Active Learning using Node Embeddings in Partially Observed Networks.

Richeek Das[1], Saumya Goyal[1], Adhikari Tirthankar Taraknath[2]

1 Department of Computer Science, IIT Bombay
2 Department of Electrical Engineering, IIT Bombay

Group Members

Name	Roll Number	Contact
Richeek Das	**190260036**	**richeek@cse.iitb.ac.in**
Saumya Goyal	**180050092**	**saumyagoyal@cse.iitb.ac.in**
Adhikari Tirthankar Taraknath	**190070003**	**190070003@iitb.ac.in**

**** work done as a part of CS768 - Learning with Graphs, IIT Bombay***
**** equal contribution***

Abstract

Incomplete data is very common in network science. It is well recognized that, if missing data is simply ignored, network analysis results will be severely skewed. A lot of study has been done in inferring cleaned output network from a noisy, incomplete input graph. There have been several studies on the effects of non-random missing data on common network measures such as centrality, homophily, topology and centralization, including some interesting work in Compressed Sensing for reconstructing information flow networks with only access to end-to-end information in social networks. However, very little attention has been put on investigating network embedding and developing robust algorithms for Active Learning for large-scale incomplete networks. Annotating data for such networks is often costly, i.e. requires human intelligence or some expensive experiment. This calls for us to maximise our annotation utility with Active Learning. In this work we investigate large-scale Partially Observed Networks (PON) and discuss how we can combine ideas from SINE: Scalable Incomplete Network Embedding [3], ALPINE: Active Link Prediction Using Network Embedding [2] and CNE: Conditional Network Embeddings [1], to design a robust algorithm for Active Learning in Partially Observed Networks.

Introduction

Here we start off by formally defining Active Learning, the Link Prediction problem, Partially Observed Networks and the problem we want to solve:

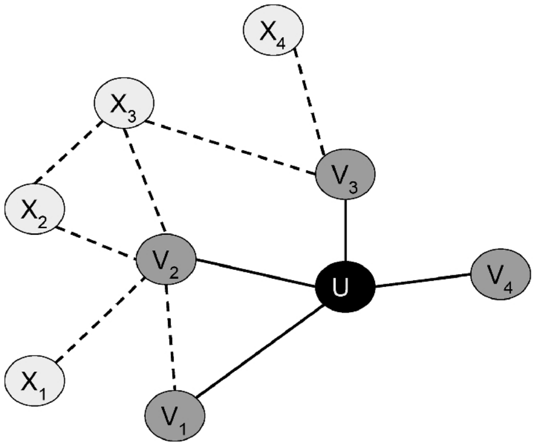

Figure 1. Example of a Undirected Partially Observed Network. Solid edges represent linked edges. Dashed edges represent unobserved edges. No connections represent unlinked edges.

Active Learning and Design

Active learning is a sub-field of machine learning, which aims to exploit the situation where learning algorithms are allowed to actively choose (part of) the training data from which they learn, in order to perform better. In many of these cases there's a budget available to query an *oracle* (say a human or expensive experiment). These queries are typically of high cost. Thus, it is of interest to identify those node pairs for which link status is unobserved but knowing which would add the most value to our training and inference task.

This work mainly discusses *pool-based* active learning, where a pool of unlabeled data points and is provided and a subset of points to be labeled is selected from this pool to be labelled by an *oracle*. Therefore the link status of only a selected few nodes can be queried to decide which labels need to be annotated for maximum benefit of the final inference task (here, our task is Link Prediction).

Embedding and Link Prediction Task

In this work we discuss Active Learning to maximise the accuracy of Link Prediction task with the budget amount of queries we can do to an oracle. One way to do this is to build a robust network embedding which can capture the information about the nodes in some latent space which can be easily translated to the various inference tasks. Formally, the goal of a network embedding model is to find a mapping $f : \mathrm{V} \to \mathbb{R}^d$, where V refers to the nodes. The embedding of a network is denoted as $\mathbf{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n}]^T \in \mathbb{R}^{n \times d}$.

The link prediction task, uses the network embeddings and maps to a probability distribution. A function $g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ evaluated on $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ to compute the probability of node i and j being linked. The target in this case is to learn a good function g which can take up the job as a classifier.

