Multi-Class and Multi-Task Strategies for Neural Directed Link Prediction

deda.ai

Claudio Moroni(✉)[1,2], Claudio Borile[2],Carolina Mattsson[2], Michele Starnini[2,3], André Panisson[2]

[1]Dedagroup, Torino, Italy, claudio.moroni@dedagroup.it

[2]CENTAI Institute, Torino, Italy, {name.surname}@centai.eu

[3]Department of Engineering, Universitat Pompeu Fabra, Barcelona, 08018, Spain

Hello, we will be presenting the paper [title], which fills a gap in training neural models for directed link prediction.

Executive Summary

1) Naively porting ULP testing techniques to DLP **is not sound → need for a "three-way" testing setup.**
2) Naively porting ULP training techniques to DLP **leads to subotpimal performances → gap**
3) We fill this gap by **developing three DLP training strategies** perform better in the three-way testing setup

Before diving in, the next two slides will give a brief overview of the entire talk. In sum, we will determine that the standard testing technique for ULP models cannot be meaningfully applied to DLP settings, for which a "three-way" testing setup has already been developed. We will next observe that naively porting ULP training techniques to the directed case leads to suboptimal performances in the more principled three-way testing technique. Existing solutions are either partial or specialized models. Therefore, we'll develop three innovative training techniques that allow any GNN capable of DLP to perform well on the three-way testing setup.

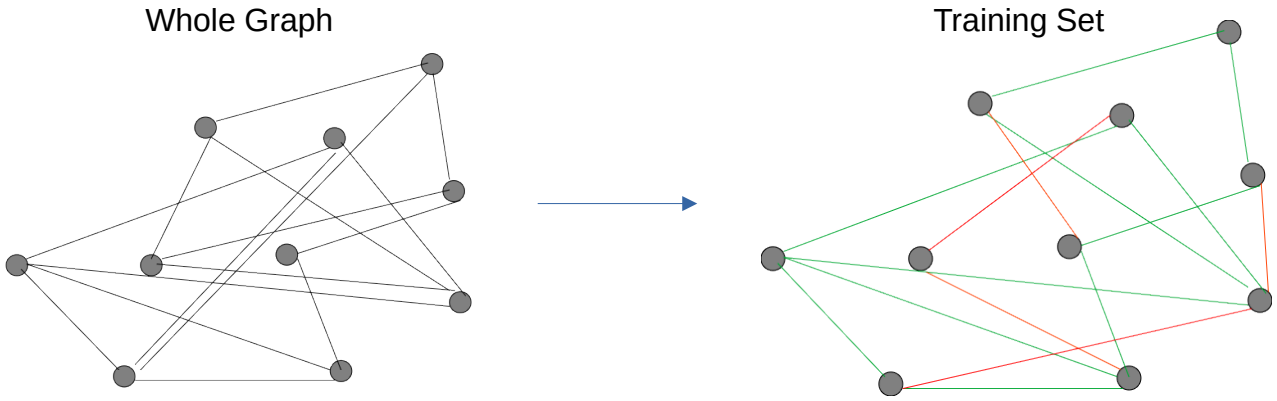
Outline

1) Undirected Link Prediction
2) Naive Directed Link Prediction
3) New Training Strategies & Results

We'll therefore follow this order: we'll first review important concepts behind ULP, then we'll see how naive training for DLP is performed and the performances we get, and finally we'll describe our new techniques.

