RESEARCH ARTICLE

SimNet: Similarity-based network embeddings with mean commute time

Moein Khajehnejad *

Max Planck Institute for Software Systems (MPI-SWS), Saarbrücken, Germany

* mkhajehn@mpi-sws.org

Abstract

In this paper, we propose a new approach for learning node embeddings for weighted undirected networks. We perform a random walk on the network to extract the latent structural information and perform node embedding learning under a similarity-based framework. Unlike previous works, we apply a different criterion to capture the proximity information between nodes in a network, and use it for improved modeling of similarities between nodes. We show that the mean commute time (MCT) between two nodes, defined as the average time a random walker takes to reach a target node and return to the source, plays a crucial role in quantifying the actual degree of proximity between two nodes of the network. We then introduce a novel definition of a similarity matrix that is based on the pair-wise mean commute time captured, which enables us to adequately represent the connection of similar nodes. We utilize pseudoinverse of the Laplacian matrix of the graph for calculating such a proximity measure, capturing rich structural information out of the graph for learning more adequate node representations of a network. The results of different experiments on three real-world networks demonstrate that our proposed method outperforms existing related efforts in classification, clustering, visualization as well as link prediction tasks.

Citation: Khajehnejad M (2019) SimNet: Similarity-based network embeddings with mean commute time. PLoS ONE 14(8): e0221172. https://doi.org/10.1371/journal.pone.0221172

Editor: Enrique Hernandez-Lemus, Instituto Nacional de Medicina Genomica, MEXICO

Received: February 23, 2019

Accepted: July 31, 2019

Published: August 15, 2019

Data Availability Statement: All BlogCatalog files are available from the "http://socialcomputing.asu.edu/datasets/BlogCatalog" database. All 20-New Group files are available from the "http://qwone.com/~jason/20Newsgroups/" database. All DBLP Citation files are available from the "https://aminer.org/citation" database.

Funding: This work was supported by MOE Tier 1 Grant SUTDT12015008. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The author has declared that no competing interests exist.

1 Introduction

Information network analysis has become ubiquitous in the past recent decades. From biology to computer sciences and from chemistry to sociology, the world is filled with networks. Building models that can effectively capture the information associated with network data has thus become increasingly important. Such models can lead to systems that are capable of performing tasks such as node ranking [1], community detection [2], classification [3] and link prediction [4].

One of the approaches towards mining graph information that recently received a significant amount of attention is the learning of graph representations, or network embeddings. The main goal of such a line of research is to learn for each node in the network a vector representation that conveys useful and meaningful information. One of the simplest approaches to learning the network embeddings can be done through the use of the adjacency matrix of the graph. For a graph of n nodes, the i-th row of the adjacency matrix corresponds to the i-th node, which gives us a n-dimensional vector representation for the node. While reasonable, one limitation with such an approach is it captures simple first-order proximity information between neighboring nodes which are directly connected to each other, ignoring higher-order proximity information amongs nodes. Furthermore, when n is a very large number, the dimension of the resulting embeddings become very large. Dealing with such high dimensional data becomes challenging.

Therefore it is important to develop methods that can represent the nodes in a graph with *low-dimensional* vectors, which can capture meaningful structural, semantic and relational information conveyed by the graph. Motivated by this, there has been a surge of interest in learning low-dimensional node representations for graphs in recent years. The skip-gram model [5] proposed an effective approach for learning representations for graphs with a special topology—linear chains. The model has been successfully applied to the task of learning word representations (or word embeddings) from natural language data and the model was implemented in the widely used toolkit `word2vec`. The model was also recently shown to have relation with an approach for learning embeddings based on factorizing a positive pairwise mutual information (PPMI) matrix [6]. DeepWalk [7], another recent work, employs *truncated random walk* to transform graph structures into linear sequences of nodes. It then makes use of the skip-gram model for learning representations for the nodes. Both of these approaches are able to capture higher k-th order proximity information between nodes (*i.e.*, the path between two nodes consists of k consecutive edges) rather than the simple first-order information as conveyed by the adjacency matrix. Such k-th order proximity information captures more global aspects of the network's structure and can play a crucial role in the process of learning graph representations. In fact, in another recent study—the LINE model [8], although it largely focuses on learning local first-order and second-order proximity information, the authors also show that it is important to integrate higher-order proximity information into their model.

The importance of learning the global structural information of a graph has led to the recently proposed GraRep [9] model. In GraRep different k-th order proximity information between nodes is captured separately with different matrices. The model then concatenates all such information to form the low-dimensional representations for nodes. The most recent work node2vec [10] proposes to optimize a custom graph-based objective function using stochastic gradient descent motivated by [5]. The model makes use of a second-order random walk with different sampling strategies to generate neighborhood nodes for each node in the network. Next it applies the skip-gram model and optimizes the log-probability of observing a specific set of neighbors conditioned on the source node's feature representation. The model requires the assumption that the neighboring nodes are conditionally independent of one another given the current node. We note that the above related works can all capture only the local information embedded in the network. GraRep has made an effort to gather more global information by exploring higher-order proximity information. However, the acquired global information is still bounded by the maximal length of the paths connecting two nodes.

In this paper, we propose a novel approach SimNet, which tackles the problem of learning graph representations from the perspective of measuring the global similarities between arbitrary nodes in the networks. When focusing on the network embedding approaches, one could divide them in three major groups of a. Random walk based, b. Matrix factorization based, and c. Deep learning based approaches [11]. One of the main contributions of this work is to combine the two Random walk based and Matrix factorization based approaches and use a similarity based measure to perform a highly beneficial embedding of small or large real-world networks such as DBLP network or Blocatalog datasets. Here, we first formally formulate the notion of global similarity between different two nodes based on how similar they really are. Such a similarity measure takes global structural information about the *complete network* into account, essentially involving all possible nodes rather than a local collection of neighboring nodes only. One essential component required for defining the similarities between nodes in a graph is the proximity measure between different nodes. Unfortunately, we show that existing approaches make simple assumptions when quantifying such an important measure, ignoring rich structural information conveyed by the graph. We argue that while it is true that a shorter path between two nodes indicates a higher proximity between them (and thus can lead to a higher similarity score), the length of the path connecting two nodes is not the only factor quantifying their proximity. For example, the number of possible different paths between the two nodes is also a strong factor that one needs to consider when measuring the proximity between nodes. We show that, specifically, the mean commute time (MCT) measure, among the other evaluated measures, is adequate for assessing the relative proximity between different nodes in a network when measuring their similarities. It is also demonstrated how MCT outperforms previously utilized similarity measures when exploited in the embedding process. Empirically, through extensive experiments on various datasets across different tasks, we illustrate the effectiveness of our proposed approach.

2 Background

In this section, we discuss background relevant to this work. We use $G = (V, E)$ to denote a graph, where $V = (v_1, v_2, \ldots, v_N)$ is the set of nodes and $E = \{e_{i,j}\}$ is the set of edges. The edge $e_{i,j}$ indicates a connection between two nodes v_i and v_j.

The adjacency matrix for a weighted graph is defined as a matrix A where $[A]_{ij} = w_{ij}$ if and only if nodes v_i and v_j are connected by an edge with weight w_{ij} and $[A]_{ij} = 0$ if they are not connected by an edge. The degree of a node v_i, denoted by $d(v_i)$, is:

$$d(v_i) = \sum_j [A]_{ij} \tag{1}$$

In a graph with N nodes the total degree of all nodes is equal to $\Sigma_{i=1}^{N} d(v_i) = 2l$ (note that A is always symmetric, and l is the total weight of all edges in the graph) and the average degree is $2l/N$. The degree matrix D is defined as the following diagonal matrix, where the i-th diagonal element is $d(v_i)$:

$$D = diag[d(v_1), d(v_2), \ldots, d(v_N)] \tag{2}$$

2.1 Random walk

Random walk has been a subject of intensive study in the past decades. It was found useful when solving problems such as ranking [12], clustering [13, 14], synchronization [15, 16] and modeling diffusion processes [17, 18]. Today it has become an important class of probabilistic models. In this section we will briefly explain how a random walker navigates on a graph.

In a random walk, the walker currently at a node v can move from v to any of its neighbouring node with a probability proportional to the weight of the edge between them. The probability of the walker stepping into node v_j from v_i is denoted by $P(v_j|v_i)$. Therefore the stochastic process of random walk is characterized with this transition matrix P which is defined as follows:

$$P = D^{-1}A \tag{3}$$

where A is the adjacency matrix and D the degree matrix defined above. Each element of P

follows the following equation:

$$[P]_{ij} = \frac{[A]_{ij}}{[D]_{ii}} = P(v_j|v_i) \tag{4}$$

Let P^t be the t-th power of P. Then $[P^t]_{ij}$ represents the probability of the walker to arrive at node v_j with exactly t steps, starting from node v_i.