Partially Observed Networks

Definition 1. An Undirected Partially Observed Network (PON) is a tuple $\mathcal{G} = (V, E, D)$ where V is a set of $n = |V|$ nodes and $E \subseteq \binom{V}{2}$ and $D \subseteq \binom{V}{2}$ the sets of node pairs with observed link and observed unlinked status, respectively, where $E \cap D = \Phi$. Thus, $K \triangleq E \cap D$ represents the observed part, and $\mathrm{U} \triangleq \binom{V}{2} \backslash K$ is the set of node pairs for which the link status is unobserved. Figure 1.

The Problem we wish to attack

Problem. Given a partially observed network $\mathcal{G} = (V, E, D)$, a network embedding model, a budget k, a query-pool $P \subseteq U$, and a target set $T \subseteq U$ containing all node pairs for which the link statuses are of primary interest, how can we select k node pairs from the pool P such that, after querying the link status of these node pairs, adding them to the respective set E or D depending on the status, and retraining the model, the link predictions made by the network embedding model for the target set T are as accurate as possible?

Materials and Methods

In this section we first introduce the key ideas from CNE [1], ALPINE [2], and SINE [3]. Then we proceed to suggest improvements and additional possibilities for the querying strategies used by the ALPINE framework. We finally propose an algorithm to combine the SINE framework with certain key ideas of ALPINE to build the final Active Learning framework.

The CNE model.

The Conditional Network Embedding Model aims to find an embedding $\mathbf{X}$ which is maximally informative about the given network G. They formalise it as a Maximum Likelihood (ML) estimation problem:

$$\arg\max_{\mathbf{X}} P(G|\mathbf{X})$$

They do not propose the likelihood function $P(G|\mathbf{X})$ directly, but instead use Bayes Rule modelling the prior distribution of the network as $P(G)$ and the density function conditional on network as $P(\mathbf{X}|G)$. Thus, they allow prior knowledge of the network into the formulation.

They model $P(G)$ based on prior known constraints like knowledge of overall network density, knowledge of the individual node degrees, etc. Such constraints do not determine $P(G)$ fully, so we model $P(G)$ as the maximum entropy distribution satisfying these constraints. They model the resulting distribution as a product of independent Bernoulli distributions, one for each element of its adjacency matrix.

$$P(G) = \prod_{\{i,j\} \in \binom{V}{2}} P_{ij}^{\hat{a}_{ij}} (1 - P_{ij})^{1-\hat{a}_{ij}}$$

For the distribution of the data conditioned on the network, we are mostly interested in the distance of the pair of nodes in the embedding. The density should also reflect the fact that connected node pairs tend to be embedded to nearby points, while disconnected node pairs tend to be embedded to more distant points. Define $d_{ij} \triangleq \|\boldsymbol{x_i} - \boldsymbol{x_j}\|_2$. The CNE model assumes d_{ij} when conditioned on the knowledge of $\hat{a}_{ij}$, that is we know if it is a edge or a non-edge, is conditionally independent of the rest of the adjacency matrix. They model the data probability given the network as half-normals:

$$p(d_{ij}|\{i,j\} \in E) = \mathcal{N}_+(d_{ij}|\sigma_1^2)$$

where $\mathcal{N}_+(d_{ij}|\sigma_1^2) = \frac{\sqrt{2}}{\sigma_1\sqrt{\pi}} \exp\left(-\frac{x^2}{2\sigma_1^2}\right)$, $x > 0$. They model $P(\mathbf{X}|G)$ as an improper density function, a product of marginal densities for all pairwise distances:

$$p(\mathbf{X}|G) = \prod_{\{i,j\}\in E} \mathcal{N}_+(d_{ij}|\sigma_1^2) \cdot \prod_{\{k,l\}\notin E} \mathcal{N}_+(d_{kl}|\sigma_2^2)$$

The improper marginal density is calculated as:

$$p(\mathbf{X}) = \sum_G p(\mathbf{X}|G)P(G)$$

Thus, with these three quantities they have access to $P(G|\mathbf{X}) = \frac{p(\mathbf{X}|G)\cdot P(G)}{p(\mathbf{X})}$. Maximising this is a non-convex optimization problem. But $P(G|\mathbf{X})$ being a well differentiable quantity, we can solve it using a block stochastic descent approach.

Thus, with this CNE model we find an embedding which is maximally informative about the given network.

The ALPINE framework.

