


RESEARCH ARTICLE

A multi-purposed unsupervised framework for comparing embeddings of undirected and directed graphs

Bogumił Kamiński¹, Łukasz Kraiński¹, Paweł Pralat^{2*}  and François Thériberge³

¹Decision Analysis and Support Unit, SGH Warsaw School of Economics, Warsaw, Poland, ²Department of Mathematics, Toronto Metropolitan University, Toronto, ON, Canada, and ³Tutte Institute for Mathematics and Computing, Ottawa, ON, Canada

*Corresponding author. Email: pralat@ryerson.ca

Action Editor: Ulrik Brandes

Abstract

Graph embedding is a transformation of nodes of a network into a set of vectors. A good embedding should capture the underlying graph topology and structure, node-to-node relationship, and other relevant information about the graph, its subgraphs, and nodes themselves. If these objectives are achieved, an embedding is a meaningful, understandable, and often compressed representation of a network. Unfortunately, selecting the best embedding is a challenging task and very often requires domain experts. In this paper, we extend the framework for evaluating graph embeddings that was recently introduced in [15]. Now, the framework assigns two scores, local and global, to each embedding that measure the quality of an evaluated embedding for tasks that require good representation of local and, respectively, global properties of the network. The best embedding, if needed, can be selected in an unsupervised way, or the framework can identify a few embeddings that are worth further investigation. The framework is flexible and scalable and can deal with undirected/directed and weighted/unweighted graphs.

Keywords: graph embeddings; null models; evaluation of node embeddings

1. Introduction

Network Geometry is a rapidly developing approach in Network Science (Hoff et al., 2002) which enriches the system by modeling the nodes of the network as points in a geometric space. There are many successful examples of this approach that include latent space models (Krioukov, 2016), and connections between geometry and network clustering and community structure (Gastner & Newman, 2006; Zuev et al., 2015). Very often, these geometric embeddings naturally correspond to physical space, such as when modeling wireless networks or when networks are embedded in some geographic space (Expert et al., 2011; Janssen, 2010).

Another important application of geometric graphs is in graph embeddings (Aggarwal & Murty, 2021). In order to extract useful structural information from graphs, one might want to try to embed them in a geometric space by assigning coordinates to each node such that nearby nodes are more likely to share an edge than those far from each other. Moreover, the embedding should also capture global structure and topology of the associated network, identify specific roles of nodes, etc.

Due to their spectacular successes in various applications, graph embedding methods are becoming increasingly popular in the machine learning community. They are widely used for

tasks such as node classification, community detection, and link prediction, but other applications such as anomaly detection are currently explored. As reported in Chen *et al.* (2021), the ratio between the number of papers published in top three conferences (ACL, WWW, and KDD) closely related to Computational Social Science (CSS) applying symbol-based representations and the number of papers using embeddings decreased from 10 in 2011 to 1/5 in 2020. We expect this trend to continue with embeddings eventually playing a central role in many machine learning tasks.

There are over 100 algorithms proposed in the literature for node embeddings. The techniques and possible approaches to construct the desired embedding can be broadly divided into the following three families: linear algebra algorithms, random walk-based algorithms, and deep learning methods (Aggarwal & Murty 2021; Kamiński *et al.*, 2022). All of these algorithms have plenty of parameters to tune, the dimension of the embedding being only one of them but an important one. Moreover, most of them are randomized algorithms which means that even for a given graph, each time we run them we get a different embedding, possibly of different quality. As a result, it is not clear which algorithm and which parameters one should use for a given application at hand. There is no clear winner and, even for the same application, the decision which algorithm to use might depend on the properties of the investigated network (Dehghan-Kooshkghazi *et al.*, 2020).

In the initial version of the framework, as detailed in Kamiński *et al.* (2020), only undirected, unweighted graphs were considered. A null model was introduced by generalizing the well-known Chung–Lu model (Chung *et al.*, 2006) and, based on that, a global divergence score was defined. The global score can be computed for each embedding under consideration by comparing the number of edges within and between communities (obtained via some stable graph clustering algorithm) with the corresponding expected values under the null model. This global score measures how well an embedding preserves the global structure of the graph. In order to handle huge graphs, a landmark-based version of the framework was introduced in Kamiński *et al.* (2020), which can be calibrated to provide a good trade-off between performance and accuracy.

In this paper, we generalize the original framework in several ways. Firstly, we add the capability of handling both directed and undirected graphs as well as taking edge weights into account. Moreover, we introduce a new, local score which measures how well each embedding preserves local properties related to the presence of edges between pairs of nodes. The global and local scores can be combined in various ways to select good embedding(s). We illustrate the usefulness of our framework for several applications by comparing those two scores, and we show various ways to combine them to select embeddings. After appropriate adjustments to directed and/or weighted graphs, landmarks can be used the same way as in the original framework to handle huge graphs.

The paper is structured as follows. The framework is introduced in Section 2. The geometric Chung–Lu (GCL) model that is the heart of the framework is introduced and discussed in Section 3. Section 4 presents experiments justifying the usefulness of the framework and investigating the quality of a scalable approximation algorithm. Finally, some conclusions and future work are outlined in Section 5. Some extra details and experiments are available in the companion “supplementary document.”

Finally, let us mention that standard measures such as various correlations coefficients, the accuracy, and the adjusted mutual information (AMI) score are not formally defined in this paper. The reader is directed to any book on data science or machine learning (e.g. to Kamiński *et al.* (2022)) for definitions and more.

2. The framework

In this section, we introduce the unsupervised framework for comparing graph embeddings, the main contribution of this paper. Section 2.1 is devoted to high-level description and intuition

behind the two embedding quality scores, local and global one, returned by the framework. The algorithm that computes them is formally defined in Section 2.2. Looking at the two scores to make an informed decision which embedding to choose is always recommended but if one wants to use the framework to select the best embedding in an unsupervised manner, then one may combine the two scores into a single value. We discuss this process in Section 2.3. The description of the scoring algorithm assumes that the graph is unweighted and directed. Generalizing it to undirected or weighted graphs is straightforward, and we discuss it in Section 2.4.

2.1 Intuition behind the algorithm

The proposed framework is multi-purposed, that is, it independently evaluates embeddings using two approaches.

The first approach looks at the network and the associated embeddings “from the distance,” trying to see a “big picture.” It evaluates the embeddings based on their ability to capture global properties of the network, namely, edge densities. In order to achieve it, the framework compares edge density between and within the communities that are present (and can be easily recovered) in the evaluated graph G with the corresponding expected edge density in the associated random null model. This score is designed to identify embeddings that should perform well in tasks requiring global knowledge of the graph such as node classification or community detection. We will call this measure a *global score*.

The second approach looks at the network and embeddings “under the microscope,” trying to see if a local structure of a graph G is well reflected by the associated embedding. The *local score* will be designed in such a way that it is able to evaluate if the embedding is a strong predictor of (directed) adjacency between nodes in the network. In general, this property could be tested using any strong supervised machine learning algorithm. However, our objective is to test not only predictive power but also explainability of the embedding (sometimes referred to as interpretability). Namely, we assume that the adjacency probability between nodes should be monotonically linked with their distance in the embedding and their in and out degrees. This approach has the following advantage: embeddings that score well should not only be useful for link prediction, but they should perform well in any task that requires a local knowledge of the graph. To achieve this, we use the same random null model as we use to calculate the global score to estimate the probability of two nodes to be adjacent. This question is the well-known and well-studied problem of link prediction in which one seeks to find the node pairs most likely to be linked by an edge. When computing the local score, we calculate a ranking of the node pairs from most to least likely of being linked. A common evaluation method for such problem is to compute the AUC (area under the ROC curve). It is important to note that the AUC is independent of the ratio between the number of edges and the number of non-edges in the graph. As a result, it can be approximated using random sampling.

Despite the fact that the global and local scores take diametrically different points of view, there is often correlation between the two. Good embeddings tend to capture both global and local properties and, as a result, they score well in both approaches. On the other hand, poor embeddings have problems with capturing any useful properties of graphs and so their corresponding scores are both bad. Nevertheless, they are certainly not identical, and we will show examples in which the two scores are different.

2.2 The algorithm

In this section, and later in the paper, we use $[n]$ to denote the set of natural numbers less than or equal to n , that is, $[n] := \{1, \dots, n\}$. Given a directed graph $G = (V, E)$ (in particular, given its in-degree and out-degree distributions \mathbf{w}^{in} and \mathbf{w}^{out} on V) and an embedding $\mathcal{E}: V \rightarrow \mathbb{R}^k$ of its nodes

in k -dimensional space, we perform the steps detailed below to obtain $(\Delta_{\mathcal{E}}^g(G), \Delta_{\mathcal{E}}^l(G))$, a pair of respectively global and local *divergence scores* for the embedding. Indeed, as already mentioned the framework is multi-purposed and, depending on the specific application in mind, one might want the selected embeddings that preserve global properties (density-based evaluation) and/or pay attention to local properties (link-based evaluation). As a result, we independently compute the two corresponding divergence scores, $\Delta_{\mathcal{E}_i}^g(G)$ and $\Delta_{\mathcal{E}_i}^l(G)$, to provide the users of the framework with a more complete picture and let them make an informative decision which embedding to use. We typically apply this algorithm to compare several embeddings $\mathcal{E}_1, \dots, \mathcal{E}_m$ of the same graph and select the best one via $\operatorname{argmin}_{i \in [m]} \Delta_{\mathcal{E}_i}(G)$, where $\Delta_{\mathcal{E}_i}(G)$ is the *combined divergence score* that takes into account the scores of the competitors and that can be tuned for a given application at hand.

Note that our algorithm is a general framework, and some parts have flexibility. We clearly identify these below and provide a specific, default, approach that we applied in our implementation. In the description below, we assume that the graph is directed and unweighted, and we then discuss how the framework deals with undirected and/or weighted graphs.

The code can be accessed at the GitHub repository¹. The first version of the framework (c.f. Kamiński *et al.* (2020)) was designed for undirected, unweighted graphs, and only used the density-based evaluation. Since it was used for experiments reported in various papers, for reproducibility purpose, the code can still be accessed on GitHub².