As pointed out in [19], this kind of random walk is an example of an ergodic Markov chain [20, 21], whose stationary distribution can be characterized by a unique N-dimensional row vector π where the i-th element $\pi_i = d(v_i)/2l$—the expected probability of reaching the i-th node in the random walk process.

2.2 Mean commute time

In this work, we would like to introduce a new method to measure the closeness or proximity between two different nodes in a network. Specifically, we introduce the mean commute time (MCT) measure to quantify this closeness. Different from many previous approaches where a simple k-th order proximity information is used for quantifying the closeness, we argue that MCT is a more appropriate measure whose soundness can be theoretically justified. There are two basic quantities that can be computed from the definition of the Markov chain, that is, from its transition probability matrix: the *mean first-passage time* (MFPT) [22] and then the *mean commute time* (MCT) [23, 24]. The mean first-passage time can be defined as the average number of steps (time required) that a random walker takes to reach the target node j for the first time starting from the source node i. In a close relation to MFPT, the mean commute time is defined as the average time that a random walker, starting from the source node i, will take to reach the target node j for the first time and then return to i. Intuitively, as it can be understood from this definition, the mean commute time between two nodes has the desirable property of decreasing when the number of paths connecting the two nodes increases and when the length of paths decreases, as opposed to the usual shortest path distance between nodes which does not capture the fact that strongly connected nodes are closer than weakly connected nodes. Therefore this quantity can give us a valuable and meaningful measure for the proximity information between nodes.

2.2.1 Calculation of MCT. There are several different approaches for computing the MCT quantity based on algorithms introduced in the Markov-chain community or on iterative procedures. Here we will use the Moore-Penrose pseudoinverse of the Laplacian matrix of the graph [25] for our purpose of computing MCT.

The symmetric Laplacian matrix $\mathbf{L}$ of the graph is defined in the usual manner, $\mathbf{L} = D - A$. Now we can introduce the Moore-Penrose pseudoinverse of $\mathbf{L}$ which will be denoted as $\mathbf{L}^+$ and is defined as:

$$\mathbf{L}^+ = \left(\mathbf{L} - \frac{\mathbf{e}\mathbf{e}^T}{N}\right)^{-1} + \frac{\mathbf{e}\mathbf{e}^T}{N} \tag{5}$$

where $\mathbf{e}$ is an N-dimensional vector consisting of all 1's.

Pirotte *et al.* [25] showed that the MCT matrix C can be derived from $\mathbf{L}^+$ using the following procedure:

$$[C]_{ij} = 2l([\mathbf{L}^+]_{ii} + [\mathbf{L}^+]_{jj} - 2[\mathbf{L}^+]_{ij}) \tag{6}$$

$$= 2l(\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{L}^+ (\mathbf{e}_i - \mathbf{e}_j) \qquad (7)$$

where $\mathbf{e}_k$ is an N-dimensional basis vector whose k-th entry is 1 and 0 elsewhere.

3 SimNet

3.1 Similarity matrix

Central to our SimNet model is the definition of a similarity matrix over the network, which captures pair-wise similarities between any two nodes in the network. The key question is how to define such a similarity matrix that is capable of capturing global similarity between two nodes. Inspired by the work of Katz *et al.* [26] which led to the definition of a type of similarity based on centrality and using the adjacency matrix [27], we use the following recursive thesis to define what is a global similarity between two nodes in a network: *Two nodes are similar to each other if one has neighbors that are similar to the other.* This statement shows that to measure the similarity between two nodes, one needs to look into the neighbors for similarity. Mathematically, we can express the above as follows:

$$sim_{i,j} = \sum_k f([P]_{ik} sim_{k,j}) + \delta_{i,j} \qquad (8)$$

where $sim_{i,j}$ is the similarity between nodes i and j, and $\delta_{i,j}$ is the indicator function which is 1 if $i = j$ and 0 otherwise.

This expression essentially says that the similarity between two nodes v_i and v_j is defined based on a sum of functions that involve similarities defined over the neighbors of v_i and the node v_j (plus the fact whether v_i and v_j are identical). One typical choice is $f(x) = \alpha x$ where $\alpha < 1$ (we will see why below), which leads to the following expression in matrix form:

$$S = \alpha P S + I \qquad (9)$$

Repeatedly replacing the S in the RHS using the above recurrence relation leads to the following:

$$S = \lim_{n \to \infty} (I + \alpha P + \alpha^2 P^2 + \cdots + \alpha^n P^n) = \sum_{k=0}^{\infty} g(k) P^k \qquad (10)$$

where $g(k) = \alpha^k$ is called the *proximity function*. As mentioned in [27], for the above limit to converge, α should have a value less than $||P||_2^{-1}$ where $||P||_2$ is the spectral radius of matrix P. Since P is the normalized transition matrix, this means $\alpha < 1$ indeed is sufficient to guarantee convergence. In fact, we have:

$$S = (I - \alpha P)^{-1} \qquad (11)$$

Now let us take a closer look at the above definition of similarity. The global similarity measure between two nodes is a combination of individual k-th order proximity information each augmented with a proximity function α^k. However, what does this damping factor α in this function mean? Essentially it says that when calculating the current similarity matrix, how much of the similarity information captured by the neighbors we would like to rely on. A higher α indicates more information from the neighbors we would like to leverage on. Intuitively, it is the level of "trust" the current node would like to put to its neighbors, or how "close" a node is to its neighbors. The proximity function $g(k) = \alpha^k$ decreases exponentially as

k increases. It was previously shown that such a function plays an important role when learning graph and word representations [5, 28–30].

Assigning a constant α in the above formula, we have a fixed coefficient applied to all pairs of elements of each P^k. This means for two different pairs of nodes with the same distance, the level of closeness or proximity considered when measuring the similarities is the same regardless of their other properties or relations that can be captured by the graph. Thus using a constant α we have managed to see how a longer distance affects the similarity measure. Although this definition itself can allow us to capture a certain amount of global information, we believe the "closeness" between two nodes in a graph should not be defined by simply counting the number of steps required to reach one of them from the other. Instead, we believe the closeness measure should be defined in a more flexible manner, where different node pairs should be assigned a different damping factor, reflecting the true closeness between them.

This is what motivates us to use variable damping factors which are different for different node pairs, leading us to the use of MCT measure for quantifying the closeness between nodes in a graph when measuring the global similarity. The MCT between nodes contains rich structural information associated with the complete graph, which can be adequately used to measure how close two nodes are in a given graph. It is a theoretically sound measure that conveys not only the information regarding shortest path connecting two nodes, but also the number of all possible paths.

As discussed in the previous section, a lower $[C]_{ij}$ value indicates less time required for traveling from v_i to v_j, hence a higher proximity level. We thus define each entry of the S matrix as:

$$[S]_{ij} = \sum_{k=0}^{\infty} g(k, i, j)[P^k]_{ij} \tag{12}$$

where $g(k, i, j)$ is used to replace the original proximity function $g(k)$ involving the constant damping factor α. As it was explained earlier, here we can utilize matrix $\mathbf{L}^+$ knowing that it captures all the information we need from the graph. So we define g as follows:

$$g(k, i, j) = \left(\frac{\arctan([\mathbf{L}^+]_{ij})}{\pi} + \frac{1}{2}\right)^k \tag{13}$$

Now let us see how the S matrix can be calculated under such a new proximity function. Let us define a matrix G where $[G]_{ij} = g(1, i, j)$. We have:

$$S = I + (G \circ P) + (G \circ G) \circ P^2 + (G \circ G \circ G) \circ P^3 + \ldots \tag{14}$$

where $\circ$ is the element-wise Hadamard product operation.

