A Broader Picture of Random-walk Based Graph Embedding

Zexi Huang
University of California
Santa Barbara, CA, USA
zexi_huang@cs.ucsb.edu

Arlei Silva
Rice University
Houston, TX, USA
arleilps@gmail.com

Ambuj Singh
University of California
Santa Barbara, CA, USA
ambuj@cs.ucsb.edu

ABSTRACT

Graph embedding based on random-walks supports effective solutions for many graph-related downstream tasks. However, the abundance of embedding literature has made it increasingly difficult to compare existing methods and to identify opportunities to advance the state-of-the-art. Meanwhile, existing work has left several fundamental questions—such as how embeddings capture different structural scales and how they should be applied for effective link prediction—unanswered. This paper addresses these challenges with an analytical framework for random-walk based graph embedding that consists of three components: a random-walk process, a similarity function, and an embedding algorithm. Our framework not only categorizes many existing approaches but naturally motivates new ones. With it, we illustrate novel ways to incorporate embeddings at multiple scales to improve downstream task performance. We also show that embeddings based on autocovariance similarity, when paired with dot product ranking for link prediction, outperform state-of-the-art methods based on Pointwise Mutual Information similarity by up to 100%.

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**; • **Information systems** → **Social networks**.

KEYWORDS

Graph representation learning; Node embedding; Random-walk

ACM Reference Format:
Zexi Huang, Arlei Silva, and Ambuj Singh. 2021. A Broader Picture of Random-walk Based Graph Embedding. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467300

1 INTRODUCTION

Random-walk based graph embedding enables the application of classical algorithms for high-dimensional data to graph-based downstream tasks (e.g., link prediction, node classification, and community detection). These embedding methods learn vector representations for nodes based on some notion of topological similarity (or proximity). Since DeepWalk [45], we have witnessed a great interest in graph embedding both by researchers and practitioners. Several regular papers [8, 10, 11, 19, 26, 32, 42, 47, 48, 51, 56, 59, 63, 66] and a few surveys [7, 9, 14, 25, 28] have attempted to not only advance but also consolidate our understanding of these models. However, the abundance of literature also makes it increasingly difficult for practitioners and newcomers to the field to compare existing methods and to contribute with novel ones.

On the other hand, despite the rich literature, several fundamental questions about random-walk based graph embedding still remain unanswered. One such question is *(Q1) how do embeddings capture different structural scales?* Random-walks of different lengths are naturally associated with varying scales [16]. However, downstream task performance of embeddings has been shown to be insensitive to random-walk lengths [26, 45]. Another relevant question is *(Q2) how should random-walk embeddings be used for link prediction?* Following node2vec [26], several works train classifiers to predict missing links based on a set of labelled pairs (edges and non-edges) [30, 40, 62]. This is counter-intuitive given that the embedding problem is often defined in terms of dot products. In fact, dot products are sometimes also applied for link prediction and for the related task of network reconstruction, where the entire graph is predicted based on the embeddings [25, 42, 61, 65].

With these questions in mind, we shall take a closer look at how embeddings are produced in random-walk based methods. It starts with selecting a *random-walk process*. DeepWalk [45] applies standard random-walks, while node2vec [26] considers biased random-walks and APP [66] adopts rooted PageRank, among others. Then, a *similarity function* maps realizations of the random-walk process into real values that represent some notion of node proximity. Most existing methods rely on the skip-gram language model [39], which has been shown to capture the Pointwise Mutual Information (PMI) similarity [36]. Finally, an *embedding algorithm* is used to generate vector representations that preserve the similarity function via optimization. This can be based on either sampled random-walks and gradient descent (or one of its variations) [26, 45], or (explicit) matrix factorization (e.g., Singular Value Decomposition) [47, 48].

This breakdown of random-walk based embedding methods allows us to build a simple, yet powerful analytical framework with three major components: *random-walk process*, *similarity function*, and *embedding algorithm*. As shown in Figure 1, our framework both categorizes existing approaches and facilitates the exploration of novel ones. For example, we will consider embeddings based on the autocovariance similarity, which has been proposed for multiscale community detection [15, 16], and compare it against the more popular PMI similarity on multiple downstream tasks.

Our framework also provides tools to answer the aforementioned questions. For Q1, we will not only illustrate how past work implicitly combines multiple scales and *its implications to node-level tasks*, but also propose novel ways to incorporate scales to *improve edge-level task performance*. To answer Q2, we will show that to

[redacted] *18, 2021, Virtual Event, Singapore.*

ACM ISBN 978-1-4503-8332-5/21/08.
https://doi.org/10.1145/3447548.3467300

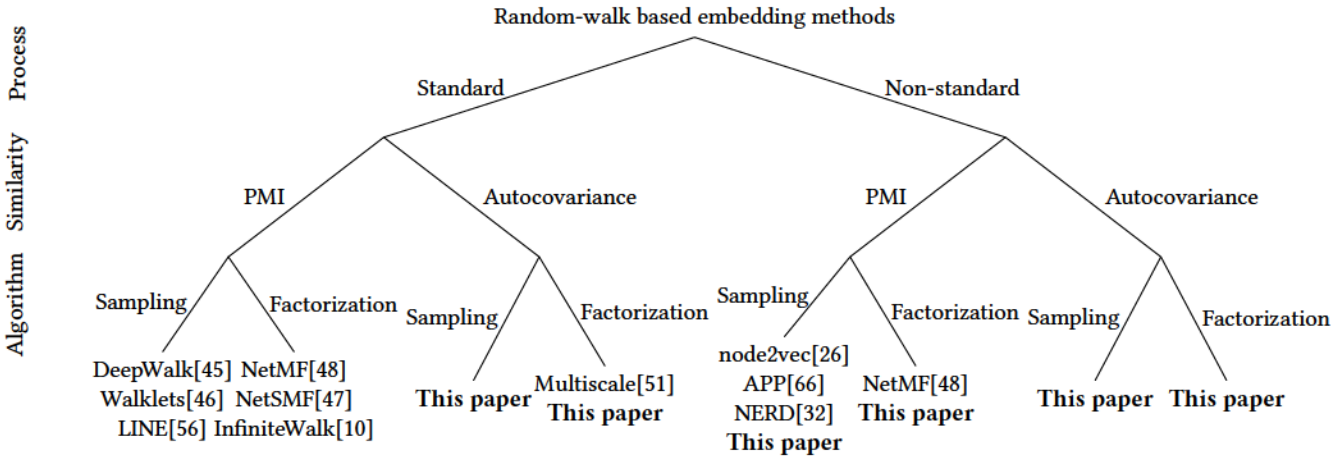

Figure 1: Different random-walk based embedding methods (old and new) classified according to our analytical framework—with process, similarity, and algorithm as main components. A key contribution of this paper is to integrate autocovariance as a similarity metric and show that it outperforms Pointwise Mutual Information (PMI) in link prediction.

optimize performance, *embedding methods should be designed with task settings in mind*. Specifically, we find that embeddings based on autocovariance, when paired with dot products, lead to a two-fold improvement over existing methods. Our analysis shows the reason to be that the particular combination enables the embedding to capture heterogeneous degree distributions [4] in real graphs. One could argue that link prediction is the most relevant downstream task for (positional) node embeddings, as graph neural networks often outperform embedding approaches in node classification [53].

To summarize, the main contributions of our paper are:

- We present a unified view of random-walk based graph embedding, incorporating different random-walk processes, similarity functions, and embedding algorithms.
- We show how autocovariance can be applied as a similarity function to create novel embeddings that outperform state-of-the-art methods using PMI by up to 100%.
- We illustrate ways to exploit the multiscale nature of random-walk similarity to further optimize embedding performance.
- We conduct an extensive experimental evaluation of our framework on various downstream tasks and provide theoretical insights behind the results.

2 METHOD

Consider an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \dots n\}$ denotes the set of n nodes and $\mathcal{E}$ denotes the set of m edges. The graph is represented by a weighted symmetric adjacency matrix $A \in \mathbb{R}^{n \times n}$, with $A_{uv} > 0$ if an edge of weight A_{uv} connects nodes u and v, and $A_{uv} = 0$, otherwise. The (weighted) degree of node u is defined as $\deg(u) = \sum_v A_{uv}$.

