TransN: Heterogeneous Network Representation Learning by Translating Node Embeddings

Zijian Li #1, Wenhao Zheng *2, Xueling Lin #3, Ziyuan Zhao †4, Zhe Wang #5,
Yue Wang ‡6, Xun Jian #7, Lei Chen #8, Qiang Yan †9 and Tiezheng Mao †10

# *Computer Science and Engineering Department, HKUST, Hong Kong, China*
* *National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China*
† *Wechat Group, Tencent Corporation, Guangzhou, China*
‡ *Shenzhen Institute of Computing Sciences, Shenzhen University, Shenzhen, China*

{1 zlicb, 3 xlinai, 5 zwangec, 6 ywangby, 7 xjian, 8 leichen}@ust.hk
2 zhengwh@lamda.nju.edu.cn
{4 joshuazhao, 9 rolanyan, 10 woodtmao}@tencent.com

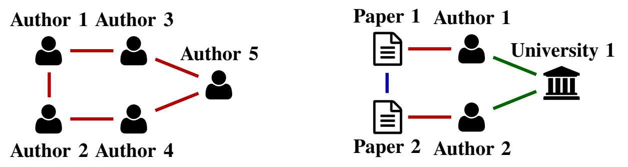
***Abstract*—Learning network embeddings has attracted growing attention in recent years. However, most of the existing methods focus on homogeneous networks, which cannot capture the important type information in heterogeneous networks. To address this problem, in this paper, we propose TransN, a novel multi-view network embedding framework for heterogeneous networks. Compared with the existing methods, TransN is an unsupervised framework which does not require node labels or user-specified meta-paths as inputs. In addition, TransN is capable of handling more general types of heterogeneous networks than the previous works. Specifically, in our framework TransN, we propose a novel algorithm to capture the proximity information inside each single view. Moreover, to transfer the learned information across views, we propose an algorithm to translate the node embeddings between different views based on the dual-learning mechanism, which can both capture the complex relations between node embeddings in different views, and preserve the proximity information inside each view during the translation. We conduct extensive experiments on real-world heterogeneous networks, whose results demonstrate that the node embeddings generated by TransN outperform those of competitors in various network mining tasks.**

***Index Terms*—Heterogeneous Network Embedding, Representation Learning, Multi-View Network Embedding, Dual Learning**

I. Introduction

Recently, there has been growing interest in learning network embeddings, i.e., low-dimensional vectorized representations of networks [4], [12]. Specifically, most of the existing network embedding methods [13], [19], [33], [41], [45] are designed for homogeneous networks, i.e., networks with only a single type of nodes and edges. Such methods focus on preserving the structure information and nodes proximity in their generated embeddings, which are demonstrated to be useful features in downstream network mining tasks such as node classification [41] and link prediction [45].

However, such methods on homogeneous networks are not suitable for learning embeddings of heterogeneous networks, i.e., networks with multiple types of nodes and edges, such as academic networks [8], social networks [34] and newsgroup networks [29]. The reason is that these methods do not consider the types of nodes and edges in their learning processes. However, the type information could be crucial for understanding heterogeneous networks. For example, the coauthor network in Figure 1(a) reveals the coauthor relations



(a) Coauthor Network (Homogeneous) (b) Academic Network (Heterogeneous)

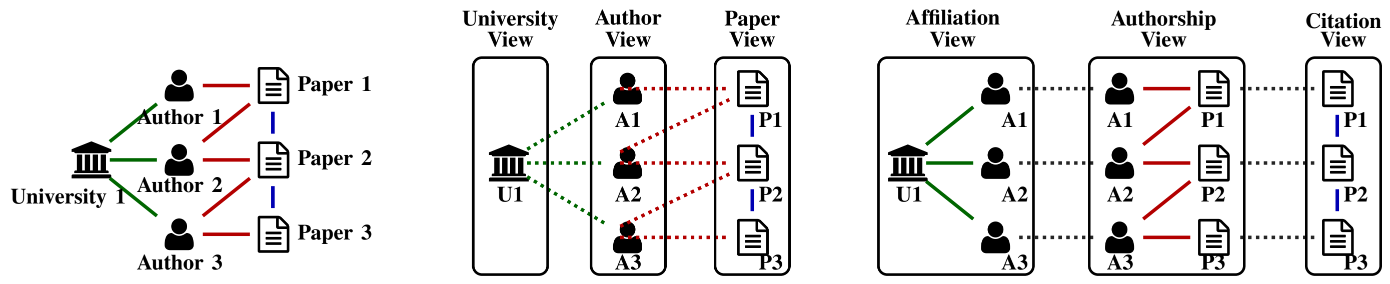
Fig. 1: Homogeneous and Heterogeneous Networks

among five authors, while the academic network in Figure 1(b) shows that two authors from the same university publish two papers sharing common keywords. Clearly, these two networks have different meanings. However, without node types, the two networks in Figure 1 will become exactly the same, and the information of the academic network is totally lost. Therefore, the network embedding methods should preserve both the structure and type information on heterogeneous networks.

