MARINE: Multi-relational Network Embeddings with Relational Proximity and Node Attributes

Ming-Han Feng
Graduate Institute of Networking and Multimedia
National Taiwan University
Taipei, Taiwan
r04944010@ntu.edu.tw

Chin-Chi Hsu
Institute of Information Science
Academia Sinica
Taipei, Taiwan
chinchi@iis.sinica.edu.tw

Cheng-Te Li
Institute of Data Science,
Department of Statistics
National Cheng Kung University
Tainan, Taiwan
chengte@mail.ncku.edu.tw

Mi-Yen Yeh
Institute of Information Science,
Research Center for IT Innovation
Academia Sinica
Taipei, Taiwan
miyen@iis.sinica.edu.tw

Shou-De Lin
Dept. of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
sdlin@csie.ntu.edu.tw

ABSTRACT

Network embedding aims at learning an effective vector transformation for entities in a network. We observe that there are two diverse branches of network embedding: for homogeneous graphs and for multi-relational graphs. This paper then proposes MARINE, a unified embedding framework for both homogeneous and multi-relational networks to preserve both the proximity and relation information. We also extend the framework to incorporate existing features of nodes in a graph, which can further be exploited for the ensemble of embedding. Our solution possesses complexity linear to the number of edges, which is suitable for large-scale network applications. Experiments conducted on several real-world network datasets, along with applications in link prediction and multi-label classification, exhibit the superiority of our proposed MARINE.

CCS CONCEPTS

• **Information systems** → **Data mining**; *Social recommendation*; • **Computing methodologies** → **Knowledge representation and reasoning**.

KEYWORDS

Homogeneous Network Embedding; Multi-relational Network Embedding; Knowledge Graph embedding

ACM Reference Format:
Ming-Han Feng, Chin-Chi Hsu, Cheng-Te Li, Mi-Yen Yeh, and Shou-De Lin. 2019. MARINE: Multi-relational Network Embeddings with Relational Proximity and Node Attributes. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3308558.3313715

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313715

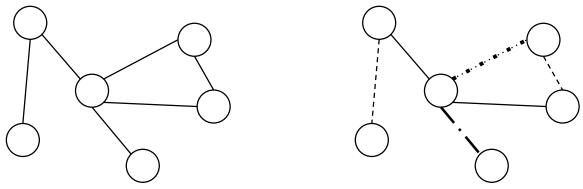


Figure 1: Homogeneous graphs (left) and multi-relational graphs (right). The latter consists of multiple types of link relationships (marked by various types of dashed lines) between node entities.

1 INTRODUCTION

The task of network embedding aims at learning a representation for each entity in a network. The idea is to *embed* some properties of a network into a vector space so that each entity is represented by a low-dimensional feature vector. Such form of representation is more ready for machines to perform further analysis, since most of the machine learning and data mining models accept unstructured feature vectors for training and prediction.

We observe that there are two branches of studies for network embedding. One is to propose an embedding method for *homogeneous graphs* (Figure 1 left) [1, 7, 8, 10, 12, 17, 29–31, 33, 36–38, 40–42, 45, 50–52], in which there is only one single type of nodes and links. The major goal of this branch of work is to preserve certain properties in the linking structure of a network. *Proximity* is the most emphasized property in the embedding of homogeneous graphs. Roughly speaking, proximity estimates the closeness between nodes, and thus a proper network embedding model shall assign similar embeddings to nodes close to each other in a graph. Preserving proximity will allow the model to accurately perform some relevant tasks such as community detection (e.g. closer nodes shall be grouped together) and link discovery (e.g. nodes with common neighbors should be linked). The other branch tries to obtain the embeddings of nodes in a *multi-relational graph*,

i.e., *knowledge graph* (Figure 1 right) [3–5, 9, 11, 13–15, 18–20, 25–28, 34, 35, 39, 44, 47–49]. A multi-relational network assumes there are different types of links (e.g. different relationships between users) in the graph, where key information depicting entities are encoded. Each directed edge can be represented by a triplet (i, k, j), where node i connects to node j via relation k. For example, a node X may connect to a node Y with relation *teacher_of*, but it can link to another node Z with relation *colleague_of*. The majority of the models for multi-relational graph embedding use an individual embedding transformation for each type of edge relation. Then based on certain algebraic operation (usually through addition or multiplication) to combine embeddings of nodes and relations, the multiple relationships in the embedding space can be recovered.

Even though a homogeneous network can be regarded as a special case of multi-relational network with single type of relation, our preliminary experiments have found that existing multi-relational graph embedding methods do not necessary perform well on homogeneous networks. It is probably because most of the existing studies do not emphasize on preserving the proximity information in the network.

Thus, in this paper we propose a multi-relational graph embedding model that preserves not only the relation information but also the proximity information, with the goal to create a unified model that is suitable for embedding both homogeneous and multi-relational networks. Although these two branches of studies respectively propose embedding methods in its own domain, to our knowledge, there has not yet been an effective unified model that works for both homogeneous and heterogeneous scenarios, which is the goal of this paper. Moreover, we also realize that directly applying the existing homogeneous proximity models shall not work for our situation since they do not consider the existence of a variety of relations. Here we propose a novel idea to regularize the importance of each embedding dimension based on the relation in generating the score function for each relation triplet. That is, we argue that some embedding dimensions are more important in predicting a type of relation while others are more important for other relations. For instance, to determine whether an edge belongs to 'classmate' relationship, a latent dimension that represents 'school attended' is probably more important than a latent dimension that represents 'favorite movies'. However, the later shall be more critical to determine the 'friendship' relationship between nodes than the former. To achieve such goal, we propose a tensor-factorization solution to regularize the importance of each embedding dimension based on the type of relations, trained based on the observed relations in the network data.

Besides a general framework to embed either type of the networks, we further intend to incorporate node attribute information into the embedding space. In practice one can usually observe attributes (e.g. demographics or metadata) of nodes in a network. Since such attributes provide semantic depiction about nodes, they shall be considered together in the embeddings of nodes. For instance, in a social network a user node can sometimes be associated with some demographic information such as age or gender. There is no reason to not incorporate such information in the node embedding. For example, we hope the embeddings of two female nodes are closer than the embeddings of two opposite-sex nodes, given the other conditions are similar. Mathematically, such non-graphical information attached to a node can be formulated as a real-valued attribute vector for each node (note that a categorical attribute can always be expanded into multiple binary ones). We notice that some network embedding studies [12, 17, 30, 40, 45, 50, 51] have considered node attributes in learning embeddings, but most of the existing approaches require at least a few labels to guide the learning. Furthermore, a very limited number of multi-relational network embedding methods [18, 19, 48] consider node attributes, as we have not yet seen any benchmark evaluation dataset that contains attributes for nodes in multi-relational networks.

In this paper, we propose the **M**ulti-relation, **A**ttribute, p**R**oximity **I**nformed **N**etwork **E**mbedding (MARINE)[1], a probabilistic network embedding model whose goal is to preserve *the proximity property, the edge relation and the node attributes information* in a low-dimensional embedding space. To achieve the goal, MARINE maximizes the likelihood of not only the existing graph structure but also the observed node attributes. To elaborate, we assign a function to score each (source_embedding, relation_embedding, target_embedding) tuple and try to maximize the difference of scores between the observed tuples and the unobserved ones. The scoring function leverages the modeling of tensor factorization and embedding translation. We show that the tensor factorization component can preserve the proximity information for each type of relation, which aligns well with the goal of most homogeneous network embedding methods. Inspired by the results of word embedding [23], we also consider the embedding translation, realizing the relation information with the relative positions among embeddings. In addition, we require the learned node embeddings to maximize the generation likelihood of node attributes, which assumes the embeddings of two nodes shall be closer if they have similar attributes. Finally, we discover an effective usage of our framework as an ensemble framework for multiple embeddings, and conduct preliminary experiments to verify its usage in this direction.