2.2.1 Global (Density-based) evaluation: $\Delta_{\mathcal{E}}^g(G)$

Step 1: Run some stable *graph* clustering algorithm on G to obtain a partition \mathbf{C} of the set of nodes V into ℓ communities C_1, \dots, C_ℓ .

Note: In our implementation, we used the ensemble clustering algorithm for unweighted graphs (ECG) which is based on the Louvain algorithm (Blondel *et al.*, 2008) and the concept of consensus clustering (Poulin & Th  berge 2018) and is shown to have good stability. For weighted graphs, by default we use the Louvain algorithm.

Note: In some applications, the desired partition may be provided together with a graph (e.g., when nodes contain some natural labeling and so some form of a ground-truth is provided). The framework is flexible and allows for communities to be provided as an input to the framework instead of using a clustering algorithm. In the supplementary document, we present a number of experiments that show that the choice of clustering algorithm does not affect the score in a significant way (provided the clustering is stable and of good quality).

Step 2: For each $i \in [\ell]$, let c_i be the proportion of directed edges of G with both end points in C_i . Similarly, for each $1 \leq i, j \leq \ell$, $i \neq j$, let $c_{i,j}$ be the proportion of directed edges of G from some node in C_i to some node in C_j . Let

$$\begin{aligned}\bar{\mathbf{c}} &= (c_{1,2}, c_{2,1}, \dots, c_{1,\ell}, c_{\ell,1}, c_{2,3}, c_{3,2}, \dots, c_{\ell-1,\ell}, c_{\ell,\ell-1}) \\ \hat{\mathbf{c}} &= (c_1, \dots, c_\ell)\end{aligned}\tag{1}$$

and let $\mathbf{c} = \bar{\mathbf{c}} \oplus \hat{\mathbf{c}}$ be the concatenation of the two vectors with a total of $2\binom{\ell}{2} + \ell = \ell^2$ entries which together sum to one. This *graph vector* \mathbf{c} characterizes the partition \mathbf{C} from the perspective of the graph G .

Note: The embedding \mathcal{E} does *not* affect the vectors $\bar{\mathbf{c}}$ and $\hat{\mathbf{c}}$ (and so also \mathbf{c}). They are calculated purely based on G and the partition \mathbf{C} .

Step 3: For a given parameter $\alpha \in \mathbb{R}_+$ and the same partition of nodes \mathbf{C} , we consider the GCL directed graph model $\mathcal{G}(\mathbf{w}^{in}, \mathbf{w}^{out}, \mathcal{E}, \alpha)$ presented in Section 3 that can be viewed in this context

as the associated null model. For each $1 \leq i, j \leq \ell, i \neq j$, we compute $b_{i,j}$, the expected proportion of directed edges of $\mathcal{G}(\mathbf{w}^{in}, \mathbf{w}^{out}, \mathcal{E}, \alpha)$ from some node in C_i to some node in C_j . Similarly, for each $i \in [\ell]$, let b_i be the expected proportion of directed edges within C_i . That gives us another two vectors:

$$\begin{aligned} \bar{\mathbf{b}}_{\mathcal{E}}(\alpha) &= (b_{1,2}, b_{2,1}, \dots, b_{1,\ell}, b_{\ell,1}, b_{2,3}, b_{3,2}, \dots, b_{\ell-1,\ell}, b_{\ell,\ell-1}) \\ \hat{\mathbf{b}}_{\mathcal{E}}(\alpha) &= (b_1, \dots, b_{\ell}) \end{aligned} \tag{2}$$

and let $\mathbf{b}_{\mathcal{E}}(\alpha) = \bar{\mathbf{b}}_{\mathcal{E}}(\alpha) \oplus \hat{\mathbf{b}}_{\mathcal{E}}(\alpha)$ be the concatenation of the two vectors with a total of ℓ^2 entries which together sum to one. This *model vector* $\mathbf{b}_{\mathcal{E}}(\alpha)$ characterizes the partition \mathbf{C} from the perspective of the embedding \mathcal{E} .

Note: The structure of graph G does *not* affect the vectors $\bar{\mathbf{b}}_{\mathcal{E}}(\alpha)$ and $\hat{\mathbf{b}}_{\mathcal{E}}(\alpha)$; only its degree distribution $\mathbf{w}^{in}, \mathbf{w}^{out}$, and embedding \mathcal{E} are used.

Note: We used the GCL directed graph model, but the framework is flexible. If, for any reason (perhaps there are some restrictions for the maximum edge length; such restrictions are often present in, for example, wireless networks) it makes more sense to use some other model of random geometric graphs, it can be easily implemented here. If the model is too complicated and computing the expected number of edges between two parts is challenging, then it can be approximated via simulations.

Step 4: Compute the distance Δ_{α} between the two vectors, \mathbf{c} and $\mathbf{b}_{\mathcal{E}}(\alpha)$, in order to measure how well the model $\mathcal{G}(\mathbf{w}^{in}, \mathbf{w}^{out}, \mathcal{E}, \alpha)$ fits the graph G .

Note: We used the well-known and widely used Jensen–Shannon divergence (JSD) to measure the dissimilarity between two probability distributions, that is, $\Delta_{\alpha} = JSD(\mathbf{c}, \mathbf{b}_{\mathcal{E}}(\alpha))$. The JSD was originally proposed in Lin (1991) and can be viewed as a smoothed version of the Kullback–Leibler divergence.

Note: Alternatively, one may independently treat internal and external edges to compensate the fact that there are $2\binom{\ell}{2} = \Theta(\ell^2)$ coefficients related to external densities, whereas only ℓ ones related to internal ones. Then, for example, after appropriate normalization of the vectors, a simple average of the two corresponding distances can be used, that is,

$$\Delta_{\alpha} = \frac{1}{2} \cdot \left(JSD(\bar{\mathbf{c}}, \bar{\mathbf{b}}_{\mathcal{E}}(\alpha)) + JSD(\hat{\mathbf{c}}, \hat{\mathbf{b}}_{\mathcal{E}}(\alpha)) \right)$$

Depending on the application at hand, other weighted averages can be used if more weight needs to be put on internal or external edges.

Step 5: Select $\hat{\alpha} = \operatorname{argmin}_{\alpha} \Delta_{\alpha}$ and define the *global (density-based) score* for embedding \mathcal{E} on G as $\Delta_{\mathcal{E}}^g(G) = \Delta_{\hat{\alpha}}$.

Note: The parameter α is used to define a distance in the embedding space, as we detail in Section 3. In our implementation, we simply checked values of α from a dense grid, starting from $\alpha = 0$ and finishing the search if no improvement is found for five consecutive values of α . Clearly, there are potentially faster ways to find an optimum value of α but, since our algorithm is the fast-performing grid search, this approach was chosen as both easy to implement and robust to potential local optima.

2.2.2 Local (Link-based) evaluation: $\Delta_{\mathcal{E}}^{\ell}(G)$

Step 6: We let

$$S^+ = \{(u, v) \in V \times V, u \neq v; uv \in E\},$$

$$S^- = \{(u, v) \in V \times V, u \neq v; uv \notin E\}.$$

For a given parameter $\alpha \in \mathbb{R}_+$, we again consider the GCL directed graph model $\mathcal{G}(\mathbf{w}^{in}, \mathbf{w}^{out}, \mathcal{E}, \alpha)$ detailed in Section 3. Let $p(u, v)$ be the probability of a directed edge $u \rightarrow v$ to be present under this model.

The receiver operating characteristic (ROC) is a curve showing the performance of a classification model at all classification thresholds (for the edge probabilities in the present context). The AUC (area under the ROC curve) provides an aggregate measure of performance across all possible classification thresholds which can be interpreted as the probability that a randomly chosen positive sample (here, a pair of nodes connected by an edge) is ranked higher than a negative sample (a pair of nodes without an edge). Thus, the AUC can be expressed as follows:

$$p_{\alpha} = \frac{\sum_{(s,t) \in S^+} \sum_{(u,v) \in S^-} \mathbb{1}\{p(s, t) > p(u, v)\}}{|S^+| \cdot |S^-|}$$

As a result, the AUC measures how much the model is capable of distinguishing between the two classes, S^+ and S^- . In other words, it may be viewed as the probability that $p(s, t) > p(u, v)$, provided that a directed edge $s \rightarrow t$ and a directed non-edge $u \not\rightarrow v$ are selected uniformly at random from S^+ and, respectively, S^- .

Note: In practice, there is no need to investigate all $|S^+| \cdot |S^-|$ pairs of nodes. Instead, we can randomly sample (with replacement) k pairs $(s_i, t_i) \in S^+$ and k pairs $(u_i, v_i) \in S^-$ and then compute

$$\hat{p}_{\alpha} = \frac{\sum_{i=1}^k \mathbb{1}\{p(s_i, t_i) > p(u_i, v_i)\}}{k}$$

to approximate p_{α} . The value of k is adjusted so that the approximate 95% confidence interval, namely,

$$\left[\hat{p}_{\alpha} - 1.96\sqrt{\hat{p}_{\alpha}(1 - \hat{p}_{\alpha})/k}, \hat{p}_{\alpha} + 1.96\sqrt{\hat{p}_{\alpha}(1 - \hat{p}_{\alpha})/k} \right]$$

is shorter than some precision level. In our implementation, the default value of k is set to $k = 10,000$ so that the length of the interval is guaranteed to be at most 0.02.

Step 7: Select $\hat{\alpha} = \operatorname{argmin}_{\alpha}(1 - \hat{p}_{\alpha})$ and define the *local (link-based) score* for embedding \mathcal{E} on G as $\Delta_{\mathcal{E}}^{\ell}(G) = 1 - \hat{p}_{\hat{\alpha}}$.

2.3 Combined divergence scores for evaluating many embeddings

As already mentioned a few times, the framework is multi-purposed and, depending on the specific application in mind, one might want the selected embeddings that preserve global properties (global, density-based evaluation) or pay more attention to local properties (local, link-based evaluation). That is the reason, we independently compute the two corresponding divergence scores, $\Delta_{\mathcal{E}_i}^g(G)$ and $\Delta_{\mathcal{E}_i}^{\ell}(G)$.