A node embedding is a function $\phi : \mathcal{V} \mapsto \mathbb{R}^d$ that maps each node v to a d-dimensional ($d \ll n$) vector $\mathbf{u}_v$. We refer to the embedding matrix of $\mathcal{V}$ as $U = (\mathbf{u}_1, \dots, \mathbf{u}_n)^T \in \mathbb{R}^{n \times d}$. For some embedding algorithms, another embedding matrix $V \in \mathbb{R}^{n \times d}$ is also generated. Random-walk based embedding methods use a random-walk process to embed nodes u and v such that a similarity metric is preserved by dot products $\mathbf{u}_u^T \mathbf{u}_v$ (or $\mathbf{u}_u^T \mathbf{v}_v$). In the next section, we will formalize random-walks on graphs.

2.1 Random-walk Process

A random-walk is a Markov chain over the set of nodes $\mathcal{V}$. The transition probability of the walker jumping to node v is based solely on its previous location u and is characterized by the adjacency matrix A. For the *standard random-walk process*, the transition probability is proportional to the edge weight A_{uv}:

$$p(x(t+1)=v | x(t)=u) = \frac{A_{uv}}{\deg(u)} \tag{1}$$

where $x(t) \in \mathcal{V}$ is the location of the walker at time t.

Transition probabilities between all pairs of nodes are represented by a transition matrix $M \in \mathbb{R}^{n \times n}$:

$$M = D^{-1} A \tag{2}$$

where $D = \text{diag}([\deg(1), \dots, \deg(n)]) \in \mathbb{R}^n$ is the degree matrix.

For a connected non-bipartite graph, the standard random-walk is ergodic and has a unique stationary distribution $\pi \in \mathbb{R}^n$:

$$\pi_u = \frac{\deg(u)}{\sum_v \deg(v)} \tag{3}$$

Standard random-walks provide a natural way to capture node neighborhood in undirected connected graphs. One can also design *biased random-walks* to explore different notions of neighborhood [26]. For directed graphs, a *PageRank process* [43] is often applied in lieu of standard random-walks to guarantee ergodicity.

2.2 Similarity Function

A node similarity metric is a function $\varphi : \mathcal{V} \times \mathcal{V} \mapsto \mathbb{R}$ that maps pairs of nodes to some notion of topological similarity. Large positive values mean that two nodes are similar to each other, while large negative values indicate they are dissimilar.

Random-walk based similarity functions are based on co-visiting probabilities of a walker. An important property of random-walks that has been mostly neglected in the embedding literature is their ability to capture similarity at different structural scales (e.g., local vs. global). This is achieved via a Markov time parameter $\tau \in \mathbb{Z}_+$, which corresponds to the distance between a pair of nodes in a walk in terms of jumps. One of the contributions of this paper is to show the effect of different Markov time scales on the embedding and ways to exploit them to optimize downstream task performance.

We will describe two random-walk based similarity functions in this section, *Pointwise Mutual Information (PMI)* and *autocovariance*. PMI has become quite popular in the graph embedding literature due to its (implicit) use by word2vec [36, 39] and DeepWalk [45, 48]. On the other hand, autocovariance is more popular in the context of multiscale community detection [15, 16]. As one of the contributions of this paper, we will demonstrate the effectiveness of autocovariance based embeddings on edge-level tasks.

2.2.1 Pointwise Mutual Information (PMI). Denote $X_v(t) \in \{0, 1\}$ as the event indicator of $x(t) = v$, which can be true (1) or false (0). The PMI between events $X_u(t) = 1$ and $X_v(t + \tau) = 1$ is defined as:

$$\begin{aligned} R_{uv}(\tau) &= \mathrm{PMI}(X_u(t) = 1, X_v(t + \tau) = 1) \\ &= \log \frac{p(X_u(t) = 1, X_v(t + \tau) = 1)}{p(X_u(t) = 1)p(X_v(t + \tau) = 1)} \end{aligned} \tag{4}$$

PMI provides a non-linear information theoretic view of random-walk based proximity. For an ergodic walk with the stationary distribution π as starting probabilities, PMI can be computed based on π and the τ-step transition probability:

$$R_{uv}(\tau) = \log(\pi_u p(x(t+\tau)=v|x(t)=u)) - \log(\pi_u \pi_v) \tag{5}$$

where $R_{uv}(\tau) \in [-\infty, -\log(\pi_v)]$. In matrix form:

$$R(\tau) = \log(\Pi M^\tau) - \log(\pi\pi^T) \tag{6}$$

where $\Pi = \mathrm{diag}(\pi) \in \mathbb{R}^{n\times n}$ and $\log(\cdot)$ is the element-wise logarithm. The PMI matrix is symmetric due to the time-reversibility of undirected random-walks.

It is noteworthy that LINE [56] and DeepWalk [45]—two random-walk embedding methods—implicitly factorize PMI, as follows:

$$R_{LINE} = R(1) - \log b \tag{7}$$

$$R_{DW} = \log\left(\frac{1}{T}\sum_{\tau=1}^{T}\exp(R(\tau))\right) - \log b \tag{8}$$

where b and T are the number of negative samples and context window size, respectively, and $\exp(\cdot)$ is the elementwise exponential. The proof of these facts is included in the Appendix. As we see, LINE preserves the PMI similarity at Markov time 1, while DeepWalk factorizes the *log-mean-exp*—a smooth approximation of the average—of the PMI matrices from Markov time 1 to T.

2.2.2 Autocovariance. The autocovariance is defined as the covariance of $X_u(t)$ and $X_v(t + \tau)$ within a time interval τ:

$$\begin{aligned} R_{uv}(\tau) &= \mathrm{cov}(X_u(t), X_v(t + \tau)) \\ &= \mathrm{E}\left[(X_u(t) - \mathrm{E}[X_u(t)])\,(X_v(t + \tau) - \mathrm{E}[X_v(t + \tau)])\right] \end{aligned} \tag{9}$$

The value of $R_{uv}(\tau)$ is a linear measure of the joint variability of the walk visiting probabilities for nodes u and v within time τ. Similar to PMI, for an ergodic walk and starting probabilities π:

$$R_{uv}(\tau) = \pi_u p(x(t+\tau)=v|x(t)=u) - \pi_u\pi_v \tag{10}$$

where $R_{uv}(\tau) \in [-\pi_u\pi_v, \pi_u(1 - \pi_v)]$. In the matrix form:

$$R(\tau) = \Pi M^\tau - \pi\pi^T \tag{11}$$

The autocovariance (Equation 11) and PMI (Equation 6) matrices share a very similar form, differing only by the logarithm operation. However, this distinction has broad implications for graph embedding. PMI is a non-linear metric closely related to the *sigmoid function*, which is a quite popular activation in deep learning, while autocovariance is related to a *piecewise linear function*—see Section 2.3.2 for details. We will compare PMI and autocovariance in multiple tasks in Section 3 and provide theoretical and empirical insights on their different performance in Section 4.

2.3 Embedding Algorithm

The goal of an embedding algorithm is to generate vector representations that preserve a given similarity metric:

$$\begin{aligned} U^* &= \arg\min_U \sum_{u,v} (\mathbf{u}_u^T\mathbf{u}_v - R_{uv})^2 \\ &= \arg\min_U \|UU^T - R\|_F^2 \end{aligned} \tag{12}$$

where $\mathbf{u}_u^T\mathbf{u}_v$ captures similarity in the embedding space, R is a similarity matrix, and $\|\cdot\|_F$ is the Frobenius norm.

In the following sections, we discuss two different techniques to optimize this embedding objective.

2.3.1 Matrix factorization. Matrix factorization is an explicit way to optimize the embedding. It generates a low-rank approximation of R as UU^T with $\mathrm{rank}(UU^T) = d$. Because R is symmetric, from the Eckart-Young-Mirsky theorem [21], the optimal $U^* = Q_d\sqrt{\Lambda_d}$, where $R = Q\Lambda Q^T$ is the Singular Value Decomposition (SVD) of R. Notice that, different from classical spectral approaches [12], factorization-based embedding is not based on the graph Laplacian.

Direct SVD of R has a complexity of $O(n^3)$, which is infeasible for most applications. However, scalable factorization is possible for sparse graphs. For autocovariance, one can apply the Lanczos method [35], which only requires sparse matrix-vector multiplications. This approach reduces the complexity to $O(nd^2 + md\tau)$. For PMI, as discussed in [47], one can construct the spectral sparsifier of the similarity matrix and apply Randomized SVD [27]. The resulting complexity for this method is $O(\bar{m}\tau \log n + \bar{m}d + nd^2 + d^3)$, where $\bar{m} = O(m\tau)$ is the number of non-zeros in the sparsifier.