Undirected Link Prediction

Training

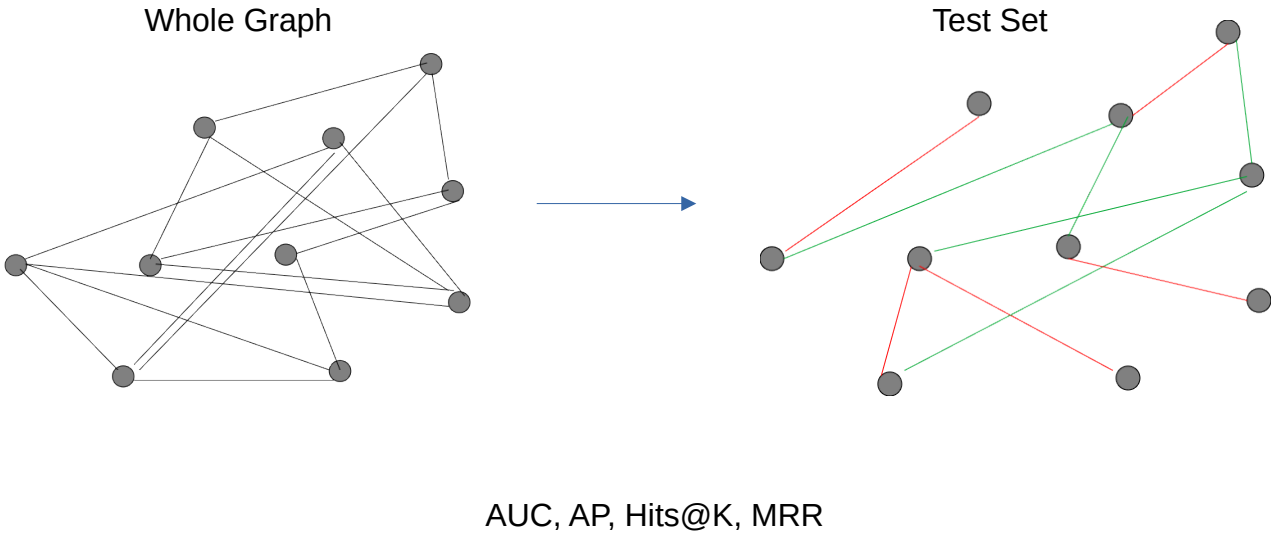


$$\mathcal{L}(\Theta) = \sum_{e \in E_{s+n}} w_p y_e \ln(\hat{p}_\Theta(e)) + (1 - y_e) \ln(1 - \hat{p}_\Theta(e))$$

When training a model for ULP, usually a graph's edges are split into a training and test sets. On the left you may see the entire undirected graph, on the right, in green, the training subset of the entire graph's edges are displayed. It is usually a random sample. If we trained our GNNs only on existing edges, they would not learn when an edge *doesn't* exist. Therefore, we also add negative edges (in red) and use a contrastive loss. Negative edges are also usually sampled randomly. Note that for performance to be balances, it is also useful to balance the contribution to the loss of positive and negative edges, here represented by the w_p weight.

Undirected Link Prediction

Testing

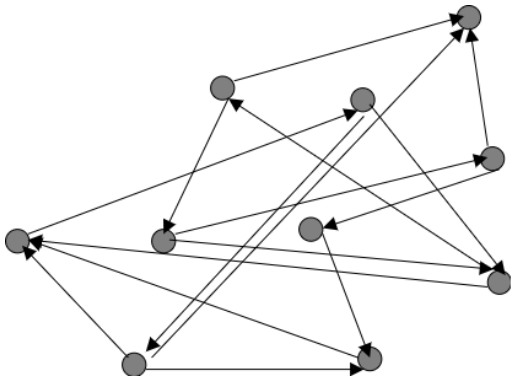


The test set of ULP models is composed similarly to the training set, in that it is a random sample of positive edges together with a random sample of the negatives, and test metrics are usually AUC, AP or the more ad-hoc hits@k and MRR.

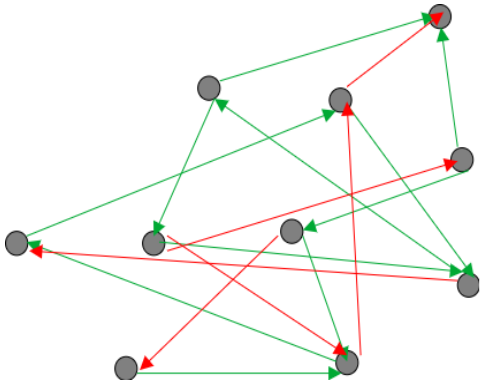
Directed Link Prediction

Naive Training

Whole Graph



Training Set



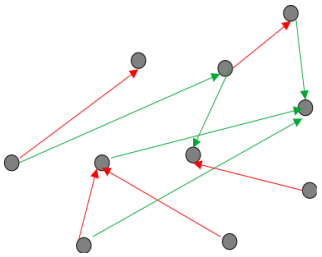
$$\mathcal{L}(\Theta) = \sum_{e \in E_{s+n}} w_p y_e \ln(\hat{p}_\Theta(e)) + (1 - y_e) \ln(1 - \hat{p}_\Theta(e))$$

Let us now switch to DLP. A naive approach to training a DLP model would consist in porting the ULP strategy as is. Therefore a random sample of positive and negative directed edges is used to train a GNN autoencoder with a contrastive and rebalanced loss.