We evaluate the model based on three different applications with eight real-world datasets. The applications include link prediction, and multi-label classification. We demonstrate that the designed objective based on maximizing the above ideas allows the proposed MARINE to outperform state-of-the-art competitors in both homogeneous networks and multi-relational networks in terms of effectiveness and robustness. With the support of additional attributes, MARINE can further boost its performance. We also provide the visualization on various embedding results as another form of evaluation. Last but not least, we show MARINE of complexity linear in the number of nodes or edges, which enables the efficient implementation in large-scale network data.

We summarize the contributions of this paper, as listed below:

- We propose a unified unsupervised embedding method, MARINE, to enhance the existing solutions for multi-relational network embedding models by preserving the proximity and attributes information in addition to various types of relations.
- Our proximity model is specifically designed for multi-relational data, which allows the dynamic adjustment of importance weights for embedding dimensions given different relations.

[1]The code is available via https://github.com/ntumslab/MARINE

- MARINE enjoys the running time linear to the number of nodes and edges. Our framework also enables the embedding ensembles. We conduct thorough experiments to justify the above claims.
- We empirically conduct several experiments on real-world networks, to examine how MARINE performs in the applications of link prediction and multi-label classification. The results show the superiority capability of MARINE in learning better network representations.

The rest of the paper is organized as follows. Section 2 briefly summarizes related work. Section 3 introduces MARINE in detail. Section 4 reports the experimental results. Finally we discuss and conclude our work in Section 5.

2 RELATED WORK

We first discuss existing studies in homogeneous network embedding. Most of the past models focus on preserving the proximity property as it captures the neighborhood topology information of nodes. Some important network mining tasks, such as link prediction and community detection, can be addressed by discovering proximity relations among nodes. Specifically, between any node pair (i, j), the h-order proximity estimates either the number of h-hop shortest paths from i to j or the probability of a h-hop random walk from i to j. Relevant studies based on proximity assume higher similarity between the embedding vectors of two nodes with high proximity. Singular value decomposition or matrix factorization is commonly used to learn such embeddings [1, 8, 12, 29, 36, 38, 40, 42, 50, 51] These models rely on a matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ pre-computed based on certain proximity estimation. Then let $\boldsymbol{A}$ be approximated by a low-rank matrix multiplication. Some models [7, 17, 41] consider deep neural network structures like auto-encoders to exploit their non-linear mappings between graph topology and embedding vectors. Inspired by recent word embedding algorithms [23], another class of homogeneous graph embedding methods [10, 30, 31, 52] gathers random walk sequences as training set. The similarity of two nodes is expected to be high if their positions are close in most of the random walk sequences. There are some common concerns about the existing approaches. First, several models [8, 12, 40, 42, 50, 51] require to factorize a dense matrix, and thus cannot be scale to large graphs as the complexity reaches $O(N^2)$. Second, some approaches [1, 10, 12, 30, 31, 38, 40–42, 45] assumes symmetric relations, that is, $\boldsymbol{x}_i^\top \boldsymbol{x}_j = \boldsymbol{x}_j^\top \boldsymbol{x}_i$ for edge (i, j), and therefore cannot be used to handle directed graphs. Third, as having been pointed out in LINE [37], it is hard for random walk-based methods to explain explicitly what graph properties are preserved in the embedding space. Our model is developed to address the above issues.

On the other hand, multi-relational network embedding approaches often focus on learning a vector $\boldsymbol{r}_k$ that represents the embedding of edge relation k. Based on different assumptions, previous studies propose various scoring functions given a relation embedding $\boldsymbol{r}_k$ and the embeddings $\boldsymbol{x}_i, \boldsymbol{x}_j$ of two adjacent nodes. It is further assumed that an observed edge (i, k, j) with nodes (i, j) and relation k shall obtain a higher score value than an unobserved edge, and a multi-relational graph can be regarded as a set of multi-relational edges. There are two major classes of the

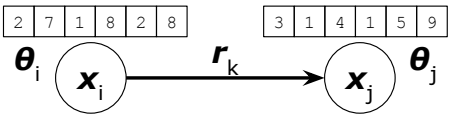


Figure 2: The notations in MARINE. There is an edge with relation k linking from node i to node j, whose attributes are θ_i and θ_j, respectively.

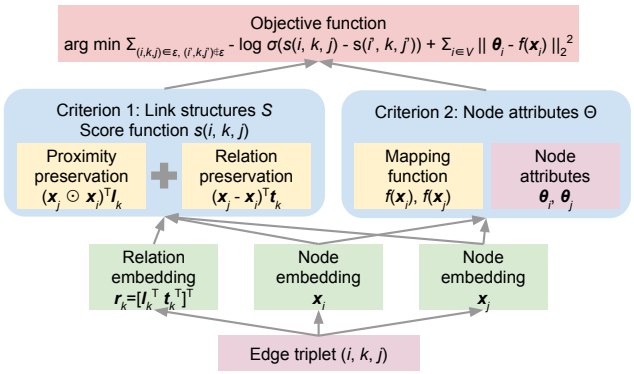


Figure 3: The overview of our proposed MARINE.

scoring functions. One is to build a bilinearity-like form $\boldsymbol{x}_i^\top \boldsymbol{R}_k \boldsymbol{x}_j$ or its non-linear extensions [3, 13, 27, 28, 34, 35, 39, 49] to learn the correlations between nodes (i, j) with respect to a relation k. By viewing an embedding vector as a point in multi-dimensional space, another set of scoring functions aim to constrain the positions of every embedding points [4, 5, 9, 11, 14, 15, 19, 20, 26, 44, 47, 48]. Given the transition scale $\boldsymbol{r}_k$ of a specific edge relation k, let it be fitted by the translation scale from point $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$. In other words, they try to minimize distance $\|f(\boldsymbol{x}_i) + \boldsymbol{r}_k - g(\boldsymbol{x}_j)\|$, where certain mapping functions f and g are added to map the node embedding space to the edge embedding space. We refer readers to a survey paper [25] that introduces recent multi-relational network embedding methods in details. To the best of our knowledge, the existing multi-relational network embedding works has not yet considered the proximity information.

3 METHODOLOGY

3.1 Problem Definition

Since homogeneous graph is a special case of multi-relational graph and undirected graph is a special case of directed graph (i.e. undirected graph can be regarded as a bi-directional directed graph), here our task is formulated over the general directed multi-relational graph. We also assume the nodes are associated with some attributes. Out task can be formulated as:

Definition 3.1. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ denote a directed multi-relational graph, where $\mathcal{V}$ is a set of nodes, $\mathcal{R}$ is a set of edge relations, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ is a set of edges. Each node $i \in \mathcal{V}$ is appended with attributes represented as a vector $\boldsymbol{\theta}_i \in \mathbb{R}^a$.

Then our goal can be defined as:

Definition 3.2. Given graph $\mathcal{G}$, we would like to learn an embedding vector $\boldsymbol{x}_i \in \mathbb{R}^d$ ($d \ll |\mathcal{V}|$) for each node $i \in \mathcal{V}$ in an *unsupervised* manner.

