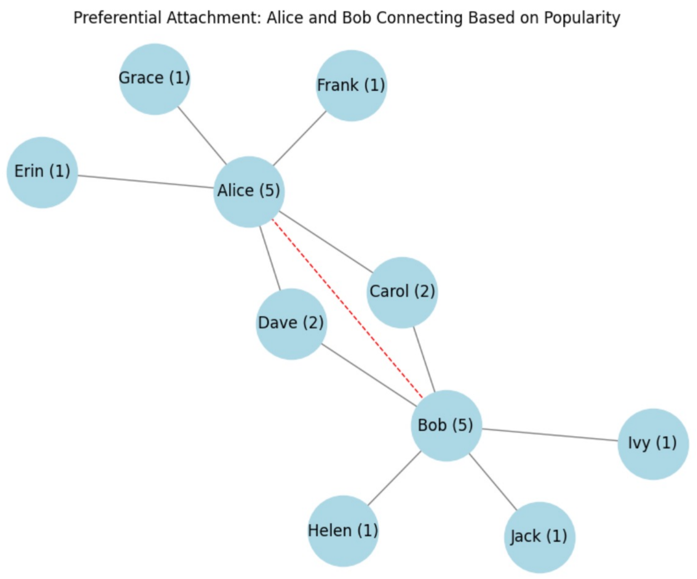


Previous Research

- **Heuristic Methods**

  Calculate similarity between two nodes using pre–defined rules based solely on the network's structure.





UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

Previous Research

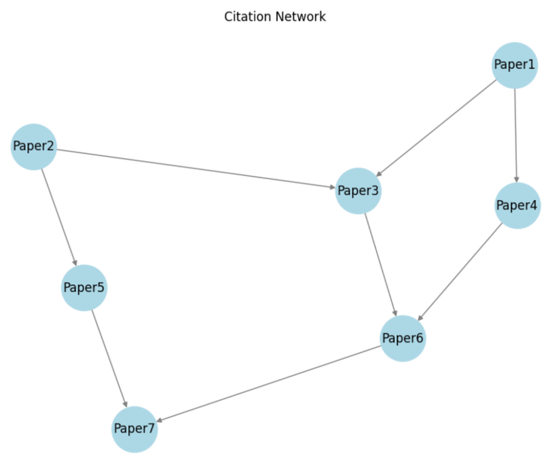
- **Issues**:

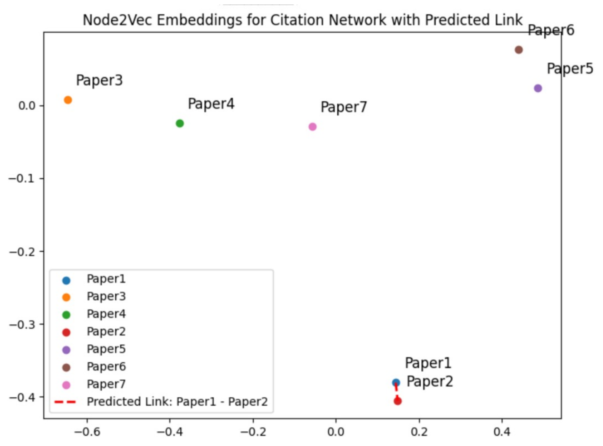
  These rules work well in some settings (like social networks) but might fail in others (e. g. , in biological networks, two proteins may share many neighbors yet not interact).

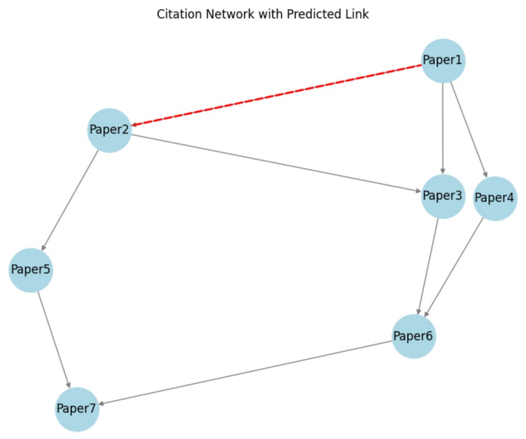
Previous Research

- **Embedding Methods**

Embedding methods, such as node2vec or DeepWalk, learn a low-dimensional vector representation for each node. The idea is that nodes with similar roles or positions in the network will have similar vectors, so you can predict links by measuring the similarity (like cosine similarity) between these vectors.