2.3.2 Sampling. Sampling-based algorithms produce embeddings that implicitly optimize Equation 12 by maximizing the likelihood of a corpus $\mathcal{D}$ generated based on samples from the process. A sample i is a random-walk sequence $\langle v_1^{(i)}, v_2^{(i)}, \ldots v_L^{(i)}\rangle$ of length L with the initial node $v_1^{(i)}$ selected according to a distribution $p(v)$—we will assume that $p(v) = \pi_v$ for the remainder of this section. From each sample, we extract pairs $(v_t^{(i)}, v_{t+\tau}^{(i)})$, where τ is the Markov time scale. Thus, $\mathcal{D}$ is a multiset of node pairs (u, v).

Different from matrix factorization, sampling algorithms produce two embedding matrices, U and V, the source and target embeddings, respectively. The use of two embeddings was initially

proposed by word2vec [39] and is considered a better alternative. We will focus our discussion on algorithms that exploit *negative sampling*, with b negative samples, to efficiently generate embeddings. Let z be a random variable associated with pairs (u, v) such that $z = 1$ if (u, v) appears in $\mathcal{D}$ and $z = 0$, otherwise. The log-likelihood ℓ of the corpus $\mathcal{D}$ can be expressed as:

$$\ell = \sum_{u,v} \#(u,v)(\log(p(z=1|u,v)) + b\mathbb{E}_{w\sim\pi} \log(p(z=0|u,w))) \quad (13)$$

where $\#(u, v)$ is the number of occurrences of the pair (u, v) in $\mathcal{D}$.

The form of the conditional probability function $p(z|u, v)$ is determined by the similarity function. For instance, in the case of PMI, $p(z|u, v)$ is known to take the form of the *sigmoid* function: $\sigma(x) = 1/(1 + \exp(x))$ [48]. More specifically, $p(z = 1|u, v) = \sigma(\mathbf{u}_u^T \mathbf{v}_v)$ and $p(z = 0|u, v) = \sigma(-\mathbf{u}_u^T \mathbf{v}_v)$. Here, we show how the objective in Equation 13 can be maximized for the autocovariance similarity. First, we define the following conditional probability function:

$$p(z = 1|u, v) = \rho\left(\frac{\mathbf{u}_u^T \mathbf{v}_v + \pi_u \pi_v}{\mathbf{u}_u^T \mathbf{v}_v + (b+1)\pi_u \pi_v}\right) \quad (14)$$

$$p(z = 0|u, v) = \rho\left(\frac{\pi_u \pi_v}{\mathbf{u}_u^T \mathbf{v}_v + (b+1)\pi_u \pi_v}\right) \quad (15)$$

where $\rho(x)$ is a *piecewise linear activation* with $\rho(x) = 0$, if $x < 0$, $\rho(x) = x$, if $0 \le x \le 1$, and $\rho(x) = 1$, otherwise.

The following theorem formalizes the connection between the above defined conditional probability and autocovariance:

Theorem 2.1. *For a large enough dimensionality d ($d = \Omega(n)$), the embedding that maximizes Equation 13 with a conditional probability given by Equation 14 and Equation 15 is such that:*

$$\mathbf{u}_u^T \mathbf{v}_v = \frac{1}{b}\pi_u p(x(t+\tau)=v|x(t)=u)) - \pi_u \pi_v \quad (16)$$

The proof of the theorem is provided in the Appendix. While here we only show the sampling-based algorithm for autocovariance, previous work has given a similar proof for PMI [48].

We minimize the likelihood from Equation 13 using gradient descent. The time complexity of the sampling algorithms (for PMI and autocovariance) is $O(|\mathcal{D}|b)$, where $|\mathcal{D}|$ is the size of the corpus and b is the number of negative samples—in practice, $|\mathcal{D}| = O(n)$.

3 EXPERIMENTS

In this section, we evaluate several random-walk based graph embedding methods defined according to our analytical framework on various downstream tasks and datasets.

Table 1: An overview of the datasets.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	labels
BlogCatalog	10,312	333,983	interests
Airport	3,158	18,606	countries/continents
Wiki-words	4,777	92,157	tags
PoliticalBlogs	1,222	16,717	ideologies
Cora	23,166	91,500	categories/subcategories
Wiki-fields	10,675	137,606	fields/subfields

3.1 Dataset

We apply six datasets (see Table 1) in our experiments.

- BlogCatalog [57]: Undirected social network of bloggers with (multi) labels representing topics of interest.
- Airport: Flight network among global airports (from OpenFlights [44]). Edges are weighted by the number of flights through the corresponding air routes. Labels represent the countries and continents where airports are located. The largest undirected connected component of the network is used in our experiments.
- Wiki-words [38]. Undirected co-occurrence network of words in the first million bytes of the Wikipedia dump. Labels are Part-of-Speech (POS) tags inferred by the Stanford POS-Tagger.
- PoliticalBlogs [1]. Network of hyperlinks between weblogs on US politics, recorded in 2005. Labels represent political ideologies (conservative vs liberal). The largest undirected connected component of the network is used in our experiments.
- Cora [55]: Directed citation network with categories (e.g., AI) and subcategories (e.g, Knowledge Representation) as labels.
- Wiki-fields: Subset of the Wikipedia directed hyperlink network [3] covering Computer Science, Mathematics, Physics and Biology for the year 2019. Fields and subfields are used as labels.

3.2 Experiment Setting

We evaluate embedding methods on three downstream tasks:

- Node classification. We follow the same procedure as [45]. For each dataset, we randomly sample a proportion of node labels for training and the rest for testing. We use one-vs-rest logistic regression implemented by LIBLINEAR [22] for multi-class classification (Airport and Cora) and multi-label classification (BlogCatalog, Wiki-words, and Wiki-fields). To avoid the thresholding effect [58] in the multi-label setting, we assume that the number of true labels for each node is known. Performance is reported as average *Micro-F1* and *Macro-F1* [60] for 10 repetitions.
- Link prediction. We randomly remove 20% of edges while ensuring that the residual graph is still connected and embed the residual graph. Edges are predicted as the top pairs of nodes ranked by either dot products of embeddings or classification scores from a logistic regression classifier with the concatenation of embeddings as input. We report *precision@k* [37] as the evaluation metric and also use *recall@k* in our analysis, where k is the number of top pairs, in terms of the ratio of removed edges.
- Community detection. We use k-means—with k-means++ initialization [2]—for community detection, with the number of clusters set to be the actual number of communities. Normalized Mutual Information (*NMI*) [54] between predicted and true communities is used as the evaluation metric.

For all experiments, the number of embedding dimensions is set to $d = 128$. When searching for the best Markov time, we sweep τ from 1 to 100. An implementation of our framework is available at https://github.com/zexihuang/random-walk-embedding.

3.3 Results

In this section, we evaluate how different components of the embedding methods affect their performance.