Furthermore, for each type of edge in a multi-relational graph, we will learn an embedding vector $\boldsymbol{r}_k$ for its relation k. Figure 2 describes the notations. We expect the embedding would capture not only the proximity information but also the the information from link relation as well as the node attributes. In the next sections, we begin to introduce how MARINE learns embeddings with desired properties.

3.2 Design Criteria

Since the learning of embedding is generally unsupervised, here we formulate the task as an optimization problem. As stated previously, we would like to preserve both the link structure and node attributes information. The link structure information can be further decomposed into two components: proximity and relation. We will then formulate the criteria in a probabilistic form:

Criterion 1. **Link Structures** $\mathcal{S}$ preservation: Given a relation $k \in \mathcal{R}$, let an observed edge $(i,k,j) \in \mathcal{E}$ and a non-existing edge $(i',k,j') \notin \mathcal{E}$. Our model would like to assign scores $s(i,k,j) > s(i',k,j')$. Such score reflects the query: *how likely there is a relation of type k from node i point to j*.

Criterion 2. **Node Attributes** Θ preservation: We assume for any two nodes (i,j), the distance between their embeddings $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is positively correlated with the distance between their attributes $(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$.

Given the two design criteria, our task becomes a maximum a posteriori (MAP) problem to find the optimal embedding $\boldsymbol{x}$ and $\boldsymbol{r}$, given $\mathcal{S}$ and Θ. Using Bayes' Rule:

$$\begin{aligned}\arg\max_{\boldsymbol{x},\boldsymbol{r}} & p(\boldsymbol{x},\boldsymbol{r} \mid \mathcal{S},\Theta) \\ &= \arg\max_{\boldsymbol{x},\boldsymbol{r}} \frac{p(\mathcal{S},\Theta \mid \boldsymbol{x},\boldsymbol{r})p(\boldsymbol{x},\boldsymbol{r})}{p(\mathcal{S},\Theta)} \\ &= \arg\max_{\boldsymbol{x},\boldsymbol{r}} \underbrace{p(\mathcal{S},\Theta \mid \boldsymbol{x},\boldsymbol{r})}_{\text{Likelihood}} \underbrace{p(\boldsymbol{x},\boldsymbol{r})}_{\text{Prior}}. \end{aligned} \tag{1}$$

The denominator $p(\mathcal{S},\Theta)$ can be eliminated due to no involvement in the maximization in $\boldsymbol{x},\boldsymbol{r}$. To simplify the analysis, here we assume no prior knowledge about the target embeddings, and hence the prior distribution $p(\boldsymbol{x},\boldsymbol{r})$ is assumed to be *uniform* and can be ignored in the optimization process. Nevertheless, generalization to non-uniform prior is possible with careful design on the form of distribution. To make the joint probability p tractable, we adopt an independence assumption $\mathcal{S} \perp \Theta$ between $\mathcal{S}$ and Θ. We also assume that the edge relation embedding $\boldsymbol{r}$ is independent of to the node attributes given the node embedding, i.e., $\boldsymbol{r} \perp \Theta \mid \boldsymbol{x}$. Therefore we can simplify (1) as:

$$\begin{aligned} p(\mathcal{S},\Theta|\boldsymbol{x},\boldsymbol{r}) &= p_{\mathcal{S}}(\mathcal{S}|\boldsymbol{x},\boldsymbol{r})p_{\Theta}(\Theta|\boldsymbol{x},\boldsymbol{r}) \\ &= \underbrace{p_{\mathcal{S}}(\mathcal{S}|\boldsymbol{x},\boldsymbol{r})}_{\text{Criterion 1}} \underbrace{p_{\Theta}(\Theta|\boldsymbol{x})}_{\text{Criterion 2}}. \end{aligned} \tag{2}$$

We will elaborate $p_{\mathcal{S}}$ and p_{Θ} in Section 3.3 and 3.5.

The overview of our proposed MARINE is presented in Figure 3. Given an edge triplet, MARINE learns their representations based on the observations on link structures $\mathcal{S}$ and node attributes Θ. We will elaborate the details of each component in the following subsections.

3.3 Preserving Link Structures $\mathcal{S}$

$p_{\mathcal{S}}$ in Criterion 1 depends on the link structure. To fulfill this criterion, we formulate the following likelihood function with respect to edges:

$$\begin{aligned} p_{\mathcal{S}}(\mathcal{S} \mid \boldsymbol{x},\boldsymbol{r}) &= \prod_{\substack{(i,k,j)\in\mathcal{E} \\ (i',k,j')\notin\mathcal{E}}} \Pr\left(s(i,k,j) > s(i',k,j') \mid \boldsymbol{x},\boldsymbol{r}\right) \\ &= \prod_{\substack{(i,k,j)\in\mathcal{E} \\ (i',k,j')\notin\mathcal{E}}} \sigma\left(s(i,k,j) - s(i',k,j')\right). \end{aligned} \tag{3}$$

Inspired by [32], (3) can be also approximated by differentiable sigmoid functions $\sigma(z) = \frac{1}{1+\exp(-z)}$. As we maximize the likelihood (3), the gap between $s(i,k,j)$ and $s(i',k,j')$ is enlarged consequently, which satisfies the requirement of Criteria 1. The scoring function s is defined based on the corresponding embeddings $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_{i'}, \boldsymbol{x}_{j'}, \boldsymbol{r}_k$ (see (4)) to preserve the proximity and relation information, as will be elaborated next.

3.4 Score Function Design

Criterion 1 demands the designed score function s to generate higher scores for an observed edge $(i,k,j) \in \mathcal{E}$, and lower ones for the non-existing counterpart $(i',k,j') \notin \mathcal{E}$. To incorporate both proximity and relation information, we design the score function as follows:

$$s(i,k,j) = \underbrace{(\boldsymbol{x}_j \odot \boldsymbol{x}_i)^\top \boldsymbol{l}_k}_{\text{Proximity preservation}} + \underbrace{(\boldsymbol{x}_j - \boldsymbol{x}_i)^\top \boldsymbol{t}_k}_{\text{Relation preservation}}, \tag{4}$$

where $\odot$ denotes the Hadamard product. We express edge relation embedding $\boldsymbol{r}_k = \begin{bmatrix}\boldsymbol{l}_k^\top & \boldsymbol{t}_k^\top\end{bmatrix}^\top \in \mathbb{R}^{2d}$ as the concatenation of two embedding vectors $\boldsymbol{l}_k \in \mathbb{R}^d, \boldsymbol{t}_k \in \mathbb{R}^d$, to achieve two different preservation targets in (4). The first term of (4) is essentially a tensor factorization to preserve proximity information. The second term implies that translation direction between embeddings shall be closer for the same edge relation. We do not set a relevant weight between two terms of (4), since the weight is absorbed in $\boldsymbol{l}_k$ or $\boldsymbol{t}_k$. In the following paragraphs, we will discuss the insight of (4).