Previous Research

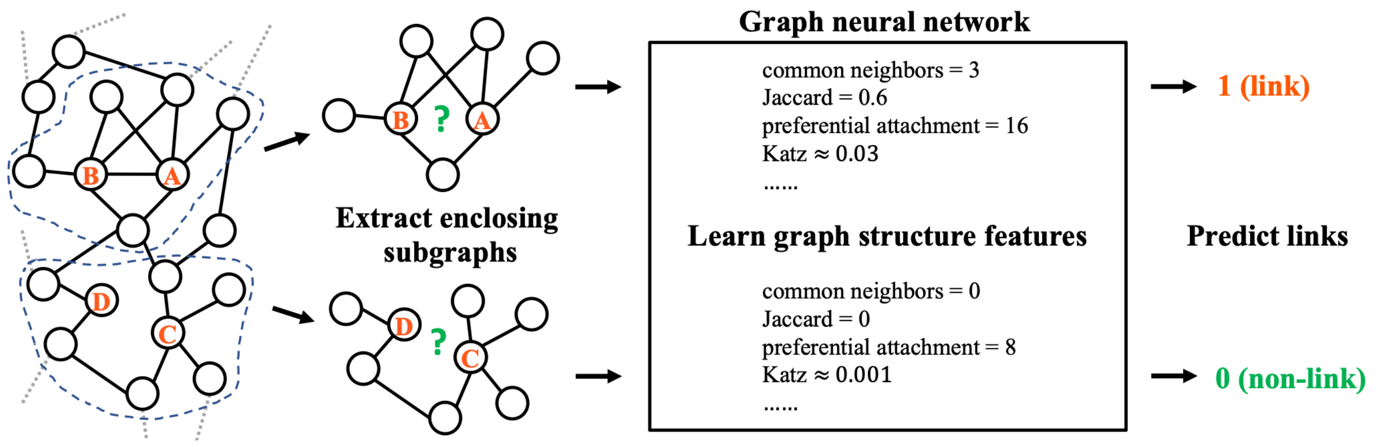
- **Issues**:

  In a sparse network (say, a niche scientific field with few papers), the random walks might not capture enough information about the structure, and the learned embeddings may not be very meaningful, leading to poor link prediction performance.

Previous Research

- **Deep learning Methods** (**SEAL**)

  Deep learning approaches, particularly those using Graph Neural Networks (GNNs), learn to predict links by extracting features directly from subgraphs around the nodes. A popular example is the SEAL framework.



Previous Research

- **Issues**:

  In methods like SEAL, after you update each node’s feature using GNN layers, you need to pool these node features into one fixed-size vector to represent the subgraph. Pooling (whether it's mean, max, or another aggregation) can lose fine-grained details about the individual nodes and their specific relationships.

Outline

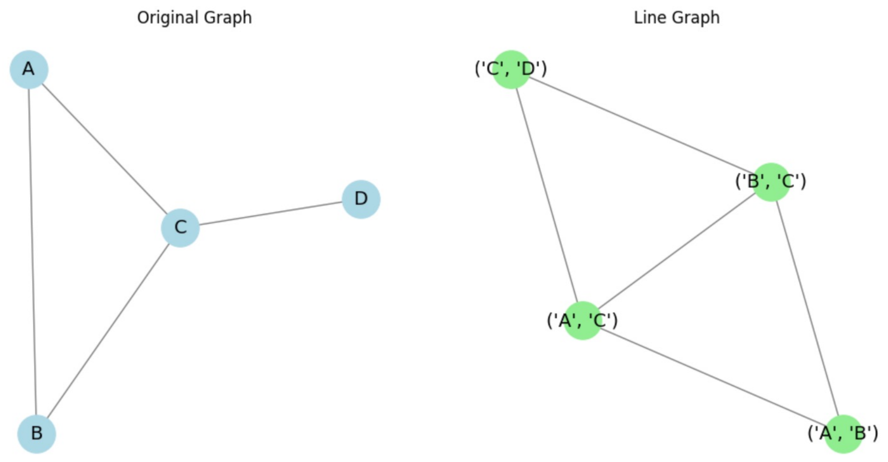
Key Contributions

1. **Reframing Link Prediction as Node Classification** (**Rather than graph classification task**)
   - In the line graph, each node represents an edge from the original graph. This reframing allows the problem to be solved as node classification, directly learning features for each link.
2. **Preservation of Detailed Local Structural Information**
   - By avoiding pooling operations, the line graph method maintains the fine-grained local structure of the original graph.
3. **Improved Efficiency and Performance**
   - The line graph neural network model is simpler and has fewer parameters compared to traditional subgraph-based methods.