To resolve this problem, several methods [6], [8], [10], [21]–[24], [29], [34] are proposed to learn embeddings on heterogeneous networks, which could be generally divided into two categories: (1) the path-based methods and (2) the multi-view methods.

The *path-based* methods [8], [10], [22], [23] learn the network embeddings by exploiting the proximity information of paths sampled from networks. However, these path-based methods either focus on specific network mining tasks such as the proximity search [22], [23], or require special meta-paths [39] as inputs [8], [10] which are difficult to be specified in practice [29].

The other methods [6], [21], [24], [29], [34] adopt the idea of *multi-view* learning [51], which separate the heterogeneous network into multiple views (i.e., subnetworks). Then, they learn the view-specific node embeddings inside each view, and transfer the learned information across multiple views. However, some multi-view methods [21], [24], [34] only consider networks with a single type of nodes. Therefore, they cannot deal with general heterogeneous networks with multiple node types, such as the academic network in Figure 2(a). In addition, some multi-view methods [6], [29] separate the network according to the node types. Therefore, when there is no edge between a single type of nodes, the views generated by these methods [6], [29] will only contain isolated nodes,



(a) An academic network. A blue edge is between two papers if one paper cites another. The red edges between authors and papers represent authorships. The green edges between authors and the universities reveal the affiliation of the authors.

(b) Multi-view methods HNE [6] and DMNE [29] divide views by node types, which could generate views with only isolated nodes, e.g., the university view and the author view. Embeddings cannot be learned from such views since they have no structure information.

(c) Our framework TransN divides views by edge types, which ensures that there is no isolated node in our generated views. Therefore, TransN can handle more general heterogeneous networks than the existing multi-view methods HNE and DMNE, since our generated views always contain structure information for any heterogeneous networks.

Fig. 2: Different View Separation Strategies on Heterogeneous Networks

where node embeddings cannot be learned inside such views due to the lack of structural information. For example, given the academic network in Figure 2(a), the views generated by the multi-view methods HNE [6] and DMNE [29] are shown in Figure 2(b), where the university view and the author view only contain isolated nodes. Therefore, embeddings cannot be learned in these two views since they have no structure information.

To address the problems above, in this paper, we propose a novel multi-view network embedding framework called Heterogeneous Network Representation Learning by **Trans**lating **N**ode Embeddings (**TransN**), to learn the node embeddings on heterogeneous networks. Basically, TransN has three advantages. (1) Compared with the path-based methods [8], [10], [22], [23], TransN can produce node embeddings for general downstream networks mining tasks, and does not require user-specified meta-paths as inputs. (2) TransN can learn embeddings on heterogeneous networks with multiple types of nodes and edges, which is more general than the existing multi-view methods [21], [24], [34] that can only handle one type of nodes. (3) TransN models a view as a collection of relations (rather than entities) with a specific type, which ensures that its generated views do not contain isolated nodes. Therefore, TransN is capable of dealing with heterogeneous networks which cannot be well handled by some of the previous multi-view methods [6], [29].

Specifically, to handle more general types of heterogeneous networks, there are two challenges to be addressed in this paper.

Firstly, a view in the existing multi-view network embedding methods [6], [21], [24], [29], [34] is always a homogeneous network which only contains a single type of nodes and edges. However, a view in our framework TransN could be either a homogeneous network, or a heterogeneous network with two types of nodes and one type of edges. Therefore, to learn the embeddings in our generated views, we cannot simply apply the existing methods designed for homogeneous views. To resolve this problem, in this paper, we propose a novel method to learn the node embeddings for both homogeneous and heterogeneous views within a unified framework.

Secondly, many multi-view network embedding methods [21], [34] assume that every node shares a consistent embedding in different views, since the views are usually complementary to each other in their studied networks [34]. However, the networks discussed in this paper is more general, and therefore, the information learned from different views contradicts each other. As shown in Figure 2(c), the authors A1 and A3 are similar under the affiliation view since they are in the same university, while they are dissimilar under the authorship view since they never publish a paper together. Therefore, it is difficult to determine whether their consistent embeddings shared across all views should be similar or not. To resolve this problem, in this paper, we propose a novel method to model the correlation (instead of the equality) between node embeddings in different views, which can capture more complex relations between nodes in different views than using shared node embeddings for all the views.

To summarize, we make the following contributions in this paper.

- We propose a novel multi-view network embedding framework, TransN, which can handle general heterogeneous networks with multiple types of nodes and edges.
- We propose a novel algorithm to learn the node embeddings inside each view, which can preserve the proximity information for both homogeneous and heterogeneous views within a unified framework.
- We propose a novel algorithm to translate (i.e., project) the node embeddings between different views based on the dual-learning mechanism [14], which can both capture the complex relations between nodes in different views, and preserve the proximity information inside each view during the translation.
- We conduct extensive experiments to compare our framework TransN with the state-of-the-art network embedding methods, where the experimental result shows that our method TransN is more effective than the competitors in various network mining tasks.