In order to compare several embeddings for the same graph G , we repeat steps 3–7 above, each time computing the two scores for a given embedding. Let us stress again that steps 1–2 are done only once, that is, we use the same partition of the graph into ℓ communities for all embeddings. In order to select (in an unsupervised way) the best embedding to be used, one may simply compute

the combined divergence score, a linear combination of the two scores:

$$\Delta_{\mathcal{E}_i}(G) = q \cdot \frac{(\Delta_{\mathcal{E}_i}^g(G) + \varepsilon)}{\min_{j \in [m]} (\Delta_{\mathcal{E}_j}^g(G) + \varepsilon)} + (1 - q) \cdot \frac{(\Delta_{\mathcal{E}_i}^\ell(G) + \varepsilon)}{\min_{j \in [m]} (\Delta_{\mathcal{E}_j}^\ell(G) + \varepsilon)} \tag{3}$$

for a fixed parameter $q \in [0, 1]$, carefully selected for a given application at hand, and $\varepsilon = 0.01$, introduced to prevent rare but possible numerical issues when one of the scores is close to zero. Note that $\Delta_{\mathcal{E}_i}(G) \geq 1$ and $\Delta_{\mathcal{E}_i}(G) = 1$ if and only if a given embedding \mathcal{E}_i does not have a better competitor in *any* of the two evaluation criteria. Of course, the lower the score, the better the embedding is. The winner \mathcal{E}_j can be identified by taking $j = \operatorname{argmin}_i \Delta_{\mathcal{E}_i}(G)$.

Let us briefly justify the choice of function (3). First note that both $\Delta_{\mathcal{E}_i}^g(G)$ and $\Delta_{\mathcal{E}_i}^\ell(G)$ are in $[0, 1]$. However, since they might have different orders of magnitude (and typically they do), the corresponding scores need to be normalized. In decision theory, one typically simply tunes q to properly take this into account. While we allow for the more advanced user to change the value of q , we believe that it is preferable to provide a reasonable scaling when the default value of q , namely, $q = 1/2$ is used. When choosing a normalization by minimum, we were guided by the fact that it is not uncommon that most of the embeddings score poorly in both dimensions; if this is so, then they affect for example the average score but they ideally should not influence the selection process. On the other hand, the minimum clearly should not be affected by bad embeddings. In particular, the normalization by the minimum allows us to distinguish the situation in which two embeddings have similar but large scores (indicating that both embeddings are bad) from the situation in which two embeddings have similar but small scores (one of the two corresponding embeddings can still be significantly better).

Finally, let us mention that while having a single score assigned to each embedding is useful, it is always better to look at the composition of the scores,

$$\left(\frac{(\Delta_{\mathcal{E}_i}^g(G) + \varepsilon)}{\min_{j \in [m]} (\Delta_{\mathcal{E}_j}^g(G) + \varepsilon)}, \frac{(\Delta_{\mathcal{E}_i}^\ell(G) + \varepsilon)}{\min_{j \in [m]} (\Delta_{\mathcal{E}_j}^\ell(G) + \varepsilon)} \right)$$

to make a more informative decision. See Section 4.3 for an example of such a selection process.

2.4 Weighted and undirected graphs

For simplicity, we defined the framework for unweighted but directed graphs. Extending the density-based evaluation to weighted graphs can be easily and naturally done by considering the sum of weights of the edges instead of the number of them, for example, c_i is the proportion of the total weight concentrated on the directed edges with both end points in C_i . For the link-based evaluation, we need to adjust the definition of the AUC so that it is equal to the probability that $p(s, t) > p(u, v)$ times the weight of a directed edge $s \rightarrow t$ (scaled appropriately such that the average scaled weight of all edges investigated is equal to one), provided that a directed edge $s \rightarrow t$ is selected from S^+ (the set of all edges) and a directed non-edge $u \not\rightarrow v$ is selected from S^- (the set of non-edges), both of them uniformly at random. As before, such quantity may be efficiently approximated by sampling.

On the other hand, clearly undirected graphs can be viewed as directed ones by replacing each undirected edge uv by two directed edges uv and vu . Hence, one can transform an undirected graph G to its directed counterpart and run the framework on it. However, the framework is tuned for a faster running time when undirected graphs are used but, of course, the divergence score remains unaffected.

3. GCL model

The heart of the framework is the associated random graph null model that is used to design both the global and the local score. The GCL model, a generalization of the original Chung–Lu model (Chung *et al.*, 2006), was introduced in Kamiński *et al.* (2020) to benchmark embeddings of undirected graphs (at that time only from the global perspective). Now, we need to generalize it even further to include directed and weighted graphs. We do it in Section 3.1. A scalable implementation is discussed in Section 3.2.

3.1 Geometric directed model

In the GCL directed graph model, we are not only given the expected degree distribution of a directed graph G

$$\mathbf{w}^{in} = (w_1^{in}, \dots, w_n^{in}) = (\deg_G^{in}(v_1), \dots, \deg_G^{in}(v_n))$$

$$\mathbf{w}^{out} = (w_1^{out}, \dots, w_n^{out}) = (\deg_G^{out}(v_1), \dots, \deg_G^{out}(v_n))$$

but also an embedding \mathcal{E} of nodes of G in some k -dimensional space, $\mathcal{E}: V \rightarrow \mathbb{R}^k$. In particular, for each pair of nodes, v_i, v_j , we know the distance between them:

$$d_{i,j} = \text{dist}(\mathcal{E}(v_i), \mathcal{E}(v_j))$$

It is desired that the probability that nodes v_i and v_j are adjacent to be a function of $d_{i,j}$, that is, to be proportional to $g(d_{i,j})$ for some function g . The function g should be a decreasing function as long edges should occur less frequently than short ones. There are many natural choices such as $g(d) = d^{-\beta}$ for some $\beta \in [0, \infty)$ or $g(d) = \exp(-\gamma d)$ for some $\gamma \in [0, \infty)$. We use the following, normalized function $g: [0, \infty) \rightarrow [0, 1]$: for a fixed $\alpha \in [0, \infty)$, let

$$g(d) := \left(1 - \frac{d - d_{\min}}{d_{\max} - d_{\min}}\right)^\alpha = \left(\frac{d_{\max} - d}{d_{\max} - d_{\min}}\right)^\alpha$$

where

$$d_{\min} = \min\{\text{dist}(\mathcal{E}(v), \mathcal{E}(w)) : v, w \in V, v \neq w\}$$

$$d_{\max} = \max\{\text{dist}(\mathcal{E}(v), \mathcal{E}(w)) : v, w \in V\}$$

are the minimum, and respectively the maximum, distance between nodes in embedding \mathcal{E} . One convenient and desired property of this function is that it is invariant with respect to an affine transformation of the distance measure. Clearly, $g(d_{\min}) = 1$ and $g(d_{\max}) = 0$; in the computations, we can use clipping to force $g(d_{\min}) < 1$ and/or $g(d_{\max}) > 0$ if required. Let us also note that if $\alpha = 0$ (that is, $g(d) = 1$ for any $d \in [d_{\min}, d_{\max})$ with the convention that $g(d_{\max}) = 0^0 = 1$), then the pairwise distances are neglected. As a result, in particular, for undirected graphs we recover the original Chung–Lu model. Moreover, the larger parameter α is, the larger the aversion to long edges is. Since this family of functions (for various values of the parameter α) captures a wide spectrum of behaviors, it should be enough to concentrate on this choice, but one can experiment with other functions. So, for now we may assume that the only parameter of the model is $\alpha \in [0, \infty)$.

The *GCL directed graph* model is the random graph $G(\mathbf{w}^{in}, \mathbf{w}^{out}, \mathcal{E}, \alpha)$ on the set of nodes $V = \{v_1, \dots, v_n\}$ in which each pair of nodes v_i, v_j , independently of other pairs, forms a directed edge from v_i to v_j with probability $p_{i,j}$, where

$$p_{i,j} = x_i^{out} x_j^{in} g(d_{i,j})$$

for some carefully tuned weights $x_i^{in}, x_i^{out} \in \mathbb{R}_+$. The weights are selected such that the expected in-degree and out-degree of v_i is w_i^{in} and, respectively, w_i^{out} ; that is, for all $i \in [n]$

$$w_i^{out} = \sum_{j \in [n], j \neq i} p_{i,j} = x_i^{out} \sum_{j \in [n], j \neq i} x_j^{in} g(d_{i,j})$$

$$w_i^{in} = \sum_{j \in [n], j \neq i} p_{j,i} = x_i^{in} \sum_{j \in [n], j \neq i} x_j^{out} g(d_{i,j})$$

Additionally, we set $p_{i,i} = 0$ for $i \in [n]$ which corresponds to the fact that the model does not allow loops.

In the supplementary document, we prove that there exists the unique selection of weights, unless G has an independent set of size $n - 1$, that is, G is a star with one node being part of every edge. (Since each connected component of G can be embedded independently, we always assume that G is connected.) This very mild condition is satisfied in practice. Let us mention that in the supplementary document, it is assumed that $g(d_{i,j}) > 0$ for all pairs i, j . In our case, $g(d_{i,j}) = 0$ for a pair of nodes that are at the maximum distance. It causes no problems in practice but, as mentioned earlier, one can easily scale the outcome of function $g(\cdot)$ to move away from zero without affecting the divergence score in any nonnegligible way.

Finally, note that it is not clear how to find weights explicitly, but they can be efficiently approximated numerically to any desired precision. In the supplementary document, we prove that, if the solution exists, which is easy to check, then the set of right-hand sides of the equations, considered as a function from \mathbb{R}^{2n} to \mathbb{R}^{2n} , is a local diffeomorphism everywhere in its domain. As a result, standard gradient root-finding algorithms should be quite effective in finding the desired weights. In our implementation, we use even simpler numerical approximation procedure.

Note: The specification of the GCL directed graph model implies that the probability of having a directed edge from one node to another one increases with their out-degree and in-degree, respectively, and decreases with the distance between them. This is a crucial feature that ensures that the local divergence score is explainable. For instance, potentially one could imagine embeddings where nodes that are far apart are more likely to be connected by an edge. However, under our framework, such embeddings would get a low local score.