What is a Line Graph ?

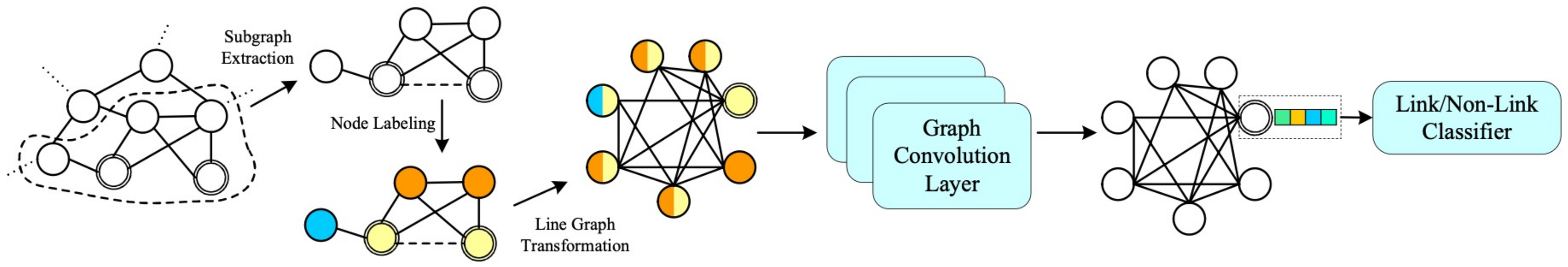
- A **line graph** is a different way to look at a network.

  - In original graph, you have nodes connected by edges.
  - In the line graph, each edge from the original graph becomes a node.
  - Two nodes in the line graph are connected if the corresponding edges in the original graph share a common node.

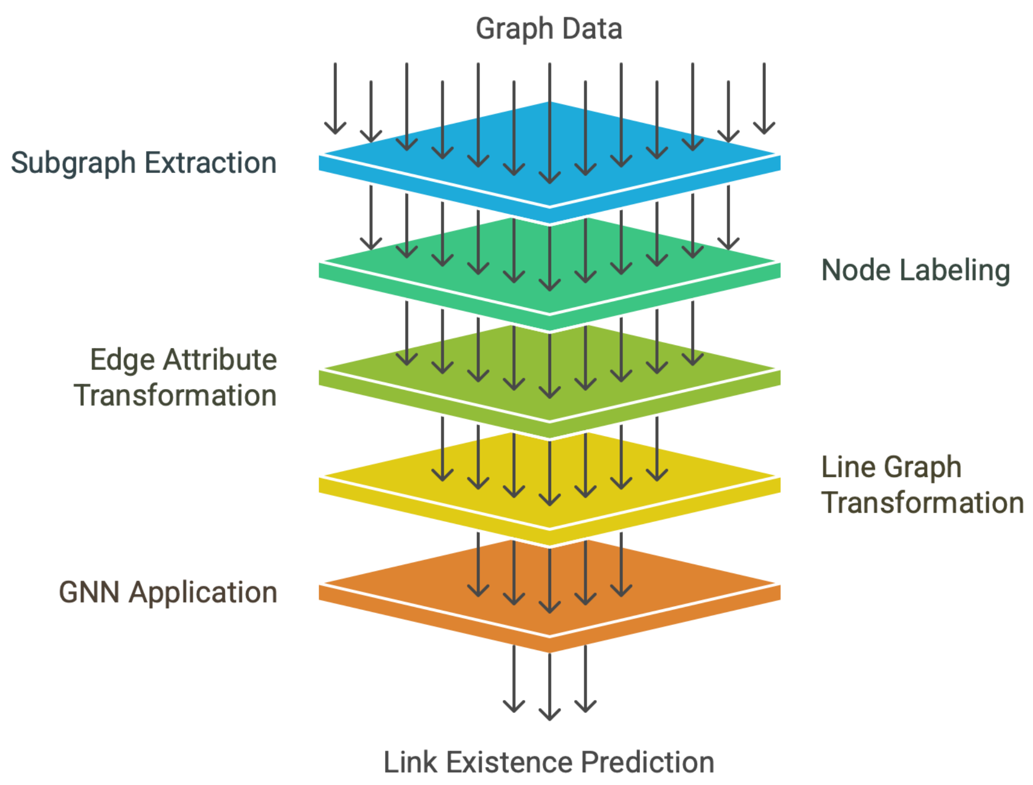


Methodology

Approach	Process Description	Feature Extraction Focus
SEAL	Extracts a labeled subgraph, processes it with GNN layers, then applies pooling to aggregate node features.	Aggregates node features into a fixed-size subgraph vector.
Line Graph	Extracts a labeled subgraph, transforms it into a line graph (edges become nodes), then applies GNN.	Directly learns edge (link) features without pooling.