The rest of this paper is organized as follows. In Section II, we formally define the concepts and symbols in this paper, and define the heterogeneous network embedding problem. In Section III, we illustrate our framework TransN in detail.

TABLE I: Table of Notations

Notation	Description
$G = \{V, E, C_V, C_E\}$	A heterogeneous network with nodes V, edges E, node types C_V, and edge types C_E
$\phi_i = \{V_i, E_i\}$	The i-th view of a heterogeneous network
$\eta_{i,j}$	A view-pair consisting of views ϕ_i and ϕ_j
$\phi'_i = \{V'_i, E'_i\}$	A paired-subview of view ϕ_i
$\vec{n}$	The embedding of node n
$\vec{n}_i$	The view-specific embedding of node n in the view ϕ_i of a heterogeneous network

In Section IV, we demonstrate the effectiveness of TransN through extensive experiments. We discuss the related works in Section V. Finally, we conclude the paper in Section VI.

II. Problem Definition

We first introduce the concepts and symbols used in this paper, which are summarized in Table I.

Definition 1. *A **heterogeneous network** is an undirected graph $G = \{V, E, C_V, C_E\}$ with $|V|$ nodes and $|E|$ edges, where each node $v \in V$ (resp., each edge $e \in E$) is associated with a type $\zeta(v) \in C_V$ (resp., $\zeta(e) \in C_E$). Here, C_V and C_E are the sets of node types and edge types, respectively, where $|C_V| + |C_E| > 1$.*

Given a heterogeneous network G with $|C_E|$ types of edges, we separate the network G into $|C_E|$ views, where the i-th view of the network G is defined as follows.

Definition 2. *A **view** $\phi_i = \{V_i, E_i\}$ is a subnetwork, with $|V_i|$ nodes and $|E_i|$ edges, of the heterogeneous network $G = \{V, E, C_V, C_E\}$, where $E_i \subseteq E$ is the set of all edges with the i-th type in network G, and $V_i \subseteq V$ is the set of the end-nodes of edges in E_i.*

Based on the definition of views, given a heterogeneous network $G = \{V, E, C_V, C_E\}$ and its views $\{\phi_1, \phi_2, ...\phi_{|C_E|}\}$, we have:

$$\forall i \neq j, E_i \cap E_j = \emptyset, \text{ and } \bigcup_{i=1}^{|C_E|} E_i = E, \tag{1}$$

where E_i and E_j are the sets of edges in views ϕ_i and ϕ_j, respectively, while E is the set of edges in network G.

Although the sets of edges in different views are disjoint, nodes could be shared between views. Specifically, we define a pair of views which share some common nodes as a *view-pair*.

Definition 3. *A **view-pair** $\eta_{i,j}$ is a pair of views ϕ_i and ϕ_j in a heterogeneous network G, whose sets of nodes V_i and V_j satisfy that $V_i \cap V_j \neq \emptyset$.*

Particularly, the type of an edge implicitly restricts its end-nodes' types. As shown in Figure 2(a), the end-nodes of an authorship edge (in red color) must have the type *author* or *paper*, and the end-nodes of a citation edge (in blue color) always represent *papers*. Therefore, a view must either be a homogeneous network with a single type of nodes and edges, or be a heterogeneous network with two types of nodes and one type of edges. To distinguish these two kinds of views, we introduce two new concepts as follows.

Definition 4. *A **homo-view** is a view containing only a single type of nodes and edges; and a **heter-view** is a view containing two types of nodes and one type of edges.*

The key idea of our cross-view algorithm (see Section III-B) is to transfer information between different views. Specifically, the information transfer only makes sense between relevant views, and two views are relevant only when they share some common nodes, which are the natural bridges to transfer information between views. Therefore, in our cross-view algorithm, we focus on the common nodes (and their neighbor nodes), and remove the less important parts from each pair of views. Formally, given a view-pair $\eta_{i,j}$, we reduce the views ϕ_i and ϕ_j to a paired-subviews ϕ'_i and ϕ'_j as below.

Definition 5. *Given a view-pair $\eta_{i,j}$, the **paired-subview** ϕ'_i (resp., ϕ'_j) is sub-network of view ϕ_i (resp,. ϕ_j) formed by the set of nodes $M_{ij} \cap A_{ij}$ and edges between them, where M_{ij} is the set of common nodes in views ϕ_i and ϕ_j, and A_{ij} is the set of nodes which are adjacent to any nodes in M_{ij}.*

Finally, we formally define the problem as follows.