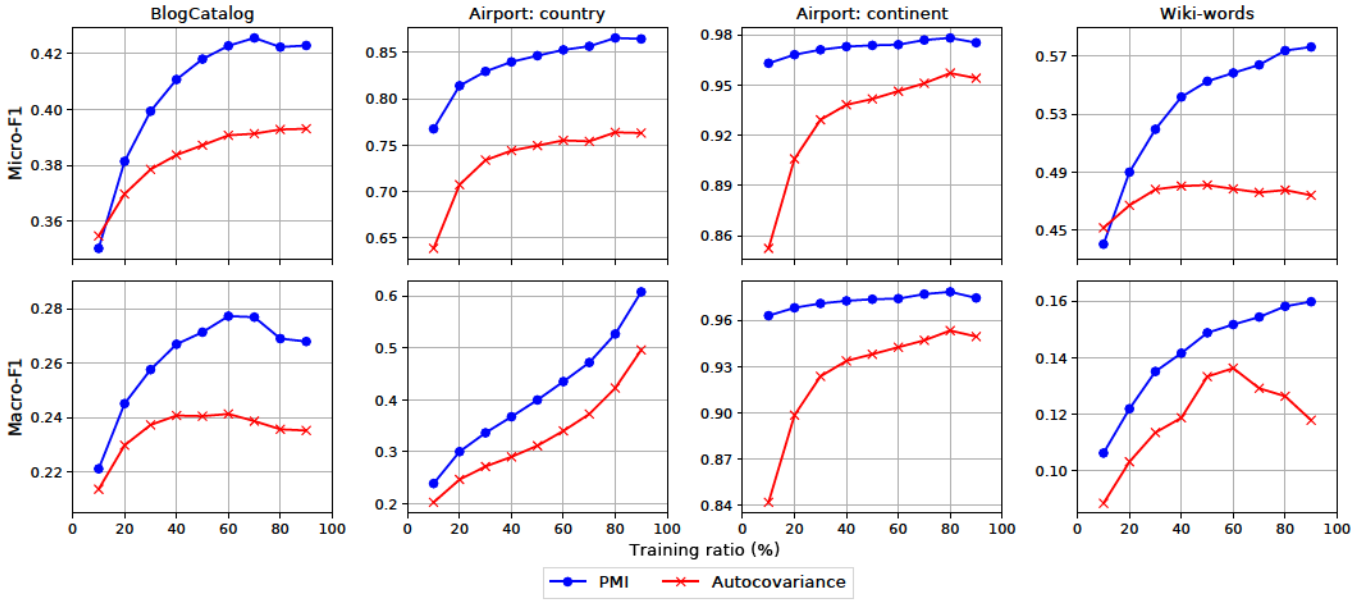

Figure 2: Node classification performance comparison between PMI and autocovariance on varying training ratios. PMI consistently outperforms autocovariance in all datasets.

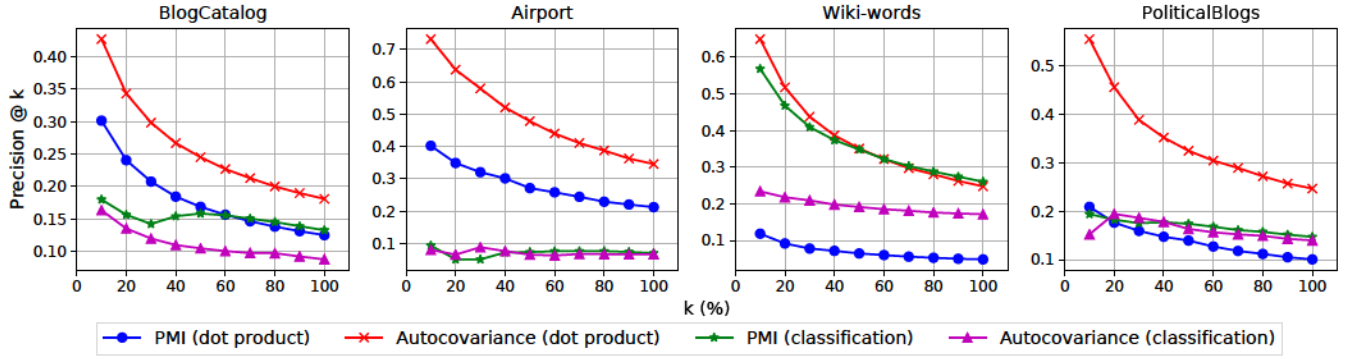

Figure 3: Link prediction performance comparison between PMI and autocovariance on varying percentages of top predictions (k). Autocovariance with dot product ranking consistently outperforms PMI (with either ranking scheme) in all datasets.

3.3.1 PMI vs autocovariance. We apply standard random-walks and the matrix factorization algorithm and analyze the difference between PMI and autocovariance similarity.

Figure 2 shows the node classification performance on undirected datasets (excluding PoliticalBlog as it is a simple binary classification task). We select the Markov time τ with the best performance for the 50% training ratio. Results show that PMI consistently outperforms autocovariance for both *Micro-F1*/*Macro-F1* scores. The average gain is 6.0%/11.3% for BlogCatalog, 14.2%/24.5% and 4.6%/5.2% for Airport with country and continent labels, and 12.9%/20.0% for Wiki-words. This is a piece of evidence that PMI, which is non-linear, is more effective at node-level tasks.

Figure 3 shows the results for link prediction, where we select the best Markov time for $k = 100\%$. Autocovariance with dot product ranking consistently outperforms PMI with either ranking scheme in all datasets. The average gains over the best ones are 44.1% in BlogCatalog, 72.9% in Airport, 2.2% in Wiki-words, and 101.4% in PoliticalBlogs. To the best of our knowledge, we are the first to observe the effectiveness of autocovariance on edge-level tasks.

While both dot product-based and classification-based link prediction have been widely used in the embedding literature, our results show that dot product is a clearly superior choice for autocovariance embedding. It also leads to better or similar performance for PMI embedding for all but the Wiki-words dataset. Thus, we will only show results based on dot products in later experiments.

Results in this section beg the deeper question of why specific similarities and ranking schemes lead to better performance. We will provide theoretical and empirical insights on this in Section 4.

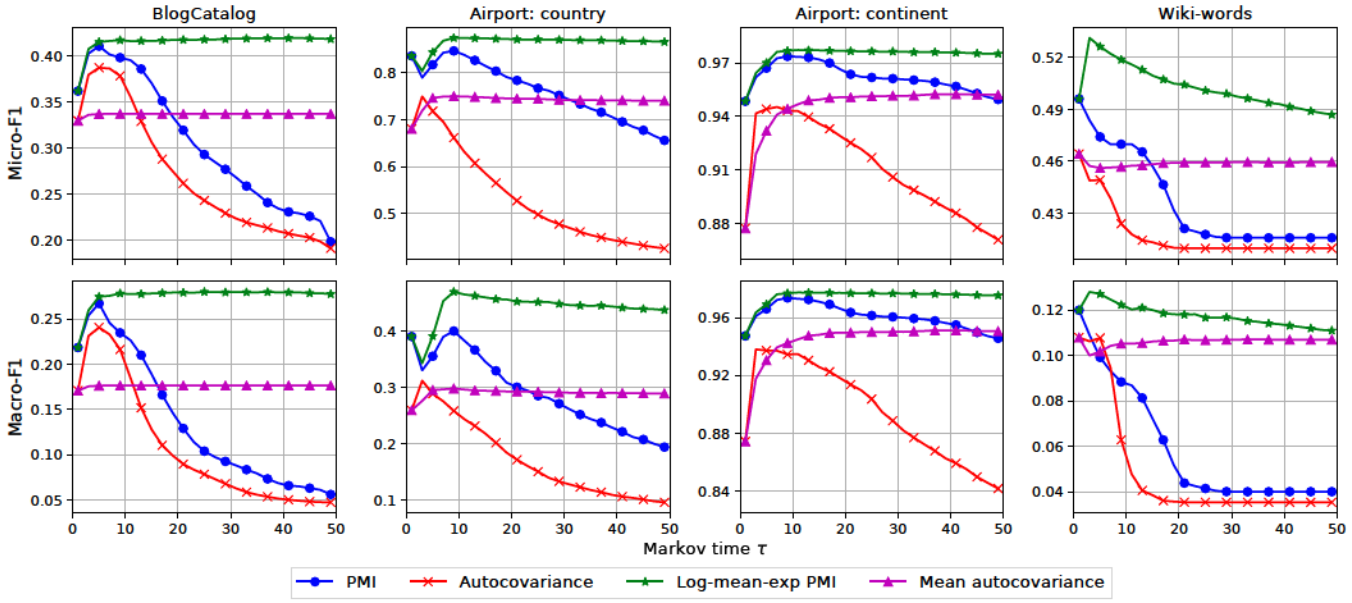

Figure 4: Node classification results for PMI, autocovariance, and their moving means on varying Markov times. While both means stabilize the performance for large Markov times, only *log-mean-exp* PMI consistently increases the peak performance.

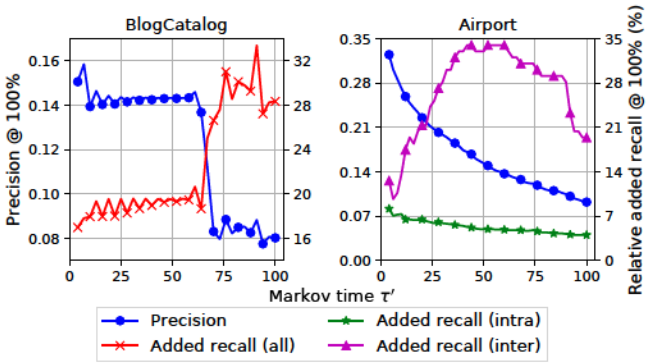

Figure 5: Relative added recall for link prediction after adding predictions from a larger Markov time τ' to the best Markov time τ^*. While the precision drops at larger Markov times, they predict a distinct set of true (inter-community) edges from those revealed at the best (and small) time.