Methodology


Graph Data
Subgraph Extraction
Node Labeling
Edge Attribute Transformation
Line Graph Transformation
GNN Application
Link Existence Prediction

Methodology

1. **Subgraph Extraction**

Given an undirected graph

$$G = (V, E),$$

and a target link between two nodes v_1 and v_2, we extract the h-hop enclosing subgraph $G_h(v_1, v_2)$ defined as:

$$G_h(v_1, v_2) = (V_h, E_h),$$

where

$$V_h = \{v \in V \mid \min(d(v, v_1),\, d(v, v_2)) \leq h\},$$

and E_h contains all edges among nodes in V_h. This subgraph captures the local neighborhood around the target link.

Methodology

2. **Node Labeling**

Each node v in $G_h(v_1, v_2)$ is assigned a structural label $f_l(v)$ that encodes its relative distance to v_1 and v_2. For example, a Double-Radius Node Labeling (DRNL) scheme can be:

$$f_l(v) = 1 + \min\big(d(v, v_1),\, d(v, v_2)\big) + \left\lfloor \frac{d(v, v_1) + d(v, v_2)}{2} \right\rfloor,$$

with the target nodes v_1 and v_2 assigned a distinct label (e.g., 1). This labeling preserves positional information in the subgraph.

Methodology

3. **Edge Attribute Transformation**

For each edge $e = (v_i, v_j)$ in $G_h(v_1, v_2)$, we define an edge attribute that aggregates the labels of its endpoints. A simple, order-invariant transformation is:

$$l(v_i, v_j) = \text{concatenate}\Big(\min\left(f_l(v_i), f_l(v_j)\right), \max\left(f_l(v_i), f_l(v_j)\right)\Big).$$

If node attributes X_v are available, one can extend this to:

$$l(v_i, v_j) = \text{concatenate}\Big(\min\left(f_l(v_i), f_l(v_j)\right), \max\left(f_l(v_i), f_l(v_j)\right), X_{v_i} + X_{v_j}\Big).$$

This attribute serves as the initial feature for the edge.

Methodology

4. **Line Graph Transformation**

The subgraph $G_h(v_1, v_2)$ is transformed into its line graph $L(G_h(v_1, v_2))$ as follows:

- **Nodes:** Each edge $e = (v_i, v_j) \in E_h$ becomes a node n_e in $L(G_h)$.
- **Edges:** Two nodes n_{e_1} and n_{e_2} in $L(G_h)$ are connected if and only if their corresponding edges e_1 and e_2 share a common endpoint in G_h.

The initial feature for node n_e is set to:

$$Z^{(0)}(e) = l(v_i, v_j).$$

Methodology

5. **Feature Learning Via GNN** (**on Line Graph Now**)

We apply graph convolution layers on the line graph $L(G_h)$ to learn higher-level representations for each edge. The update rule for the $(k+1)^{\text{th}}$ layer is:

$$Z^{(k+1)}(e) = \left(Z^{(k)}(e) + \beta \sum_{d \in \mathcal{N}(e)} Z^{(k)}(d) \right) W^{(k)},$$

where:

- $Z^{(k)}(e)$ is the feature vector for node n_e (edge e) at layer k,
- $\mathcal{N}(e)$ is the set of neighbors of n_e in $L(G_h)$ (i.e., edges sharing a common endpoint with e),
- β is a normalization coefficient,
- $W^{(k)}$ is the weight matrix at layer k.

After K layers, the final embedding for the edge is:

$$Z^{(K)}(e).$$

Methodology

6. **Link Prediction**

Finally, the final embedding $Z^{(K)}(e)$ is used to predict the existence of the link corresponding to edge e. This is achieved via a classifier:

$$p_e = \sigma\Big(f\big(Z^{(K)}(e)\big)\Big),$$

where:

- $f(\cdot)$ is a fully connected layer (or a multi-layer perceptron),
- σ is an activation function (e.g., the sigmoid function for binary classification).