Problem Definition. Given a heterogeneous network $G = \{V, E, C_V, C_E\}$ and a positive integer $d \ll |V|$, the problem of **heterogeneous network embedding** is to represent each node n in network G by a d-dimensional vector $\vec{n} \in \mathbb{R}^d$.

Note that, a node could have different embeddings in different views, where the *view-specific* embedding of node n in the view ϕ_i is denoted by $\vec{n}_i$.

To address the heterogeneous network embedding problem, we propose a novel multi-view network embedding framework, TransN, which is elaborated in the following section.

III. TransN: The Proposed Framework

In this section, we introduce our proposed framework, TransN, for learning embeddings on heterogeneous networks. As shown in Figure 3, for each view of a heterogeneous network (step (a)), we propose a **single-view algorithm** (see Section III-A) to preserve the proximity information inside each single view, whose inputs are random walks sampled from the view (step (c)). Specifically, our random walks are **biased** with respect to node degrees, i.e., nodes with higher degrees are more likely to be sampled since they are usually more important in a network. On heter-views, our random walks are **correlated** random walks [2], since each step's direction is correlated to the previous step.

Meanwhile, for each view-pair (step (b)), we propose a **cross-view algorithm** (see Section III-B) to transfer the relations between views by translators (step (f)). Specifically, each translator is a stack of encoders modeled by neural networks, and the inputs of each translator are random walks sampled from paired-subviews (step (e)), instead of the original views.

Formally, TransN learns the node embeddings of the input network by minimizing the overall loss function (step (h)):

$$L_{\text{overall}} = L_{\text{single}} + L_{\text{cross}}, \tag{2}$$

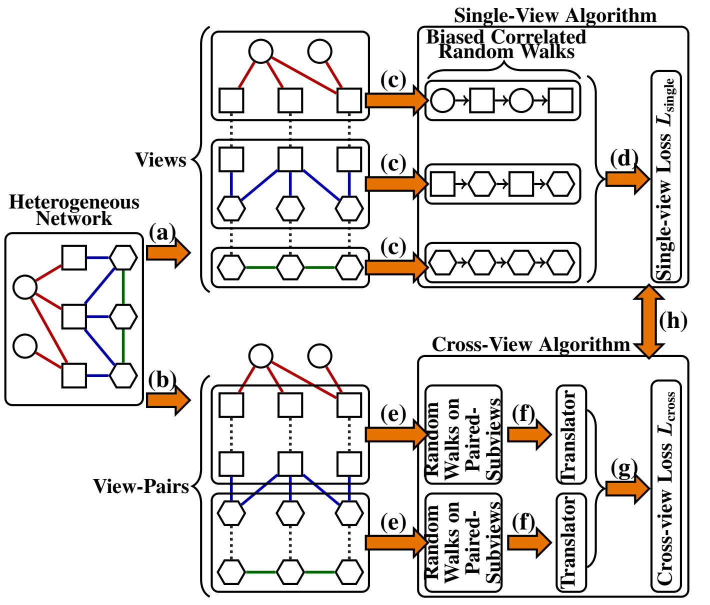

Heterogeneous Network
(a)
(b)
Views
View-Pairs
Single-View Algorithm
Biased Correlated Random Walks
(c)
(d)
Single-view Loss L_{single}
(h)
Cross-View Algorithm
(e)
Random Walks on Paired-Subviews
(f)
Translator
(g)
Cross-view Loss L_{cross}

Fig. 3: The proposed TransN framework. The shapes (i.e., circles, squares, and hexagons) represent nodes with various types. Solid lines are edges whose colors represent types, and dotted lines represent common nodes shared by view-pairs. The steps (a)∼(h) are illustrated in Section III.

where L_{single} and L_{cross} are the loss functions of our single-view and cross-view algorithms (steps (d) and (g)), respectively.

In Sections III-A and III-B, we illustrate the details of the single-view and cross-view algorithms in our framework TransN, respectively. Then, in Section III-C, we elaborate the optimization algorithm of our framework TransN. Finally, we analyze the time complexity of training our proposed TransN framework in Section III-D.

A. The Single-View Algorithm

The goal of our single-view algorithm is to preserve the node proximities inside each view. Inspired by the path-based network embedding methods [8], [13], [33], for each view $\phi_i = \{V_i, E_i\}$, we preserve the node proximities in view ϕ_i by minimizing the single-view loss function [13]:

$$L_{\text{single}} = -\log\left(\prod_{n\in V_i}\prod_{c\in\mathcal{S}(n)} p(c|\vec{n}_i)\right), \tag{3}$$

where $\mathcal{S}(n)$ is the set of context nodes of node n, and $p(c|\vec{n}_i)$ is the conditional probability that node c is node n's context node given the view-specific embedding $\vec{n}_i$ of node n in view ϕ_i, which is commonly formulated as a softmax function [8], [13], [33]. Specifically, Equation (3) aims to capture the proximity information in view ϕ_i, by promoting each node n to predict each of its context nodes $c \in \mathcal{S}(n)$ given its view-specific embedding $\vec{n}_i$. Therefore, the information encoded into the embedding $\vec{n}_i$ heavily depends on the selection of node n's context nodes in view ϕ_i.