We consider the eigendecomposition of P as $P = T\Lambda T^{-1}$, where Λ is a diagonal matrix whose diagonal entries are eigenvalues and the columns of T are the eigenvectors. We have:

$$P^n = T\Lambda^n T^{-1} \tag{15}$$

and

$$[P^n]_{ij} = \sum_{k=1}^{N} T_{ik}(\Lambda_{kk})^n T^{-1}_{kj} \tag{16}$$

Now for each element $[S]_{ij}$ of matrix S we have:

$$[S]_{ij} = \sum_{l=0}^{\infty} [G]_{ij}^{l} [P^{l}]_{ij} = \sum_{k=1}^{N} T_{ik} T_{kj}^{-1} \sum_{l=0}^{\infty} \left([G]_{ij} (\Lambda_{kk})\right)^{l} \tag{17}$$

$$= \sum_{k=1}^{N} T_{ik} T_{kj}^{-1} \left(1 - [G]_{ij} (\Lambda_{kk})\right)^{-1} \tag{18}$$

Finally to obtain a symmetric matrix for the similarity between each pair of nodes we define $\bar{S}$ as follows:

$$\bar{S} = \frac{1}{2}(S + S^{T}) \tag{19}$$

This matrix conveys the necessary pair-wise similarity information that is required for building our SimNet model. The matrix also preserves all the properties sufficient for it to reside in a metric space. Namely, identity of indiscernibles, symmetry, and subadditivity are preserved. Hence, one could deduce that it can be utilized to form a transition matrix and in the learning process without the concerns about metric space residency. Our approach essentially performs some refinement to the conventional notion of closeness/proximity for a network. We use Fig 1 to illustrate the high-level ideas. In most conventional approaches, as shown on the left, the closeness measure between two nodes are calculated based on the number of steps required to reach one node from the other. In our approach, however, we introduce a new perspective to measuring the closeness. Our new notion of closeness between two nodes is based on the MCT measure that quantify the average time required for a random walker to reach one node from the other and return back in a principled manner. One can imagine that this is equivalent to say that for any arbitrary two nodes, there is a "virtual edge" that connects them which reveals the strength of closeness between them, as illustrated on the

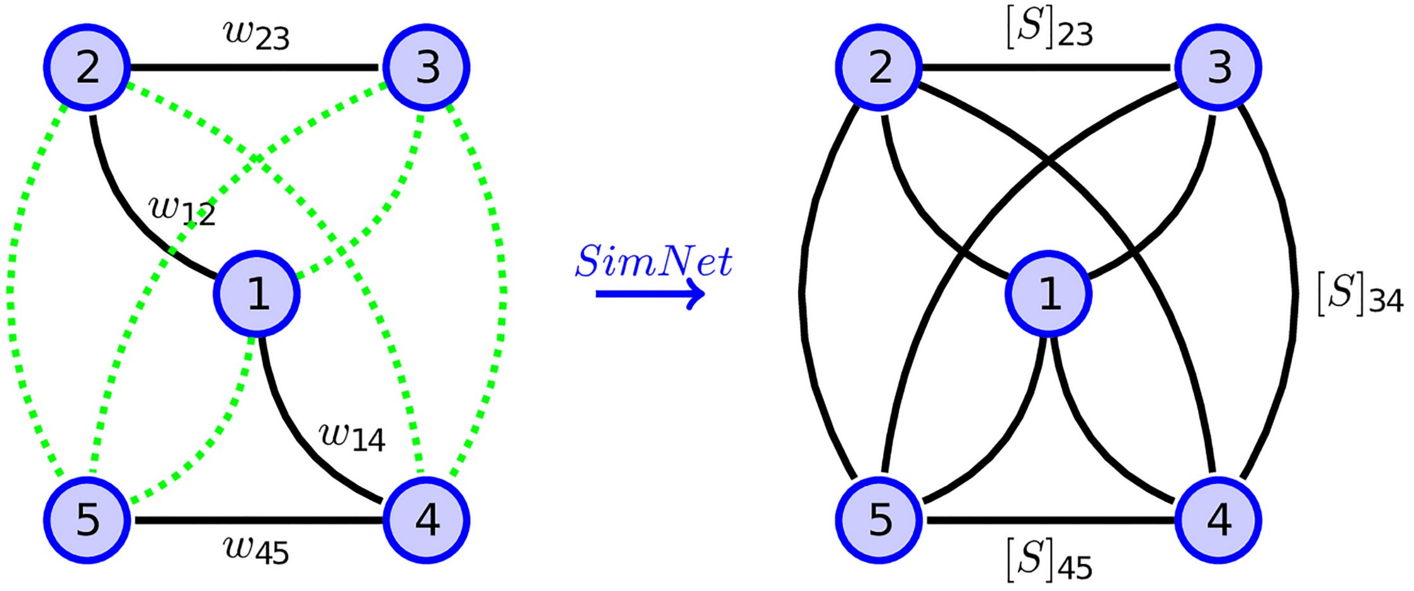

Fig 1. SimNet method. Defining a similarity matrix based on a new way of measuring the closeness between nodes.

https://doi.org/10.1371/journal.pone.0221172.g001 right of the same figure. This allows a richer amount of global structural information to be captured in the above similarity matrix $\bar{S}$.

3.2 Learning

Once we have obtained the similarity matrix, we will be able to perform the learning of the embeddings for the nodes based on it. Our first step is to normalize the resulting matrix, arriving at the following new transition matrix P_S:

$$D_S = diag[\,[\bar{S}]_{11}, [\bar{S}]_{22}, \ldots, [\bar{S}]_{NN}] \tag{20}$$

$$P_S = D_S^{-1}\bar{S} \tag{21}$$

Based on this matrix P_S, we next follow Levy *et al.* [6] and our previous work by Cao *et al.* [9] to define a proper loss function, and learn the node representations via optimization of such a loss function using matrix factorization. We describe the details next.

3.2.1 Loss function. Recall our goal is to learn low-dimensional vector representations for nodes in a network. We slightly abuse the notation here by also using v_i to denote the vector representation of the node v_i. Inspired by the negative sampling model by Mikolov *et al.* [5], we define the following loss function for v_i:

$$L(v_i) = \left[\sum_{v_j \in V} [P_S]_{ij} \log \sigma(\vec{v}_i \cdot \vec{v}_j)\right] + \lambda \mathbb{E}_{v_{j'} \sim P'(V)}[\log \sigma(-\vec{v}_i \cdot \vec{v}_{j'})] \tag{22}$$

where λ is the number of negative samples considered. $P'(V)$ is a distribution over all nodes in the network. $\mathbb{E}$ is the expected value over this distribution. The distribution $P'(V)$ is defined as:

$$P'(v_j) = 1/N \sum_{v_{i'} \in V} [P_S]_{i'j} \tag{23}$$

This leads to defining a local loss function over a specific v_i and v_j which we will later apply in the next step for optimization.

$$l(v_i, v_j) = [P_S]_{ij} \log \sigma(\vec{v}_i \cdot \vec{v}_j) + \lambda/N \sum_{v_{i'} \in V} [P_S]_{i'j} \log \sigma(-\vec{v}_i \cdot \vec{v}_j) \tag{24}$$

where $[P_S]_{ij}$ is the relation between v_i and v_j. $\sigma(.)$ is the sigmoid function: $\sigma(x) = (1+e^{-x})^{-1}$. The larger the similarity (dot product) between two node representations v_i and v_j is, the higher the value of the sigmoid function. Our goal is to learn such node representations.

3.2.2 Optimization. Our next step is to optimize the loss function. Following [6], to optimize Eq 24, we set $\partial l/\partial x = 0$ where we define $x = \vec{v}_i.\vec{v}_j$.

Solving $\partial l/\partial x = 0$ leads to:

$$e^x = \frac{[P_S]_{ij}}{\lambda/N \sum_{v_{i'} \in V} [P_S]_{i'j}} \tag{25}$$

Therefore,

$$x = \log\left(\frac{[P_S]_{ij}}{\lambda/N\sum_{v_{i'}\in V}[P_S]_{i'j}}\right) \tag{26}$$

So since $x = \vec{v}_i \cdot \vec{v}_j$, we can introduce a matrix R with the (i, j) entry being $\vec{v}_i \cdot \vec{v}_j$ as defined in the above equation:

$$[R]_{ij} = \log\left(\frac{[P_S]_{ij}}{\lambda/N\sum_{v_{i'}\in V}[P_S]_{i'j}}\right) \tag{27}$$

Following the work of Levy *et al.* [6], to reduce noise, we consider the non-negative matrix R^+ as follows:

$$R^+_{i,j} = \max(0, R_{i,j}) \tag{28}$$

Amongs various methods for matrix factorization we choose to perform singular value decomposition (SVD) [31] due to its simplicity and effectiveness as shown in our previous work [9]. For a given matrix R^+, its SVD yields the following:

$$R^+ = U\Sigma V^T \tag{29}$$

where U and V are orthogonal matrices from the space $\mathbb{R}^{N\times N}$, and $\Sigma = diag(\sigma_1, \ldots, \sigma_N)$ is a diagonal matrix whose diagonal entries are singular values of R^+, satisfying: $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_n \geq 0$.

Our primary purpose in this paper is to represent low-dimensional vectors for a network's nodes. Therefore we use an alternative matrix $R^+_d \in \mathbb{R}^{d\times d}$ which is a low-rank approximation of R^+, defined as follows:

$$R^+_d = U_d\Sigma_d V^T_d \tag{30}$$

where U_d and V_d are matrices constructed from the first d columns of matrices U and V, and Σ_d is a diagonal matrix constructed from its first d singular values: $diag[\sigma_1, \ldots, \sigma_d]$.

The above leads to factorization of matrix R^+ to two separate matrices which we call W_1 and W_2.

$$R^+ \approx W_1 W_2 \tag{31}$$

where $W_1 = U_d(\Sigma_d)^{1/2}$ and $W_2 = (\Sigma_d)^{1/2}V^T_d$.