3.2 Scalable implementation

The main bottleneck of the algorithm is the process of tuning $2n$ weights $x_i^{out}, x_i^{in} \in \mathbb{R}_+$ ($i \in [n]$) in the GCL Graph (both in directed and in undirected counterpart). This part requires $\Theta(n^2)$ steps and so it is not feasible for large graphs. Fortunately, one may modify the algorithm slightly to obtain a scalable approximation algorithm that can be efficiently run on large networks. It was done for the framework for undirected graphs in Kamiński et al. (2020) to obtain the running time of $O(n \ln n)$ which is practical.

The main idea behind our approximation algorithm is quite simple. The goal is to group together nodes from the same part of the partition \mathbf{C} obtained in Step 1 of the algorithm that are close to each other in the embedded space. Once such refinement of partition \mathbf{C} is generated, one may simply replace each group by the corresponding auxiliary node (that we call a *landmark*) that is placed in the appropriately weighted center of mass of the group it is associated with. The reader is directed to Kamiński et al. (2020) for more details. Minor adjustments are only needed for the density-based evaluation to accommodate directed graphs and approximating the link-based score is easy. Both issues are discussed in the supplementary document.

The framework, by default, uses the approximation algorithm for networks with 10,000 nodes or more. In Section 4.4, we show how well the approximation algorithm works in practice.

4. Experiments

In this section, we detail some experiments we performed showing that the framework works as desired. In Section 4.1, we provide descriptions of both synthetic and real-world networks that we used for the experiments. A few embedding algorithms for directed graphs are introduced in Section 4.2. We used two of them for our experiments. In Section 4.3, we present results of an experiment with a small “toy example” to illustrate how one can use the framework to select the best embedding among several ones. As explained earlier, in order to have a scalable framework that can be used to evaluate huge graphs, we introduced landmarks to provide approximated scores. We show in Section 4.4 that this approximation works well. Finally, we ran experiments showing the usefulness of the framework. In Section 4.5, we show that the global score may be used to predict how good the evaluated embedding is for algorithms that require good global properties such as node classification or community detection algorithms. Similarly, the correlation between the local score and the ability of embeddings to capture local properties (for example, with link prediction algorithm) is investigated in Section 4.6. A few more experiments showing the differences between the two scores are presented in Section 4.7.

Experiments were conducted using **SOSCIP**³ Cloud infrastructure. We used Compute G4-x8 (8 vCPUs, 32 GB RAM) machines and Ubuntu 18.04 operating system. Computation used for experimentation and calibration of the scripts took approximately 1500 vCPU-hours. For reproducibility purpose, the scripts and results presented in this paper can be found on GitHub repository⁴.

4.1 Synthetic and real-world graphs

In order to test the framework, we performed various experiments on both synthetic and real-world networks.

4.1.1 Stochastic block model

To model networks with simple community structure, we used the classical stochastic block model (SBM) (Holland *et al.*, 1983) (see Funke & Becker (2019) for an overview of various generalizations). The model takes the following parameters as its input: the number of nodes n , a partition of nodes into ℓ communities, and an $\ell \times \ell$ matrix P of edge probabilities. Two nodes, u from i th community and v from j th community, are adjacent with probability P_{ij} , and the events associated with different pairs of nodes are independent. The model can be used to generate undirected graphs in which case matrix P should be symmetric, but one can also use the model to generate directed graphs.

For our experiments, we generated directed **SBM** graphs with $n = 10,000$ nodes and 30 communities of similar size. Nodes from two different communities are adjacent with probability $P_{ij} = 0.001 = 10/n$, and nodes from the same community are adjacent with probability $P_{ii} = 0.025 = 250/n$. As a result, about 54% of the edges ended up between communities.

4.1.2 LFR model

The LFR (Lancichinetti, Fortunato, Radicchi) model (Lancichinetti & Fortunato, 2009; Lancichinetti *et al.*, 2008) generates networks with communities, and at the same time it allows for the heterogeneity in the distributions of both node degrees and of community sizes. As a result, it became a standard and extensively used method for generating artificial networks. The original model (Lancichinetti *et al.*, 2008) generates undirected graphs, but it was soon after generalized to directed and weighted graphs (Lancichinetti & Fortunato, 2009). The model has various parameters: the number of nodes n , the mixing parameter μ that controls the fraction of edges that are

between communities, power law exponent γ for the degree distribution, power law exponent β for the distribution of community sizes, average degree d , and the maximum degree Δ .

For our experiments, we generated two families of directed graphs that we called **LFR** and **noisy-LFR**. To generate **LFR** graphs, we used $n = 10,000$, $\mu = 0.2$, $\gamma = 3$, $\beta = 2$, $d = 100$, and $\Delta = 500$, whereas for **noisy-LFR** we used $n = 10,000$, $\mu = 0.5$, $\gamma = 2$, $\beta = 1$, $d = 100$, and $\Delta = 500$.

4.1.3 ABCD model

In order to generate undirected graphs, we used an “LFR-like” random graph model, the Artificial Benchmark for Community Detection (ABCD graph) (Kamiński et al., 2021) that was recently introduced and implemented⁵, including a fast implementation that uses multiple threads (ABCDe)⁶. Undirected variant of LFR and ABCD produce graphs with comparable properties, but ABCD/ABCDe is faster than LFR and can be tuned to allow the user to make a smooth transition between the two extremes: pure (independent) communities and random graph with no community structure.

For our experiments, we generated **ABCD** graphs on $n = 10,000$ nodes, $\xi = 0.2$ (the counterpart of $\mu \approx 0.194$ in LFR), $\gamma = 3$, $\beta = 2$, $d = 8.3$, and $\Delta = 50$. The number of edges generated was $m = 41,536$, and the number of communities was $\ell = 64$.

4.1.4 EU email communication network

We also used the real-world network that was generated using email data from a large European research institution (Paranjape et al., 2017). The network is made available through Stanford Network Analysis Project (Leskovec & Sosič, 2016)⁷. Emails are anonymized, and there is an edge between u and v if person u sent person v at least one email. The dataset does not contain incoming messages from or outgoing messages to the rest of the world. More importantly, it contains “ground-truth” community memberships of the nodes indicating which of 42 departments at the research institute individuals belong to. As a result, this dataset is suitable for experiments aiming to detect communities, but we ignore this external knowledge in our experiments.

The associated **EMAIL** directed graph consists of $n=1,005$ nodes, $m = 25,571$ edges, and $\ell = 42$ communities.

4.1.5 College Football network

In order to see the framework “in action,” in Section 4.3, we performed an illustrative experiment with the well-known College Football real-world network with known community structures. This graph represents the schedule of U.S. football games between Division IA colleges during the regular season in Fall 2000 (Girvan & Newman, 2002). The associated **FOOTBALL** graph consists of 115 teams (nodes) and 613 games (edges). The teams are divided into conferences containing 8–12 teams each. In general, games are more frequent between members of the same conference than between members of different conferences, with teams playing an average of about seven intra-conference games and four inter-conference games in the 2000 season. There are a few exceptions to this rule, as detailed in Lu et al. (2018): one of the conferences is really a group of independent teams, one conference is really broken into two groups, and three other teams play mainly against teams from other conferences. We refer to those as outlying nodes.

4.2 Node embeddings

As mentioned in the introduction, there are over 100+ node embedding algorithms for undirected graphs. There are also some algorithms explicitly designed for directed graphs, or that can handle

both types of graphs, but their number is much smaller. We selected two of them for our experiments, **Node2Vec** and **HOPE**, but there are more to choose from. Embeddings were produced for 16 different dimensions between 2 and 32 (with a step of 2).

4.2.1 Node2Vec

Node2Vec⁸ (Grover & Leskovec, 2016) is based on random walks performed on the graph, an approach that was successfully used in Natural Language Processing. In this embedding algorithm, biased random walks are defined via two main parameters. The return parameter (p) controls the likelihood of immediately revisiting a node in the random walk. Setting it to a high value ensures that we are less likely to sample an already visited node in the following two steps. The in-out parameter (q) allows the search to differentiate between inward and outward nodes, so we can smoothly interpolate between breadth-first search (BFS) and depth-first search (DFS) exploration. We tested three variants of parameters p and q : ($p = 1/9, q = 9$), ($p = 1, q = 1$), and ($p = 9, q = 1/9$).

4.2.2 HOPE

HOPE (High-Order Proximity preserved Embedding) (Ou *et al.*, 2016) is based on the notion of asymmetric transitivity, for example, the existence of several short directed paths from node u to node v makes the existence of a directed edge from u to v more plausible. The algorithm learns node embeddings as two concatenated vectors representing the source and the target roles. HOPE can also be used for undirected graphs, in which case the source and target roles are identical, so only one is retained. Four different high-order proximity measures can be used within the same framework. For our experiments, we used three of them: Katz, Adamic-Adar, and personalized PageRank (denoted in our experiments as *katz*, *aa* and, *ppr*, respectively).

4.2.3 A few other ones

Similarly to HOPE, **APP** (Asymmetric Proximity Preserving) (Zhou, 2021) is also using asymmetric transitivity and is based on directed random walks and preserved rooted page rank proximity between nodes. It learns node embedding as a concatenation of two vectors representing the node's source and target roles. **NERD** (Node Embeddings Respecting Directionality) (Khosla *et al.*, 2019) also learns two embeddings for each node as a source or a target, using alternating random walks with starting nodes used as a source or a target; this approach can be interpreted as optimizing with respect to first-order proximity for three graphs: source-target (directed edges), source-source (pointing to common nodes), and target-target (pointed to by common nodes). Other methods for directed graphs try to learn node embeddings as well as some directional vector field. For example, **ANSE** (Asymmetric Node Similarity Embedding) (Dernbach & Towsley, 2020) uses skip-gram-like random walks, namely forward and reverse random walks, to limit dead-end issues. It also has an option to embed on a hypersphere. In another method that also try to learn a directional vector field (Perrault-Joncas & Meila, 2011), a similarity kernel on some learned manifold is defined from two components: (i) a symmetric component, which depends only on distance, and (ii) an asymmetric one, which depends also on the directional vector field. The algorithm is based on asymptotic results for (directed graph) Laplacians embedding, and the directional vector space can be decoupled.