ALPINE is the first work which introduces an algorithm for pool-based active link prediction using network embeddings. They utilise the well interpretable CNE model and modify it to accommodate Partially Observed Networks. Most general network embedding models do not distinguish between the non-linked and unobserved status of the edges. ALPINE proposes a modified CNE. They distinguish unlinked from the unobserved by maximising probability only for the observed part of the node pairs:

$$P(\mathcal{G}|\mathbf{X}) = \prod_{\{i,j\}\in E} P(a_{ij} = 1|\mathbf{X}) \cdot \prod_{\{k,l\}\in D} P(a_{kl} = 0|\mathbf{X})$$

where, $\mathcal{G} = (V, E, D)$ is a PON.

$$\mathbf{X}^* = \arg\max_{\mathbf{X}} P(\mathcal{G}|\mathbf{X})$$

This is the node embedding ALPINE uses in its framework. They use this node embedding in an active learning query strategy which evaluates the informativeness of the unlabelled node pairs. They call this query strategy as *utility function*, defined as $u_{\mathbf{A},\mathbf{X}^*} : V \times V \to \mathbb{R}$. The output of this utility function is ranked and the top ones are selected to be labelled by some oracle. Therefore, query strategy will select the next query for an appropriate $u_{\mathbf{A},\mathbf{X}}$ as:

$$\arg\max_{\{i,j\}\in P} u_{\mathbf{A},\mathbf{X}^*}(i,j)$$

Thus, both the network embedding accumulating the information from network structure and query strategies for selecting the most essential-to-know node pair are very important. ALPINE inherently suggests a lot of intuitive querying strategies for this providing justification for each choice. The querying strategies suggested are:

Strategy	Definition	Utility Function
page-rank	Page Rank score sum	$u_A(i,j) = \mathrm{PR}_i + \mathrm{PR}_j$
max-deg	Degree sum	$u_A(i,j) = \sum_{k:(i,k)\in E} a_{ik} + \sum_{l:(j,l)\in E} a_{jl}$
max-prob	Link probability	$u_{A,X^*}(i,j) = P(a_{ij} = 1 \mid X^*)$
min-dis	Node pair distance	$u_{A,X^*}(i,j) = -\|x_i^* - x_j^*\|_2$
max-ent	Link entropy	$u_{A,X^*}(i,j) = -\sum_{a_{ij}=0,1} P(a_{ij}\mid X^*) \log P(a_{ij}\mid X^*)$
d-opt	Parameter variance reduction	$u_{A,X^*}(i,j) = u_{x_i^*}(i,j) + u_{x_j^*}(i,j)$
v-opt	Prediction variance reduction	$u_{A,X^*}(i,j) = \sum_{k:(i,k)\in T} u^{ik}(i,j) + \sum_{l:(j,l)\in T} u^{jl}(i,j)$

Table 1. Querying strategies used in ALPINE.

We argue, that the proposed query strategies namely [**max-deg**, **max-prob**, **min-dis**, **max-ent**] are prone to outliers and can drastically affect the model if there is noise in the input Partially Observed Network. We suggest an intuitive improvement over these base queries in Suggested improvements in querying strategies for ALPINE.

The SINE framework.

SINE algorithm proposes an algorithm for learning node representations from large scale incomplete/partially observed graphs. It tries to mitigate the negative effects of missing information on representation learning.

Problem SINE aims to solve: Given an incomplete network $G = (\mathcal{V}, \mathcal{E}, \mathcal{A}, X, \Omega)$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ set of observed edges and $\mathcal{A}$ the set of node attributes. $X \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{A}|}$ is the node feature matrix with each observed element $X_{ij} \geq 0$ indicates the occurrence times/weights of attribute $a_j \in \mathcal{A}$ at node $v_i \in \mathcal{V}$, and $\Omega \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{A}|}$ indicates whether the node attribute value X_{ij} is observed, with $\Omega_{ij} = 1$ for observed and $\Omega_{ij} = 0$ for unobserved.

The objective of incomplete network embedding is to leverage incomplete information in $\mathcal{E}$ and X to learn a mapping function $\Phi : v_i \in \mathcal{V} \rightarrow \mathbb{R}^{|\mathcal{V}|\times d}$. The learned node representations $\Phi(v_i)$ are expected to be (1) low-dimensional with $d \ll |\mathcal{V}|$, and (2) informative for the downstream tasks, such as node classification, node clustering and link prediction, etc.