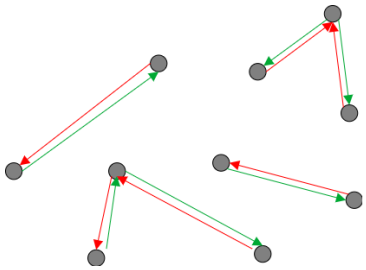
Directed Link Prediction

Testing

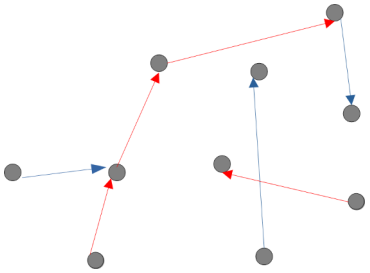
General Test Set



Directional Test Set



Bidirectional Test Set



AUC, AP, Hits@K, MRR

For testing anyway, we cannot just port the ULP testing setup. Infact, a random sample of positive and negative edges would not test the model ability to actually distinguish the edges' directions. Why is that? Consider the leftmost test set, named general test set. It consists in the direct porting of the ULP testing setup. Because positive and negative edges are sample randomly, they are uncorrelated and it is very rare that a positive edge is sampled together with its reversed negative counterpart. Therefore, models that are not or insufficiently able to distinguish an edge direction may still do well on this test set. Thus, two more test sets have been proposed in the DLP literature. One, named "directional" is a sample of unidirectional edges (meaning edges that exists in only one direction) and has their reverses as negatives. The purpose of this test set is to test the model's ability to distinguish edges' directions. The other, named "bidirectional", is a sample of on direction of true bidirectionals (meaning edges that exist in both directions) together with a sample of true unidirectionals which are treated as negatives. The purpose of this test set is to measure the degree to which a model can differentiate between an edges that exist in both directions from edges that exist in only one direction. The same metrics as ULP are used for each individual test set. Only by considering the performances of the model on all three test sets, one may judge its overall DLP performance.

Directed Link Prediction

Results with Naive Training

model	decoder	GENERAL		DIRECTIONAL		BIDIRECTIONAL	
		ROC-AUC	AUPRC	ROC-AUC	AUPRC	ROC-AUC	AUPRC
GAE	$\vec{z}_i \cdot \vec{z}_j$	84.6	88.6	50.0	50.0	62.4	64.0
GR-GAE	$\sigma(\vec{z}_v[0] - \lambda \ln(\|\|\vec{z}_u[1:] - \vec{z}_v[1:]\|\|_2^2))$	89.2	92.4	63.4	61.5	69.1	66.5
ST-GAE	$\sigma(\vec{z}_v[:\frac{L}{2}] \cdot \vec{z}_u[\frac{L}{2}:])$	87.8	90.1	60.8	64.5	74.6	74.1
DiGAE	$\sigma(\vec{z}_v^S \cdot \vec{z}_v^T)$	80.4	85.3	57.5	63.0	70.4	68.6
MLP-GAE	$\text{MLP}(\overrightarrow{z}_v \|\| \vec{z}_u)$	77.1	78.2	90.7	90.7	69.9	69.7
MAGNET	MLP-like	75.2	77.8	90.4	89.8	71.9	70.4
dMPLP	MLP-like	86.1	88.0	75.7	76.8	81.1	82.2

Anyway, the naive training strategy we discussed before does not lead, in general, to models that perform well across the three test sets we saw earlier. The table in the slides shows the name of the model, the functional form of the decoder, and its AUC and AUPRC performance across the three test set. All models are trained “naively”, or as they are trained for ULP. Let’s start from GAE. This model is structurally unable to distinguish the edge’s directions, let alone their bidirectionality, given its decoder is symmetric. Therefore, if we limited ourselves to test it on the generla test set, it would perform deceptively well. The other thre models all show the same pattern. They are DLP-capable, meaning that their decoder, in principle, allws them to distinguish the edges’ directions, they perform well on the general but less well, sometimes undesirably, on the other two tasks. This is also reflected in their original papers. We’ll later argue why. MLP-GAE and MAGNET on the other hand perform strongly on the Directional task, less well on the other two. dMPLP shows instead acceptable performance across General and Bidirectional tasks, but does less well on the Directional test set. The question is: how do we get these models to perform well on all three simultaneously? It is important since this is necessary in order to deem a model “good” at DLP.

