RASE: Relationship Aware Social Embedding

Aravind Sankar*, Adit Krishnan*, Zongjian He, Carl Yang
Department of Computer Science,University of Illinois, Urbana-Champaign
{asankar3, aditk2, zh13, jiyang3}@illinios.edu

***Abstract*—This paper studies the problem of learning latent representations or *embeddings* for users in social networks, by leveraging relationship semantics associated with each link. User embeddings are low-dimensional vector-space representations designed to preserve structural proximity indicated by the pairwise relationships. In social networks, the closeness (or proximity) between pairs of users is very different *w.r.t.* multiple social relationships and thus cannot be represented accurately using a single embedding space. Furthermore, social networks pose a unique challenge of relationship label sparsity that precludes the application of knowledge-graph embedding techniques.**

In this paper, we associate each observed link with multiple relationship types through relationship weights and learn projection matrices for each relationship type to model the social distance (or proximity) between users specific to each relationship. We propose a novel two-step mutual enhancement framework to iteratively (a) learn user embeddings preserving relationship-specific proximity, and (b) link-relationship weights capturing the role of each link in multiple relationship types. The first step learns user embeddings optimizing relationship-specific proximity, while fixing the relationship weights (or roles) for each link. In the second step, the user embeddings and corresponding projection matrices are assumed to be fixed, while the link-relationship weights are learned. We demonstrate that the relationship-aware user embeddings learned through this mutual enhancement framework, are more effective in representing the users and outperform representative baseline techniques in multi-label classification and relationship prediction tasks.

***Index Terms*—Network modeling, Social embedding, Classification and Clustering**

I. Introduction

In recent years, social networks have become increasingly popular with an exponential growth in the number of interacting users. This has led to an imminent need for mining information from social networks, to support various social media applications such as online advertising, marketing and recommendation. Thus, it is essential to accurately learn useful features or representations for users in a social network, that capture their attributes and structural relationships.

Learning latent representations for nodes in a large graph (or network), has recently gained prominence as a fundamental research problem. Each node in the network is represented as a low-dimensional vector that is designed to capture various kinds of semantic, relational and structural information conveyed by the network, *i.e.*, the goal is to reconstruct and predict the properties of the network in the learned embedding space. Such low-dimensional node embeddings have widespread applications in various machine learning tasks such as graph visualization [1], node classification [2], clustering [3], link prediction [4] and recommendation [5, 6].

The key challenges faced in learning effective latent representations for users, are non-linearity and sparsity. Existing methods employ a wide range of techniques to capture network structural properties in varying extents, which determines their effectiveness in different applications. Recent techniques have been inspired by the development of neural language models for learning latent representations of words in a text corpus. For example, DeepWalk [7] transforms a social network into a collection of paths (sequence of users) using random walks and uses the Skip-gram model [8] (originally designed for words) to learn representation of users capturing co-occurrences in the sampled random walks. Another technique that follows a principled approach to learning network representations is LINE [1], which defines a loss function to capture both 1-step and 2-step local relational information or proximity. Several recent methods have examined deep neural networks to model high-order measures of network proximity [9, 10].

In this work, we focus on social embedding, *i.e.*, learning latent representations for users in social networks, where the relationships of the links exhibit different semantics. Most existing network embedding techniques such as Deepwalk, LINE and others, are agnostic to how two users are connected in the network. As a consequence, the latent user representations learned by these techniques only provide a generic measure of user proximity in the social network. However, social networks are typically associated with rich semantics where users possess various attributes such as gender and hobbies, and participate in different relationships such as colleagues, classmates, and neighbors. The social distance between a pair of users varies significantly based on the type of relationship, *e.g.*, two users may be very close to each other in terms of classmates, while being far away from each other in the relationship of geographical neighbors.

Our objective is to *learn embeddings that capture relationship-specific proximity* between users in social networks. In order to leverage users' multiple relationships, we cannot directly employ existing network embeddings techniques because real-world relation-rich social networks pose several unique challenges. First, real-world social networks are faced with severe sparsity on the relationship labels for each link, *e.g.*, less than 50% of the links have associated relationship labels in a publicly available Linkedin social network [11]. Second, each link is typically associated with multiple relationship labels which are possibly incomplete, thus rendering the problem to be even more challenging.