3.4.1 Proximity Preservation. Most previous studies on homogeneous network embedding adopt matrix factorization (see Section 2) to learn embedding. Under such framework, given the embeddings $\boldsymbol{x}_i, \boldsymbol{x}_j$ of node pair (i,j), the existing studies mostly exploit inner product $a_{ij} = \boldsymbol{x}_i^\top \boldsymbol{x}_j = \sum_{u=1}^d x_{iu}x_{ju}$ to fit the proximity-related value a_{ij}. Random walk-based solutions also exploit the inner product operation to learn the similarities among embeddings. Using $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ presumes a strong assumption that each dimension in the embedding space is equally important. Here we argue that for different types of relations, the importance or weight of each dimension to evaluate the proximity quality shall be different. We take a multi-relation social network where nodes are people as the example. Suppose that a single embedding dimension implies "salary" of

a node. Let us compare two different relations: "colleagues" and "friends". This particular dimension shall play a more important role to determine the former relationship than it is to the later one. It is because two colleagues are more likely to earn salary in the similar range than two friends. To model the influence of each dimension of embedding given different relations, we expand two-way matrix factorization into the *three-way tensor factorization*. The axes in this tensor indicate subject node i, edge relation k and object node j. Each binary element is set to 1 for edge $(i,k,j) \in \mathcal{E}$, and 0 for $(i,k,j) \notin \mathcal{E}$ Similar to the assumption in matrix factorization, a low rank d is assumed for this tensor. The score of the embeddings of tuples $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{l}_k$ is evaluated by a tensor product:

$$\sum_{u=1}^{d} x_{iu} x_{ju} l_{ku} = (\boldsymbol{x}_i \odot \boldsymbol{x}_j)^\top \boldsymbol{l}_k. \tag{5}$$

Note that in our design, the importance of each embedding dimension is regularized by the learned relation embedding $\boldsymbol{l}_k$, meaning that different embedding dimensions can play different roles in determining whether a relation shall exist between two nodes. The next question we would like to answer is that *why such design can satisfy the proximity constraint*. Next we would like to establish the connections of our model to the first and second order proximity defined in homogeneous networks.

Below we first provide formal definitions to such proximity.

Definition 3.3. Given a pair of nodes (i,j), *the first-order proximity* for their embeddings $\boldsymbol{x}_i, \boldsymbol{x}_j$ means a high score between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the embedding space, if there is an edge adjacent to i and j in the network.

Definition 3.4. Given a pair of nodes (i,j), *the second-order proximity* for their embeddings $\boldsymbol{x}_i, \boldsymbol{x}_j$ means a high score between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the embedding space, if nodes (i,j) share the same neighbor set in the network.

Previous studies often adopt the inner product $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ or the squared Euclidean distance $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2$ as the score function to achieve proximity preservation. Our design criteria guarantees to *explicitly* preserve the first-order proximity in a homogeneous network. By Criterion 1, given fixed relation k, we have $s(i,k,j) > s(i',k,j')$ for observed edge $(i,k,j) \in \mathcal{E}$ and unobserved edge $(i',k,j') \notin \mathcal{E}$. Modeling Criterion 1 in MARINE likelihood (3) satisfies the first-order proximity. Here we would like to prove that applying (5) to a homogeneous network (or a multi-relation network with single type of relation) can *implicitly* keep the second-order proximity.

Lemma 3.5. *Considering (5), MARINE* ***implicitly*** *preserves the* ***second-order proximity*** *in a homogeneous network.*

Proof. Given a fixed relation k, let two nodes (i,j) have the same set of direct successors. That is, for any node v ($v \neq i, v \neq j$), we have either $(i,k,v) \in \mathcal{E}$, $(j,k,v) \in \mathcal{E}$ or $(i,k,v) \notin \mathcal{E}$, $(j,k,v) \notin \mathcal{E}$. With the progress of training, the objective function eventually will cause observed edges and non-existing edges to become separable in scores. Then there must exist a threshold ϵ such that

$$s(i,k,v) \geq \epsilon \text{ as } (i,k,v) \in \mathcal{E},$$
$$s(i',k,v') < \epsilon \text{ as } (i',k,v') \notin \mathcal{E}$$

, and eventually we can obtain $\epsilon = 0$. If we consider only the tensor factorization part $s(i,k,v) = (\boldsymbol{x}_i \odot \boldsymbol{x}_v)^\top \boldsymbol{l}_k = (\boldsymbol{l}_k \odot \boldsymbol{x}_v)^\top \boldsymbol{x}_i$, then for the above-mentioned node v, we have:

$$s(i,k,v) \cdot s(j,k,v) = (\boldsymbol{l}_k \odot \boldsymbol{x}_v)^\top \boldsymbol{x}_i \boldsymbol{x}_j^\top (\boldsymbol{l}_k \odot \boldsymbol{x}_v) \geq \epsilon^2 = 0,$$

where $\boldsymbol{l}_k \odot \boldsymbol{x}_v \in \mathbb{R}^d$. It implies that matrix $\boldsymbol{x}_i \boldsymbol{x}_j^\top$ is a positive semi-definite matrix that has the non-negative trace property:

$$\boldsymbol{x}_i^\top \boldsymbol{x}_j = \mathrm{tr}(\boldsymbol{x}_i \boldsymbol{x}_j^\top) \geq 0.$$

In practice, with sufficient number of training epochs, our scores (4) can move farther from the zero threshold such that $s(i,k,v) \cdot s(j,k,v) \gg 0$. It could lead to a high inner product $\boldsymbol{x}_i^\top \boldsymbol{x}_j$. In other words, $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ could be close to each other in the embedding space, if nodes (i,j) share direct successors. Similar proof can applied to direct predecessors $(v,k,i), (v,k,j)$. □

3.4.2 Relation Preservation. This part was inspired by a word embedding approach [23] illustrating the distribution of word embeddings in a two-dimensional space. It implicitly reveals the two clusters of words, even though no label is provided to guide the learning. Between two clusters, there exists an implicit relation controlling the translation direction from one point to its adjacent point in the other cluster. In contrast to implicit relation learning in [23], we choose to put an explicit embedding translation term in our score metric (4), thanks to the availability of the relation information in multi-relational networks. Specifically, we expect that given an edge (i,k,j), the vector direction from $\boldsymbol{x}_i$ to $\boldsymbol{x}_j$ shall be close to the relation embedding vector $\boldsymbol{t}_k$. The idea can be modeled by the inner product:

$$(\boldsymbol{x}_j - \boldsymbol{x}_i)^\top \boldsymbol{t}_k. \tag{6}$$

In real-world multi-relational networks, similar type of nodes might connect to similar types of others via similar relation (e.g. sales to customers). They are more likely to be clustered together, as shown in our visualization experiment.

As an additional bonus, the embedding translation term brings an asymmetric similarity metric, that is, $s(i,k,j) \neq s(j,k,i)$ by our definition (4). Notice that the tensor factorization term is symmetric since $(\boldsymbol{x}_i \odot \boldsymbol{x}_j)^\top \boldsymbol{l}_k = (\boldsymbol{x}_j \odot \boldsymbol{x}_i)^\top \boldsymbol{l}_k$. In a directed network, the direction of an edge determines an asymmetric relation between two nodes. Directed graphs are quite popular in real world. For instance, the 'following' information in an online social network is directional, meaning the opposite does not always hold. We observe that a few prior approaches [31, 38, 41] cannot leverage the edge direction information due to symmetric scoring functions. Using the asymmetric nature of the embedding translation term (6), we are allowed to model embeddings for directed graphs.