Directed Link Prediction

Existing Solutions

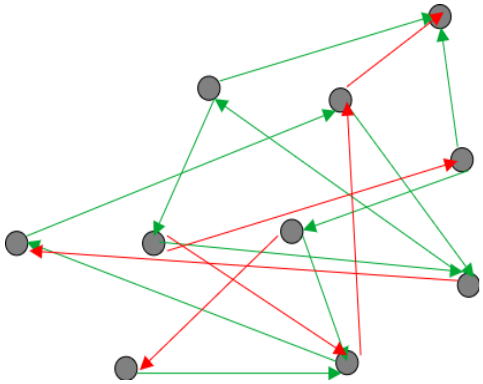
- Naive Training Kollias et al. 2022
- Train one model per task Salha et al. 2019
- Ad-hoc architectures Zhang et al. 2021

We wish to develop techniques that allow any NDLP-capable architecture to achieve good performance on all three tasks simultaneously

Let us first review how this issue was dealt with in previous literature. We identified three ways, for each of which we reported the initial or most prominent paper.
Some authors test only on the general test set and train naively. This would be a non-solution (unless the application only demands performance on the general task).
Other authors train one model per task. This clearly increases all three performances but does not yield one SINGLE model that is overall good al DLP.
Finally, ad-hoc model have been developed, that address these issues in heuristic ways, such as MAGNET. While they perform very well, we would like to develop training strategies that allow *ANY* DLP-capable model to obtain good performances across across the three tasks simultaneously.

New Training Strategies

Multi Class



We need to **balance the loss** w.r.t. **positives vs negatives** AND **unidirectionals vs bidirectionals** SIMULTANEOUSLY!

$$[\hat{p}^{nb}_{uv}, \hat{p}^{nu}_{uv}, \hat{p}^{pu}_{uv}, \hat{p}^{pb}_{uv}] = [(1-\hat{p}_{uv})(1-\hat{p}_{vu}), (1-\hat{p}_{uv})\hat{p}_{vu}, \hat{p}_{uv}(1-\hat{p}_{vu}), \hat{p}_{uv}\hat{p}_{vu}]$$

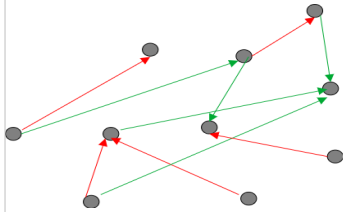
$$\mathcal{L}_{MC\text{-}NDLP}(\Theta) = -\sum_{c \in C}\sum_{uv \in T} w_{y_{uv}} \mathbb{I}(y_{uv} = c)\log(\hat{p}^{y_{uv}}_{uv}), \quad w_{y_{uv}} = \frac{n_x}{n_{y_{uv}}}$$

The first training strategy we developed stems from understanding *why* naive training fails. It fails because, like for the general test set, in a randomly sampled training set positives and negatives are uncorrelated, therefore the model very rarely learns that an edge exist in one direction but NOT in the other. It is only trained on one direction, either the existing or the non existing one. We propose to should therefore “give more weight” to those rare occasions where we get both directions. Simultaneously, we wish to weigh the loss to balance positives vs negatives. This simultaneous requirements led us to define a 4 class classification task, where an edge might be pos unidir, neg unidir, pos bidir, neg bidir, and then re-weight all classes in the loss so that they give equal contribution. To further imprve performance, we used all possible negative edges (not shown in the picture for clarity). Therefore, given a GNN autoencoder that predicts p_uv as the probability that a directed edge from node u to node v exists, we can turn it into a 4-class classifier as describes in the slide. So the probability that it is negative bidir is..etc etc. We then use standard cross entropy to train this multi-class classifier, re-weighting the loss so that all four classes give the same contribution.