*Equal contribution

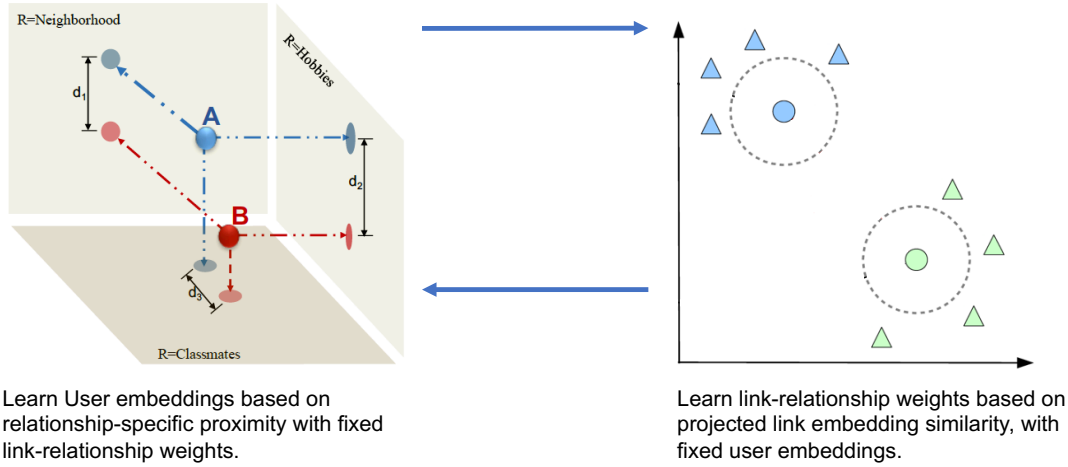


Fig. 1. Illustration of two-step learning of RASE

Finally, different relationships in social networks are not independent and may exhibit different degrees of correlation, *e.g.*, two users who are classmates are more likely to live in the same location than users who are professional acquaintances.

To address the above challenges, we additionally learn associations or roles for each link in multiple relationship types, via *link-relationship* weights. The link-relationship weights are not normalized, hence allowing independent degrees of freedom for each link to carry multiple relationships. These link-relationship weights are then utilized to learn relationship-aware user embeddings. To learn both user embeddings and link-relationship weights (or associations) simultaneously, we propose a novel mutual enhancement framework *RASE* which proceeds iteratively in an EM-style step-wise manner.

In the first step, the link-relationship weights are assumed to be known and fixed, and the user embeddings are trained by optimizing the social distance (or proximity) between users specific to each relationship type. We project the original embedding space through relationship projection matrices, which which enables us to focus on measuring distances specific to each relationship. Similar concepts have been applied in the context of knowledge graph analysis [12, 13, 14] where the objective is to learn latent representations for entities and relations given a knowledge base such as Wordnet or Freebase. The general principle used here is that two entities connected by a specific relation are close when projected onto the relation space. In our scenario, we follow a similar strategy to learn user embeddings and relationship-specific projection matrices, given fixed link-relationship weights.

In the second step, the user embeddings and corresponding projection matrices are assumed to be fixed, while the link-relationship weights are learned. Here, the key hypothesis is that two links have similar associations to a given relationship r (*i.e.*, possess similar link-relationship weights), if their corresponding projections onto the relationship space of r, are similar. We define an objective function to learn the relationship weights for each link by leveraging relationship-specific projections of user embeddings. To further reduce the training complexity, we introduce a clustering algorithm to cluster links based on their relationship-specific representations, which leads to a reduction in complexity from quadratic to linear in the number of links.

We evaluate the learned embedding representations on two real-world social network datasets, *i.e.*, Facebook [15] and Linkedin Ego-networks [11] on two different tasks, *i.e.*, Multi-label classification and Relationship prediction. We find that our algorithm *RASE* outperforms baselines techniques by significant margins on these datasets. The main contributions of this work are summarized as follows:

1) In this work, we draw attention to the problem of relationship aware social network embedding where we propose to capture users' relationship specific distances in the corresponding projected subspaces of pairwise user relationships on social networks.
2) We develop a simple and elegant two-step iterative embedding learning framework (*i.e.*, *RASE*) for relationship aware network embedding through considering both first- and second-order network proximity to learn relationship specific link weights and latent user representations.
3) Through extensive experiments on two real-world social network datasets, we demonstrate the effectiveness of *RASE* for the important yet challenging tasks, multi-label user classification and link-level relationship prediction.

II. Related Work

In this work, we address the problem of learning latent representations for users in a social network scenario, by leveraging the semantics of different relationships between users. First, we discuss existing literature on learning network embeddings. The earliest techniques such as Principal component analysis [16] and Laplacian Eigenmaps [17] were based on dimensionality reduction. They optimize an objective function that works on an affinity matrix of the network such that it maximizes the variance of the latent representation.

Such techniques typically encounter the huge cost of eigen-decomposition of the affinity matrix which is computationally very expensive and may not even be possible for massive social networks. Probabilistic graphical models have also been introduced to extract multi-faceted representations of social graphs, and more recently for social behavior analysis [18, 19].