3.5 Node Attributes Θ

Based on Criteria 2, we need to design p_Θ such that node attributes $\boldsymbol{\theta} \in \mathbb{R}^a$ can affect the learning of embeddings $\boldsymbol{x} \in \mathbb{R}^d$. To satisfy the criterion, we define a bijective mapping function $\boldsymbol{\theta} = f(\boldsymbol{x})$ to bridge $\boldsymbol{\theta}$ and $\boldsymbol{x}$. Thus embedding $\boldsymbol{x}_i = f^{-1}(\boldsymbol{\theta}_i)$ could approach $\boldsymbol{x}_j = f^{-1}(\boldsymbol{\theta}_j)$ if the corresponding attribute vectors $\boldsymbol{\theta}_i \approx \boldsymbol{\theta}_j$. Here

we impose a multi-variate normal distribution for $\boldsymbol{\theta}$:

$$p_\Theta(\Theta \mid \boldsymbol{x}) = \prod_{i \in \mathcal{V}} \mathcal{N}(\boldsymbol{\theta}_i \mid f(\boldsymbol{x}_i), \frac{1}{2\alpha} I)$$
$$= \prod_{i \in \mathcal{V}} \left(\frac{\pi}{\alpha}\right)^{-\frac{a}{2}} \exp\left(-\alpha \|\boldsymbol{\theta}_i - f(\boldsymbol{x}_i)\|_2^2\right). \quad (7)$$

where $\mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \Sigma)$ denotes a multi-variate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ. Since the maximum probability density appears in the mean for normal distributions, (7) can model the approximate mapping $\boldsymbol{\theta} \approx f(\boldsymbol{x})$ with uncertainty as variance $\frac{1}{2\alpha}$. Any differentiable bijective functions f are allowed, though in our experiments a naive linear mapping $f(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ works well, which alleviates the need to tune potential hyperparameters of complex functions f. In the following sections, we use notation $\boldsymbol{\phi} \in \{\boldsymbol{W}, \boldsymbol{b}\}$ as the parameters in function f.

3.6 Objective Function and Optimization

Combining (3) and (7), when minimizing the negative logarithm of (2), we finally aim at solving:

$$\arg\min_{\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{\phi}} L = \sum_{\substack{(i,k,j) \in \mathcal{E} \\ (i',k,j') \notin \mathcal{E}}} -\log \sigma\left(s(i,k,j) - s(i',k,j')\right)$$
$$+ \alpha \sum_{i \in \mathcal{V}} \|\boldsymbol{\theta}_i - f(\boldsymbol{x}_i)\|_2^2. \quad (8)$$

We remove the constant terms in parameters $\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{\phi}$. For large-scale graphs the complexity is a concern, as there may exist quadratic number of edge pairs $((i,k,j),(i',k,j'))$ in (8). Fortunately we can adopt the commonly used *negative sampling* technique to sample a small subset of negative edge pairs for training. Referring to the idea in [32], we sample $Q \in \mathbb{N}$ negative edges $(i',k,j') \notin \mathcal{E}$ for each positive edge $(i,k,j) \in \mathcal{E}$, given relation k fixed. Viewing an edge pair $((i,k,j),(i',k,j'))$ as a training instance, we can apply Stochastic Gradient Descent (SGD) to efficiently update $\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{\phi}$ in differentiable (8). For our objective function $\tilde{L}$, with respect to one training instance $((i,k,j),(i',k,j'))$, SGD optimizes a parameter z using the repetition of following update rule:

$$z \leftarrow z - \eta \frac{\partial \tilde{L}}{\partial z}, \quad (9)$$

where η denotes the learning rate that can be automatically adjusted by certain optimizers like Adam [16]. Here we present the derivatives of objective function $\tilde{L}$ over $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{l}_k, \boldsymbol{t}_k, \boldsymbol{\phi}$ for SGD optimization.

$$\frac{\partial \tilde{L}}{\partial \boldsymbol{x}_i} = -\sigma\left(s(i',k,j') - s(i,k,j)\right)\left((\boldsymbol{x}_j \odot \boldsymbol{l}_k) - \boldsymbol{t}_k\right)$$
$$- 2\alpha\left(\boldsymbol{\theta}_i - f(\boldsymbol{x}_i)\right)\frac{\partial f}{\partial \boldsymbol{x}_i} \quad (10)$$

$$\frac{\partial \tilde{L}}{\partial \boldsymbol{x}_j} = -\sigma\left(s(i',k,j') - s(i,k,j)\right)\left((\boldsymbol{x}_i \odot \boldsymbol{l}_k) + \boldsymbol{t}_k\right)$$
$$- 2\alpha\left(\boldsymbol{\theta}_j - f(\boldsymbol{x}_j)\right)\frac{\partial f}{\partial \boldsymbol{x}_j} \quad (11)$$

$$\frac{\partial \tilde{L}}{\partial \boldsymbol{l}_k} = -\sigma\left(s(i',k,j') - s(i,k,j)\right)\left(\boldsymbol{x}_j \odot \boldsymbol{x}_i\right) \quad (12)$$

$$\frac{\partial \tilde{L}}{\partial \boldsymbol{t}_k} = -\sigma\left(s(i',k,j') - s(i,k,j)\right)\left(\boldsymbol{x}_j - \boldsymbol{x}_i\right) \quad (13)$$

$$\frac{\partial \tilde{L}}{\partial \boldsymbol{\phi}} = -2\alpha\left(\boldsymbol{\theta}_j - f(\boldsymbol{x}_j)\right)\frac{\partial f}{\partial \boldsymbol{\phi}} \quad (14)$$

3.7 Model Analyses

3.7.1 Complexity. The execution time of MARINE is determined by its SGD optimization as discussed previously. Given Q negative sample edges for each positive edge, we have $Q|\mathcal{E}|$ positive-negative edge pairs to optimize our parameters. Therefore the overall time complexity is $O(Q|\mathcal{E}|\omega)$ where ω is the number of epochs. Our experiments show $\omega = 500$ is sufficient for convergence. Since $Q \ll |\mathcal{E}|$ ($Q = 5$ in our experiments) and $\omega \ll |\mathcal{E}|$ in practice, MARINE enjoys time complexity that is linear to the number of positive edges $|\mathcal{E}|$, satisfying the scalability requirement. The training data contain $|\mathcal{E}|$ edges and $a|\mathcal{V}|$ attributes where each node has a attributes. As $a \ll |\mathcal{V}|$, MARINE possesses space complexity $O(|\mathcal{E}| + a|\mathcal{V}|)$ is efficient for practical applications.

3.7.2 Hyparameters. Observing the objective function in (8), there is only one hyperparameter α to be tuned by human, while all the other parameters can be learned during optimization. Note that a more complex mapping function f could lead to more hyperparameters. However our experiment results suggest that a linear mapping function without any hyperparameter can yield satisfiable results.

3.7.3 Embedding Ensemble. Even the external node attributes are not available, (7) can still be effective if we extract node attributes from the input graphs (e.g. degree information). Furthermore, the formula can be used as an embedding ensemble framework. That is, to treat the embedding outcomes of one model as the existing attributes Θ of the nodes. As will be described in the next section, positive experiment results support such idea.

4 EXPERIMENTS

4.1 Experimental Setup

Following a similar setup as most of network embedding studies, we conduct *Link Prediction* and *Multi-label Classification* tasks for evaluations. Three widely compared classic models (DeepWalk, LINE, TransE) and two of the start-of-the-art models (SDNE, ProjE) in network embedding are chosen to compare with MARINE. Note that DeepWalk, LINE and SDNE are models for homogeneous networks, while the other two are models for multi-relational networks. All the embedding dimensions are fixed to **128**; for the methods requiring negative sampling, we set the number of negative samples to **five** times of the positive samples (i.e, $Q = 5$) as mentioned in [10, 37]. Other hyperparameters are set to the default values suggested by the original papers.

- Competitors from homogeneous network embedding:
  - **DeepWalk** [31]: walk length $t = 80$; number of walks per source $\gamma = 10$; window size $w = 10$.

- **LINE** [37]: total number of samples $T = 10$ billion; 64-dimensional first-order proximity embeddings concatenated with 64-dimensional second-order proximity embeddings.
- **SDNE** [41]: first-order proximity weight $\alpha = 0.2$; reconstruction penalty $\beta = 10$; dimensionality of hidden layers $D = [1000]$.