4.3 Illustration of the framework

In order to illustrate the application of the framework, we ran the two embedding algorithms (**HOPE** and **Node2Vec**) in different dimensions and sets of parameters on the **FOOTBALL** graph. For each combination of the parameters, an embedding was produced and assessed with

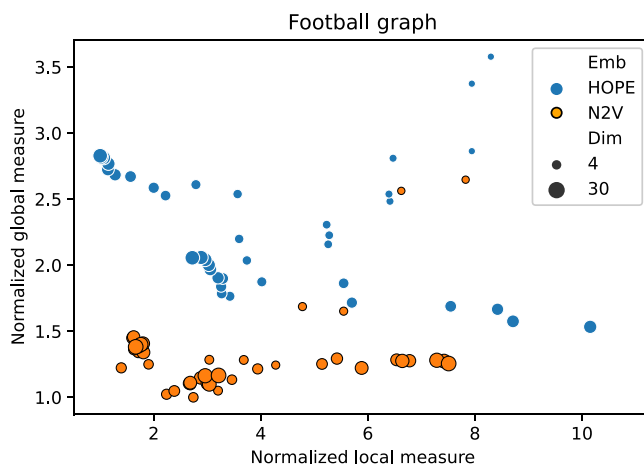


Figure 1. Normalized global and local scores for FOOTBALL graph.

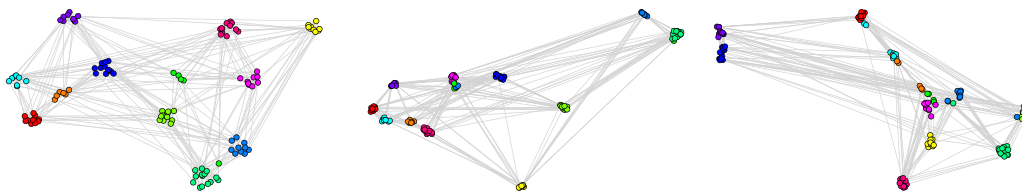


Figure 2. Two-dimensional projections of embeddings of FOOTBALL graph according to the framework: the best (left), the worst with respect to the local score (middle), and the worst with respect to the global score (right).

our framework. Local and global scores were normalized, as explained in Section 2.3, and are presented in Figure 1. In particular, good embeddings are concentrated around the auxiliary point (1, 1) that corresponds to the embedding that is the best from both local and global perspective. (Such embedding might or might not exist.)

We recommend to select the best embedding by careful investigation of both scores but, alternatively, one can ask the framework to decide which embedding to use in an unsupervised fashion. In order to do it, one may combine the global and the local scores, as explained in Equation (3), to make a decision based on a single combined divergence score. This way we identified the best and the worst embeddings. To visualize the selected embeddings in high dimensions, we needed to perform dimension reduction that seeks to produce a low-dimensional representation of high-dimensional data that preserve relevant structure. For that purpose, we used the Uniform Manifold Approximation and Projection (UMAP)⁹ (McInnes et al., 2018), a novel manifold learning technique for dimension reduction. UMAP is constructed from a theoretical framework based in Riemannian geometry and algebraic topology; it provides a practical scalable algorithm that applies to real-world datasets. The results are presented in Figure 2. The embedding presented on the left-hand side shows the winner that seems to not only separate nicely the communities but also puts pairs of communities far from each other if there are few edges between them; as a result, both scores are good. The embedding in the middle separates communities, but the nodes within communities are clumped together resulting in many edges that are too short. There are also a lot of edges that are too long. As a result, the local score for this embedding is bad. Finally, the embedding on the right has a clear problem with distinguishing communities, and some of them are put close to each other resulting in a bad global score.

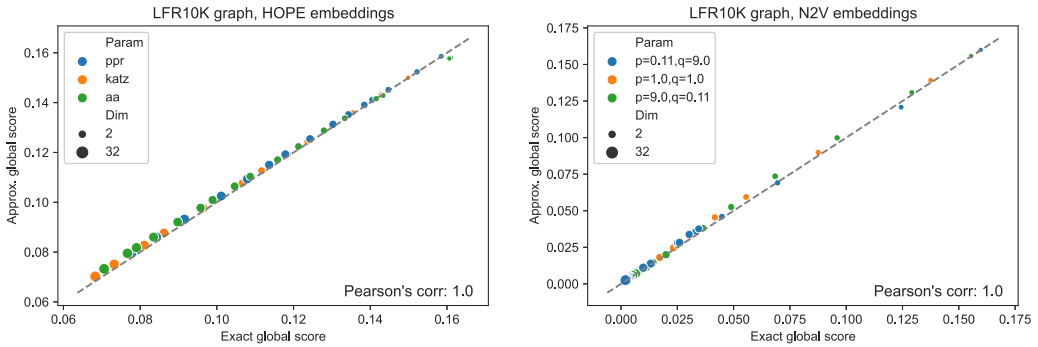


Figure 3. Approximated vs. exact global scores for LFR graphs and HOPE (left) and Node2Vec (right) embeddings.

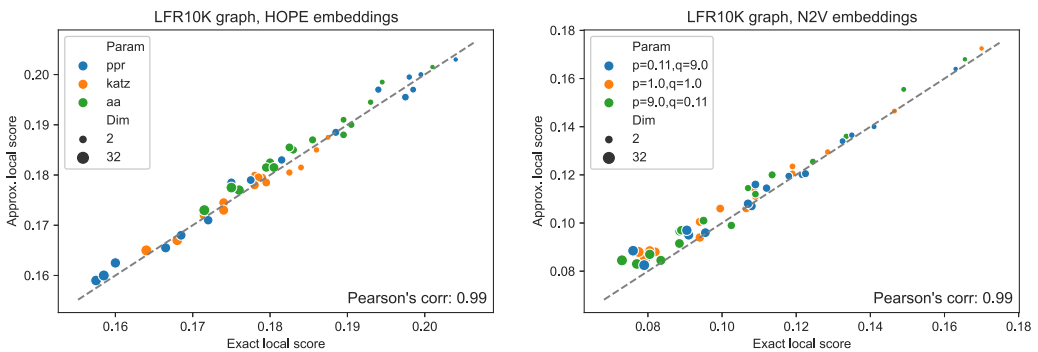


Figure 4. Approximated vs. exact local scores for LFR graphs and HOPE (left) and Node2Vec (right) embeddings.

4.4 Approximating the two scores

Let us start with an experiment testing whether our scalable implementation provides a good approximation of both measures, the density-based (global) score $\Delta_{\mathcal{E}}^g(G)$ and the link-based (local) score $\Delta_{\mathcal{E}}^l(G)$. We tested **SBM**, **LFR**, and **noisy-LFR** graphs with the two available embeddings (using the parameters and selected dimensions as described above). The number of landmarks was set to be five times larger than the number of communities in each graph, namely **SBM** graph used $30 \times 5 = 150$ landmarks, **LFR** had $75 \times 5 = 375$ landmarks, and for **noisy-LFR** the number of landmarks was set to $54 \times 5 = 270$.

Figures 3 and 4 present the results of experiments for **LFR** graph and both embedding algorithms. (Experiments for other graphs can be found in the supplementary document.) The first figure shows how well the global score is approximated by a scalable algorithm and the second figure concentrates on the local score. Recall that graphs on 10,000 nodes are, by default, the smallest graphs for which the framework uses scalable algorithm. The default setting in the framework is to use at least 4 landmarks per community and at least $4\sqrt{n}$ of them overall, so 400 landmarks for graphs of size 10,000 and many more for larger graphs. In our experiment, we see good results with even fewer landmarks. This was done in purpose to test approximation precision in a challenging corner case. As expected, global properties reflected by the global score are easier to approximate than local properties investigated by the local score that are more sensitive to small perturbations. Nevertheless, both scores are approximated to a satisfactory degree. The Pearson correlation coefficients for both **HOPE** and **Node2Vec** are close to 1 for the global score and roughly 0.99 for the local score. For other standard correlation measures, see Tables 1 and 2.

Table 1. Correlation between approximated and (exact) global scores

Graph embedding	Pearson	Spearman	Kendall-Tau
SBM10K-HOPE	0.98	0.98	0.94
SBM10K-N2V	1.0	1.0	0.97
LFR10K-HOPE	1.0	1.0	0.99
LFR10K-N2V	1.0	1.0	0.99
nLFR10K-HOPE	1.0	1.0	1.0
nLFR10K-N2V	1.0	0.96	0.87

Table 2. Correlation between approximated and (exact) local scores

Graph embedding	Pearson	Spearman	Kendall-Tau
SBM10K-HOPE	0.99	0.99	0.92
SBM10K-N2V	0.99	0.97	0.87
LFR10K-HOPE	0.99	0.99	0.93
LFR10K-N2V	0.99	0.98	0.90
nLFR10K-HOPE	1.0	1.0	0.98
nLFR10K-N2V	1.0	0.99	0.94

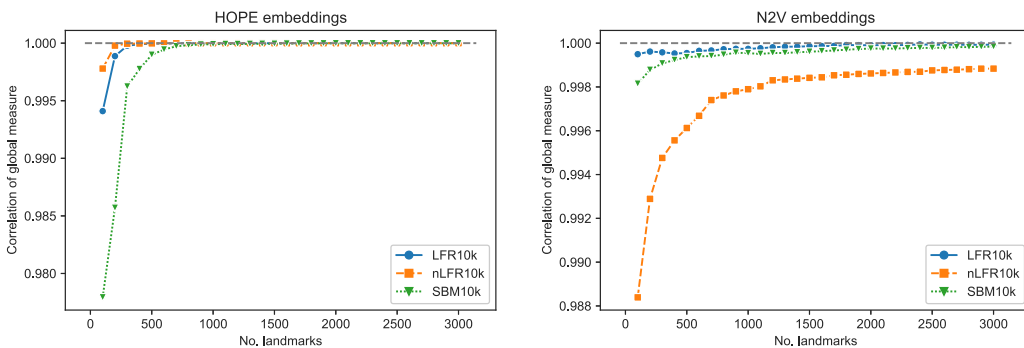


Figure 5. Pearson’s correlation between approximated and exact global scores for SMB, LFR, noisy-LFR graphs, and HOPE (left) and Node2Vec (right) embeddings.