The i-th blackrow of W_1 gives us the final vector representation of the i-th node v_i in the network.

4 Experiments

In this section we conduct experiments on several real-world datasets to assess the effectiveness of the chosen similarity measure and our graph representation method. To understand how effective our method is in general, we conduct experiments on four different tasks where three different types of networks are involved. These datasets include a language network, a social network and a citation network. We also compare our method against various existing baseline approaches.

Table 1. Statistics of the real-world networks.

	Social Network	Language Network	Citation Network
Name	**Blogcatalog**	**20-NewsGroup (200 samples)**	**DBLP (author citation)**
Type	unweighted	weighted	wighted
$\#(V)$	10,312	600, 1200 and 1800	7,314
$\#(E)$	333,983	Fully connected	72,927
Avg. degree	64.78	—	19.94
# Labels	39	3, 6 and 9	3
Task	Classification & Link Prediction	Clustering	Visualization

https://doi.org/10.1371/journal.pone.0221172.t001

4.1 Experiment Setup

4.1.1 Datasets. We provide a brief discussion on the three types of datasets that we use for our experiments. In Table 1, we list down the detailed summaries of such datasets.

Blogcatalog is a social network. Each node indicates a blog author and each edge demonstrates the relation between two bloggers. The labels show the different topics which the bloggers talk about. We will use this dataset to conduct experiments in a supervised setting, where we consider the multi-label classification task using the learned node representations. Moreover, Blogcatalog network will be used in a link prediction task where we score the possibility of a link existing between each pair of nodes given their embeddings.

DBLP Network is a citation network. Each node indicates an author and each edge between two authors has a weight illustrating the number of citations from one to another. First we choose to look at authors from six popular conferences. Next we group them into three categories: *1. data mining*, *2. computer vision*, and *3. machine learning*. Specifically, *WWW* and *KDD* are from the first category, *CVPR* and *ICCV* belong to the second category and *NIPS* and *ICML* fall into the third category. This is a weighted network that will be used for the visualization task.

20-Newsgroups is a language network with approximately 20,000 different newsgroup documents from 20 categories. Each word in each document is represented by its tf-idf score which together build up a vector for the whole document. Each document is then regarded as a node in such a language network, and the weight over an edge between any two node is defined as the cosine similarity between their respective vectors. We follow [32] to randomly sample 200 documents from a topic and form three networks from 3, 6 and 9 different newsgroups. Specifically,

3-Newsgroups includes the following news categories:
comp.graphics, comp.graphics, and *talk.politics.guns*

6-Newsgroups includes:
comp.sys.mac.hardware, rec.motorcycles, rec.sport.hockey,
soc.religion.christian, alt.atheism, and *omp.sys.ibm.pc.hardware*

9-Newsgroups includes:
talk.politics.mideast, talk.politics.misc, comp.os.mswindows.misc,
sci.crypt, sci.med, sci.space, sci.electronics, misc.forsale, and
talk.religion.misc

These are weighted and also fully-connected networks which will be used for performing clustering tasks.

4.1.2 Baseline methods. In this section, we will first compare our choice of MCT measure with other popular similarity measures which quantify the global similarities between nodes in a network. Next, after recognizing the most efficient candidate among similarity measures and justifying our original choice, we will move on to evaluate the performance of MCT-based SimNet against previous network embedding approaches.

Similarity measure variants. We choose three other well-known similarity measures to replace the MCT measure used in Section 3.1. The learning process is then carried out on these variants of SimNet.

1. **Personalized PageRank**: Personalized PageRank (PPR) [33] is a common similarity measure among nodes, practically used for graph mining tasks. We call this variant **SimNet-PPR**.
2. **Maximal Entropy Random Walk**: Maximal Entropy Random Walk (MERW) [34] is based on the nodes tendency to be linked to central nodes in a network and tries to model this behaviour. This method aims to maximize the entropy rate of the random walk. This approach will be referred to as **SimNet-ME**.
3. **Katz index**: Katz [26] is an index which sums the influence of all present paths between each pairs of nodes while penalizing paths by their length. When using this index as our similarity measure, we call the resulting approach **SimNet-Katz**.

In the rest of the paper, **SimNet** refers to the default version of our method which utilizes MCT as the similarity measure. Next, we consider the following 5 previous approaches for network embedding as our baselines.

1. **Spectral Clustering**: spectral clustering [35] is an algorithm that aims at minimizing normalized cut (NCut). It also uses matrix factorization methods, but it focuses on a different matrix—the Laplacian matrix.
2. **DeepWalk**: DeepWalk [7] is an approach for learning latent representations of nodes in a network using local information obtained from truncated random walks. It is originally only applicable to unweighted networks.
3. **LINE**: LINE [8] is a method for learning graph representations on large-scale information networks. Its loss function is based on first-order and second-order relational information between each pair of nodes among the network. The model can also make use of certain higher-order proximity information by using an extended neighborhood. The learning procedure involves several steps. In the first step, it learns the first d dimensions by performing a BFS-style sampling over first-order neighbors. This is followed by the second step, where it samples nodes strictly at a 2-hop distance for learning the next d dimensions.
4. **GraRep**: GraRep [9] is our recently proposed method for learning node representations of weighted graphs. It captures the k-th order proximity information between a node and its k-th order neighbors using the skip-gram model. GraRep constructs its representation of the graph by concatenating the results obtained from each step. However the value k can not be any arbitrarily large number and is usually empirically set to less than 6. Increasing this value linearly will also lead to a linear increasing of the resulting node representations' dimensions. The dimension of the resulting learned node representations will grow linearly as we increase the value k.
5. **node2vec**: node2vec [10] is a recent approach for learning representations for nodes in networks. It maps nodes to a low-dimensional space of features. node2vec designs an objective that seeks to preserve local neighborhoods of nodes. The objective is then optimized using stochastic gradient descent (SGD). A second order random walk approach is applied to generate neighborhoods for nodes.

4.1.3 Parameter settings. Following [8, 9], we set the dimension (dim) of representations to 128 and report results for the Blogcatalog network, so as to make a fair comparison. For the two previous models LINE and GraRep, final representations are constructed by concatenating smaller vector representations. We thus also report the results when the dimension of the smaller vectors are set to 128 (for GraRep, we set $k = 6$, following [9]).

For the clustering experiments on the 20-Newsgroup dataset, we report results under 3 different dimensions for the representations: 64 (which was used in [32]), 192 (which is 64×3—the dimension that leads to the optimal results for GraRep), and 72 (which yields the best performance for SimNet).

There are also some model-specific parameters. For DeepWalk, we followed [7] to set the window size to 10 and the walk length to 40. For LINE, as suggested in [8], we set the order to 2 so that both first-order and second-order proximity information can be concatenated to form the representations. We also employed the reconstruction strategy for nodes with small degrees so as to achieve the optimal performance. First, it learns d dimensions by BFS-style simulations over immediate neighbors of nodes which is then followed by learning the next d dimensions by sampling nodes strictly at a 2-hop distance from the source node. It eventually yields representations with a total dimension of $2d$. For GraRep, as suggested in [9], we set the parameter β to $1/N$. We used the following combinations of k and d to obtain representations of different dimensions (note that $dim = kd$): we set $k = 4$ and $d = 16$ for $dim = 64$, $k = 4$ and $d = 32$ for $dim = 128$, then $k = 3$ and $d = 24$ for $dim = 72$, and as suggested in [9], we set $k = 3$ and $d = 64$ for $dim = 192$. For node2vec, we followed their paper [10] to set the window size and the number of walks per node to 10, while the length of random walk is set to 80. We tune different values for p and q to get the best performance, which is when $p = q = 0.25$. We also repeat the experiments for 10 random seed initializations.

For different variants of SimNet, we choose the corresponding damping factors to reach the best performance. This leads us to the choice of $\alpha = 0.85$ for SimNet-PPR and SimNet-ME while SimNet-Katz performs best with $\alpha = 0.5$.

SimNet-c. For our SimNet model, to understand the effectiveness of using MCT as the closeness measure, we also developed a different, simplified version of our model. Specifically, this version uses a similarity matrix with a constant dumping factor α, which we will call **SimNet-c**. Hence, there will be no dynamic damping factor involved which allows us to better compare and capture the power of MCT-based damping factor. We make a rational choice by selecting $\alpha = 0.5$ in our experiments for the SimNet-c model. Our experiments show that although increasing α can lead to slightly better results for the clustering tasks, doing so at the same time leads to much worse results in the classification task. So to have a well-made decision we choose $\alpha = 0.5$. Our main model **SimNet** utilizes MCT-based damping factors as described in this paper to calculate the similarities between nodes.