- Competitors from multi-relational network embedding:
  - **TransE** [4]: margin in score function $\gamma = 1$; dissimilarity measurement $d = L_2$.
  - **ProjE** [34]: the original algorithm, denoted as 'ProjE-ori' in our tables, uses 25% of the total entities as negative samples suggested by the authors. However, to be fair in the comparison, we further compare with the performance of five negative samples per training instance (denoted as 'ProjE-Q5'). In addition, the *list-wise* version is selected since it reaches the best performance as reported in the paper.

While training our model, Adam optimizer [16] with suggested values (learning rate 0.001, parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$) is adopted. In all experiments, MARINE is trained for at most 500 epoches. All embeddings are initialized from samples of a uniform distribution ranged in $[-0.05, 0.05]$. For all experiment tasks, MARINE learns the representation without using node attributes, except for the experiments in Section 4.4.

4.2 Unsupervised Link Prediction

We use the same evaluation protocol as proposed in [4]. For each pair of an edge relation k and an object node j, we rank all triplets $(i, k, j)\ \forall\ i$ in the descending order of scores, based on the scoring function $s(i, k, j)$ of each model. The same procedure is then repeated for $(i, k, j)\ \forall\ j$. The performance is evaluated with respect to the hit rate at 10 and Mean Reciprocal Rank (MRR). According to the objective functions in the referenced papers, we review the scoring functions of the baseline models in Table 1.

We use three datasets for link prediction evaluation:

- Datasets used in previous approaches for multi-relational networks:
  - **WordNet (WN18)** [24] is a lexical database for English. It groups English words into sets of synonyms called synsets, and records a number of relations among these synonym sets. The relationships include *hypernyms*, *hyponyms*, *meronym*, and other lexical relations. WN18 has 18 relations, 40,943 nodes, and 141,442 edges.
  - **Freebase (FB15K)** [2] is a large collaborative knowledge base of general facts. For instance, a triplet *(Curtis Armstrong, place_of_birth, Detroit)* indicates that a human entity '*Curtis Armstrong*' and a city entity '*Detroit*' have the relationship '*place_of_birth*'. FB15k has 14,951 nodes, 483,142 edges, and 1,345 relations.
- Datasets used in previous approaches for homogeneous networks:
  - **Protein-Protein Interactions (PPI)** [6] is the dataset used in [10], which is the subgraph of the PPI network for Homo Sapiens. The original network has 3,890 nodes (proteins), 76,584 edges (interactions), and 50 labels representing the biological states. For link prediction task, we randomly remove 10% of existing edges for testing data while ensuring the residual graph is connected.

The experimental results are listed in Table 2. The relatively weak performance of DeepWalk, LINE, and SDNE is expected, since the information of relations (and their direction in edges) is not utilized. The results show that MARINE significantly outperforms all other competitors when 5 negative links are sampled. It is competitive to the state-of-the-art ProjE-ori which samples 25 percentage of the negative samples. We further increase the negative sample size of MARINE to 100, denoted as MARINE-Q100, and found the performance can further be improved. We also realized that our model performs significantly better on homogeneous network PPI, which reflects that the current multi-relational solution might not be the best solution for homogeneous network embedding.

Note that ProjE's performance drops significantly while training with fewer number of negative samples. The requirement of massive negative samples in ProjE results in quadratic training complexity, which is less suitable for large-scale networks.

From Table 2, we can find that the proximity preservation part in MARINE dominantly contributes to the performance. As the relation preservation part used in MARINE, the performance of MARINE in WN18 and FB15K, where edge muti-relations are provided, is relatively better than that in PPI. Besides, the overall performance can be significantly enhanced with the exploitation of both terms together. The relation term mainly focuses on learning different relations, while the link prediction evaluates the model to distinguish different nodes.

4.3 Multi-label Classification

We follow the same evaluation protocol as in [10]: The embeddings generated unsupervisedly by each model are used as the input training features to train a multi-label classification, using one-vs-rest logistic regression with L2 regularization[2]. The training and testing data are split equally, and the procedure is repeated 10 times to get the average. Micro F1-score and Macro F1-score are used to evaluate the performance. The regularization strength parameter $C = 1.0$ is not tuned, leaved as default value.

Three datasets are used for multi-label classification evaluation. The first two networks are homogeneous and the last one contains multiple relations.

- Homogeneous graphs including labels:
  - **PPI** [6] has been introduced in previous subsection.
  - **Wikipedia** [22] is the first billion bytes of the English Wikipedia dump. The processed data [10] builds a co-occurrence network of words, and the labels represent the Part-of-Speech tags. The network has 4,777 nodes, 184,812 edges and 40 labels.
- Multi-relational networks including labels:
  - **Movies** [46] is a complex network containing 41,412 entities with 20 labels (e.g., films, actors, directors) and 134,938 directed edges from 34 social relations (e.g., 'lived-with' and 'married-to').

[2] scikit-learn package LogisticRegression's LibLinear solver is used.

Table 1: Score functions of our competitors. σ here denotes the softmax function.

Model	Score function	Ranking order
DeepWalk	$\boldsymbol{x}_i^\top \boldsymbol{x}_j$	descending
LINE	$\boldsymbol{x}_i^\top \boldsymbol{x}_j$	descending
SDNE	$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$	ascending
TransE	$\|\boldsymbol{x}_i + \boldsymbol{r}_k - \boldsymbol{x}_j\|_2$	ascending
ProjE	$\sigma(\boldsymbol{x}_i^\top \tanh(\boldsymbol{u} \odot \boldsymbol{x}_j + \boldsymbol{v} \odot \boldsymbol{r}_k + \boldsymbol{b}) + c)$	descending

Table 2: Link Prediction results. Models are evaluated by Hit Rate at top 10 (HR@10) and Mean Reciprocal Rank (MRR).

Dataset	PPI		WN18		FB15k	
	HR@10	MRR	HR@10	MRR	HR@10	MRR
DeepWalk	0.0131	0.0187	0.4165	0.1503	0.0369	0.0187
LINE	0.1726	0.0819	0.0926	0.0347	0.0757	0.0345
SDNE	0.1049	0.0571	0.1951	0.0621	0.0720	0.0291
TransE	0.2101	0.0934	0.9215	0.3678	0.6089	0.3449
ProjE-Q5	0.3905	0.2345	0.8557	0.5901	0.3076	0.1636
MARINE-Q5	**0.8602***	**0.5671***	**0.9253***	**0.6211***	**0.7202***	**0.4861***
ProjE-ori	0.7441	0.5339	0.9389	0.7664	0.6957	0.5126
MARINE-Q100	0.8973	0.6665	0.9402	0.7034	0.7424	0.4044

* outperforms the second-best model at 0.01 level paired t-test.

Table 3: Multi-label classification results with respect to Micro F1-score and Macro F1-score. The last row shows the case MARINE considering LINE's embeddings as attributes (see Section 4.4).