New Training Strategies

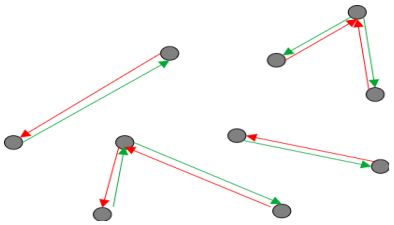
Multi Task - MGDA

General Training Set



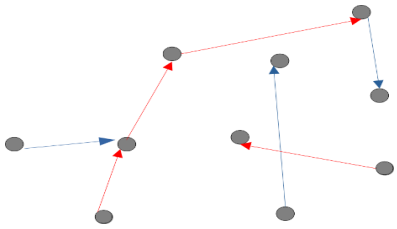
$\nabla \mathcal{L}_G$

Directional Training Set



$\nabla \mathcal{L}_D$

Bidirectional Training Set



$\nabla \mathcal{L}_B$

$$\vec{g}_{\text{MGDA}} = \min_{||\sum_i \alpha_i \nabla \mathcal{L}_i||_2^2} \sum_i \alpha_i \nabla \mathcal{L}_i \quad \text{s.t} \quad \sum_i \alpha_i = 1$$

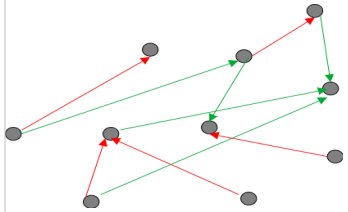
$$\theta' = \theta - \eta \vec{g}_{\text{MGDA}}$$

Another technique we developed approached the matter backwards. Since we want the model to perform well on three different tasks, it might be a good idea to simultaneously train the model on all three tasks. Therefore we split the training set in three, reflecting the compositions of the three test sets, and during training, at each epoch, we get a gradient from each of these training sets. Summing them is not mathematically guaranteed to result in a gradient that optimizes all three losses simultaneously, therefore we employed MGDA, a mathematical technique that allow to compute a gradient that will optimize all three losses simultaneously just by knowing the individual gradients computed from each loss. It consists in computing the shortest convex linear combination of the input gradients. The (negated) result is the update we are looking for.

New Training Strategies

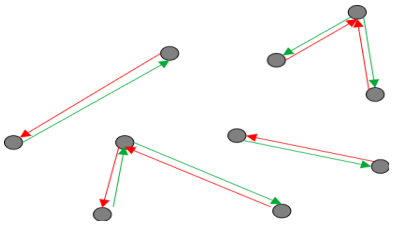
Multi Task - Scalarization

General Training Set



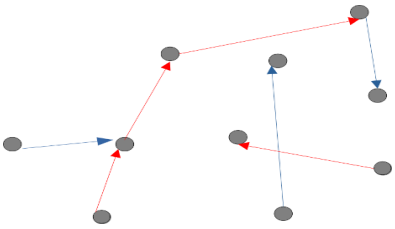
$\nabla \mathcal{L}_G$

Directional Training Set



$\nabla \mathcal{L}_D$

Bidirectional Training Set



$\nabla \mathcal{L}_B$

$$\vec{g}_{\mathrm{S}} = \sum_i \alpha_i \nabla \mathcal{L}_i$$

$$\theta' = \theta - \eta \vec{g}_{\mathrm{S}}$$

The last technique is very similar to MGDA, only that we heuristically sum the gradients. It has been shown to perform very well even if it lacks formal convergence guarantees. We just weighted the individual gradients using the test metrics as computed on a validation set we kept aside.