4.2 Tasks and results

In this section we empirically demonstrate the effectiveness of our SimNet model by using the learned representations for performing classification, clustering, visualization, and link prediction tasks on real-world networks. We make the source code of SimNet available at https://github.com/Moein-Khajehnejad/SimNet/.

4.2.1 Classification. The first experiment is conducted on a multi-label classification task. Our focus is on a supervised classification task for a social network such as Blogcatalog, where we classify different nodes into different classes by using their learned node representations as input features. We use the LibLinear package [36] which is an efficient implementation of linear classifiers that is capable of handling data with millions of instances and features. During

Table 2. Results on Blogcatalog.

Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	SimNet (*dim* = 128)	**39.61**	**42.22**	**42.98**	**43.95**	**44.47**	**45.01**	**45.32**	**45.40**	**46.31**
	SimNet-c (*dim* = 128)	38.54	41.12	42.40	43.28	43.67	44.12	44.23	44.46	45.03
	SimNet-PPR (*dim* = 128)	37.95	40.36	41.50	43.56	43.94	44.51	44.82	45.03	45.51
	SimNet-ME (*dim* = 128)	35.87	38.36	39.44	40.45	40.93	41.19	41.32	41.68	42.72
	SimNet-Katz (*dim* = 128)	38.11	40.93	42.09	42.91	43.35	43.87	43.96	44.18	44.61
	node2vec (*dim* = 128)	35.56	38.72	40.32	41.33	42.11	42.73	42.85	43.00	43.59
	GraRep (*k* = 4, *d* = 32, *dim* = 128)	35.61	39.98	41.47	42.12	42.38	43.58	42.74	43.08	43.20
	GraRep (*k* = 6, *d* = 128, *dim* = 768)	38.31	40.44	41.29	41.92	42.51	42.96	43.54	43.61	44.33
	LINE (*d* = 64, *dim* = 128)	33.98	37.49	38.05	39.76	40.51	41.21	41.48	41.89	42.02
	LINE (*d* = 128, *dim* = 256)	37.12	39.57	40.71	41.66	42.27	42.63	43.11	43.44	43.79
	DeepWalk (*dim* = 128)	36.22	38.71	39.75	40.79	41.28	41.54	41.67	41.70	42.83
	Spectral Clustering (*dim* = 128)	37.23	39.33	40.39	40.76	41.32	41.47	41.52	41.65	42.16
Macro-F1	SimNet (*dim* = 128)	**23.83**	**27.88**	**29.63**	30.71	31.50	**32.43**	32.66	32.92	33.67
	SimNet-c (*dim* = 128)	23.07	27.21	28.61	29.48	30.27	30.80	30.92	31.53	31.77
	SimNet-PPR (*dim* = 128)	22.84	26.73	28.44	30.12	30.68	31.12	31.55	31.93	32.26
	SimNet-ME (*dim* = 128)	19.87	22.98	24.02	24.57	25.29	26.07	26.88	27.25	27.89
	SimNet-Katz (*dim* = 128)	22.55	26.91	28.24	29.28	30.19	30.68	30.75	31.44	31.59
	node2vec (*dim* = 128)	23.44	26.38	29.16	**30.85**	**31.76**	32.18	**32.73**	**33.19**	**33.80**
	GraRep (*k* = 4, *d* = 32, *dim* = 128)	20.81	24.89	27.33	27.95	28.43	28.78	28.94	29.03	29.31
	GraRep (*k* = 6, *d* = 128, *dim* = 768)	23.13	25.68	26.58	27.72	28.33	28.90	29.59	30.08	30.89
	LINE (*d* = 64, *dim* = 128)	16.23	18.99	21.12	21.57	21.85	22.01	22.35	22.68	22.96
	LINE (*d* = 128, *dim* = 256)	19.71	22.92	24.67	25.54	26.38	27.34	28.03	28.89	29.52
	DeepWalk (*dim* = 128)	21.47	23.42	24.39	24.93	25.69	26.56	27.08	27.22	28.14
	Spectral Clustering (*dim* = 128)	19.15	22.38	23.44	24.46	24.91	25.07	25.41	25.63	26.49

https://doi.org/10.1371/journal.pone.0221172.t002

training, a certain fraction of nodes (with labels) are selected to form the training set. Next the task is to predict the labels of the remaining nodes. We use one-vs-rest logistic regression as the classification method, and repeat the experiments 10 times and report the averaged Micro-F1 and Macro-F1 scores. We perform this task for randomly selected samples of 10 to 90 percent of the nodes in the network which are used for training and in each round the rest of the nodes are used for prediction and evaluation. The results are reported in Table 2. While SimNet proves to be consistently better than all its variants, we also evaluate its performance against previously introduced network embedding baselines. We can see that when the Macro-F1 score is considered, node2vec and SimNet perform the best both giving really satisfying scores compared to the rest. As the training set increases, node2vec starts to yield better results under this Macro-F1 though SimNet is still very close and performs almost as efficiently. The better results of node2vec under this metric indicate its better predictive power for rare classes. However, when Micro-F1 is considered, the results show that our proposed SimNet consistently yields the best scores across different setups. From the experiments we can also observe that the simpler version SimNet-c, which makes use of a constant closeness measure, consistently yields lower results. Our results (even SimNet-c) are also better than GraRep, which we believe can be largely due to its ability to capture rich higher-order proximity information.

4.2.2 Clustering. To assess the effectiveness of SimNet in a clustering task and compare it to the other baselines, following the previous research effort [9], we evaluate and report the average *normalized mutual information* (NMI) score [37] over 20 consecutive runs for all

Table 3. Clustering results on 20-newsgroups (200 samples).

Algorithm	*dim* = 64			*dim* = 72			*dim* = 192		
	3NG	6NG	9NG	3NG	6NG	9NG	3NG	6NG	9NG
SimNet	**78.62**	**67.77**	**59.55**	**79.34**	**68.63**	**60.19**	78.86	**68.57**	**59.97**
SimNet-c	76.87	65.93	57.18	77.27	67.39	58.55	76.95	68.08	59.61
SimNet-PPR	69.24	65.37	49.21	73.14	66.81	59.32	70.81	68.35	55.74
SimNet-ME	50.33	51.46	47.28	46.91	50.73	46.18	58.25	50.84	46.92
SimNet-Katz	76.62	65.68	57.76	77.01	67.13	58.29	76.73	68.64	59.30
GraRep	76.25	65.51	56.03	77.82	66.28	57.74	**81.12**	67.54	59.39
node2vec	22.17	18.86	16.33	21.31	18.79	15.12	16.75	15.55	14.03
LINE	75.24	63.99	53.62	76.41	64.78	54.10	79.25	63.05	52.17
DeepWalk	65.64	63.67	48.89	65.16	62.98	47.56	60.88	60.04	47.14
Spectral Clustering	49.12	51.03	46.93	46.37	50.43	39.13	28.46	27.81	35.98

https://doi.org/10.1371/journal.pone.0221172.t003

baselines and all three groups of 20-Newsgroups network. This is a network with each node having a single label and having an overall number of three classes. The results are shown in Table 3. When the commonly used setting *dim* = 64 is selected, our SimNet model returns the best results. When *dim* = 192 is used, GraRep performs better than our model on the 3NG dataset, but our model is better on 6NG and 9NG datasets. Our model achieves the optimal results when the setting *dim* = 72 is used, under which it also outperforms all previous approaches. SimNet clearly holds its stronger position against all its variants in this task as well. The relative effectiveness of each model was also reported in the previous work [9].

We also note that the representations produced by node2vec do not perform well for this clustering task. We believe this is partly because the biased random walk employed by node2-vec as discussed in their paper does not fit the network topology that we consider in this task very well. To get a better insight, consider a walker that just arrived at node v from t and now has to decide on the next step. Since 20-Newsgroup is a fully-connected network the distance between t and every other possible node is always 1, unless the walker is returning to the source node t. This essentially makes their in-out parameter q vacuous (which is used to bias the walker towards exploring nodes which are closer to or further away from t). Applying node2-vec on such a graph leads to a matrix no different than the original transition matrix itself with just its diagonal entries being changed. Also because of the very large number of choices the walker has in each step (since all the nodes are at distance 1 from the source and there are no neighbours of second order or more), node2vec requires more time to surf the network while performing the BFS strategy to form the set of neighbours.

The simpler version of our model SimNet-c, that makes use of higher-order proximity information of the graph, performs competitively when compared with the previous model GraRep. By using the MCT-based proximity function, our full model SimNet significantly out-performs SimNet-c.

4.2.3 Visualization. After SimNet proved to be most efficient in its default form based on an MCT similarity measure, we will exclude its variants from the results in the remaining tasks as they fall short in providing competing results.