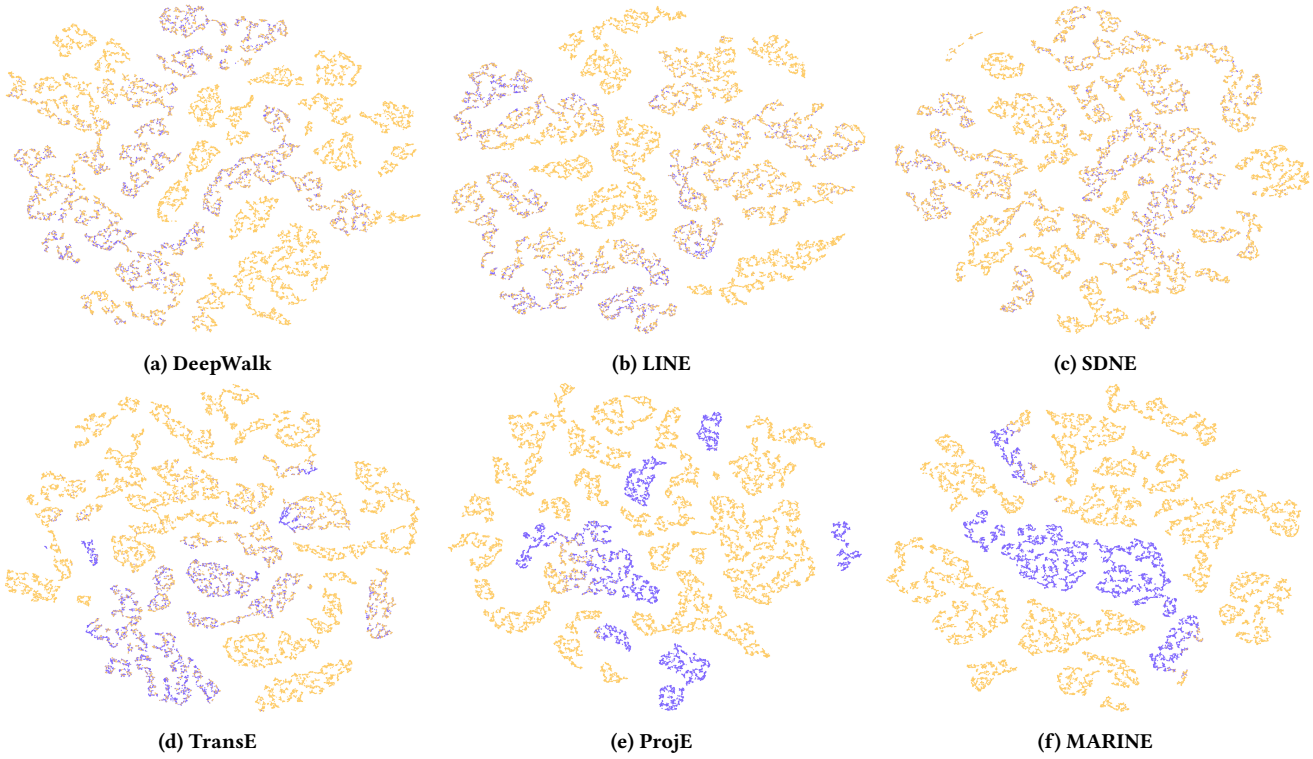
Dataset	PPI		Wikipedia		Movies	
	Micro	Macro	Micro	Macro	Micro	Macro
DeepWalk	0.0839	0.0631	0.3485	0.0595	0.0260	0.0156
LINE	0.0527	0.0332	0.3794	0.0694	0.6958	0.1136
SDNE	0.0021	0.0014	0.2811	0.0215	0.4871	0.0635
TransE	0.0355	0.0209	0.3312	0.0401	0.8932	0.3479
ProjE-Q5	0.0855	0.0683	0.3142	0.0678	0.9315	0.7181
MARINE-Q5	**0.0954***	**0.0723***	**0.4033***	**0.1044***	**0.9511***	**0.7315***
ProjE-ori	0.1083	0.0849	0.4134	0.1338	0.9049	0.6243
MARINE-Q100	0.1064	0.0829	0.4117	0.1219	0.9587	0.7925
Ensemble(MARINE, LINE)	0.1085	0.0822	0.4132	0.1137	0.9502	0.6617

* outperforms the second-best model at 0.01 level paired t-test.

The results are listed in Table 3. Again, given micro F1-score and macro F1-score, it is observed that MARINE significantly outperforms the others. In particular for the movie datasets that contain multiple relations, even MARINE-Q5 can outperform ProjE-ori significantly. For homogeneous networks PPI and Wikipedia, we can see that the most contribution to MARINE performance lies in the proximity preservation part, which consists with the perspective of previous homogeneous network embedding papers. By contrast, in the multi-relational network Movies the relation preservation part devotes more to MARINE in classification problems. We notice that using the original negative sampling strategy, ProjE can slightly outperform our MARINE. However we emphasize that ProjE samples $O(|\mathcal{V}|^2)$ negative node pairs to achieve high performance, which is not scalable at all. Under the same condition of sampling only 5 negative edges per positive edge example, MARINE shall significantly perform better than ProjE.

4.4 Exploiting Node Attributes

In this section, we discuss whether considering the node attributes allows our model to further boost the performance. As the datasets we used previously do not contain attributes, here we exploit the HepTh co-author citation network consisting of 5,501 author and 17,286 paper entities to predict the future citation counts using the network embeddings. The attributes are the ages of papers in HepTh. Following the same setup as [43], we train the embeddings on citations before 1999 and predict the number of citations after 2000. Random Forest Regressor (number of estimators = 32) is used



(a) DeepWalk (b) LINE (c) SDNE

(d) TransE (e) ProjE (f) MARINE

Figure 4: Visualization of HepTh. The orange spots indicate papers, while the purple spots indicate authors.

to train on embeddings for prediction. Five-fold cross-validation experiment is conducted; MAE (Mean Absolute Error) is chosen as the evaluation metric. The baseline usage of attributes relies simply on concatenating the embeddings with the attributes for training. We can see from Table 4 that not all models can benefit from additional attributes, while our model yields significantly better results than simply concatenating attributes with embeddings.

We also want to evaluate the performance of embedding ensemble as described in Section 3.7.3. We try to combine our model with LINE by treating its embedding results as our features, since LINE performs well among classic baseline embedding models. The last row in Table 3 shows that based on the ensemble of MARINE and LINE, we can gain improvement on two of the three datasets. The only exception is for the Movies dataset, probably due to the large performance gap between these two models.

Table 4: MAE evaluation of predicting citation counts. The comparison of the usage of node attributes in each embedding model.

usage	w/o attributes	concatenate
DeepWalk	6.6568	6.1724
LINE	6.8721	6.8994
SDNE	6.0891	6.1112
TransE	6.2739	6.3335
ProjE-Q5	6.5164	6.4319
ProjE-ori	5.9996	5.8402
MARINE-Q5	5.9525	5.8201
MARINE utilizes attributes as Θ		5.7042

4.5 Visualization

Here we want to show how the embedding results are visualized in the low-dimensional space. We apply visualization tool t-SNE [21] on the embeddings learned from, for the ease of presentation, HepTh dataset. All the models learn the embeddings without attributes. As Figure 4 suggests, the models that consider multi-relational networks are able to better distinguishing between different types of nodes. Both MARINE and ProjE can separate different types of nodes, and MARINE further groups the same type of nodes (purple spots) together. It shows that MARINE considers the relations of edges while preserving proximity simultaneously.

5 CONCLUSIONS

In this paper, we consider the two main branches of network embedding models: for homogeneous networks and for multi-relational networks, or knowledge graphs. Our conclusion is that although the existing multi-relational network embedding methods can be adopted to homogeneous networks, the performance can further be boosted while considering the relation-based proximity, as proposed in this paper. Our experimental results suggest that by considering graph proximity, relations, and attributes altogether, it is possible to yield a more robust network embedding model suitable for different types of networks. Finally, comparing with the state-of-the-art

solution ProjE that requires significant amount of negative links sampled, our model enjoys linear time and space complexity during computation.