In our single-view algorithm, we propose a random-walk-based method to select context nodes for each single view, which jointly deals with homo-views and heter-views within a unified framework. Formally, we define the context nodes of each node on a sampled path as follows.

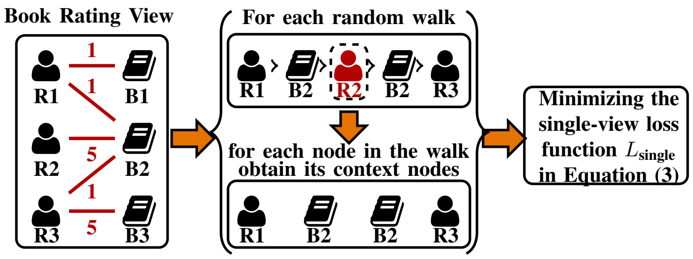

Book Rating View
R1
R2
R3
B1
B2
B3
1
1
5
1
5
For each random walk
R1
B2
R2
B2
R3
for each node in the walk obtain its context nodes
R1
B2
B2
R3
Minimizing the single-view loss function L_{single} in Equation (3)

Fig. 4: An example of our single-view algorithm on a book rating view, which contains three readers (R1, R2, and R3) and three books (B1, B2, and B3). The edge weights are the readers' rating scores of books (from one to five, the higher the better). The length of each random walk is 5.

Definition 6. *Given a sampled path $\lambda = \{n_1, n_2, ..., n_r\}$,*

- *when path λ is sampled from a homo-view, the **context nodes** of each node $n_k \in \lambda$ are nodes n_{k-1} (if $k > 1$), and n_{k+1} (if $k < r$);*
- *when path λ is sampled from a heter-view, the **context nodes** of each node $n_k \in \lambda$ are nodes n_{k-2} (if $k > 2$), n_{k-1} (if $k > 1$), n_{k+1} (if $k < r$), and n_{k+2} (if $k < r-1$).*

According to Definition 6, for each node n_k in a sampled path $\lambda = \{n_1, n_2, ..., n_r\}$, we consider nodes n_{k-1} and n_{k+1} as its context nodes to preserve the relations between node n_k and its neighbors. Moreover, in heter-views, we additionally take nodes n_{k-2} and n_{k+2} as node n_k's context nodes, which aims to capture the proximity between node n_k and its neighbors-of-neighbors (i.e., indirect neighbors). The reason is that a node in a heter-view is not only related to its direct neighbors by explicit edge connections, but also correlated with its indirect neighbors by sharing common end-nodes. Take the book rating view in Figure 4 as an example, the reader R1 is correlated with books B1 and B2 since she reads both books (i.e., they are neighbors in the view). In addition, although there is no edge between readers R1 and R3, these two readers are similar since they both read the book B2 (i.e., they are indirect neighbors in the view).

In the remaining part of this section, we illustrate how we control the direction of our random walks. Given the first k steps in a path $\lambda_k = \{n_1, n_2, ..., n_k\}$ sampled by a random walk, the probability of choosing a specific node n_{k+1} in the $(k+1)$-th step is:

$$\pi(n_{k+1}|\lambda_k) \propto \begin{cases} \pi_1(n_{k+1}|\lambda_k), & \text{if } k=1 \text{ or } \Delta = 0 \\ & \text{or on homo-views,} \\ \pi_1(n_{k+1}|\lambda_k)\cdot\pi_2(n_{k+1}|\lambda_k), & \text{otherwise,} \end{cases} \tag{4}$$

where:

$$\Delta = \max\{w_{v,n_k} : v\in\mathcal{N}(n_k)\} - \min\{w_{v,n_k} : v \in \mathcal{N}(n_k)\}, \tag{5}$$

and

$$\pi_1(n_{k+1}|\lambda_k) \propto \frac{w_{n_{k+1},n_k}}{\sum_{v\in\mathcal{N}(n_k)} w_{v,n_k}}, \tag{6}$$

$$\pi_2(n_{k+1}|\lambda_k) \propto 1 - \tfrac{1}{\Delta}\cdot(w_{n_{k+1},n_k} - w_{n_k,n_{k-1}}), \tag{7}$$