3.3.2 Multiscale. In this section, we analyze the effect of different Markov time scales on the embedding performance using standard random-walks and matrix factorization.

Figure 4 compares the node classification performance for different similarity measures on varying Markov time scales from 1 to 50. The training ratio is fixed at 50% for all datasets. The metrics have a peak performance at a certain Markov time. However, notice that this aspect has not been given much relevance by previous work on embedding. This is a legacy of DeepWalk [45], which, as we have shown, implicitly applies the *log-mean-exp* of the PMI matrix within a range of Markov times. Thus, we also show the performance for the moving *log-mean-exp* PMI and mean autocovariance (it is linear) in Figure 4. Interestingly, while both mean versions have smoother results, only *log-mean-exp* PMI has a higher peak performance—with gains of 2.2%/4.8% for BlogCatalog, 3.3%/17.4% and 0.4%/0.4% for Airport with country and continent labels, and 7.1%/6.7% for Wiki-words. This shows *log-mean-exp* is indeed an effective approach to combine PMI similarity at different scales. Conversely, we cannot apply a similar strategy for autocovariance.

We observe similar results for community detection (see Figure 9 in the Appendix). Also included are results for Markov Stability [16], which applies clustered autocovariance for multiscale community detection. For both countries and continents in Airport, (*log-mean-exp*) PMI achieves the best performance. This is another piece of evidence for the effectiveness of PMI on node-level tasks.

We then evaluate the effect of different Markov time scales on link prediction using BlogCatalog and Airport. We first note that a moving mean does not improve the performance for either PMI or autocovariance (see Figure 10 in the Appendix). We hypothesize the reason to be that each edge plays a structural role that is specific to a few relevant scales (e.g., connecting two *mid-sized* communities). To validate it, we first find the best Markov time τ^* in terms of *precision@100%* for autocovariance. Then, for every Markov time τ' larger than τ^*, we compute its added *recall@100%*—i.e., the proportion of correctly predicted edges at τ' that are not predicted at τ^*. We show the relative gain of added recall compared to τ^* and the *precision*@100% for different values of τ' in Figure 5. For BlogCatalog, while precision drops as τ' increases, the relative added recall increases to up to 33.4% at $\tau' = 91$ (4,042 new edges added to the 12,104 edges correctly predicted at $\tau^* = 3$). To further

understand the roles of those edges, we show the relative added recall for intra-continent edges and inter-continent edges separately for Airport. Most of the gain is for inter-community edges (up to 34.0% at $\tau' = 42$). This observation suggests that link prediction can be improved by accounting for the scale of edges.

3.3.3 Factorization vs sampling. We now switch our focus to evaluating the performance of sampling and matrix factorization algorithms. For PMI, we apply a publicly available node2vec implementation[1] with parameters that make it equivalent to DeepWalk. For autocovariance, we implement our algorithm using the PyTorch framework with Adam optimizer [33]. Parameters are set based on [26]: 10 walks per node with length 80, 400 epochs for convergence, 1000 walks per batch, and 5 negative samples.

Figure 6 shows the link prediction performance where the context window size for both PMI and autocovariance is set to the best Markov time for *precision@100%*. We focus on link prediction because [48] has already shown evidence that matrix factorization is superior to sampling for node classification. Factorization achieves better performance for both datasets and similarity functions. The average gain of *precision@k* for PMI is 277.0%/553.7% on BlogCatalog/Airport, and 240.7%/121.5% for autocovariance.

Figure 13 (in the Appendix) shows how dot products of 16-D embeddings for Zachary's karate club generated using sampling and matrix factorization algorithms approximate the entries of similarity matrices. We notice that while embeddings produced by both sampling and factorization approaches are correlated with the similarities, matrix factorization achieves a higher correlation.

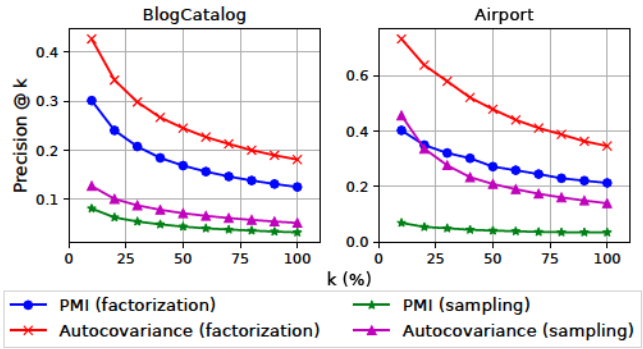

Figure 6: Link prediction performance for PMI and autocovariance using matrix factorization and sampling algorithms on varying percentages of top predictions (k). Factorization algorithms achieve the best performance.

3.3.4 Directed vs undirected. The final part of our evaluation compares embeddings for undirected and directed graphs, which requires different random-walk processes, as discussed in Section 2.1. The results for Cora and Wiki-fields are shown in Figures 11 and 12 (in the Appendix). Overall, directed embeddings outperform undirected ones for the very top ranked pairs in link prediction, while undirected embeddings are better in node classification.

[1] https://github.com/eliorc/node2vec/blob/master/node2vec/node2vec.py

4 INSIGHTS ON PMI VS AUTOCOVARIANCE

This section provides both theory and experiments supporting the differences in performance achieved by autocovariance and PMI. We will focus on link prediction, for which autocovariance was shown to achieve the best performance. Our analysis might benefit the design of novel embedding methods using our framework.

We will assume that the graph G has community structure [24] and heterogeneous degree distribution [4], which hold for many real graphs and for generalizations of Stochastic Block Models (SBM) [31]. Moreover, let the number of dimensions d be large enough to approximate the similarities well. For simplicity, we consider an SBM instance with two clusters and intra-cluster edge probability p significantly higher than the inter-cluster probability q.

We will use the dot product setting for link prediction [52]. Specifically, we estimate the probability of an edge as $P(e_{u,v}) \propto \max(0, \mathbf{u}^T\mathbf{v})$, where we have conveniently dropped the embedding subscripts. Further, for $p \gg q$, $\mathbf{u}^T\mathbf{v} > 0$ iff $C(u) = C(v)$, where $C(v)$ is the cluster v belongs to. We then have the following observations.

Observation 1. *Link prediction based on dot products correlates predicted node degrees and the 2-norms of embedding vectors:*

$$\widetilde{\deg}(u) \approx \sum_{v \in C(u)-\{u\}} \mathbf{u}^T\mathbf{v} = \|\mathbf{u}\|_2 \sum_{v \in C(u)-\{u\}} \|\mathbf{v}\|_2 \cos(\theta_{\mathbf{u},\mathbf{v}})$$

where $\cos(\theta_{u,v})$ *is the cosine of the angle between* $\mathbf{u}$ *and* $\mathbf{v}$.

The above property follows from the community structure, as the majority of the edges will be within communities. We now look at the norms of vectors generated by autocovariance and PMI.

Observation 2. *For autocovariance similarity:*

$$\|\mathbf{u}\|_2^2 = \pi(u)([M^\tau]_{u,u} - \pi(u)) = \frac{\deg(u)}{2m}\left([M^\tau]_{u,u} - \frac{\deg(u)}{2m}\right)$$

Norms for autocovariance depend on two factors. The first is proportional to the actual node degree, while the second expresses whether u belongs to a community at the scale τ. For SBM, there exists a τ such that $[M^\tau]_{u,u}$ is close to $\deg(u)/m$ in expectation. That is the case when the process is almost stationary within $C(u)$, with $m/2$ expected edges, but not in the entire graph. It implies that the embedding norms are proportional to actual node degrees. Combining this with Observation 1, we can conclude that autocovariance embedding predicts degrees related to the actual ones.

Observation 3. *For PMI similarity:*

$$\|\mathbf{u}\|_2^2 = \log\left(\pi(u)[M^\tau]_{u,u}\right) - \log\left(\pi^2(u)\right) = \log\left(\frac{2m[M^\tau]_{u,u}}{\deg(u)}\right)$$

Notice that as $[M^\tau]_{u,u}$ approaches to $\deg(u)/m$, $\|\mathbf{u}\|_2$ becomes constant. As a consequence, different from autocovariance, norms for PMI embeddings are not directly related with node degrees.