Recent methods for learning network embeddings are based on stochastic methods or optimization of simpler well-defined objective functions. Deepwalk [7], one of the first techniques in this line of work was inspired by neural language models and uses random walks of nodes to sample paths from the network. The sampled paths are used to model proximity between nodes in the network using the SkipGram formulation introduced in word2vec [8]. LINE [1] defines two objective functions, modeling first order proximity between neighboring nodes and second order proximity which models the presence of common neighbors for a pair of nodes. node2vec [20] is an extension of Deepwalk based on the design of a biased random walk controlled by two parameters that allows exploration of both neighborhoring nodes and node sequences farther from the root node. Recently, deep learning techniques have been explored by SDNE [9] which introduces a semi-supervised deep model using an auto-encoder to preserve both local and global measures of network structural proximity. Continuing on this line of work, many specifically designed embedding algorithms have been developed to capture more complex node attributes and link structures [10, 21, 22, 23, 24, 25, 26, 27]. Further, contemporary techniques have examined embeddings methods for dynamic graphs [28]. However, none of them consider the embedding of different social relationships carried by the seemingly uniform network links.

Next, we discuss knowledge graph embeddings that learn representations for entities and relations. Amongst these methods TransE [29] learns vector embeddings in a continuous space for both entities and relationships in a knowledge graph. It models the relationship between two entities by a translation between the emebeddings of the entities. Later on, TransH [30] is proposed to address the limitations of TransE in 1-to-N, N-to-1 and N-to-N relations, by employing relation specific hyperplanes. Adversarial learning has also been explored in the context of neighbor-homophily models [5, 31]. The most related work to ours is TransR [32], which separates entity space and relation spaces, and constrains entity distances in the projected relation spaces. In this way, it is able to leverage multiple relationships in different semantic spaces among entities. However, all of these methods assume that each link in the network is labeled with a relation, and limit each link to contain a single relationship label, which is not the case in our problem setting.

III. Problem Definition

We assume the input as a social network which is an undirected graph with partially labeled relationships on each link of the network. The objective is to learn a latent representation of the social network where each user is represented as a low-dimensional vector. We present the formal definitions below:

TABLE I
NOTATIONS.

Symbol	Dimensions	Description
$\mathbb{L}$	L (set)	Set of relationship types
$\mathbf{U}$	$\lvert\mathcal{V}\rvert \times D$	User embedding matrix
$\mathbf{M}_r$	$D \times D$	Projection matrix for relationship $r \in \mathbb{L}$
$\mathbf{W}$	$\lvert\mathcal{E}\rvert \times L$	Link-relationship weight matrix

Definition 1. ***Social Network:*** *A social network is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, R)$, where $\mathcal{V}$ is the set of users, each representing a data object, $\mathcal{E}$ is the set of links between the users, each representing a relationship between two data objects and $\mathcal{R}$ is the set of given relationship labels for each link. Each edge $e \in \mathcal{E}$ is an ordered pair $e = (u, v)$ and has an associated entry $l_e \in \mathcal{R}$ which is the set of relationship labels for the edge e. The set of all possible relationship labels for any edge is denoted by the set $\mathbb{L}$ with cardinality $L = |\mathbb{L}|$, and $\forall e \in \mathcal{E}$, $l_e \in 2^{\mathbb{L}}$. We assume that G is undirected and not all edges have associated relationship labels.*

Definition 2. ***Social Network Embedding:*** *Given a large network $G = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, the problem of Social Network Embedding aims to represent each user $v \in \mathcal{V}$ into a low-dimensional space $\mathbb{R}^d$ where $d << |\mathcal{V}|$, i.e., learn a function $f_G : \mathcal{V} \rightarrow \mathbb{R}^d$ which preserves relationship-specific proximity between the users in each relationship type in $\mathbb{L}$.*

IV. Relationship Aware Social Embedding

In this section, we present our RASE framework for learning user embeddings in a social network $\mathcal{G}$. RASE is a joint iterative learning framework comprising two major components: (a) Learning user representations preserving relationship-specific proximity, and (b) Learning relationship specific weights for each link. First, we formally define proximity between users using first-order and second-order proximities specific to each relationship in Section IV-A. Then, we will discuss how weights can be learnt for associating each link to different relationship using the embeddings in Section IV-B. Finally, we present our joint learning framework that learns embeddings of user and associations for each link to multiple relationships iteratively in Section IV-C.

A. *Learning User Representations*

In order to learn latent user representations, we are interested in modeling proximity of a pair of users specific to a relationship. To achieve this goal, we extend the notions of first and second order proximity defined by LINE [1] specific to each relationship. Traditionally, the first-order proximity in a network is the local pairwise proximity between two linked users which is defined as being proportional to the weight of the link connecting two users. Since we have partial relationship information on the links of the network, we allow each link to have weights specific to each relationship (that would be later learnt by our model).