The model is trained by minimizing the cross-entropy loss:

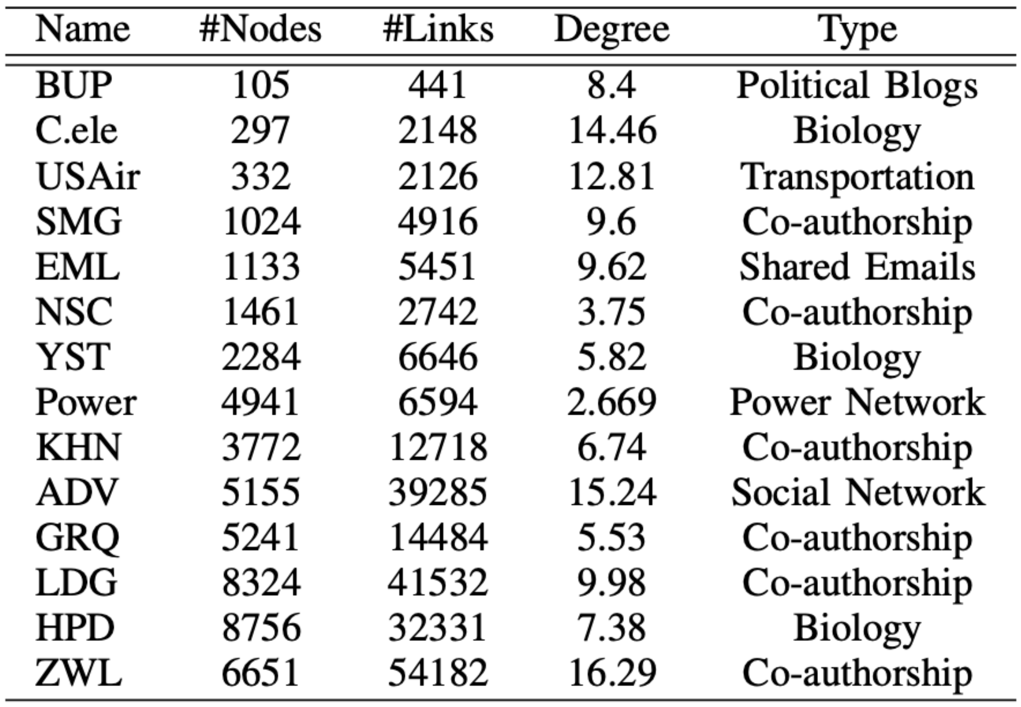
$$\mathcal{L} = -\sum_{e \in \mathcal{L}_t} \Big[y_e \log(p_e) + (1 - y_e) \log(1 - p_e)\Big],$$

where:

- $\mathcal{L}_t$ is the set of target edges (including both positive and negative examples),
- $y_e \in \{0, 1\}$ is the ground truth label for edge e.