In Figure 7, we provide empirical evidence for Observations 2 and 3—the correlation between actual degrees and embedding norms for BlogCatalog and Airport. The correlation for autocovariance is significantly higher than that for PMI, showing the ability of autocovariance to capture heterogeneous degree distributions.

Figure 8 shows the predicted links for an instance of SBM ($p = 0.15$, $q = 0.02$) with two hubs—generated by merging 20 nodes inside each community. Visualization is based on t-SNE [29] projection from 16-D embeddings. Around half of the edges (97 out of top

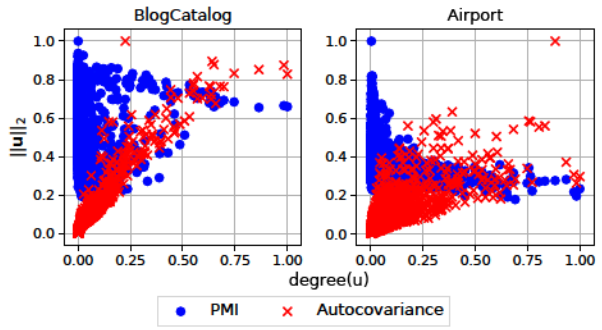

Figure 7: Correlation between (max normalized) degrees and 2-norms of embedding vectors based on PMI and autocovariance. Autocovariance produces embeddings with norms that are well correlated with node degrees in both datasets.

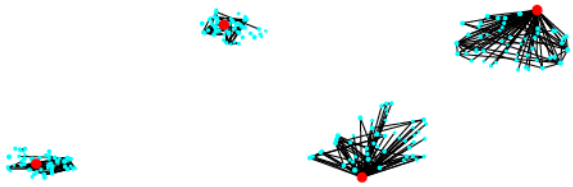

Figure 8: Edges predicted using PMI and autocovariance embeddings for a synthetic graph with community structure and hubs (one per community, shown in red). Autocovariance is more effective at capturing the hubs in the graph.

200) are predicted to be connected to the hubs using autocovariance, but none using PMI. This example shows how autocovariance with dot product ranking enables state-of-the-art link prediction.

The analysis presented here does not apply to node-level tasks (node classification and clustering), where the degree distribution does not play a major role. In fact, Figure 8 shows that PMI produces a better separation between clusters with hubs for each cluster positioned in the middle, which is desired for node-level tasks. This property might be explained by the non-linearity of PMI.

5 RELATED WORK

We refer to [7, 9, 14, 25, 28] for an overview of graph embedding and representation learning. The general problem of embedding vertices of a graph into a vector space can be traced back to much earlier work on embedding metric spaces [6], spectral graph theory [12], nonlinear dimensionality reduction [5] and graph drawing [17]. However, the more recent appeal for graph embedding methods coincided with the renewed interest in deep learning, specially the skip-gram model for text, such as word2vec [39]. DeepWalk [45], which introduced the idea of using random-walks as graph counterparts of sentences in skip-gram, is considered the pioneering work on random-walk based graph embedding. Subsequent extensions have led to an extensive literature [26, 42, 56, 66]. Random-walk based embedding models have also been proposed for heterogeneous [18], dynamic [20] and multilayer networks [64].

Previous work has proposed embeddings based on specific processes. For instance, node2vec [26] applies a biased random-walk process that is flexible enough to capture homophily and structural similarity. In [66], the authors propose a variation of DeepWalk with PageRank [43] instead of the standard random-walks. These studies motivate generalizing embedding to other processes, such as the Ruelle-Bowen random-walk, which has maximum entropy [50], and even continuous-time random-walks [34, 40].

With the exception of [51], random-walk based embeddings cannot generalize—either explicitly or implicitly—to other similarities besides Pointwise Mutual Information (PMI) [48]. As an alternative, autocovariance similarity is the basis of Markov Stability [15, 16, 34], an effective multiscale community detection algorithm validated in many real networks. In [51], the authors introduced a general embedding scheme based on control theory, of which autocovariance is a special case. However, they neither contextualized it with other existing embedding approaches nor provided any theoretical or empirical evaluation on any particular task. A link prediction algorithm based on autocovariance was introduced in [23], but it does not produce embeddings. Integrating autocovariance into the broader framework of random-walk embedding—with different processes and embedding algorithms—and investigating its properties both theoretically and empirically is a contribution of our work.

For some time, graph embedding models were divided into those based on skip-gram [26, 45, 56], which are trained using sampled walks and stochastic gradient descent (or its variations) and those based on matrix factorization [8, 42]. It was later shown that, under certain assumptions, skip-gram based text embedding models perform matrix factorization implicitly [36]. More recently, Qiu et al. [48] showed that DeepWalk, node2vec and LINE [56] also perform implicit matrix factorization. Our framework incorporates both sampling-based and matrix factorization algorithms and we show how the former can generalize to similarities beyond PMI.

Multiscale graph embedding based on random-walks has attracted limited interest in the literature [11, 63], though many real-life networks are known to have a multiscale (or hierarchical) structure [13, 49]. Notice that we focus on graphs where scales are defined by clusters/communities. This is different from graphs where vertices belong to different levels of a hierarchy, such as trees, for which hyperbolic embedding [41] is a promising approach.

6 CONCLUSION

We have introduced a framework that provides a renewed bearing on random-walk based graph embedding. This area has capitalized on the close connection with skip-gram but has also been biased by some of its design choices. Our framework has enabled us to scrutinize them, which will benefit researchers and practitioners alike. In the future, we want to explore alternative choices of components in our framework, as well as extending it to graphs with richer information, such as signed, dynamic, and attributed graphs.

ACKNOWLEDGMENTS

This work is partially funded by NSF via grant IIS 1817046 and DTRA via grant HDTRA1-19-1-0017.

REFERENCES

[1] Lada A Adamic and Natalie Glance. 2005. The political blogosphere and the 2004 US election: divided they blog. In *Workshop on Link discovery*.

[2] David Arthur and Sergei Vassilvitskii. 2006. *k-means++: The advantages of careful seeding*. Technical Report. Stanford.

[3] Nicolas Aspert, Volodymyr Miz, Benjamin Ricaud, and Pierre Vandergheynst. 2019. A graph-structured dataset for Wikipedia research. In *WebConf*.

[4] Albert-László Barabási and Eric Bonabeau. 2003. Scale-free networks. *Scientific american* 288, 5 (2003), 60–69.

[5] Mikhail Belkin and Partha Niyogi. 2002. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NeurIPS*.

[6] Jean Bourgain. 1985. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics* 52, 1-2 (1985), 46–52.

[7] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE TKDE* 30, 9 (2018), 1616–1637.

[8] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *CIKM*.

[9] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. 2020. Machine Learning on Graphs: A Model and Comprehensive Taxonomy. *arXiv:2005.03675* (2020).

[10] Sudhanshu Chanpuriya and Cameron Musco. 2020. Infinitewalk: Deep network embeddings as Laplacian embeddings with a nonlinearity. In *SIGKDD*.

[11] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. 2018. Harp: Hierarchical representation learning for networks. In *AAAI*.

[12] Fan RK Chung. 1997. *Spectral graph theory*. Number 92. AMS.

[13] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. 2008. Hierarchical structure and the prediction of missing links in networks. *Nature* 453, 7191 (2008), 98–101.

[14] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2018. A survey on network embedding. *IEEE TKDE* 31, 5 (2018), 833–852.

[15] Jean-Charles Delvenne, Michael T Schaub, Sophia N Yaliraki, and Mauricio Barahona. 2013. The stability of a graph partition: A dynamics-based framework for community detection. In *Dynamics On and Of Complex Networks*. Springer, 221–242.

[16] J-C Delvenne, Sophia N Yaliraki, and Mauricio Barahona. 2010. Stability of graph communities across time scales. *PNAS* 107, 29 (2010), 12755–12760.

[17] Josep Díaz, Jordi Petit, and Maria Serna. 2002. A survey of graph layout problems. *ACM Computing Surveys (CSUR)* 34, 3 (2002), 313–356.

[18] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD*.

[19] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning structural node embeddings via diffusion wavelets. In *SIGKDD*.

[20] Lun Du, Yun Wang, Guojie Song, Zhicong Lu, and Junshan Wang. 2018. Dynamic Network Embedding: An Extended Approach for Skip-gram based Network Embedding. In *IJCAI*.

[21] Carl Eckart and Gale Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika* 1, 3 (1936), 211–218.

[22] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *JMLR* 9, Aug (2008), 1871–1874.