Specifically, we assume that each link $e \in \mathcal{E}$ has a *link-relationship* weight vector $\mathbf{w_e}$ of dimensionality L, represented by matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{E}| \times L}$. Here, each entry w_e^r denotes the weight of the link e for the relationship $r \in \mathbb{L}$. The weight vectors $\mathbf{w_e}$ would be learnt iteratively by our model. The link-relationship weights for link e are initialized based on the given input labels for each edge $l_e \in \mathcal{R}$, *i.e.*, $\mathbf{w_e}$ would be initialized as binary vector to start. Assuming we have the weight vectors for each link, we model the proximity specific to each relationship using the projections of the embedding vectors of the nodes onto the corresponding relationship space. For this purpose, we define projection matrices $\mathbf{M}_r \in \mathbb{R}^{D \times D}$ for each relationship $r \in L$. The set of all user representations are denoted by embedding matrix $\mathbf{U} \in \mathbb{R}^{|\mathcal{V}| \times D}$. Table I depicts the set of all notations used in RASE. Now, we define first and second order proximity specific to each relationship.

1) RASE - first-order proximity: We capture the proximity of two users specific to each relationship by defining first order proximity on the relation space obtained by projecting the embedding vectors of the two users in a link. For each link (v_i, v_j), we define the joint probability between users v_i and v_j specific to relation $r \in \mathbb{L}$ as follows:

$$p_1{}^r(v_i, v_j) = \frac{1}{1 + \exp(-(\mathbf{M}_r \mathbf{u}_i)^T (\mathbf{M}_r \mathbf{u}_j))} \tag{1}$$

where $\mathbf{u}_i \in \mathbb{R}^d$ is the embedding representation of user v_i and $\mathbf{M}_r$ is the projection matrix for relationship r. Corresponding to this probability, we can define the empirical probability distribution by:

$$\hat{p_1}{}^r(e = (v_i, v_j)) = \frac{W_e^r}{W^r} \qquad W^r = \sum_{e=(v_i,v_j) \in \mathcal{E}} W_e^r \tag{2}$$

where W_e^r is the link-relationship weight for link e and relationship r. To preserve first-order we minimize the KL-divergence between these two probability distributions and sum over all possible relationship types which gives us:

$$O_1 = -\sum_{r \in L} \sum_{(v_i,v_j) \in \mathcal{E}} \hat{p_1}{}^r(v_i, v_j) \log(p_1^r(v_i, v_j)) \tag{3}$$

Assuming we know the weight vectors for each link, we can optimize this objective function using gradient descent to obtain the embedding vector $\mathbf{u}_i$ for each user v_i. Similar to LINE [1], we also consider second order proximity specific to each relationship, as defined below.

2) RASE - second-order proximity: In general, second-order proximity assumes that a pair of users that share many connections to other users are similar to each other. We extend this definition to relationships by specifying that a pair of users that are related to many other users in a similar way are similar to each other specific to that relationship. To model this, we define context and target vectors for each user to model the different roles played by each user. Specifically, we assume each user v_i has two vector representations $\mathbf{u_i}$ and $\mathbf{u}_i{}'$ where $\mathbf{u}_i$ and $\mathbf{u}_i'$ are the representations of v_i when it is treated as a target and context respectively. For each link (v_i, v_j), we first define the probability of context v_j generated by vertex v_i through relationship r as:

$$p_2^r(v_j \mid v_i) = \frac{exp((\mathbf{M}_r \mathbf{u}_j')^T (\mathbf{M}_r \mathbf{u}_i))}{\sum_{v_k \in \mathcal{V}} exp((\mathbf{M}_r \mathbf{u}_k')^T (\mathbf{M}_r \mathbf{u_i}))} \tag{4}$$

To preserve the second-order proximity, we make the conditional distribution of the contexts $p_2^r(\cdot \mid v_i)$ specified by the low-dimensional representations to be close to the empirical distribution specified by:

$$\hat{p_2}{}^r(v_j \mid v_i) = \frac{W_{(v_i,v_j)}^r}{d_i^r} \qquad d_i^r = \sum_{v_k \in N(v_i)} W_{(v_i,v_k)}^r$$

Minimizing the KL-divergence between these two probability distributions leads to the objective function:

$$O_2 = -\sum_{r \in \mathbb{L}} \sum_{(v_i,v_j) \in \mathcal{E}} \hat{p_2}{}^r(v_j \mid v_i) \log(p_2{}^r(v_j \mid v_i)) \tag{5}$$

We optimize this by negative sampling since direct optimization of this function is very expensive [8]. For each link, we sample K negative samples specific to each relationship. For link (v_i, v_j) and relationship r, we get the following objective function :

$$\begin{aligned} O_2'(v_i, v_j, r) = &\log \sigma((\mathbf{M}_r \mathbf{u}_j')^T (\mathbf{M}_r \mathbf{u}_i)) + \\ &\sum_{i=1}^{K} E_{v_n \sim P_n(v)} \log \sigma(-(\mathbf{M}_r \mathbf{u}_n')^T (\mathbf{M}_r \mathbf{u}_i))) \end{aligned} \tag{6}$$

This is optimized using stochastic gradient descent to learn the embedding representations of the nodes. Next, we describe how to learn associations for each link to multiple relationships using the learned embeddings.