Finally, we checked if the number of landmarks used by the framework as a default value (namely, $4\sqrt{n}$) is a good choice. We see in Figures 5 and 6 that the approximation quickly stabilizes, and so there is no need for a large number of landmarks to get the desired approximation. As commented earlier, the local score is more challenging to approximate (see Figure 6 again), and the Pearson correlation between the approximated and the exact local scores does not seem to tend to 1 quickly but it is very close to 1, providing a satisfactory precision.

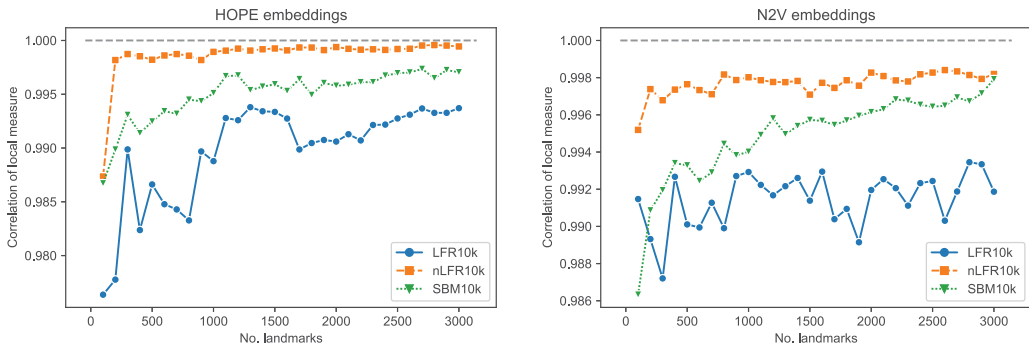


Figure 6. Pearson's correlation between approximated and exact local scores for SMB, LFR, noisy-LFR graphs, and HOPE (left) and Node2Vec (right) embeddings.

4.5 Node classification and community detection vs. the global score

It is expected and desired that the global score measures how useful a given embedding is for any application that requires a good understanding of a global structure of the network. In our next experiments, we tested two such applications: node classification and community detection.

Node classification task aims to train a model to learn to which class a node belongs to. Typically, the goal is to label each node with a categorical class (binary classification or multi-class classification) or to predict a continuous number (regression). The process is supervised in nature, that is, the model is trained using a subset of nodes that have ground-truth labels.

For each graph (**SBM**, **LFR**, **noisy-LFR**, and **EMAIL**), each of the two embeddings (**HOPE** and **Node2Vec**) was used as an input for community detection task based on XGBoost model. XGBoost¹⁰ is an open-source software library that provides a regularizing gradient boosting framework, the algorithm of choice for many winning teams of machine learning competitions. Since we aim to have a fair benchmark evaluating if an embedding extracts any useful global properties of the network instead of optimizing the quality of the outcome, we used vanilla XGBoost with default hyperparameters. For each embedding, we conducted 10 independent repetitions of the standard training process. First, nodes with the corresponding embedding was randomly partitioned into a training and a test set (25% of all observations). Then, XGBoost model was trained on the training set together with the corresponding labels indicated the ground-truth community the nodes belong to. Finally, the model was used to predict the communities on the test set, and the accuracy was reported.

Similarly, in order to investigate how much of the community structure got preserved by the evaluated embeddings, each embedding was independently clustered 20 times using the classic *k*-means algorithm, a well-known method that aims to partition *n* vectors into *k* clusters in which each vector belongs to the cluster with the nearest mean (center of mass). As before, since we aim for a fair and easy benchmark rather than a carefully tuned specific approach, we simply used the vanilla *k*-means algorithm and the value of *k* set to the true number of communities. The AMI score was then calculated based on the two partitions of the set of nodes: the clusters assignment returned by the *k*-means algorithm and the ground-truth communities. This approach is similar to the one used in the recent paper (Tandon *et al.*, 2021) in which the authors investigate if embeddings can be used to detect communities.

The results of both experiments are presented on Figure 7 (the accuracy score for node classification task is presented on the left side, whereas the AMI score for community detection task can be found on the right side). The embeddings that are identified by the framework as good (left-bottom corner, close to the auxiliary point (1, 1) representing the hypothetical perfect score) tend to perform well in both applications (large balls representing large values of the accuracy/AMI).

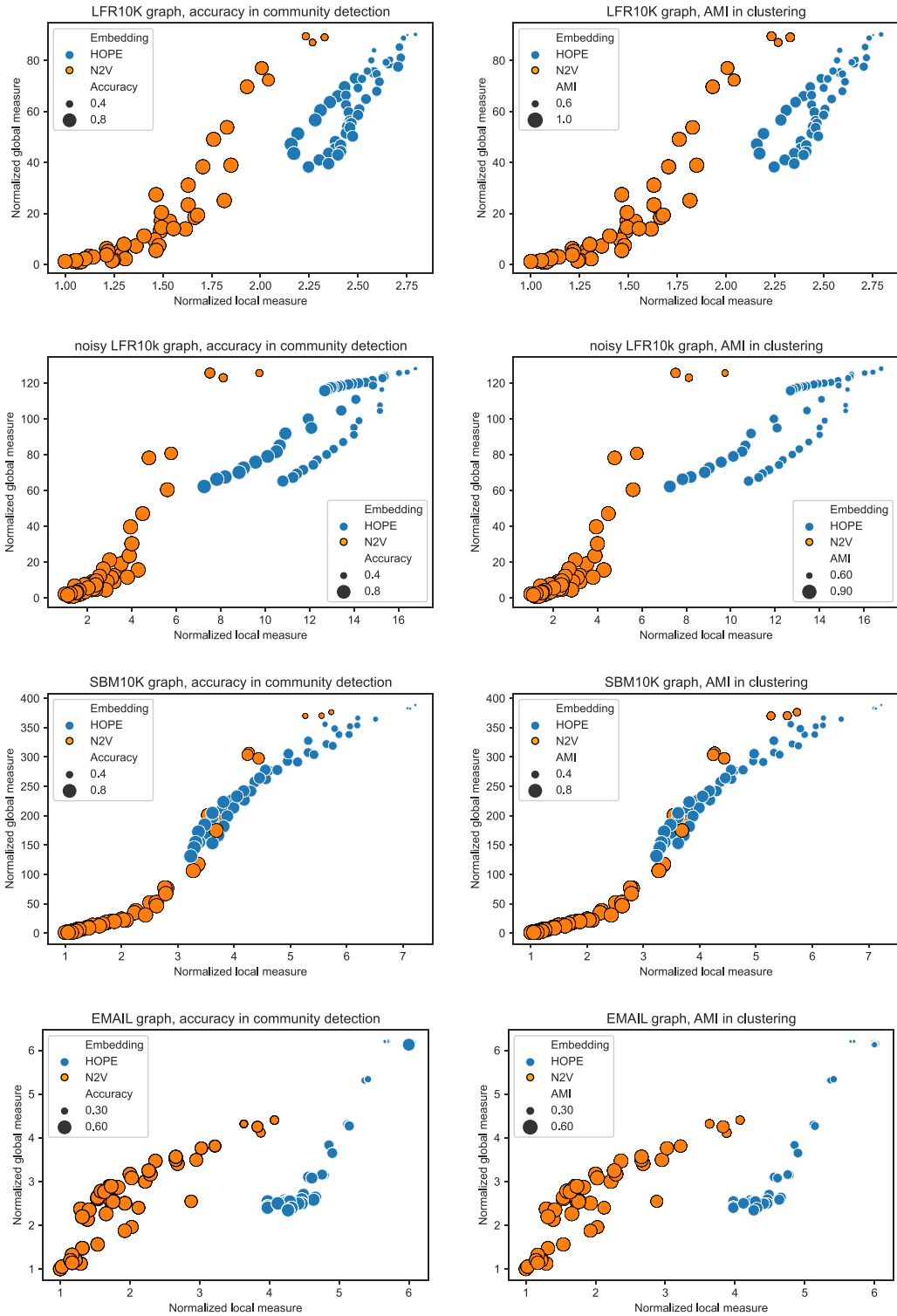


Figure 7. Global and local scores with accuracy (left) and AMI (right) overlay for SBM, LFR, noisy-LFR, EMAIL graphs, and HOPE and Node2Vec embeddings.

Table 3. Correlation coefficients between the accuracy scores in node classification task and global/local scores (averaged over all parameters)

Graph embedding	Pearson		Spearman		Kendall-Tau	
	Global	Local	Global	Local	Global	Local
SBM10K-HOPE	-0.97	-0.99	-1.0	-1.0	-0.97	-0.96
SBM10K-N2V	-0.9	-0.82	-0.57	-0.57	-0.42	-0.42
LFR10K-HOPE	-0.96	-0.93	-1.0	-0.97	-0.97	-0.89
LFR10K-N2V	-0.8	-0.71	-0.25	-0.23	-0.16	-0.14
NLFR10K-HOPE	-0.98	-0.98	-1.0	-1.0	-0.96	-0.96
NLFR10K-N2V	-0.91	-0.85	-0.14	-0.1	-0.08	-0.04
EMAIL-HOPE	-0.97	-0.68	-0.9	-0.72	-0.77	-0.61
EMAIL-N2V	-0.86	-0.87	-0.69	-0.69	-0.54	-0.54

Table 4. Correlation coefficients between the AMI scores in community detection task and global/local scores (averaged over all parameters)

Graph embedding	Pearson		Spearman		Kendall-Tau	
	Global	Local	Global	Local	Global	Local
SBM10K-HOPE	-0.98	-0.99	-1.0	-1.0	-0.97	-0.95
SBM10K-N2V	-0.93	-0.85	-0.91	-0.91	-0.8	-0.81
LFR10K-HOPE	-0.98	-0.96	-1.0	-0.97	-0.96	-0.89
LFR10K-N2V	-0.82	-0.73	-0.76	-0.75	-0.6	-0.59
NLFR10K-HOPE	-0.99	-0.99	-0.99	-0.99	-0.95	-0.95
NLFR10K-N2V	-0.92	-0.86	-0.4	-0.39	-0.29	-0.29
EMAIL-HOPE	-0.86	-0.46	-0.62	-0.48	-0.54	-0.42
EMAIL-N2V	-0.86	-0.87	-0.77	-0.80	-0.61	-0.65

On the other hand, poorly scored embeddings (top-right corner) perform poorly (small balls). The same conclusion is obtained after more rigorous investigation of the correlation coefficients between the accuracy/AMI and the global/local scores (see Table 3 and, respectively, Table 4). Let us note that the rank-based correlations (Spearman and Kendall-Tau) report a smaller correlation for **Node2Vec** embeddings as in our experiment we generated many embeddings that were comparable and of very good quality. As a result, their rankings with respect to the global/local scores and the accuracy/AMI might not be exactly the same but, in any case, the framework was able to distinguish good embeddings from bad ones. More experiments can be found in the supplementary document. In particular, since the accuracies and the AMI scores are not always easy to compare on Figure 7, we present them directly as functions of local and global scores.