[23] Hua Gao, Jianbin Huang, Qiang Cheng, Heli Sun, Baoli Wang, and He Li. 2019. Link prediction based on linear dynamical response. *Physica A: Statistical Mechanics and its Applications* 527 (2019), 121397.

[24] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *PNAS* 99, 12 (2002), 7821–7826.

[25] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94.

[26] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*.

[27] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review* 53, 2 (2011), 217–288.

[28] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv:1709.05584* (2017).

[29] Geoffrey E Hinton and Sam T Roweis. 2003. Stochastic neighbor embedding. In *NeurIPS*.

[30] Amin Javari, Tyler Derr, Pouya Esmailian, Jiliang Tang, and Kevin Chen-Chuan Chang. 2020. Rose: Role-based signed network embedding. In *WebConf*.

[31] Brian Karrer and Mark EJ Newman. 2011. Stochastic blockmodels and community structure in networks. *PRE* 83, 1 (2011), 016107.

[32] Megha Khosla, Jurek Leonhardt, Wolfgang Nejdl, and Avishek Anand. 2019. Node representation learning for directed graphs. In *ECML-PKDD*. Springer.

[33] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).

[34] Renaud Lambiotte, Jean-Charles Delvenne, and Mauricio Barahona. 2014. Random walks, Markov processes and the multiscale modular organization of complex networks. *IEEE Transactions on Network Science and Engineering* 1, 2 (2014), 76–90.

[35] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. 1998. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM.

[36] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *NeurIPS*.

[37] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications* 390, 6 (2011), 1150–1170.

[38] Matt Mahoney. 2011. Large text compression benchmark. https://www.mattmahoney.net/dc/textdata.

[39] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

[40] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In *WebConf*.

[41] Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *NeurIPS*.

[42] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *SIGKDD*.

[43] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web*. Technical Report. Stanford InfoLab.

[44] Jani Patokallio. 2020. OpenFlights.org: Flight logging, mapping, stats and sharing. https://openflights.org/data.html.

[45] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*.

[46] Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. 2017. Don't Walk, Skip! Online learning of multi-scale network embeddings. In *ASONAM*.

[47] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. 2019. Netsmf: Large-scale network embedding as sparse matrix factorization. In *WebConf*.

[48] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*.

[49] Erzsébet Ravasz and Albert-László Barabási. 2003. Hierarchical organization in complex networks. *PRE* 67, 2 (2003), 026112.

[50] David Ruelle. 2004. *Thermodynamic formalism: the mathematical structure of equilibrium statistical mechanics*. Cambridge University Press.

[51] Michael T Schaub, Jean-Charles Delvenne, Renaud Lambiotte, and Mauricio Barahona. 2019. Multiscale dynamical embeddings of complex networks. *PRE* 99, 6 (2019), 062308.

[52] C Seshadhri, Aneesh Sharma, Andrew Stolman, and Ashish Goel. 2020. The impossibility of low-rank representations for triangle-rich complex networks. *PNAS* 117, 11 (2020), 5631–5637.

[53] Balasubramaniam Srinivasan and Bruno Ribeiro. 2019. On the equivalence between positional node embeddings and structural graph representations. In *ICLR*.

[54] Alexander Strehl and Joydeep Ghosh. 2002. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *JMLR* 3, Dec (2002), 583–617.

[55] Lovro Šubelj and Marko Bajec. 2013. Model of complex networks based on citation dynamics. In *WebConf*.

[56] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WebConf*.

[57] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *SIGKDD*.

[58] Lei Tang, Suju Rajan, and Vijay K Narayanan. 2009. Large scale multi-label classification via metalabeler. In *WebConf*.

[59] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. 2018. Verse: Versatile graph embeddings from similarity measures. In *WebConf*.

[60] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2009. Mining multi-label data. In *Data mining and knowledge discovery handbook*. Springer, 667–685.

[61] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *SIGKDD*.

[62] Suhang Wang, Jiliang Tang, Charu Aggarwal, Yi Chang, and Huan Liu. 2017. Signed network embedding in social media. In *SDM*.

[63] Zhenghua Xin, Jie Chen, Guolong Chen, and Shu Zhao. 2019. Marc: Multi-Granular Representation Learning for Networks Based on the 3-Clique. *IEEE Access* 7 (2019), 141715–141727.

[64] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. 2018. Scalable Multiplex Network Embedding. In *IJCAI*.

[65] Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. 2018. Arbitrary-order proximity preserved network embedding. In *SIGKDD*.

[66] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. 2017. Scalable graph embedding for asymmetric proximity. In *AAAI*.

A APPENDIX

A.1 Proof of the Relationship between PMI Similarity and Skip-gram Based Models

Here, we show the close relationship between the PMI similarity and existing embedding models. As proved in [48], LINE and DeepWalk implicitly factorize the following matrices:

$$R_{LINE} = \log(\text{vol}(\mathcal{G})D^{-1}AD^{-1}) - \log b \tag{17}$$

$$R_{DW} = \log(\text{vol}(G)(\frac{1}{T}\sum_{r=1}^{T}(D^{-1}A)^r)D^{-1}) - \log b \tag{18}$$

where b and T are the number of negative samples and the context window size in skip-gram respectively, and $\text{vol}(\mathcal{G}) = \sum_u \deg(u)$. Now, consider the standard random-walk where $M = D^{-1}A$ and $\Pi = D/\text{vol}(\mathcal{G})$. We notice the following equivalence:

Lemma A.1. *The PMI similarity matrix $R(\tau)$ in Equation 6 for the standard random-walk can be expressed as*

$$R(\tau) = \log(\text{vol}(\mathcal{G})(D^{-1}A)^\tau D^{-1}) \tag{19}$$

Proof. We use $\circ$ and $\oslash$ to represent elementwise multiplication and division between matrices. We have:

$$\begin{aligned} R(\tau) &= \log(\Pi M^\tau) - \log(\pi\pi^T) \\ &= \log((\pi,\dots,\pi)\circ M^\tau) - \log((\pi,\dots,\pi)\circ(\pi,\dots,\pi)^T) \\ &= \log(M^\tau) - \log((\pi,\dots,\pi)^T) \\ &= \log(M^\tau \oslash (\pi,\dots,\pi)^T) \\ &= \log(M^\tau\Pi^{-1}) \\ &= \log(\text{vol}(\mathcal{G})(D^{-1}A)^\tau D^{-1}) \end{aligned} \tag{20}$$

□

Thus, R_{LINE} factorizes a shifted version of the similarity at $\tau=1$:

$$R_{LINE} = R(1) - \log b \tag{21}$$

And R_{DW} factorizes a smooth approximation of the average (*log-mean-exp*) for the shifted similarity with t from 1 to T:

$$R_{DW} = \log\left(\frac{1}{T}\sum_{\tau=1}^{T}\exp(R(\tau))\right) - \log b \tag{22}$$

A.2 Proof of Theorem 2.1

Proof. We first show that, for $b = 1$, Equations 14 and 15 can be derived from the definition of autocovariance and the Bayes Theorem. The corpus $\mathcal{D}$ is composed of samples from the random-walk process, and thus, for autocovariance-based embeddings:

$$\begin{aligned} p(u,v|z=1) &= \pi_u p(x(t+\tau)=v|x(t)=u)) \\ &= \mathbf{u}_u^T\mathbf{v}_v + \pi_u\pi_v \end{aligned} \tag{23}$$

For pairs that are not in the corpus (negative samples):

$$p(u,v|z=0) = \pi_u\pi_v \tag{24}$$

Because $b=1$, $p(z=0)=p(z=1)$, and $p(u,v)=\mathbf{u}_u^T\mathbf{u}_v+2\pi_u\pi_v$. From the Bayes Theorem, we get Equation 14 and Equation 15. We will assume that Equation 16 can be computed exactly from samples:

$$\mathbf{u}_u^T\mathbf{v}_v = \frac{\#(u,v)}{b|\mathcal{D}|} - \frac{\#(u)\#(v)}{|\mathcal{D}|^2} \tag{25}$$

Moreover, similar to [36], we will simplify Equation 13 as follows:

$$\begin{aligned} \ell &= \sum_{u,v}\#(u,v)\log(p(z=1|u,v)) + b\sum_{u,v}\#(u,v)\mathbb{E}_{w\sim\pi}\log(p(z=0|u,w)) \\ &= \sum_{u,v}\#(u,v)\log(p(z=1|u,v)) + b\sum_{u}\#(u)\frac{\#(v)}{|\mathcal{D}|}\log(p(z=0|u,v)) \\ &\quad + b\sum_{u,w\neq v}\#(u)\frac{\#(w)}{|\mathcal{D}|}\log(p(z=0|u,w)) \end{aligned}$$

where $\#(v) = \sum_w \#(v,w)$.