B. Learning Relationship-specific Weights

In the previous section, we have assumed that the link-relationship weights $\mathbf{W}$ which indicate the role of each link in different relationships, to be fixed. Here, we describe how to learn these link-relationship weights by assuming that the user embeddings and projection matrices obtained from the first step, are fixed. Our key hypothesis is that the role of two links in a relationship are similar if the two interacting users are embedded close in the corresponding relationship space. The role of a link in a relationship can be used to learn the association or weights for each edge for multiple relationships. To achieve this goal, we can define a representation for a link e specific to relationship r by applying a translation operation on the relationship-projected user representations given by:

$$\mathbf{v}_e^r = \mathbf{M}_r \mathbf{u}_i - \mathbf{M}_r \mathbf{u}_j$$

This enables us to define a representation for a link specific to each relationship. Now, two links which have similar

Algorithm 1 RASE

1: Initialize W_e based on l_e $\forall e \in \mathcal{E}$.
2: **while** user embeddings $\mathbf{u}_i$'s have not converged **do**
▷ *Learn user embeddings* $\mathbf{U}$ *and projection matrices* $\{M_r \forall r \in L\}$ *by fixing link-relationship weights* $\mathbf{W}$
3: Update $\{\mathbf{u}_i \ \forall \ v_i \in \mathcal{V}\}$ and $\{\mathbf{M}_r \ \forall r \in \mathbb{L}\}$ by optimizing $O_1 + O_2$ simultaneously (Eq. 3 and Eq. 6).
▷ *Learn link-relationship weights* $\mathbf{W}$ *by fixing user embeddings* $\mathbf{U}$ *and projection matrices* $\{\mathbf{M}_r \ \forall r \in \mathbb{L}\}$
4: Update $\{W_e^r \ \forall e \in \mathcal{E}$ and $r \in \mathbb{L}\}$ by optimizing J_2 (Eq. 8).
5: **end while**

representations in the relationship space of r are expected to have similar weights *w.r.t.* relationship r. We can capture this property by minimizing an objective function as shown below:

$$J_1 = \sum_{e_1 \in \mathcal{E}} \sum_{e_2 \in \mathcal{E}} \sum_{r \in \mathbb{L}} (W_{e_1}^r - W_{e_2}^r)(\mathbf{v}_{e_1}^r)^T(\mathbf{v}_{e_2}^r) \quad (7)$$

The above objective function ensures that two links with similar relationship-specific difference vectors have similar link-relationship weights. Note that the above objective function involves iterating through all pairs of links in the network for each relationship leading to a complexity $O(L|\mathcal{E}|^2)$ which is very expensive even if sampling is used. Furthermore, we may not need to compare every pair of links since we expect multiple links to exhibit similar behavior.

To reduce the computational complexity, we first cluster the links based on a combined representation across all the relations, *i.e.*, the embeddings specific to all the relationships are concatenated to obtain a combined representation for a link. Then, we use K-medoids to cluster the links into K clusters and identify the cluster center. We assume that the weight assigned to each cluster center (from the previous iteration) is non-zero for the different relations. (If not, we compute the next closest cluster center). Then, these cluster centers are used to approximate the objective function defined above by defining each link to be similar to its cluster center with a weight close to that of the cluster center. This leads to a simpler efficient objective function given by:

$$J_2 = \sum_{k=1}^{K} \sum_{e \in \mathcal{E}_k} \sum_{r \in \mathbb{L}} (W_e^r - W_k^r)(\mathbf{v}_e^r)^T(\mathbf{v}_k^r) + \lambda \parallel W \parallel^2 \quad (8)$$

where $\mathcal{E}_k$ is the set of all links assigned to cluster k, and $\mathbf{v}_k^r$ is the embedding of the corresponding cluster center. We can compute the gradient *w.r.t.* each weight W_e^r as:

$$\frac{\partial J_2}{\partial W_e^r} = (W_e^r - w_k^r)(\mathbf{v}_e^r)^T(\mathbf{v}_k^r) + 2\lambda W_e^r$$

C. Iterative Learning Framework

In the previous sections, we have described the separate steps of learning user embeddings and link-relationship weights. Here, we describe how to combine the two steps outlined earlier to develop a mutual enhancement framework to learn both user embeddings as well as link-relationship weights. The overall objective function is given by:

$$J = O_1 + O_2 + J_2$$

Alg. 1 depicts the pseudo code of the iterative learning framework. The algorithm proceeds in a block-coordinate descent manner in two alternating optimization steps. In the first step, the user embeddings and projection matrices are learnt by capturing relationship-specific proximity based on the weights of the links for each relationship, while keeping the link-relationship weights fixed. In the second step, the user embeddings and projection matrices are fixed while the relationship-specific weights for each link are learnt by capturing the role of each link in multiple relationships based on the translated projected embeddings of the corresponding users. The weight vectors of each link are initialized based on the available relationship labels in the dataset, *i.e.*, $\{l_e \neq \phi; \ \forall e \in \mathcal{E}\}$ by using a binary vector to initialize the weight vectors. As illustrated in Alg. 1, the two steps are repeated alternately until the learned user embeddings converge. In the next section, we describe our experiments and evaluation.