4.6 Link prediction vs. the local score

As discussed in the previous section, the global score returned by the framework captures how well the embedding preserves global properties of embedded networks. Similarly, it is expected and desired that the local score evaluates the power of embeddings to encapsulate local properties. In the next experiment, we tested one local algorithm, namely, link prediction algorithm.

Indeed, most of the methods for directed graph embeddings are validated via link prediction (Zhou et al., 2017), where one deletes a subset of edges (typically randomly selected) and then measures how well they can be recovered. A commonly used scoring method is the AUC obtained from ranking of node pairs from most to least likely to have an edge between them. A variation of that approach is known as node recommendation, where one removes some outgoing edges for a subset of nodes and try to recover the most likely missing neighbors. Another score that may potentially be used is to compare precision and recall among the top- k candidate node pairs for a specific k . This is useful, for example, if one has a limited “budget” to investigate if a given pair of nodes are actually linked by an edge or not, so one can only test a small number of pairs.

In our experiments, for each of the four graphs we tested (**SBM**, **LFR**, **noisy-LFR**, and **EMAIL**), given graph G , we randomly selected 5% of its edges and removed them to form a graph G' . Then we took another random sample of nonadjacent pairs of nodes in G , set E' . Both classes had the same number of pairs of nodes so that the test set created in such a way was balanced. Our goal was to train a model that uses one of the embeddings of graph G' to detect which pairs of nodes in $E \cup E'$ are adjacent in G . We repeated this process independently five times (with five different seed values) and reported the average AUC scores.

For each of the two algorithms (**HOPE** and **Node2Vec**), each combination of their parameters and 16 dimensions we tested, we produced an embedding of graph G' . For evaluation purpose, we computed both the global and the local divergence scores for the produced embedding. In order to create a training set for the classifier, we considered all pairs of adjacent nodes in G' (the positive class) as well as a random subset of pairs of nonadjacent nodes (the negative class) that is of the same size as the number of edges in G' (as for the test sets, to keep both classes balanced). Finally, we concatenated embeddings of pairs of nodes representing edges/non-edges into feature vectors to be used for prediction. Such training set was used to train XGBoost model with default hyperparameter values. As already mentioned, in order to measure the quality of this simple model, we computed the AUC score that provides a measure of separability, as it tells us how capable the model is of distinguishing between the two classes, edges E and non-edges E' , both coming from the original graph G . (Note that some pairs of nodes in the test set might overlap with pairs of nodes in the training set. Even more, some positive pair in the test set might be negative in the training one.)

The results of this experiment are presented on Figure 8. The embeddings that score well by the framework (left-bottom corner of the plots, near to the auxiliary point (1, 1)) turn out to perform well in predicting links (large balls representing large values of AUC), and the opposite is true for the embeddings that are identified as poor ones (right-top corner of the plots). More rigorous approach can be found in Table 5 where the correlation coefficients between the AUC and the global/local scores are reported. As before, since there are many **Node2Vec** embeddings that are almost indistinguishable, ranking-based correlation coefficients report weaker correlation. The framework has no chance to predict the exact ranking of the embeddings based on their global/local scores, but it is clearly able to identify good embeddings and separate them from bad ones. As usual, more experiments can be found in the supplementary document, including counterpart of Figure 8 presenting the AUC score as functions of local and global scores.

4.7 More challenging situations

Based on experiments in Section 4.5, we see that both global (as expected) and local (far from being obvious) scores correlate well with the quality of both node classification and

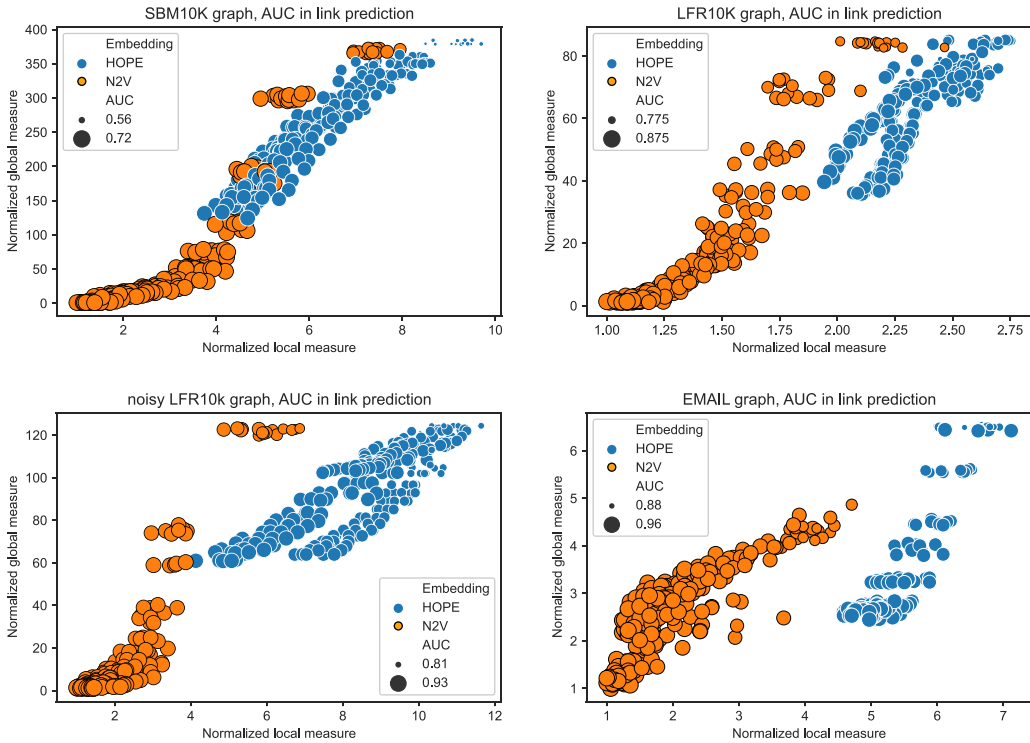


Figure 8. Global and local scores with AUC overlay for SBM, LFR, noisy-LFR, EMAIL graphs, and HOPE and Node2Vec embeddings.

community detection. Similarly, experiments in Section 4.6 imply that both scores correlate with the quality of link prediction algorithm (this time, we expect this from the local score but not necessarily from the global one). We see such behavior as the two scores often correlate with one another. Indeed, for a relatively easy graph to deal with, a good quality embedding should be able to preserve both local and global properties of the network. On the other hand, poor quality embeddings (e.g., when the dimension is too small or the parameters are not properly selected) typically fail to produce anything useful, from any of the two perspectives. Our synthetic models and **EMAIL** graph seem to confirm this. However, it is expected and plausible that large, real-world graphs are more challenging to deal with. For such graphs, it might be difficult if not impossible to find an embedding that scores well from both perspectives. In such situations, one needs to make a decision which embedding to use based on a specific application at hand (using either global or local properties). In order to illustrate such situations, we experimented with an undirected **ABCD** graph and 32-dimensional **Node2Vec** embedding (with parameters set to $p = q = 1$).

First, we rewired a specified fraction p of edges within each community. The rationale behind it is that such operation should destroy local properties of the embedding (many edges within communities are now long with respect to the associated embedding and some pairs of close nodes are not adjacent), but global properties should remain unchanged.

We independently generated five graphs with the following rewiring fractions: $p = 0.2$, $p = 0.4$, $p = 0.6$, $p = 0.8$, and $p = 1.0$; note that $p = 0$ corresponds to the original **ABCD** graph. As expected, the global score and the quality of both node classification and community detection algorithms remain the same. More importantly, the local score increases (the quality of the embedding gets worse) and the quality of link prediction algorithm (AUC) gets worse—see Figure 9 (left).

Table 5. Correlation coefficients between the AUC scores in link prediction task and global/local scores (averaged over all parameters)

Graph embedding	Pearson		Spearman		Kendall-Tau	
	Global	Local	Global	Local	Global	Local
SBM10K-HOPE	-0.92	-0.95	-0.99	-0.96	-0.91	-0.83
SBM10K-N2V	-0.81	-0.73	-0.3	-0.31	-0.21	-0.22
LFR10K-HOPE	-0.94	-0.91	-0.99	-0.95	-0.94	-0.82
LFR10K-N2V	-0.92	-0.9	-0.91	-0.87	-0.82	-0.74
NLFR10K-HOPE	-0.96	-0.87	-0.97	-0.9	-0.89	-0.73
NLFR10K-N2V	-0.91	-0.95	-0.96	-0.9	-0.86	-0.74
EMAIL-HOPE	-0.9	-0.49	-0.75	-0.43	-0.58	-0.35
EMAIL-N2V	-0.8	-0.86	-0.65	-0.76	-0.48	-0.57

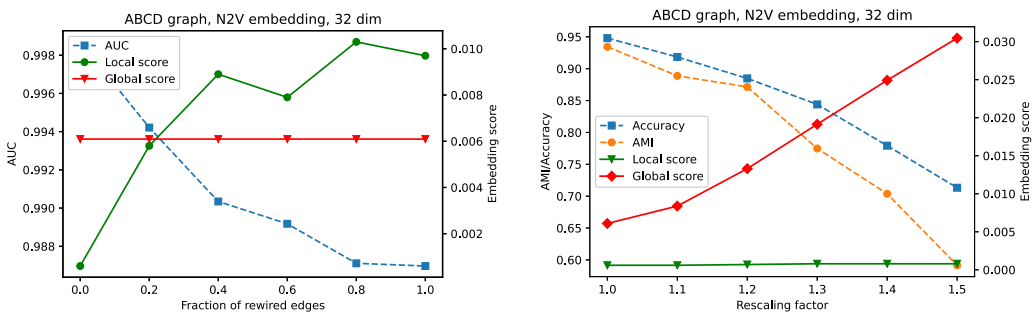


Figure 9. Comparison of local/global scores (and associated quality measures for various tasks) for rewired graphs (left) and rescaled embeddings (right).