Results and Analysis

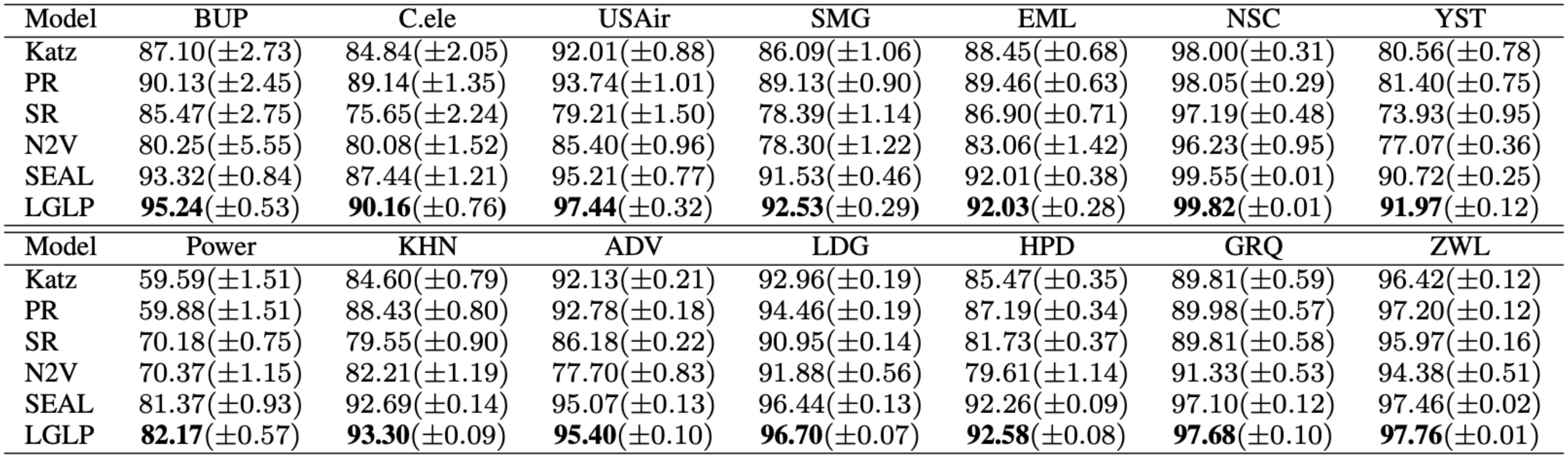
Datasets Used



Name	#Nodes	#Links	Degree	Type
BUP	105	441	8.4	Political Blogs
C.ele	297	2148	14.46	Biology
USAir	332	2126	12.81	Transportation
SMG	1024	4916	9.6	Co-authorship
EML	1133	5451	9.62	Shared Emails
NSC	1461	2742	3.75	Co-authorship
YST	2284	6646	5.82	Biology
Power	4941	6594	2.669	Power Network
KHN	3772	12718	6.74	Co-authorship
ADV	5155	39285	15.24	Social Network
GRQ	5241	14484	5.53	Co-authorship
LDG	8324	41532	9.98	Co-authorship
HPD	8756	32331	7.38	Biology
ZWL	6651	54182	16.29	Co-authorship

Results and Analysis

AUC Comparison (80% of Training Links)



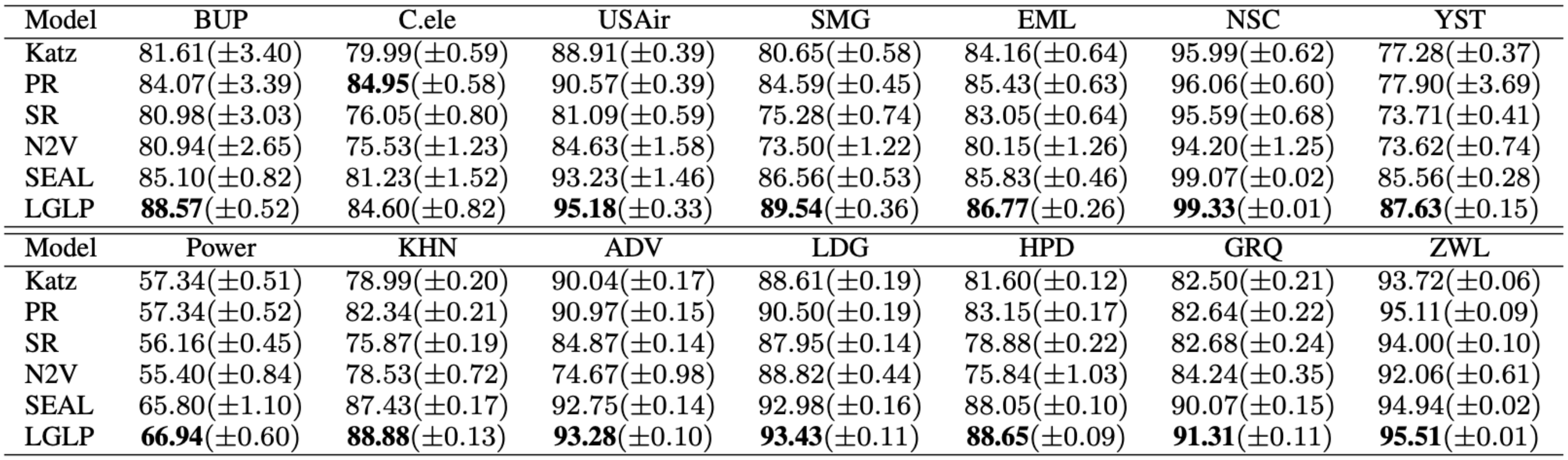
Model	BUP	C.ele	USAir	SMG	EML	NSC	YST
Katz	87.10(±2.73)	84.84(±2.05)	92.01(±0.88)	86.09(±1.06)	88.45(±0.68)	98.00(±0.31)	80.56(±0.78)
PR	90.13(±2.45)	89.14(±1.35)	93.74(±1.01)	89.13(±0.90)	89.46(±0.63)	98.05(±0.29)	81.40(±0.75)
SR	85.47(±2.75)	75.65(±2.24)	79.21(±1.50)	78.39(±1.14)	86.90(±0.71)	97.19(±0.48)	73.93(±0.95)
N2V	80.25(±5.55)	80.08(±1.52)	85.40(±0.96)	78.30(±1.22)	83.06(±1.42)	96.23(±0.95)	77.07(±0.36)
SEAL	93.32(±0.84)	87.44(±1.21)	95.21(±0.77)	91.53(±0.46)	92.01(±0.38)	99.55(±0.01)	90.72(±0.25)
LGLP	**95.24**(±0.53)	**90.16**(±0.76)	**97.44**(±0.32)	**92.53**(±0.29)	**92.03**(±0.28)	**99.82**(±0.01)	**91.97**(±0.12)