This is very similar to the type of node embeddings the ALPINE model desires for its query strategies.

Considering a DeepWalk model, define: $\boldsymbol{n(v_i, v_j)} :=$ The total occurrence times of v_j as v_i's context nodes across all generated random walks within t window size. The CNE model considers only local signals and hence information from neighbouring nodes. SINE alleviates this by allowing nodes sharing similar context nodes to have similar representations.

The learning framework of SINE is a three-layer neural network. First, it takes in a one-hot representation of the nodes $\mathbf{p}^{(i)} \in \mathbb{R}^{|V|}$, the node representation.

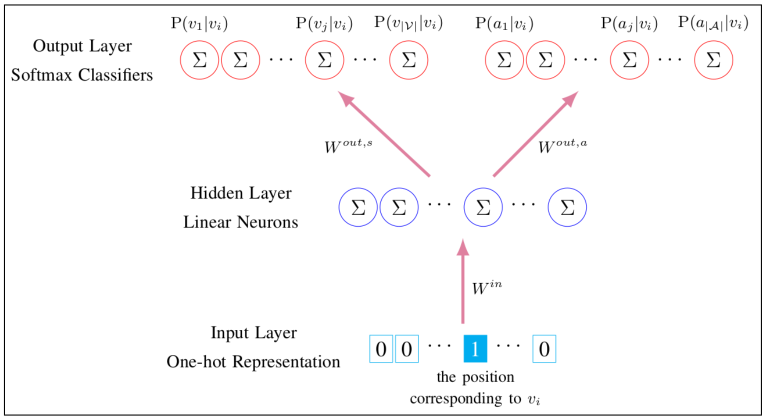

Figure 2. The model architecture of SINE.

★ Given the first layer, the node representation $\Phi(v_i) = W^{in^T}\mathbf{p}^{(i)} = \mathbf{w}_i^{in}$

★ In the output layer, for node context pair (v_i, v_j), the probability $P(v_j|v_i)$ is modeled by the softmax signal:

$$P(v_j|v_i) = \frac{\exp(\Phi(v_i)\cdot \mathbf{w}_j^{out,s})}{\sum_{k=1}^{|\mathcal{V}|}\exp(\Phi(v_i)\cdot \mathbf{w}_k^{out,s})}$$

where, $\mathbf{w}_j^{out,s}$ is the j-th column of $W^{out,s} \in \mathbb{R}^{d\times|\mathcal{V}|}$, the weight matrix from the hidden layer to the output layer for predicting node context.

★ Similarly, the node attribute co-occurrence pair (v_i, a_j):

$$P(a_j|v_i) = \frac{\exp(\Phi(v_i)\cdot \mathbf{w}_j^{out,a})}{\sum_{k=1}^{|\mathcal{A}|}\exp(\Phi(v_i)\cdot \mathbf{w}_k^{out,a})}$$

where, $\mathbf{w}_j^{out,a}$ is the j-th column of $W^{out,a} \in \mathbb{R}^{d\times|\mathcal{A}|}$, the weight matrix from the hidden layer to the output layer for predicting node attribute.

Based on the context and information preserving network, we can solve the following optimisation problem to get the node-embeddings:

$$\min_{\Phi} \mathcal{O}$$

where,

$$\mathcal{O} = -\alpha_1 \sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|} n(v_i, v_j)\log P(v_j|v_i) - \alpha_2 \sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{A}|}\Omega_{ij}X_{ij}\log P(a_j|v_i)$$

where α_1 and α_2 are the tradeoff parameters used for normalisation of each of the quantities.

In this optimization we are only concerned about non-zero values of $n(v_i, v_j)$ and $\Omega_{ij}X_{ij}$ i.e., the node context pairs collected from observed links and the observed node attribute co-occurrence pairs. In this way, the available network structure and node content information can be fully utilized, and the negative impacts of missing information is diminished.

This model enables us to use the encoded information about the graph structure and global signals. On top of that it also considers vector node attributes and tackles partial knowledge about node attributes too. This is definitely a strong point over a Maximum Likelihood approach like CNE. **Note:** that Φ is the network embedding.

Suggested improvements in querying strategies for ALPINE.