ACKNOWLEDGMENTS

This work is supported by Ministry of Science and Technology (MOST) of Taiwan under grants 108-2636-E-006-002, 107-2218-E-006-040, 107-2221-E-001-009-MY3, and 106-3114-E-002-008, by Air Force Office of Scientific Research and Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-17-1-4038, and also by Academia Sinica under grant AS-TP-107-M05.

REFERENCES

[1] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. [n. d.]. Distributed Large-scale Natural Graph Factorization *(WWW '13)*.
[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. [n. d.]. Freebase: a collaboratively created graph database for structuring human knowledge *(SIGMOD '08)*.
[3] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A Semantic Matching Energy Function for Learning with Multi-relational Data. *Machine Learning*.
[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. [n. d.]. Translating embeddings for modeling multi-relational data *(NIPS '13)*.
[5] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. [n. d.]. Learning Structured Embeddings of Knowledge Bases *(AAAI'11)*.
[6] Bobby-Joe Breitkreutz, Chris Stark, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, Michael Livstone, Rose Oughtred, Daniel H. Lackner, JÃijrg BÃđhler, Valerie Wood, Kara Dolinski, and Mike Tyers. 2008. The BioGRID Interaction Database: 2008 update. *Nucleic Acids Research* (2008), D637–D640.
[7] Shaosheng Cao, Wei Lu, and Qiongkai Xu. [n. d.]. Deep Neural Networks for Learning Graph Representations *(AAAI'16)*.
[8] Shaosheng Cao, Wei Lu, and Qiongkai Xu. [n. d.]. GraRep: Learning Graph Representations with Global Structural Information *(CIKM '15)*.
[9] Miao Fan, Qiang Zhou, Emily Chang, and Thomas Fang Zheng. [n. d.]. Transition-based Knowledge Graph Embedding with Relational Mapping Properties *(PACLIC '14)*.
[10] Aditya Grover and Jure Leskovec. [n. d.]. node2vec: Scalable feature learning for networks *(KDD '16)*.
[11] Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. [n. d.]. Learning to Represent Knowledge Graphs with Gaussian Embedding *(CIKM '15)*.
[12] Xiao Huang, Jundong Li, and Xia Hu. [n. d.]. Label Informed Attributed Network Embedding *(WSDM '17)*.
[13] Rodolphe Jenatton, Nicolas Le Roux, Antoine Bordes, and Guillaume Obozinski. [n. d.]. A Latent Factor Model for Highly Multi-relational Data *(NIPS '12)*.
[14] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. [n. d.]. Knowledge Graph Embedding via Dynamic Mapping Matrix. *(ACL '15)*.
[15] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. [n. d.]. Knowledge Graph Completion with Adaptive Sparse Transfer Matrix *(AAAI'16)*.
[16] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv* (2014).
[17] Thomas N. Kipf and Max Welling. [n. d.]. Semi-Supervised Classification with Graph Convolutional Networks *(ICLR '17)*.
[18] Denis Krompaβ, Stephan Baier, and Volker Tresp. [n. d.]. Type-Constrained Representation Learning in Knowledge Graphs *(ISWC'15)*.
[19] Yankai Lin, Zhiyuan Liu, and Maosong Sun. [n. d.]. Knowledge Representation Learning with Entities, Attributes and Relations *(IJCAI'16)*.
[20] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. [n. d.]. Learning Entity and Relation Embeddings for Knowledge Graph Completion *(AAAI'15)*.
[21] Laurens van der Maaten and Geoffrey Hinton. [n. d.]. Visualizing data using t-SNE *(JMLR '08)*.
[22] Matt Mahoney. 2011. Large Text Compression Benchmark. http://www.mattmahoney.net/dc/textdata
[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. [n. d.]. Distributed Representations of Words and Phrases and their Compositionality *(NIPS '13)*.
[24] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* (1995).
[25] Dat Quoc Nguyen. 2017. An overview of embedding models of entities and relationships for knowledge base completion. *ArXiv* (2017).
[26] Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. [n. d.]. STransE: a novel embedding model of entities and relationships in knowledge bases *(NAACL '16)*.
[27] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. [n. d.]. Holographic Embeddings of Knowledge Graphs *(AAAI'16)*.
[28] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. [n. d.]. A Three-way Model for Collective Learning on Multi-relational Data *(ICML'11)*.
[29] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. [n. d.]. Asymmetric Transitivity Preserving Graph Embedding *(KDD '16)*.
[30] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. [n. d.]. Tri-party Deep Network Representation *(IJCAI'16)*.
[31] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. [n. d.]. Deepwalk: Online learning of social representations *(KDD '14)*.
[32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. [n. d.]. BPR: Bayesian personalized ranking from implicit feedback *(UAI '09)*.
[33] Leonardo F.R. Ribeiro, Pedro H.P. Saverese, and Daniel R. Figueiredo. [n. d.]. Struc2Vec: Learning Node Representations from Structural Identity *(KDD '17)*.
[34] Baoxu Shi and Tim Weninger. [n. d.]. ProjE: Embedding Projection for Knowledge Graph Completion *(AAAI '17)*.
[35] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. [n. d.]. Reasoning With Neural Tensor Networks for Knowledge Base Completion.
[36] Han Hee Song, Tae Won Cho, Vacha Dave, Yin Zhang, and Lili Qiu. [n. d.]. Scalable Proximity Estimation and Link Prediction in Online Social Networks *(IMC '09)*.
[37] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. [n. d.]. Line: Large-scale information network embedding *(WWW '15)*.
[38] Lei Tang and Huan Liu. [n. d.]. Relational Learning via Latent Social Dimensions *(KDD '09)*.
[39] Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Falk Brauer. [n. d.]. Random Semantic Tensor Ensemble for Scalable Knowledge Graph Link Prediction *(WSDM '17)*.
[40] Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, and Maosong Sun. [n. d.]. Max-margin Deepwalk: Discriminative Learning of Network Representation *(IJCAI'16)*.
[41] Daixin Wang, Peng Cui, and Wenwu Zhu. [n. d.]. Structural deep network embedding *(KDD '16)*.
[42] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. [n. d.]. Community Preserving Network Embedding *(AAAI'17)*.
[43] Yujing Wang, Yunhai Tong, and Ming Zeng. 2013. Ranking Scientific Articles by Exploiting Citations, Authors, Journals, and Time Information.
[44] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. [n. d.]. Knowledge Graph Embedding by Translating on Hyperplanes *(AAAI '14)*.
[45] Xiaokai Wei, Linchuan Xu, Bokai Cao, and Philip S. Yu. [n. d.]. Cross View Link Prediction by Learning Noise-resilient Representation Consensus *(WWW '17)*.
[46] Gio Wiederhold. 1999. Movies. https://kdd.ics.uci.edu/databases/movies/movies.html
[47] Han Xiao, Minlie Huang, and Xiaoyan Zhu. [n. d.]. TransG : A Generative Model for Knowledge Graph Embedding *(ACL '16)*.
[48] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. [n. d.]. Representation Learning of Knowledge Graphs with Hierarchical Types *(IJCAI'16)*.
[49] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. [n. d.]. Embedding Entities and Relations for Learning and Inference in Knowledge Bases *(ICLR '15)*.
[50] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. [n. d.]. Network Representation Learning with Rich Text Information *(IJCAI'15)*.
[51] D. Zhang, J. Yin, X. Zhu, and C. Zhang. [n. d.]. Homophily, Structure, and Content Augmented Network Representation Learning *(ICDM'16)*.
[52] Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. [n. d.]. Scalable Graph Embedding for Asymmetric Proximity *(AAAI'17)*.