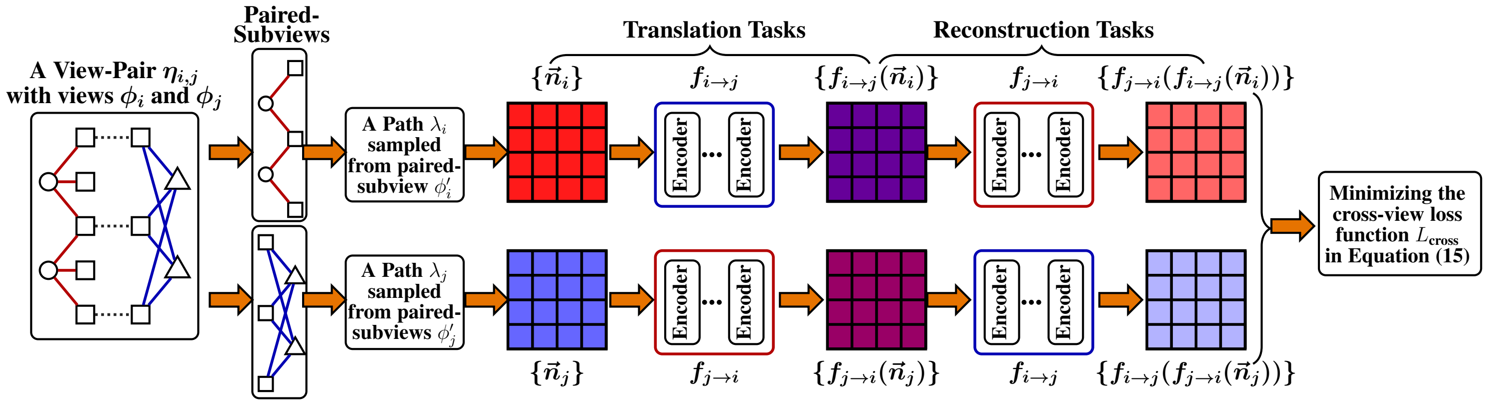

Paired-Subviews
Translation Tasks
Reconstruction Tasks
A View-Pair $\eta_{i,j}$ with views ϕ_i and ϕ_j
$\{\vec{n}_i\}$
$f_{i\to j}$
$\{f_{i\to j}(\vec{n}_i)\}$
$f_{j\to i}$
$\{f_{j\to i}(f_{i\to j}(\vec{n}_i))\}$
A Path λ_i sampled from paired-subview ϕ'_i
Encoder ... Encoder
Minimizing the cross-view loss function L_{cross} in Equation (15)
A Path λ_j sampled from paired-subviews ϕ'_j
$\{\vec{n}_j\}$
$f_{j\to i}$
$\{f_{j\to i}(\vec{n}_j)\}$
$f_{i\to j}$
$\{f_{i\to j}(f_{j\to i}(\vec{n}_j))\}$

Fig. 5: An overview of our cross-view algorithm on a view-pair $\eta_{i,j}$ with views ϕ_i and ϕ_j whose common node type is represented by the square. The sampled paths λ_i and λ_j only contain square nodes, and each row of the matrices is an embedding vector corresponding to a node on the sampled paths. In addition, $\vec{n}_i$ and $\vec{n}_j$ are view-specific embeddings of node n in views ϕ_i and ϕ_j, respectively, while $f_{i\to j}(\cdot)$ (resp., $f_{j\to i}(\cdot)$) is the translator (i.e., stack of encoders) to translate (i.e., project) node embeddings from view ϕ_i to view ϕ_j (resp., from view ϕ_j to view ϕ_i). Note that, each translator is used in two different places in this figure.

where $w_{x,y}$ is the weight of the edge between nodes x and y, and $\mathcal{N}(n)$ is the set of neighbors of node n, while Δ is the maximal difference between two weights of the edges adjacent to node n_k.

Specifically, in Equation (4), the probability $\pi_1(n_{k+1}|\lambda_k)$ makes our random walks prefer the edges with higher weights, since higher weights usually represents stronger relations between nodes in a network. Meanwhile, on heter-views, the probability $\pi_2(n_{k+1}|\lambda_k)$ allows our random walks to be more likely to choose an edge whose weight w_{n_{k+1},n_k} is close to the weight $w_{n_k,n_{k-1}}$ of the previous edge on the random walk, since the relation between two indirect neighbors is mainly dependent on their relations with the common end-nodes. For example, as shown in Figure 4, the reader R1 is more similar to the reader R3 than to the reader R2, since both readers R1 and R3 offer a low rating score to the book B2 (i.e., they dislike the book B2), while the reader R2 give a high rating score to the book B2 (i.e., she likes the book B2). Therefore, it is more reasonable to select node R3, instead of node R2, as node R1's context node.

B. The Cross-View Algorithm

The node embeddings learned by our single-view algorithm can only preserve the information inside each view of a heterogeneous network. However, the information inside each view could be biased and inaccurate. For instance, in the authorship view of Figure 2(c), our single-view algorithm will judge that the authors A1 and A3 are completely dissimilar since they never co-author a paper. In fact, these two authors are correlated since they serve the same university, and their papers P1 and P2 have mutual citations. Such information cannot be learned inside the authorship view solely, which requires the information transferred from the affiliation and citation views. Therefore, in this section, we propose a cross-view algorithm to transfer information between different views.