TABLE II
STATISTICS OF THE DATASETS.

Dataset	# nodes	# links	# rel labels
Facebook	4,039	88,234	8
Linkedin	29,040	118,302	3

V. EXPERIMENTS

In this section, we present our experiments on two real-world publicly available social network datasets. We first describe the datasets in Section. V-A, then introduce the baselines in Section. V-B and finally present our results on multi-label classification and relationship prediction in Section. V-C.

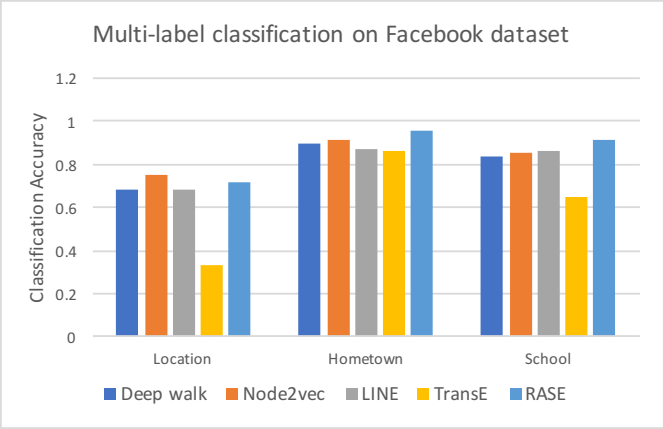
A. Datasets

We conduce experiments on two popular real world social networks from Facebook and LinkedIn. An overview of the dataset statistics is provided in Table II :

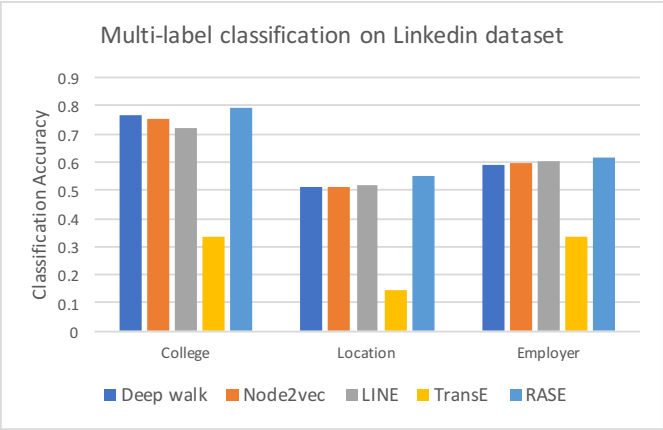
- Facebook [15] : This dataset has 10 ego-networks of friends with multiple attributes for each user, which were used to construct meaningful relationship labels. We constructed 8 relationship labels based on the attributes that were not sparse. These include major, college, school, hometown, location, employer, work-location and work-position. In this dataset, only 44,900 links had at least one relationship label out of 88,234 links, which corresponds to a fraction of about 50%.
- Linkedin [11] : This dataset contains a set of ego-networks on Linkedin with about 250 ego networks. It has three relationship labels between pairs of users which are employer, location and college. In this dataset, only

TABLE III
RELATIONSHIP PREDICTION RESULTS ON THE FACEBOOK DATASET.

Relationship type	Algorithm	Average	Hadamard	Weighted-L_1	Weighted-L_2
location	Deep walk	0.609	0.613	0.531	0.527
	Node2vec	0.649	0.658	0.557	0.568
	LINE	0.588	0.586	0.529	0.539
	TransE	0.636	0.594	0.564	0.565
	RASE	**0.657**	**0.667**	**0.582**	**0.581**
hometown	Deep walk	0.658	0.654	0.567	0.580
	Node2vec	0.681	0.687	0.590	0.594
	LINE	0.636	0.638	0.547	0.558
	TransE	0.691	0.631	0.593	0.588
	RASE	**0.697**	**0.721**	**0.624**	**0.627**
school	Deep walk	0.622	0.620	0.541	0.550
	Node2vec	0.654	0.654	0.561	0.558
	LINE	0.636	0.633	0.542	0.541
	TransE	0.624	0.621	0.574	0.576
	RASE	**0.657**	**0.662**	**0.578**	**0.577**



(a) Facebook



(b) LinkedIn

Fig. 2. **Classification results on Facebook and LinkedIn datasets**

58,276 links had at least one relationship label out of 118,302 links, which give a fraction of about 49%.