New Training Strategies

Results - Cora

		GENERAL		DIRECTIONAL		BIDIRECTIONAL	
model	strategy	ROC-AUC	AUPRC	ROC-AUC	AUPRC	ROC-AUC	AUPRC
GAE	BASELINE	84.6 ± 0.4	88.6 ± 0.3	50.0 ± 0.0	50.0 ± 0.0	62.4 ± 3.0	64.0 ± 3.1
GR-GAE	BASELINE	**89.2 ± 0.4**	**92.4 ± 0.2**	63.4 ± 2.5	61.5 ± 2.7	69.1 ± 3.1	66.5 ± 3.3
	MO-NDLP	84.5 ± 1.1	86.3 ± 1.1	80.6 ± 0.7	80.2 ± 0.9	79.6 ± 4.3	84.6 ± 3.5
	MC-NDLP	88.6 ± 0.4	90.0 ± 0.4	82.1 ± 0.5	**81.8 ± 0.7**	77.3 ± 2.2	76.3 ± 1.7
	S-NDLP	87.8 ± 0.6	89.5 ± 0.5	**82.3 ± 0.5**	81.6 ± 0.4	**89.6 ± 1.6**	**92.4 ± 1.1**
ST-GAE	BASELINE	**87.8 ± 0.7**	**90.1 ± 0.5**	60.8 ± 0.5	64.5 ± 0.6	74.6 ± 1.8	74.1 ± 2.2
	MO-NDLP	86.3 ± 0.5	86.2 ± 0.4	**79.3 ± 1.0**	80.0 ± 0.9	79.3 ± 0.5	79.5 ± 1.9
	MC-NDLP	80.7 ± 2.0	80.1 ± 2.1	79.0 ± 2.3	**81.6 ± 1.9**	70.3 ± 3.0	68.1 ± 2.1
	S-NDLP	84.5 ± 0.4	84.9 ± 0.7	75.8 ± 1.0	78.4 ± 0.9	**81.1 ± 0.9**	**80.4 ± 1.6**
DiGAE	BASELINE	**80.4 ± 1.1**	**85.3 ± 0.8**	57.5 ± 1.3	63.0 ± 1.4	70.4 ± 2.2	68.6 ± 1.2
	MO-NDLP	70.2 ± 3.8	72.6 ± 3.6	73.6 ± 5.4	76.0 ± 4.2	67.3 ± 4.6	69.6 ± 4.1
	MC-NDLP	75.4 ± 0.9	77.4 ± 1.0	**84.3 ± 0.6**	**85.4 ± 0.8**	68.9 ± 1.5	69.3 ± 1.1
	S-NDLP	72.5 ± 4.0	77.4 ± 4.4	61.6 ± 1.3	69.2 ± 1.4	**72.1 ± 5.6**	**74.4 ± 5.7**
MLP-GAE	BASELINE	**77.1 ± 0.9**	**78.2 ± 0.6**	90.7 ± 0.6	90.7 ± 0.6	69.9 ± 3.2	69.7 ± 3.7
	MO-NDLP	76.0 ± 0.8	76.4 ± 0.7	93.4 ± 0.6	93.5 ± 0.6	**80.7 ± 1.6**	**79.2 ± 2.4**
	MC-NDLP	74.5 ± 0.7	75.6 ± 0.7	**94.3 ± 0.6**	**94.4 ± 0.5**	71.7 ± 2.4	65.7 ± 1.8
	S-NDLP	74.7 ± 1.0	74.9 ± 0.9	90.5 ± 0.7	90.0 ± 0.9	72.0 ± 2.6	70.5 ± 2.9
MAGNET	BASELINE	**75.2 ± 1.4**	**77.8 ± 1.0**	90.4 ± 0.9	89.8 ± 0.8	**71.9 ± 2.3**	**70.4 ± 2.8**
	MO-NDLP	74.4 ± 1.4	77.4 ± 1.1	91.3 ± 1.0	90.9 ± 1.0	70.6 ± 2.7	68.6 ± 2.7
	MC-NDLP	74.4 ± 1.0	77.4 ± 1.0	**92.1 ± 0.7**	**91.6 ± 0.7**	71.8 ± 2.6	70.0 ± 2.6
	S-NDLP	74.6 ± 1.3	77.5 ± 1.1	91.0 ± 1.0	90.4 ± 1.0	71.8 ± 2.8	70.2 ± 2.9
dMPLP	BASELINE	**86.1 ± 0.5**	**88.0 ± 0.9**	75.7 ± 2.2	76.8 ± 1.6	81.1 ± 3.6	82.2 ± 5.3
	MO-NDLP	83.5 ± 0.6	85.1 ± 0.6	**89.1 ± 1.7**	**89.0 ± 2.1**	**85.8 ± 3.3**	**89.3 ± 2.5**
	MC-NDLP	81.4 ± 1.7	82.0 ± 1.5	83.7 ± 4.2	83.7 ± 3.6	70.0 ± 4.3	71.5 ± 3.9
	S-NDLP	85.6 ± 1.0	86.9 ± 1.0	84.8 ± 2.7	86.3 ± 2.3	83.6 ± 4.7	87.0 ± 4.3

The results show that our techniques produce more even performances across all three subtasks for nearly al models, except MAGNET that being already ad-hoc for directed graphs does not benefit as much. The first three models show dramatic improvements on Directional and Bidirection with little cost for the General, while MLP GAE and dMPLP respectively improve on the bidirectional and Directional tasks, again with little cost for the General task. These “costs” in the General tasks, albeit small compared to the gains on the others, show that the three tasks are actually in slight trade-offs w.r.t. one another, at least with the model we tried. In sum, our strategies allow for obtaining even and good performances simultaneously on all the three subtasks DLP is composed of with ANY DLP-capable GNN autoencoder.

Thanks!

deda.ai

That was it! I’ll be happy to take any question!