Model	Power	KHN	ADV	LDG	HPD	GRQ	ZWL
Katz	59.59(±1.51)	84.60(±0.79)	92.13(±0.21)	92.96(±0.19)	85.47(±0.35)	89.81(±0.59)	96.42(±0.12)
PR	59.88(±1.51)	88.43(±0.80)	92.78(±0.18)	94.46(±0.19)	87.19(±0.34)	89.98(±0.57)	97.20(±0.12)
SR	70.18(±0.75)	79.55(±0.90)	86.18(±0.22)	90.95(±0.14)	81.73(±0.37)	89.81(±0.58)	95.97(±0.16)
N2V	70.37(±1.15)	82.21(±1.19)	77.70(±0.83)	91.88(±0.56)	79.61(±1.14)	91.33(±0.53)	94.38(±0.51)
SEAL	81.37(±0.93)	92.69(±0.14)	95.07(±0.13)	96.44(±0.13)	92.26(±0.09)	97.10(±0.12)	97.46(±0.02)
LGLP	**82.17**(±0.57)	**93.30**(±0.09)	**95.40**(±0.10)	**96.70**(±0.07)	**92.58**(±0.08)	**97.68**(±0.10)	**97.76**(±0.01)

https://arxiv.org/pdf/2010.10046

Results and Analysis

AUC Comparison (50% of Training Links)



Model	BUP	C.ele	USAir	SMG	EML	NSC	YST
Katz	81.61(±3.40)	79.99(±0.59)	88.91(±0.39)	80.65(±0.58)	84.16(±0.64)	95.99(±0.62)	77.28(±0.37)
PR	84.07(±3.39)	**84.95**(±0.58)	90.57(±0.39)	84.59(±0.45)	85.43(±0.63)	96.06(±0.60)	77.90(±3.69)
SR	80.98(±3.03)	76.05(±0.80)	81.09(±0.59)	75.28(±0.74)	83.05(±0.64)	95.59(±0.68)	73.71(±0.41)
N2V	80.94(±2.65)	75.53(±1.23)	84.63(±1.58)	73.50(±1.22)	80.15(±1.26)	94.20(±1.25)	73.62(±0.74)
SEAL	85.10(±0.82)	81.23(±1.52)	93.23(±1.46)	86.56(±0.53)	85.83(±0.46)	99.07(±0.02)	85.56(±0.28)
LGLP	**88.57**(±0.52)	84.60(±0.82)	**95.18**(±0.33)	**89.54**(±0.36)	**86.77**(±0.26)	**99.33**(±0.01)	**87.63**(±0.15)

Model	Power	KHN	ADV	LDG	HPD	GRQ	ZWL
Katz	57.34(±0.51)	78.99(±0.20)	90.04(±0.17)	88.61(±0.19)	81.60(±0.12)	82.50(±0.21)	93.72(±0.06)
PR	57.34(±0.52)	82.34(±0.21)	90.97(±0.15)	90.50(±0.19)	83.15(±0.17)	82.64(±0.22)	95.11(±0.09)
SR	56.16(±0.45)	75.87(±0.19)	84.87(±0.14)	87.95(±0.14)	78.88(±0.22)	82.68(±0.24)	94.00(±0.10)
N2V	55.40(±0.84)	78.53(±0.72)	74.67(±0.98)	88.82(±0.44)	75.84(±1.03)	84.24(±0.35)	92.06(±0.61)
SEAL	65.80(±1.10)	87.43(±0.17)	92.75(±0.14)	92.98(±0.16)	88.05(±0.10)	90.07(±0.15)	94.94(±0.02)
LGLP	**66.94**(±0.60)	**88.88**(±0.13)	**93.28**(±0.10)	**93.43**(±0.11)	**88.65**(±0.09)	**91.31**(±0.11)	**95.51**(±0.01)

Results and Analysis

Simple (**Plain**) **Graph vs**. **Attributed Graph**:

- In the **plain graph** setting (where only the structure is used without node attributes LGLP performs better.
- In the **attributed graph** setting, additional node attributes (or latent features) are incorporated along with the structural labels. Experiments show that SEAL's performance may degrade when node attributes are directly concatenated with node labels, whereas LGLP is robust and maintains its performance.