We use the attributes of the users in both datasets to evaluate the performance of the embeddings on classification.

B. *Baselines*

To validate the performance of our approach we compare it against a number of state-of-the-art baselines from both general network and knowledge graph embedding methods as listed below:

- Deepwalk [7] : This approach learns d-dimensional feature representations by learning user co-occurrences from uniformly sampled random walks.
- LINE [1] This approach learns d-dimensional feature representations in two separate phases. In the first phase, it learns $d/2$ dimensions using first-order proximity while in the second phase, it learns the next $d/2$ dimensions using second-order proximity.
- Node2vec [20] : This approach extends Deepwalk by using a biased random walk strategy. It simulates both BFS and DFS exploration of the network in order to perform path sampling.
- TransE [29]: This is a knowledge-graph embedding method designed to operate on entities and relations. To use this method, we provide a fully labeled social graph as input, using an extra relationship type indicating unknown/absent relationships.

C. *Experimental Results*

We evaluate the performance of different network embedding methods on two standard downstream prediction tasks.

TABLE IV
RELATIONSHIP PREDICTION RESULTS ON THE LINKEDIN DATASET.

Relationship type	Algorithm	Average	Hadamard	Weighted-$L1$	Weighted-L_2
college	Deep walk	0.744	0.680	0.544	0.546
	Node2vec	0.707	0.715	0.577	0.567
	LINE	0.664	0.648	0.518	0.505
	TransE	0.704	0.601	0.569	0.542
	RASE	**0.766**	**0.736**	**0.581**	**0.576**
location	Deep walk	0.614	0.610	0.521	0.518
	Node2vec	0.576	0.617	0.531	0.531
	LINE	0.571	0.564	0.508	0.512
	TransE	0.598	0.559	0.554	0.538
	RASE	**0.646**	**0.625**	**0.544**	**0.551**
employer	Deep walk	0.623	0.621	0.523	0.528
	Node2vec	0.602	**0.663**	0.535	0.545
	LINE	0.599	0.644	0.521	0.502
	TransE	0.619	0.588	0.556	0.531
	RASE	**0.665**	0.662	**0.564**	**0.547**

1) Multi-label classification: In the multi-label classification setting, every user is assigned one or more labels from a finite set $\mathcal{L}$ which is the set of user attributes in our case. During the training phase, a certain fraction of labels are observed and the task is to predict the labels for the remaining nodes. We split the input data into train and test with 80% to 20% ratio and evaluate the performance based on 5-fold cross-validation. We report the classification accuracy obtained using cross-validation. For all models we use a one-vs-rest logistic regression implemented by LibLinear [33] extended to return the most probable labels. We use the embedding dimension $d = 128$ for all methods and use parameters such as window size - 10, walk length - 20 and walks per vertex - 20 for both Deepwalk and node2vec. For LINE, we evaluate the model using both first-order and second-order proximity independently and report the best results. The classification results obtained on the two datasets are shown in Figures 2.

Our proposed algorithm *RASE* constantly outperforms the compared baselines by significant margins, indicating the sanity of considering relationship-aware embeddings and the effectiveness of our particular model design. We find that TransE performs significantly worse than general network embedding methods for = classification, which is primarily due to its inability to model second order proximity *w.r.t.* different relationships. On the other hand, during the experiments, we do observe slight overfitting of relationship projection matrices. Part of the reason for this could be due to the fact that the projection matrix has a dimensionality of $D \times D$ where D is the dimensionality of the embedding space, which leads to a large number of parameters learned. It may be prudent to place restrictions and narrow the domain of the learning algorithm to obtain better results, which we leave as a promising line of future work. A possible exploration could be to enforce sparsity norm constraints on the projection matrices.

TABLE V
BINARY OPERATORS USED FOR LEARNING LINK FEATURES IN RELATIONSHIP PREDICTION TASK.

Operator	Symbol	Definition
Average	$\boxplus$	$[f(u) \boxplus f(v)]_i = \frac{f_i(u)+f_i(v)}{2}$
Hadamard	$\boxdot$	$[f(u) \boxdot f(v)]_i = f_i(u) \times f_i(v)$
Weighted-L_1	$\|\cdot\|_1$	$\| f(u) \cdot f(v) \|_{1i} = \|f_i(u) - f_i(v)\|$
Weighted-L_2	$\|\cdot\|_2$	$\| f(u) \cdot f(v) \|_{1i} = \|f_i(u) - f_i(v)\|^2$

2) Relationship prediction: In the relationship prediction problem, we sample a fraction of all labeled links (subset S $\subset \mathcal{E}$) and learn embeddings on this reduced set of links. We perform relationship prediction on unlabeled links using the user embeddings learned. Given an unlabeled link (v_i, v_j) we predict the existence of each relationship label, $l \in \mathbb{L}$, the set of all relation labels in the network on the link. We use logistic regression implemented by LibLinear [33] to perform binary classification for every relationship label. Input features are chosen as Average, Hadamard distance, L_1 distance and L_2 distance, of the user embeddings corresponding to the end points of each link, similar to [20]. The relationship prediction results are tabulated in Tables 2 & 4.