Query strategies like Uncertainty Sampling (mentioned as **max-ent** in Table 1), Degree sum, Link Probability, Node pair distance are prone to outliers. Other than these even Query-by-Committee (QBC) strategy is drastically affected by outliers. The least certain instance lies on the classification boundary, but is not "representative" of other instances in the distribution, so knowing its label is unlikely to improve accuracy on the data as a whole. So we must have a workaround so that such outliers don't perturb the model a lot. To address this we can use an **active learning approach called information density (ID):**

$$\phi^{ID}(\mathbf{x}) = \phi^{SE}(\mathbf{x}) \times \left(\frac{1}{U} \sum_{u=1}^{U} \text{sim}(\mathbf{x}, \mathbf{x}^{(u)}) \right)^{\beta}$$

The informativeness of $\mathbf{x}$ is weighted by its average similarity to the other node-pairs in the unobserved set, subject to the β which controls the relative importance of the density term.

In our implementation we use the cosine similarity as the similarity measure:

$$\text{sim}_{cos}(\mathbf{x}, \mathbf{x}^{(u)}) = \frac{\mathbf{x} \cdot \mathbf{x}^{(u)}}{\|\mathbf{x}\| \times \|\mathbf{x}^{(u)}\|}$$

We report our observations by weighing each of the query measures mentioned above (as ϕ^{SE}) with this average measure of similarity.

The proposed framework combining the key ideas from the above discussion.

In this section we discuss how we can combine the SINE: Scalable Incomplete Network Embedding framework with ideas from ALPINE: Active Link Prediction Using Network Embedding. We also see how we can use distance-weighting improvements to utility functions to construct a robust algorithm for Active Learning Link Prediction task using Node Embeddings in Partially Observed Networks.

We summarize our algorithm as follows:

⇒ We are given a PON $\mathcal{G} = (V, E, D)$, network embedding model, a query strategy defined by a utility function $u_{A,X}$, a pool $P \subseteq U$, a target set $T \subseteq U$, a step size s, a budget k (number of links that can be queried to the oracle).

⇒ At iteration $i = 0$ we initialize the pool as $P_0 = P$, and the set of node pairs with known link status as $K_0 = E \cup D$, and initialize $G_0 = G$ and $A_0 = A$.

⇒ Repeat:

- ⇒ Construct a `SINE` model and `fit()` it on the graph given by `A`
- ⇒ Train a link prediction algorithm and get all the probabilities of the link being present according to the trained `SINE` embeddings. `[This forms our posterior probability when compared with the CNE model]`
- ⇒ We find the set of queries $Q_i \subseteq P_i$ of size $|Q_i| = \min(s, k)$ with the largest utilities according to u_{A,X^*} (with or without distance-weighting)
- ⇒ Query the oracle for the link statuses of node pairs in Q_i , set $P_{i+1} \leftarrow P_i \backslash Q_i$ and set $\mathcal{G}_{i+1}$ equal to $\mathcal{G}_i$ with node pairs $\{i, j\} \in Q_i$ added to the set of known linked or unlinked node pairs (depending on the query result), then set A_{i+1} accordingly
- ⇒ Set $k \leftarrow k - |Q_i|$ and break if k is 0

We run several experiments and find interesting results when we vary the % of the network observed and try out different query strategies. We report our results in Results.

Results

Comparing our algorithm with the proposed ALPINE algorithm.

We compare the average ROC-AUC scores for the different Node embedding models wrt to the change in the % of observed network. Unfortunately, we do not observe improvements in accuracy with the SINE embedding model. Despite **SINE** having the ability to capture global signals and complex networks we see relatively poor results as compared to a much simpler model like **CNE**. We postpone further tests and tweaks as possible future work.