As discussed in Section I, the common nodes in different views are the natural bridges to transfer information between views. Therefore, our cross-view algorithm focus on modeling the relations between common nodes in each pair of views. Specifically, as shown in Figure 5, we capture the relations between two views ϕ_i and ϕ_j by two translators (i.e., stacks of encoders) $f_{i\to j}$ and $f_{j\to i}$, whose training objectives are both (1) to reduce the translation error (i.e., the *translation task*) and (2) to preserve the self-consistency of node embeddings (i.e., the *reconstruction task*).

In the remaining part of this section, we first introduce the overall process of our cross-view algorithm. Then, we illustrate the details of translators in our model. Finally, we propose two kinds of training tasks, the translation and reconstruction tasks, to be accomplished by our cross-view algorithm.

1) Overview: The process of our cross-view algorithm is shown in Figure 5. Specifically, given a view-pair $\eta_{i,j}$ with two views ϕ_i and ϕ_j, we first reduce it into two paired-subviews ϕ'_i and ϕ'_j (see Definition 5). Then, we sample some paths from each paired-subview by the random-walk-based method proposed in Section III-A. Moreover, on each sampled path, we remove the nodes which are not shared between the paired-subviews ϕ'_i and ϕ'_j, which makes our cross-view algorithm concentrate on the common nodes of two views. Finally, for each sampled path, our cross-view algorithm aims to accomplish two tasks, i.e., the translation and reconstruction tasks, to transfer the information between views and update the node embeddings.

2) Translator: The key component of our cross-view algorithm is the translator, denoted by $\mathcal{T}_{i\to j}$ (resp,. $\mathcal{T}_{j\to i}$), which aims to translate (i.e., project) the node embeddings on a sampled path from view ϕ_i to ϕ_j (resp,. ϕ_j to ϕ_i). Formally, the node embeddings on a sampled path λ is a matrix of size $|\lambda| \times d$, where d is the number of dimensions of node embeddings, and the k-th row of the matrix is the embedding of the k-th node on the sampled path.

A translator is a stack of encoders, where each encoder takes a matrix as the input, and outputs a new matrix which becomes the input of the next encoder in the translator. Specifically, each encoder is comprised by two neural network layers, i.e., a self-attention layer [44] and a feed-forward layer [11], where each layer is equivalent to a function projecting a matrix A into a new matrix defined as below.

$$\textbf{Self-attention layer: } \mathcal{S}(A) = \zeta(AA^T/\sqrt{d}) \cdot A \quad (8)$$

$$\textbf{Feed-forward layer: } \mathcal{F}(A) = relu(W \cdot A + b) \quad (9)$$

Here, the function $\zeta(\cdot)$ applies the softmax function [49] to each row of the matrix $AA^T/\sqrt{d}$, where d is the number of dimensions of each row in the input matrix A. In Equation (9), $W^{|\lambda|\times|\lambda|}$ and $b^{|\lambda|\times 1}$ are trainable parameters, where $|\lambda|$ is the length of the sampled path. In addition, $relu(x) = max\{0, x\}$ is the rectified linear units (RELU) [1] activation function of the feed-forward layer.

Therefore, given an embedding matrix A as the input, a translator $\mathcal{T}_{i\to j}(A)$ with $\mathcal{H}$ encoders can be represented by:

$$\mathcal{T}_{i\to j}(A) = \overbrace{\mathcal{F}(\mathcal{S}(\cdots\mathcal{F}(\mathcal{S}(\mathcal{F}(\mathcal{S}(A)\cdots)}^{2\mathcal{H}\text{ layers}}, \quad (10)$$

where functions $\mathcal{S}(\cdot)$ and $\mathcal{F}(\cdot)$ are defined in Equations (8) and (9), respectively.

Note that, our translators utilize the self-attention layers [44] for better capturing the proximity of nodes inside each sampled path since they could be context nodes of each other. It is because that the self-attention layer has been widely applied to learn the correlations inside sequence data [7], [38], [44].

3) Translation Tasks: We propose the translation tasks in order to capture the relations between commons nodes shared by different views, which is achieved by reducing the translation error. Inspired by the dual-learning [14], for each view-pair $\eta_{i,j}$ with views ϕ_i and ϕ_j, we define two symmetric **translation tasks**:

Task T1: For each path λ_i sampled from the paired-subview ϕ'_i, we translate its node embedding matrix A_i to a new matrix $\mathcal{T}_{i\to j}(A_i)$, such that the translated embedding matrix $\mathcal{T}_{i\to j}(A_i)$ is similar to the target embedding matrix A'_i;