In these experiments, RASE outperforms the baselines, although by smaller margins, specifically in the Hadamard, Weighted-L_1 and L_2 feature spaces. A probable reason could be the reorientation of the user embedding space to better align user embeddings to capture proximities upon projection. This could lead to the user translation vectors having more pronounced relationship specific dimensions which benefits this task. In this setting, we find TransE to be a very competitive baseline, however it still suffers from relationship sparsity unlike RASE which learns soft link-relationship weights.

VI. Conclusion

In this work, we focus learning relationship aware network embedding for social networks, where the semantics of each link and relationship specific proximities are preserved in the corresponding projected subspaces of latent relationships. A two-step inference algorithm considering both first- and second-order proximities to learn relationship specific link weights is designed and shown to be effective in multiple real-world social network datasets. Future work includes improvements on the effectiveness and efficiency of the proposed algorithm, interpretation and discrimination of the learned embedding regarding target relationships of the network, as well as its applications towards various downstream applications.

References

[1] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077. ACM, 2015.

[2] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011.

[3] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. In *NIPS*, volume 14, pages 849–856, 2001.

[4] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the Association for Information Science and Technology*, 58(7):1019–1031, 2007.

[5] Adit Krishnan, Ashish Sharma, Aravind Sankar, and Hari Sundaram. An adversarial approach to improve long-tail performance in neural collaborative filtering. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1491–1494. ACM, 2018.

[6] Iftikhar Ahamath Burhanuddin, Payal Bajaj, Sumit Shekhar, Dipayan Mukherjee, Ashish Raj, and Aravind Sankar. Similarity learning for product recommendation and scoring using multi-channel data. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1143–1152. IEEE, 2015.

[7] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710. ACM, 2014.

[8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[9] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *SIGKDD*, pages 1225–1234. ACM, 2016.

[10] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[11] Rui Li, Chi Wang, and Kevin Chen-Chuan Chang. User profiling in an ego network: Co-profiling attributes and relationships. In *WWW*, 2014.

[12] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Conference on artificial intelligence*, 2011.

[13] Adit Krishnan, P Deepak, Sayan Ranu, and Sameep Mehta. Leveraging semantic resources in diversified query expansion. *World Wide Web*, 21(4):1041–1067, 2018.

[14] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. In *Thirtieth Aaai conference on artificial intelligence*, 2016.

[15] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

[16] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *TPAMI*, 29(1), 2007.

[17] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

[18] Qiang Qu, Cen Chen, Christian S Jensen, and Anders Skovsgaard. Space-time aware behavioral topic modeling for microblog posts. *IEEE Data Eng. Bull.*, 38(2):58–67, 2015.

[19] Adit Krishnan, Ashish Sharma, and Hari Sundaram. Insights from the long-tail: Learning latent representations of online user behavior in the presence of skew and sparsity. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 297–306. ACM, 2018.

[20] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *SIGKDD*, pages 855–864. ACM, 2016.

[21] Tianshu Lyu, Yuan Zhang, and Yan Zhang. Enhancing the network embedding quality with structural similarity. In *CIKM*, pages 147–156. ACM, 2017.

[22] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *SIGKDD*, pages 385–394. ACM, 2017.

[23] Aravind Sankar, Xinyang Zhang, and Kevin Chen-Chuan Chang. Motif-based convolutional neural network on graphs. *arXiv preprint arXiv:1711.05697*, 2017.

[24] Carl Yang, Mengxiong Liu, Vincent W Zheng, and Jiawei Han. Node, motif and subgraph: leveraging network functional blocks through structural convolution. In *ASONAM*. IEEE/ACM, 2018.

[25] Carl Yang, Hanqing Lu, and Kevin Chang Chang. Cone: Community oriented network embedding. In *arXiv preprint arXiv:1709.01554*, 2017.

[26] Carl Yang, Yichen Feng, , Pan Li, Yu Shi, and Jiawei Han. Meta-graph based hin spectral embedding: Methods, analyses, and insights. In *ICDM*, 2018.

[27] Carl Yang, Mengxiong Liu, Frank He, Xikun Zhang, Jian Peng, and Jiawei Han. Similarity modeling on heterogeneous networks via automatic path discovery. In *ECML-PKDD*. Springer, 2018.

[28] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dynamic graph representation learning via self-attention networks. *arXiv preprint arXiv:1812.09430*, 2018.

[29] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

[30] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. Citeseer, 2014.

[31] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM, 2011.

[32] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187, 2015.

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