For a large enough number of dimensions, we can minimize the above Equation for each pair (u,v):

$$\ell_{u,v} = \#(u,v)\log(p(z=1|u,v)) + b\#(u)\frac{\#(v)}{|\mathcal{D}|}\log(p(z=0|u,v)) \tag{26}$$

Now, we plug the conditional probabilities $p(z=1|u,v)$ and $p(z=0|u,v)$ from Equation 14 and Equation 15 into $\ell_{u,v}$, and compute its derivative with respect to $x = \mathbf{u}_u^T\mathbf{v}_v$, and we have

$$\frac{\partial\ell_{u,v}}{\partial x} = \frac{-\#(u,v)\pi_u\pi_v}{(x+\pi_u\pi_v)(x+(b+1)\pi_u\pi_v)} + \frac{b\#(u)\#(v)}{|\mathcal{D}|(x+(b+1)\pi_u\pi_v)}$$

where we have conveniently dropped the function ρ.

Setting the derivative to zero, we get Equation 25. We emphasize that similar to [36], our theorem only holds when d is large enough to allow embeddings to be optimal pairwise, which can only be guaranteed in the general case when $d = n$. □

A.3 Additional Figures

Figure 9 and Figure 10 (both referred in Section 3.3.2) show community detection and link prediction performance for varying Markov time. In Figure 11 and Figure 12 (both referred in Section 3.3.4) we show link prediction and node classification results for directed and undirected embeddings with PageRank. And Figure 13 (referred in Section 3.3.3) shows the correlation between entries of similarity matrices and the reconstruction from the embeddings.

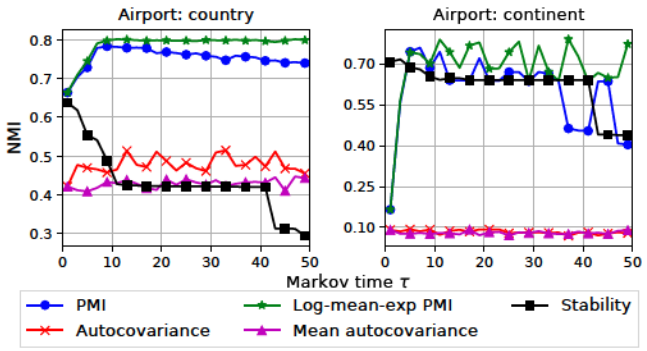

Figure 9: Community detection for PMI, autocovariance, their moving means, and Markov Stability [16] on varying Markov times. *Log-mean-exp* PMI outperforms autocovariance and Markov Stability for country and continent levels.

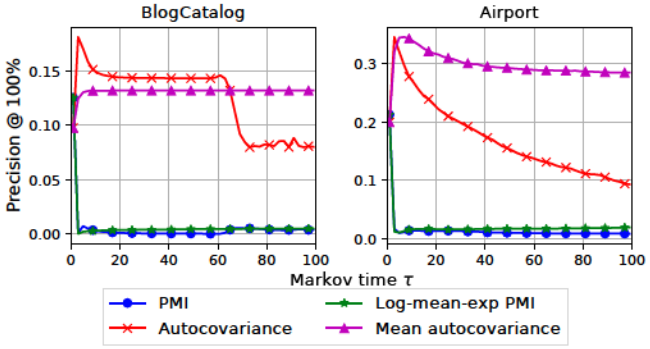

Figure 10: Link prediction for PMI, autocovariance, and their moving means on varying Markov times. Neither *log-mean-exp* PMI nor mean autocovariance increases the peak performance.

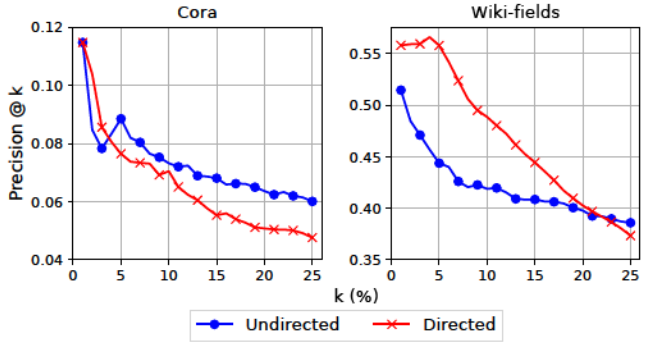

Figure 11: Link prediction for directed and undirected embeddings with PageRank on varying top pairs. Directed embedding outperforms undirected embedding for the very top ranked edges.

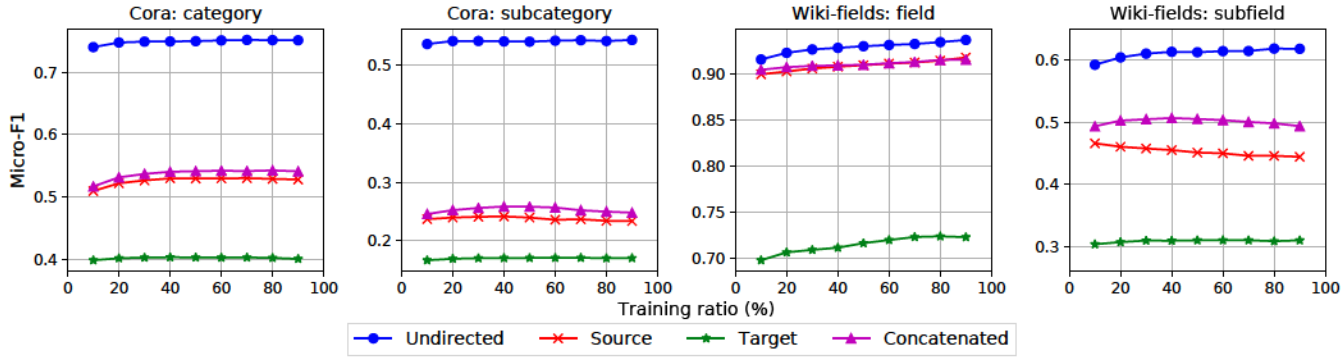

Figure 12: Node classification for directed and undirected embeddings with PageRank on varying training ratios. *Macro-F1* scores are not shown here as they follow similar patterns as *Micro-F1* scores. The undirected embedding consistently outperforms both source, target and concatenated directed embeddings in both datasets.

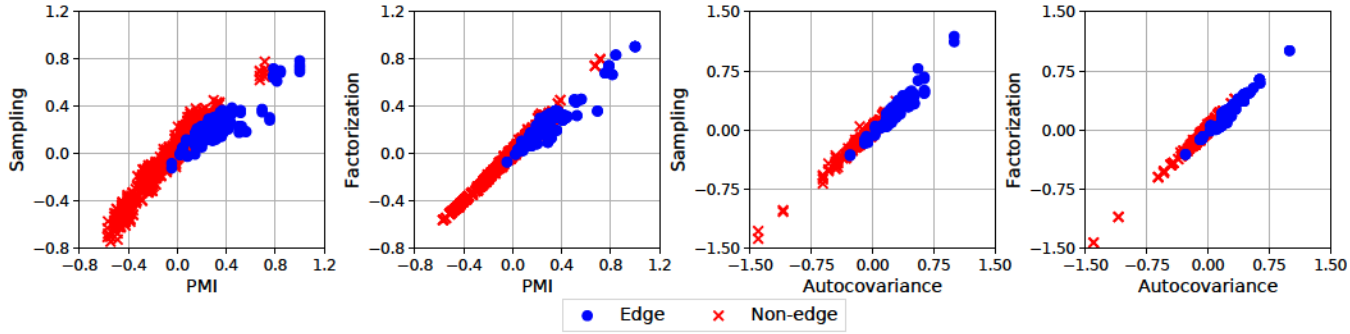

Figure 13: Correlation between entries of the similarity matrices (PMI and autocovariance) and the corresponding dot product reconstruction from the embeddings generated via sampling and matrix factorization algorithms. The results are generated using Zachary's karate club network. Both edge and non-edge pairs of nodes are shown. While a clear correlation can be noticed in all cases, matrix factorization methods provide a better approximation of the similarity metrics.