In this experiment we aim to visualize the learned representations by examining a real cita-tion network based on DBLP. We use the standard t-SNE toolkit [38] and get a visualization of the learned graph representations. A 2-dimensional visualization of the graph is obtained with each colored dot as an indicator of each node. As mentioned before, we categorize the data of the DBLP network into three main groups, and the nodes belonging to the same group share

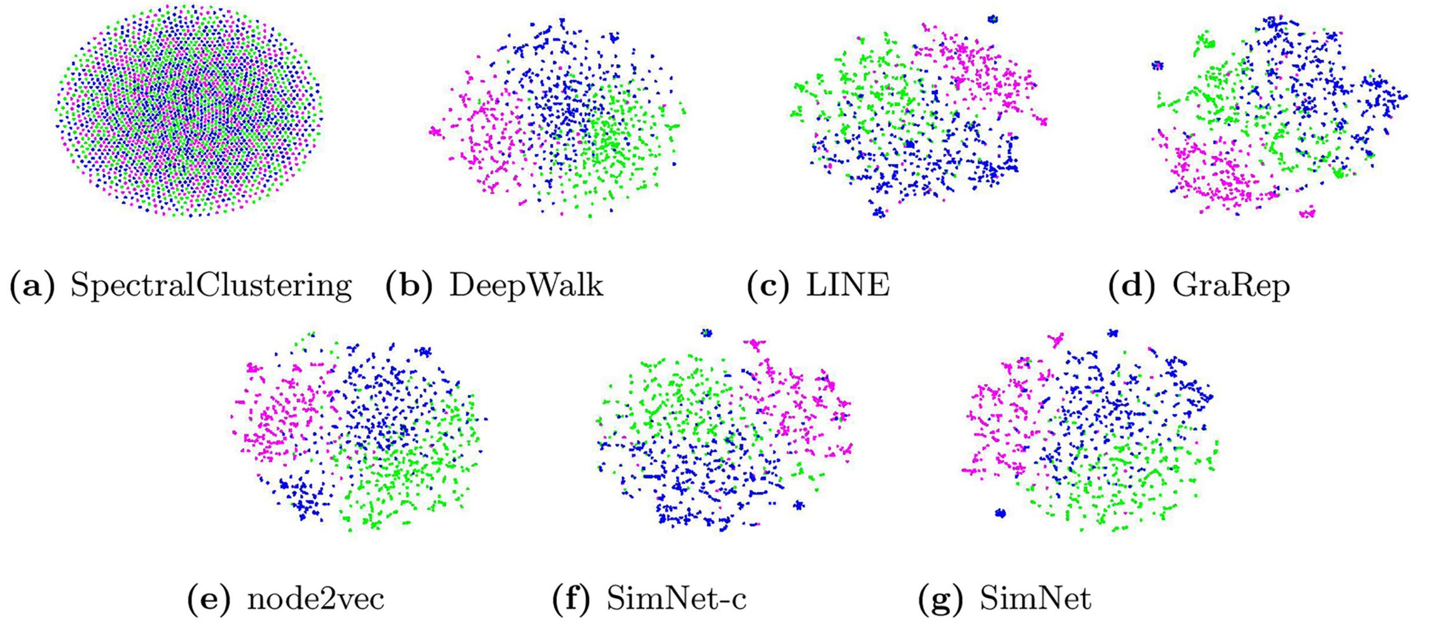

Fig 2. Visualization of author citation network. Each point indicates one author. Green: *Data Mining*, magenta: *Computer Vision* and blue:*Machine Learning*.

https://doi.org/10.1371/journal.pone.0221172.g002

the same color. This task also gives us the Kullback-Leibler divergence indicating the error between the pairwise similarities of the input and their corresponding projection in the resulting 2-dimensional mapping. Therefore a lower KL divergence indicates a better network representation.

As it can be observed from Fig 2, in Spectral Clustering the visualization is not very informative since the nodes of different groups with different colors are heavily mixed together. For both DeepWalk and node2vec there are better groupings of similar nodes, but the boundaries between the groups are still not very clear and nodes of the same color can appear at different regions to form different groups. LINE, GraRep, SimNet-c and SimNet give better results with much more precise borders between different groups, with SimNet performing the best and having the smallest KL-divergence, as shown in Table 4.

4.2.4 Link Prediction. In this last experiment, the goal is to predict the existence of an edge between any given pair of nodes. This task will reveal the edge predictability power of SimNet and the baseline methods. For this cause, we randomly hide 20% of connected node pairs and simultaneously the same number of node pairs from the disconnected ones in the Blogcatalog graph. These nodes will be our test set. The rest of the data are shuffled and the remaining connected node pairs together with an equal number of remaining disconnected pairs are kept for training. Next, the obtained representation vectors from different embedding modalities will be employed in a logistic regression method to predict the probability of edge existence for a given node pair [39]. Finally, the trained link prediction model is applied on the test set and the results are shown in Fig 3. We measure the AUC value which indicates the probability that the score of an unobserved edge is higher than that of a nonexistent one for

Table 4. Final KL divergence for the DBLP dataset.

Algorithm	SimNet	SimNet-c	node2vec	GraRep	LINE	DeepWalk
KL divergence	**0.9193**	0.9385	1.1233	0.9371	1.0054	1.0804

https://doi.org/10.1371/journal.pone.0221172.t004

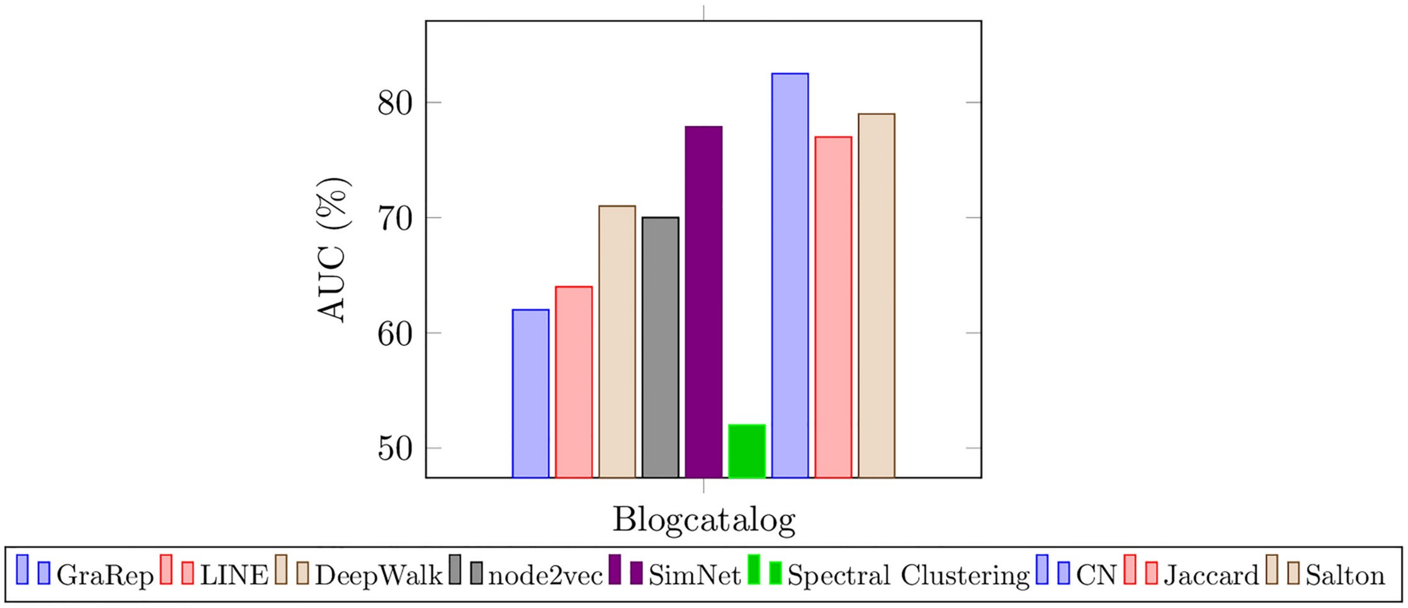

Fig 3. Experimental results on link prediction.

https://doi.org/10.1371/journal.pone.0221172.g003

SimNet as well as the baseline methods. In this specific task, we also take into account three most commonly exploited link prediction baseline: Common Neighbors (CN), Jaccard Index and Salton Index [40]. As illustrated in Fig 3, SimNet's link prediction power exceeds GraRep, node2vec, LINE, DeepWalk, and Spectral Clustering however, CN outperforms all the rest with the highest AUC score in this task.