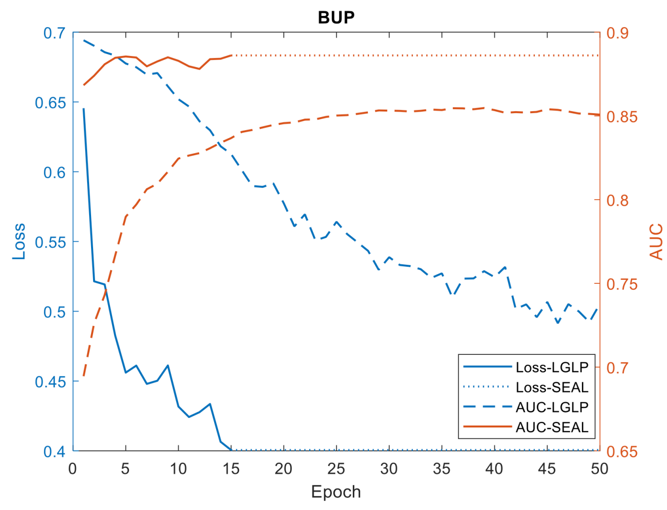
COMPARISON ON CORA DATASET USING PLAIN GRAPH AND ATTRIBUTED GRAPH (50% TRAINING LINKS).

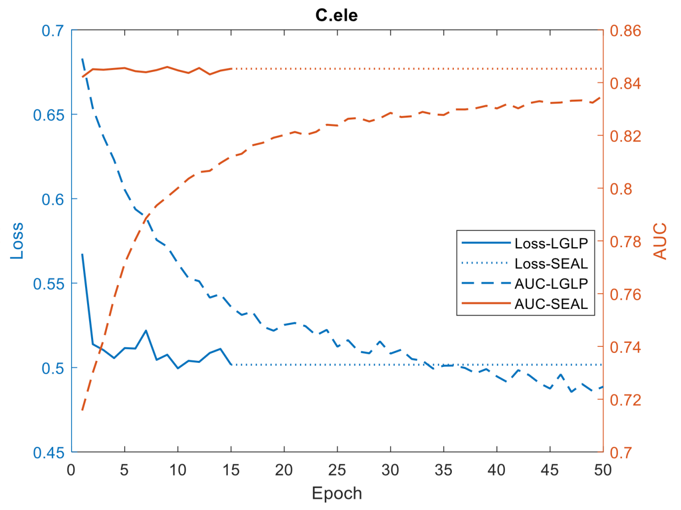
	Attribute		Plain	
	AUC	AP	AUC	AP
SEAL	75.33	77.69	79.95	82.91
LGLP	**81.45**	**81.99**	**79.96**	**83.30**

Convergence Speed Analysis

- The line graph method requires fewer parameters because it eliminates the need for pooling layers and extra 1–D convolutions typically used in SEAL.

- Experimentally, LGLP converges much faster (often in 10–15 epochs) compared to SEAL, which can take up to 50 **epochs**.





Limitation

Consider a complete bipartite graph $G = (V, E)$ with n nodes, where the nodes are equally partitioned into two sets of $\frac{n}{2}$ nodes each. In such a graph, every node in one partition is connected to every node in the other partition. Therefore, the total number of edges is given by:

$$|E| = \left(\frac{n}{2}\right) \times \left(\frac{n}{2}\right) = \frac{n^2}{4}.$$

When we transform G into its line graph $L(G)$, each edge in G becomes a node in $L(G)$. Thus, the number of nodes in the line graph is:

$$|V_{L(G)}| = \frac{n^2}{4}.$$

For example, if $n = 5000$, then the number of edges in G is:

$$|E| = \frac{5000^2}{4} = 6\,250\,000.$$

Conclusion

- **Innovative Reframing**:
  Converts link prediction from subgraph to edge (node) classification in the line graph.

- **Enhanced Feature Preservation**:
  Directly learns edge–specific features without pooling–induced information loss.

- **Efficiency Gains**:
  Simplified architecture with fewer parameters and faster convergence.

- **Promising Future Directions**:
  Explore adaptive subgraph extraction, hyperparameter tuning, and extensions to directed/dynamic graphs.

UNIVERSITY OF
WATERLOO

FACULTY OF MATHEMATICS

Thank you!