To run this experiment: `python3 exp1.py`

To plot: `python3 plt_exp1.py`

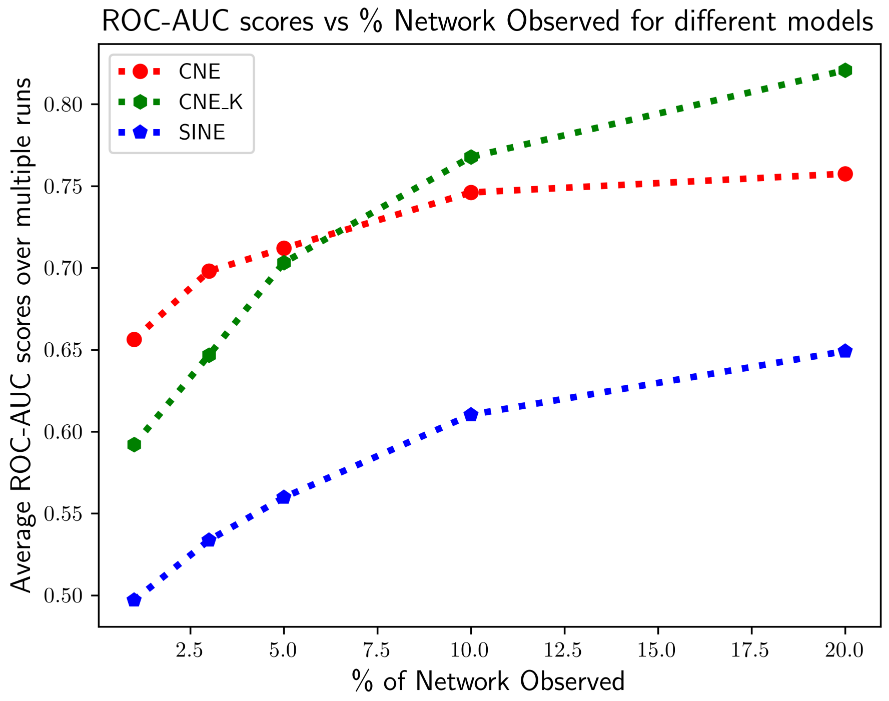

Figure 3. **Experiment 1** Comparison of the **average ROC-AUC** scores achieved by **CNE, CNE_K** and **SINE** as the Node Embedding models in **ALPINE** with respect to the change in **% of network observed**

Comparing the performance of the querying strategies with CNE-K model in place.

We compare the average ROC-AUC scores for the different querying strategies with/without the **Information Density** (Distance Weighting) addition defined in Suggested improvements in querying strategies for ALPINE. Here, we expect to see improvements in querying strategies prone to outliers like the `[''max_degree_sum'', ''max_probability'', ''min_distance'', ''max_entropy'']`.

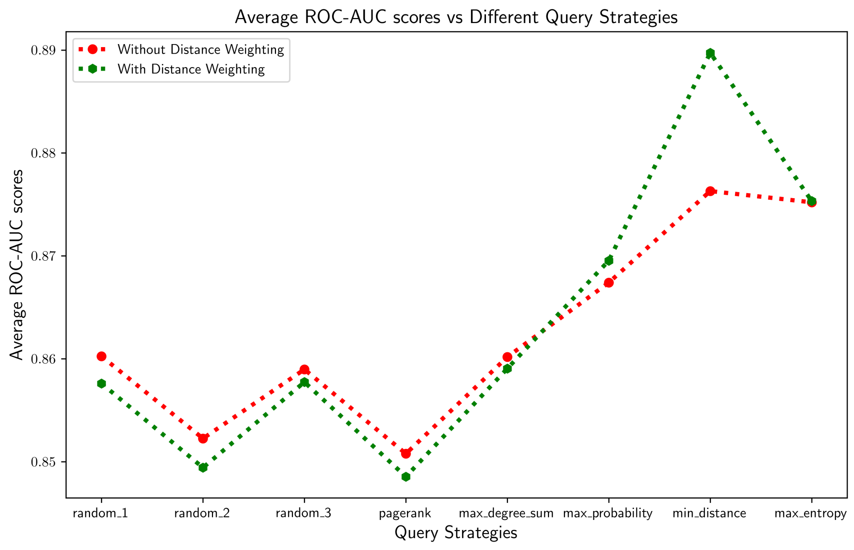

Figure 4. Experiment 2. Comparison of the **average ROC-AUC** scores achieved by **CNE_K** as the Node Embedding model with respect to the **different querying strategies** implemented, with and without the **Information Density** weighting addition.

References

1. B. Kang, J. Lijffijt, and T. D. Bie. Conditional network embeddings, 2018.
2. J. L. Xi Chen, Bo Kang and T. D. Bie. Alpine: Active link prediction using network embedding, 2021.
3. D. Zhang, J. Yin, X. Zhu, and C. Zhang. Sine: Scalable incomplete network embedding, 2018.