4.3 Parameter sensitivity

In this section we will discuss the effect of varying our model's parameters. Specifically we will investigate how different choices of dimension can affect the results and also the running time of our proposed method. Fig 4 shows the clustering performance of SimNet under different dimension settings. We can see that when dimension is small, increasing the dimension can lead to an increase in the NMI score for 20-Newgroups network, and in our experiments we get the best results when dimension is chosen as 72. However a dimension larger than 72 will not make any significant improvements in the NMI score. These results demonstrate the fact that with a larger dimension we can gather complementary information of the network. However for a *d* greater than 72 no significant additional information can be obtained.

In the left part of Fig 5, we compare the NMI scores of the SimNet model and other baselines when the dimension changes. It illustrates that either in a large or small dimension SimNet consistently outperforms other baselines with representations under the same dimension. As it can be seen in the right part of Fig 5, running time of the method increases approximately linearly as the dimension increases. In Fig 6 we have measured and plotted Micro-F1 and Macro-F1 scores for varying sizes of training sets and different dimensions in Blogcatalog network. We observed that an increase in the dimension will give us an even better performance alongside the fact that this increase will not cost us much time since the right part of Fig 5 shows us an approximately linear trend.

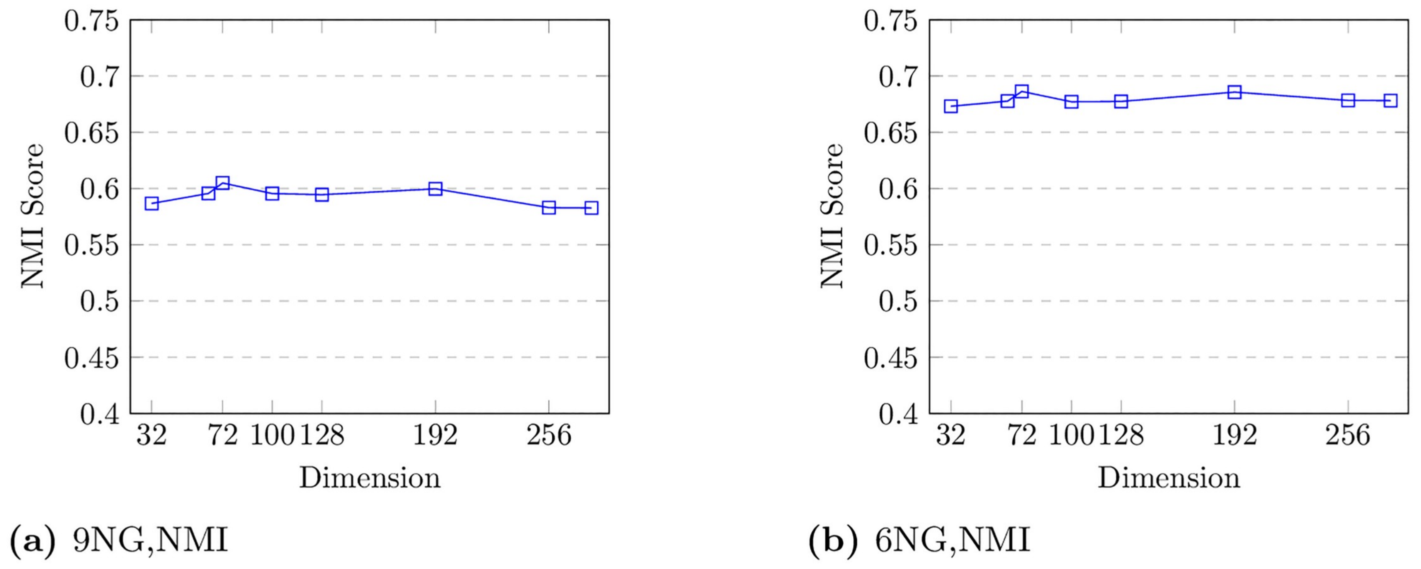

Fig 4. Performance of SimNet over different dimensions.

https://doi.org/10.1371/journal.pone.0221172.g004

Finally a complexity comparison is performed among the main competing methods in Table 5. In this tabel, *d* is the embedding dimensionality, *n* the number of nodes, *m* the number of edges, and s the number of samples used, and *t* the number of iterations in GraRep.

5 Conclusion

In this paper we proposed SimNet, a novel model for learning graph representations using latent structural information of the network. For the first time we utilized the mean commute time (MCT) measure in the learning process of network embeddings. Instead of using a conventional method involving simple constant factors for measuring closeness between nodes, we introduce the use of MCT as such a measure. By calculating MCT, we show that we are able to adequately quantify the closeness measure between nodes in a network in a principled

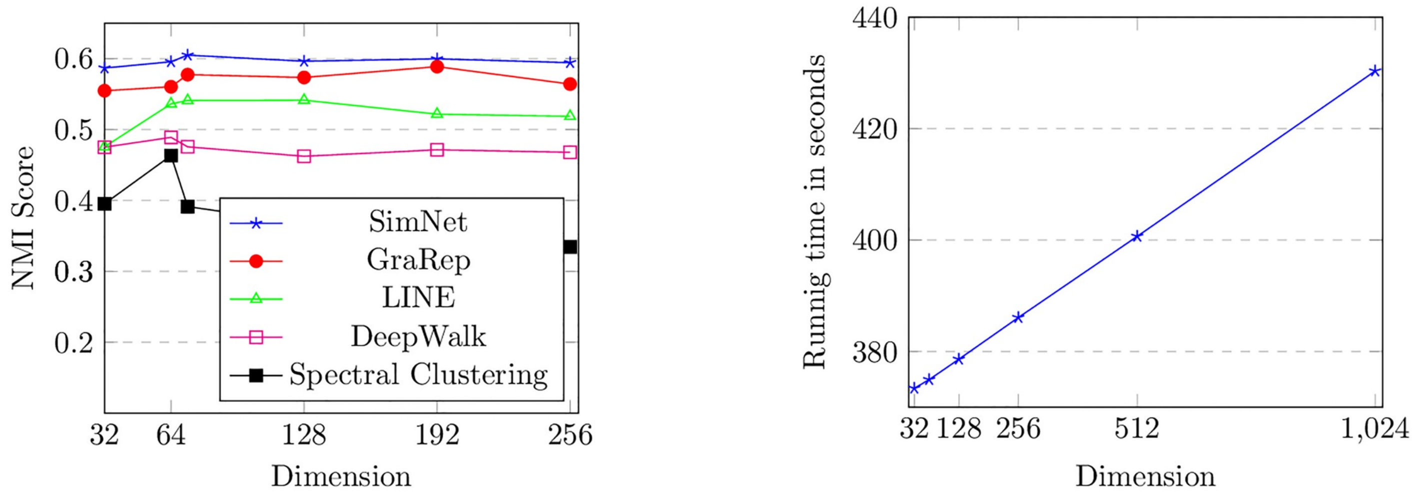

Fig 5. Left: Performance of different methods over different dimensions for 9NG network. Right: Running time.

https://doi.org/10.1371/journal.pone.0221172.g005

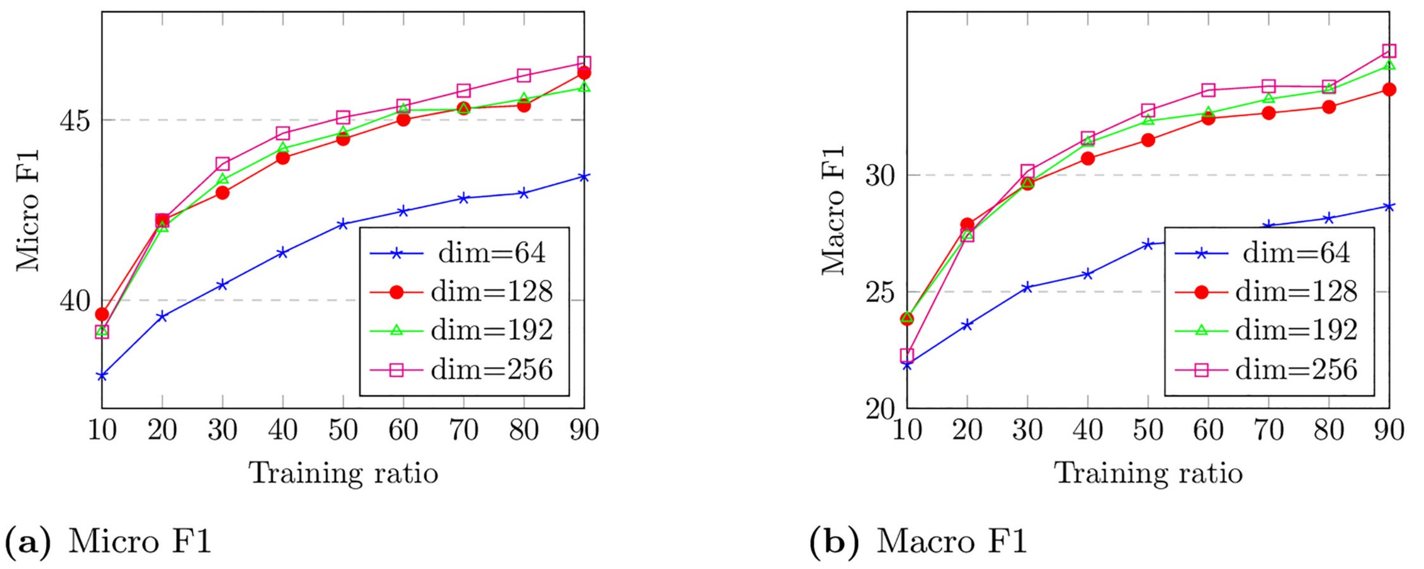

Fig 6. Performance on Blogcatalog over training ratio.

https://doi.org/10.1371/journal.pone.0221172.g006

Table 5. Comparison of network embedding methods in terms of complexity in time.