In order to test the opposite scenario, we kept the original **ABCD** graph but slightly modified the original embedding. Each community was independently rescaled by a factor of $q > 1$. Formally, all points in the embedding that are associated with nodes that belong to a given community increased their distance from the center of mass of this community while keeping the direction from it. The idea behind this approach is that, on average, the distance between nodes from different communities (most pairs of nodes) remain the same, and the distance between nodes within one community is only slightly increased (small fraction of pairs of nodes anyway). As a result, local properties should be preserved almost as well as in the original embedding. On the other hand, inflated communities tend to overlap more in the embedded space than before and, as a result, the global quality measure should decrease.

We independently generated five embeddings with the following rescaling factors: $q = 1.1$, $q = 1.2$, $q = 1.3$, $q = 1.4$, and $q = 1.5$; note that $q = 1$ corresponds to the original embedding. As expected, the local score and the quality of link prediction algorithm remain the same. More importantly, the global score increases (the quality of the embedding gets worse), and the quality of both node classification (accuracy) and community detection (AMI) get worse—see Figure 9 (right).

The conclusion can be summarized as follows. Global score might not be able to detect problems with a given embedding if these problems are local by nature. For example, despite the fact

that an embedding properly captures between-community distances, it may inaccurately reflect relationships between nodes within communities. On the other hand, local score might fail to capture the fact that in the embedding, communities (treated globally as a cloud of points) are not appropriately separated. For this reason, although in many cases global and local scores are consistent, they are in general not equivalent and it is useful to calculate them both and investigate carefully.

5. Conclusions

In this paper, we introduced a framework that assigns two scores (global and local) for embeddings of a given graph. The two scores measure the ability of analyzed embeddings to preserve global and, respectively, local properties of the embedded graph. The code is written in Julia (high-level, high-performance, and dynamic programming language) and available online¹¹. The framework is easy to use (knowledge of Julia is not required to use it; default parameters should be suitable for most cases) and flexible (more advanced users may modify parameters, if needed).

If the embeddings need to be used for unsupervised tasks, the framework is able to identify the best embedding in an unsupervised way via combined divergence score. However, the user might still want to consider a few embeddings that score well from both perspectives and, if possible, select an embedding generated by a simple/explainable algorithm. In particular, there is usually no need to use very high-dimensional embeddings if the improvement is marginal. On the other hand, if the embedding is to be used for supervised tasks, the best course of action might be to consider a number of top-scoring embeddings and select the winner by performing a standard supervised selection for a given application at hand.

Let us also discuss some potential future directions. In Dehghan-Kooshkghazi *et al.* (2020), some throughout analysis of the original framework for undirected graphs (and using only the global score) was performed. A natural next step is to test in more detail the current version of the framework (using the two scores) on both real-world and synthetic networks. More importantly, it would be valuable to show the predictive power and usefulness of the framework for a large dataset and some real-world important application. The industry partners we collaborate with provide a positive feedback so far, but having a publishable results of experiments performed on publicly available datasets will be of high value.

When analyzing the correlation between the link prediction algorithm and the local score (Section 4.6), we independently performed a few more experiments investigating the reason why embeddings are able to predict missing links. In particular, we created two additional training sets. The first training set was the same as the original one but with five additional columns: in-degree and out-degree for both nodes as well as the distance between them. The second training set consisted with only these five additional columns. The quality of models obtained by using both such training sets turned out to be comparable to the original one we used for our tests. As expected, the richer set typically gave slightly better quality results but not always. This implies that the embeddings encoded the most important information about the graph via the distances between nodes (or XGBoost was not able to extract more from other sources). This suggests that the GCL model should be able to accurately model the shape of the embedded networks as they require exactly the five columns as the parameters/input. We plan to use the GCL model, along with graph-based communities and other properties such as transitive closure, to build a better quality link prediction algorithm. Another potential and important application is to detect overlapping communities or to detect anomalies. In both cases, the GCL model equipped with a good embedding would be the heart of such an algorithm.

Competing interests. None.

Supplementary materials For supplementary material for this article, please visit <http://doi.org/10.1017/nws.2022.27>

Notes

- 1 <https://github.com/KrainskiL/CGE.jl>
- 2 <https://github.com/ftheberge/Comparing/Graph/Embeddings>
- 3 <https://www.soscip.org/>
- 4 <https://github.com/KrainskiL/UnsupervisedFrameworkForComparingGraphEmbeddings>
- 5 <https://github.com/bkamins/ABCDGraphGenerator.jl/>
- 6 <https://github.com/tolcz/ABCDeGraphGenerator.jl/>
- 7 <https://snap.stanford.edu/data/email-Eu-core.html>
- 8 <https://github.com/eliorc/node2vec>
- 9 <https://github.com/lmcinnes/umap>
- 10 <https://github.com/dmlc/xgboost>
- 11 <https://github.com/KrainskiL/CGE.jl>

References

- Aggarwal, M., & Murty, M. N. (2021). *Machine learning in social networks: Embedding nodes, edges, communities, and graphs*. Springer Nature.
- Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), P10008.
- Chen, H., Yang, C., Zhang, X., Liu, Z., Sun, M., & Jin, J. (2021). From symbols to embeddings: A tale of two representations in computational social science. *Journal of Social Computing*, 2(2), 103–156.
- Chung, F., Chung, F. R., Graham, F. C., Lu, L., & Chung, K. F. (2006). *Complex graphs and networks (No. 107)*. American Mathematical Society.
- Dehghan-Kooshkghazi, A., Kamiński, B., Krański, L., Prałat, P., & Théberge, F. (2020). Evaluating node embeddings of complex networks. *Journal of Complex Networks*, 10(4), cnac0302022.
- Dernbach, S., & Towsley, D. (2020). Asymmetric node similarity embedding for directed graphs. In *Complex networks XI* (pp. 83–91). Cham: Springer.
- Expert, P., Evans, T. S., Blondel, V. D., & Lambiotte, R. (2011). Uncovering space-independent communities in spatial networks. *Proceedings of the National Academy of Sciences*, 108(19), 7663–7668.
- Funke, T., & Becker, T. (2019). Stochastic block models: A comparison of variants and inference methods. *PLoS One*, 14(4), e0215296.
- Gastner, M. T., & Newman, M. E. (2006). The spatial structure of networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 49(2), 247–252.
- Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12), 7821–7826.
- Grover, A., & Leskovec, J. (2016, August). node2vec: scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 855–864).
- Hoff, P. D., Raftery, A. E., & Handcock, M. S. (2002). Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460), 1090–1098.
- Holland, P. W., Laskey, K. B., & Leinhardt, S. (1983). Stochastic blockmodels: First steps. *Social Networks*, 5(2), 109–137.
- Janssen, J. (2010, June). Spatial models for virtual networks. In *Conference on computability in Europe*. Berlin, Heidelberg: Springer (pp. 201–210).
- Kamiński, B., Prałat, P., & Théberge, F. (2020). An unsupervised framework for comparing graph embeddings. *Journal of Complex Networks*, 8(5), cnz043.
- Kamiński, B., Prałat, P., & Théberge, F. (2020, September). A scalable unsupervised framework for comparing graph embeddings. In *International workshop on algorithms and models for the web-graph* (pp. 52–67). Cham: Springer.
- Kamiński, B., Prałat, P., & Théberge, F. (2021). Artificial benchmark for community detection (ABCD)—Fast random graph model with community structure. *Network Science*, 9(2), 1–26.
- Kamiński, B., Prałat, P., & Théberge, F. (2022). *Mining complex networks*. New York: Chapman and Hall/CRC.
- Khosla, M., Leonhardt, J., Nejdil, W., & Anand, A. (2019, September). Node representation learning for directed graphs. In *Joint european conference on machine learning and knowledge discovery in databases*. Cham: Springer (pp.395–411).
- Krioukov, D. (2016). Clustering means geometry in networks. In *APS March meeting abstracts*, vol. 2016, (pp. Y12–005).
- Lancichinetti, A., & Fortunato, S. (2009). Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1), 016118.
- Lancichinetti, A., Fortunato, S., & Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4), 046110.
- Leskovec, J., & Sosič, R. (2016). Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIIST)*, 8(1), 1–20.

- Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1), 145–151.
- Lu, Z., Wahlström, J., & Nehorai, A. (2018). Community detection in complex networks via clique conductance. *Scientific Reports*, 8(1), 1–16.
- McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv: 1802.03426.
- Ou, M., Cui, P., Pei, J., Zhang, Z., & Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1105–1114).
- Paranjape, A., Benson, A. R., & Leskovec, J. (2017). Motifs in temporal networks. In *Proceedings of the tenth ACM international conference on web search and data mining* (pp. 601–610).
- Perrault-Joncas, D., & Meila, M. (2011). Directed graph embedding: An algorithm based on continuous limits of laplacian-type operators. *Advances in Neural Information Processing Systems*, 24, 990–998.
- Poulin, V., & Théberge, F. (2018, December). Ensemble clustering for graphs. In *International conference on complex networks and their applications*. Cham: Springer (pp.231–243).
- Tandon, A., Albeshri, A., Thayanathan, V., Alhalabi, W., Radicchi, F., & Fortunato, S. (2021). Community detection in networks using graph embeddings. *Physical Review E*, 103(2), 022316.
- Zhou, T. (2021). Progresses and challenges in link prediction. arXiv preprint arXiv: 2102.11472.
- Zhou, C., Liu, Y., Liu, X., Liu, Z., & Gao, J. (2017). Scalable graph embedding for asymmetric proximity. In *Proceedings of the AAAI conference on artificial intelligence*, 31(1).
- Zuev, K., Boguná, M., Bianconi, G., & Krioukov, D. (2015). Emergence of soft communities from geometric preferential attachment. *Scientific Reports*, 5(1), 1–9.