Method	SimNet	node2vec	GraRep	LINE	DeepWalk	Spectral Clustering
Time Complexity	$\mathcal{O}(n^3)$	$\mathcal{O}(dsn)$	$\mathcal{O}(tn^3)$	$\mathcal{O}(dsn)$	$\mathcal{O}(dn\log n)$	$\mathcal{O}(n^3)$

https://doi.org/10.1371/journal.pone.0221172.t005

manner, which can then be used to derive a better similarity matrix for learning network representations. By replacing MCT with other popular similarity measures when building the similarity matrix which is later used in the learning process, we prove that the well-chosen MCT-based similarity measure clearly yields better results.

We empirically demonstrate the effectiveness of our approach through extensive experiments across different tasks. One of the current challenges that we are facing is the computation of $\mathbf{L}^+$—the pseudoinverse of the Laplacian matrix involved in the learning process is expensive. In our future work we plan to optimize the computation of $\mathbf{L}^+$, and explore simple yet precise approximations of this measure for learning improved network embeddings.

Acknowledgments

We would like to thank the anonymous reviewers. Most of this work was done when the author was visiting Singapore University of Technology and Design (SUTD).

Author Contributions

Formal analysis: Moein Khajehnejad.

Investigation: Moein Khajehnejad.

Methodology: Moein Khajehnejad.

Writing – original draft: Moein Khajehnejad.

References

1. Cheng X, Su S, Zhang Z, Wang H, Yang F, Luo Y, et al. Virtual network embedding through topology-aware node ranking. ACM SIGCOMM Computer Communication Review. 2011; 41(2):38–47. https://doi.org/10.1145/1971162.1971168
2. Fortunato S. Community detection in graphs. Physics reports. 2010; 486(3-5):75–174. https://doi.org/10.1016/j.physrep.2009.11.002
3. Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T. Collective classification in network data. AI magazine. 2008; 29(3):93–93. https://doi.org/10.1609/aimag.v29i3.2157
4. Liben-Nowell D, Kleinberg J. The link-prediction problem for social networks. Journal of the American society for information science and technology. 2007; 58(7):1019–1031. https://doi.org/10.1002/asi.20591
5. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems; 2013. p. 3111–3119.
6. Levy O, Goldberg Y. Neural word embedding as implicit matrix factorization. In: Advances in neural information processing systems; 2014. p. 2177–2185.
7. Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2014. p. 701–710.
8. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q. Line: Large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web. International World Wide Web Conferences Steering Committee; 2015. p. 1067–1077.
9. Cao S, Lu W, Xu Q. Grarep: Learning graph representations with global structural information. In: Proceedings of the 24th ACM international on conference on information and knowledge management. ACM; 2015. p. 891–900.
10. Grover A, Leskovec J. node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2016. p. 855–864.
11. Wu Z, Pan S, Chen F, Long G, Zhang C, Yu PS. A comprehensive survey on graph neural networks. arXiv preprint arXiv:190100596. 2019;.
12. Zhu X, Goldberg A, Van Gael J, Andrzejewski D. Improving diversity in ranking using absorbing random walks. In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference; 2007. p. 97–104.
13. Newman ME. Finding community structure in networks using the eigenvectors of matrices. Physical review E. 2006; 74(3):036104. https://doi.org/10.1103/PhysRevE.74.036104
14. Yen L, Vanvyve D, Wouters F, Fouss F, Verleysen M, Saerens M. clustering using a random walk based distance measure. In: ESANN; 2005. p. 317–324.
15. Zhou C, Motter AE, Kurths J. Universality in the synchronization of weighted random networks. Physical review letters. 2006; 96(3):034101. https://doi.org/10.1103/PhysRevLett.96.034101 PMID: 16486704
16. Skardal PS, Taylor D, Sun J. Optimal synchronization of complex networks. Physical review letters. 2014; 113(14):144101. https://doi.org/10.1103/PhysRevLett.113.144101 PMID: 25325646
17. Wang H, Li Q, D'Agostino G, Havlin S, Stanley HE, Van Mieghem P. Effect of the interconnected network structure on the epidemic threshold. Physical Review E. 2013; 88(2):022801. https://doi.org/10.1103/PhysRevE.88.022801
18. Yang Z, Zhou T. Epidemic spreading in weighted networks: An edge-based mean-field solution. Physical Review E. 2012; 85(5):056106. https://doi.org/10.1103/PhysRevE.85.056106
19. Grinstead CM, Snell JL. Introduction to probability. American Mathematical Soc.; 2012.
20. Kemeny JG, Snell JL. Finite Markov Chains, by John G. Kemeny and Laurie Snell J.; 1960.
21. Aldous D, Fill J. Reversible Markov chains and random walks on graphs; 1995.
22. Zhang Z, Julaiti A, Hou B, Zhang H, Chen G. Mean first-passage time for random walks on undirected networks. The European Physical Journal B. 2011; 84(4):691–697. https://doi.org/10.1140/epjb/e2011-20834-1
23. Yen L, Fouss F, Decaestecker C, Francq P, Saerens M. Graph nodes clustering based on the commute-time kernel. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer; 2007. p. 1037–1045. https://doi.org/10.1007/978-3-540-71701-0_117
24. Yen L, Saerens M, Mantrach A, Shimbo M. A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2008. p. 785–793.

25. Pirotte A, Renders JM, Saerens M. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Transactions on Knowledge & Data Engineering. 2007;(3):355–369.

26. Katz L. A new status index derived from sociometric analysis. Psychometrika. 1953; 18(1):39–43. https://doi.org/10.1007/BF02289026

27. Fouss F, Yen L, Pirotte A, Saerens M. An experimental investigation of graph kernels on a collaborative recommendation task. In: Sixth International Conference on Data Mining (ICDM'06). IEEE; 2006. p. 863–868.

28. Pennington J, Socher R, Manning C. Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP); 2014. p. 1532–1543.

29. Cao S, Lu W, Xu Q. Deep neural networks for learning graph representations. In: Thirtieth AAAI Conference on Artificial Intelligence; 2016.

30. Levy O, Goldberg Y, Dagan I. Improving distributional similarity with lessons learned from word embeddings. Transactions of the Association for Computational Linguistics. 2015; 3:211–225. https://doi.org/10.1162/tacl_a_00134

31. Klema V, Laub A. The singular value decomposition: Its computation and some applications. IEEE Transactions on automatic control. 1980; 25(2):164–176. https://doi.org/10.1109/TAC.1980.1102314

32. Tian F, Gao B, Cui Q, Chen E, Liu TY. Learning deep representations for graph clustering. In: Twenty-Eighth AAAI Conference on Artificial Intelligence; 2014.

33. Page L, Brin S, Motwani R, Winograd T. The PageRank citation ranking: Bringing order to the web. Stanford InfoLab; 1999.

34. Burda Z, Duda J, Luck JM, Waclaw B. Localization of the maximal entropy random walk. Physical review letters. 2009; 102(16):160602. https://doi.org/10.1103/PhysRevLett.102.160602 PMID: 19518691

35. Shi J, Malik J. Normalized cuts and image segmentation. Departmental Papers (CIS). 2000; p. 107.

36. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ. LIBLINEAR: A library for large linear classification. Journal of machine learning research. 2008; 9(Aug):1871–1874.

37. Strehl A, Ghosh J, Mooney R. Impact of similarity measures on web-page clustering. In: Workshop on artificial intelligence for web search (AAAI 2000). vol. 58; 2000. p. 64.

38. Maaten Lvd, Hinton G. Visualizing data using t-SNE. Journal of machine learning research. 2008; 9 (Nov):2579–2605.

39. Wang H, Wang J, Wang J, Zhao M, Zhang W, Zhang F, et al. Graphgan: Graph representation learning with generative adversarial nets. In: Thirty-Second AAAI Conference on Artificial Intelligence; 2018.

40. Salton G, McGill MJ. Introduction to modern information retrieval. mcgraw-hill; 1983.