Task T2: For each path λ_j sampled from the paired-subview ϕ'_j, we translate its node embedding matrix A_j to a new matrix $\mathcal{T}_{j\to i}(A_j)$, such that the translated embedding matrix $\mathcal{T}_{j\to i}(A_j)$ is similar to the target embedding matrix A'_j,

where A_i (resp., A'_i) is a matrix with $|\lambda_i|$ rows, whose k-th row is view ϕ_i's (resp., ϕ_j's) specific embedding of the k-th node on the sampled path λ_i. Likewise, A_j (resp., A'_j) is a matrix with $|\lambda_j|$ rows, whose k-th row is view ϕ_j's (resp., ϕ_i's) specific embedding of the k-th node on the sampled path λ_j. In addition, $\mathcal{T}_{i\to j}$ (resp,. $\mathcal{T}_{j\to i}$) is the translator which translates node embeddings from view ϕ_i to ϕ_j (resp,. ϕ_j to ϕ_i).

Specifically, the loss functions of the translation tasks are defined as follows.

$$L_{i\to j}(\lambda_i) = \frac{1}{|\lambda_i|}\sum_{a=1}^{|\lambda_i|}\sum_{b=1}^{d}(\mathcal{T}_{i\to j}(A_i)\odot A'_i)_{ab}\,, \quad (11)$$

$$L_{j\to i}(\lambda_j) = \frac{1}{|\lambda_j|}\sum_{a=1}^{|\lambda_j|}\sum_{b=1}^{d}(\mathcal{T}_{j\to i}(A_j)\odot A'_j)_{ab}, \quad (12)$$

where $L_{i\to j}(\lambda_i)$ and $L_{j\to i}(\lambda_j)$ are the loss functions of translation tasks T1 and T2, with respect to the sampled paths λ_i and λ_j, respectively. Here, d is the number of dimensions of node embeddings, $\odot$ is the element-wise product of matrices, and M_{ab} is the element in the a-th row and the b-th column of matrix M. Essentially, the loss function $L_{i\to j}(\lambda_i)$ (resp., $L_{j\to i}(\lambda_j)$) models the similarity between the translated embedding matrix $\mathcal{T}_{i\to j}(A_i)$ (resp., $\mathcal{T}_{j\to i}(A_j)$) and the target embedding matrix A'_i (resp., A'_j) by the average inner product[1] value for each pair of row vectors in matrices $\mathcal{T}_{i\to j}(A_i)$ (resp., $\mathcal{T}_{j\to i}(A_j)$) and A'_i (resp., A'_j).

4) Reconstruction Tasks: A node could have different embeddings in various views, whereas these embeddings still represent the same entity. Therefore, the translators should preserve the self-consistency of node embeddings, that is, an embedding matrix translated from view ϕ_i to view ϕ_j should be projected to itself when being translated reversely, i.e., $A_i \approx \mathcal{T}_{j\to i}(\mathcal{T}_{i\to j})(A_i)$, where $\mathcal{T}_{i\to j}(\cdot)$ (resp., $\mathcal{T}_{j\to i}(\cdot)$) is the translator to translate node embeddings from view ϕ_i to view ϕ_j (resp., from view ϕ_j to view ϕ_i). For this purpose, we define two **reconstruction tasks** for each view-pair $\eta_{i,j}$:

Task R1: For each path λ_i sampled from the paired-subview ϕ'_i, we translate its embedding matrix A_i from view ϕ_i to view ϕ_j, and then translate it back to view ϕ_i, such that the reconstructed embedding matrix $\mathcal{T}_{j\to i}(\mathcal{T}_{i\to j}(A_i))$ is similar to the original embedding matrix A_i;

Task R2: For each path λ_j sampled from the paired-subview ϕ'_j, we translate its embedding matrix A_j from view ϕ_j to view ϕ_i, and then translate it back to view ϕ_j, such that the reconstructed embedding matrix $\mathcal{T}_{i\to j}(\mathcal{T}_{j\to i}(A_j))$ is similar to the original embedding matrix A_j,

where the symbols are the same with those in translation tasks T1 and T2. Similar to Equations (11) and (12), the loss functions of the reconstruction tasks are formulated as:

$$L_{i\to j\to i}(\lambda_i) = \frac{1}{|\lambda_i|}\sum_{a=1}^{|\lambda_i|}\sum_{b=1}^{d}(\mathcal{T}_{j\to i}(\mathcal{T}_{i\to j}(A_i))\odot A_i)_{ab}\,, \quad (13)$$

$$L_{j\to i\to j}(\lambda_j) = \frac{1}{|\lambda_j|}\sum_{a=1}^{|\lambda_j|}\sum_{b=1}^{d}(\mathcal{T}_{i\to j}(\mathcal{T}_{j\to i}(A_j))\odot A_j)_{ab}\,, \quad (14)$$

where $L_{i\to j\to i}(\lambda_i)$ and $L_{j\to i\to j}(\lambda_j)$ are the loss functions of tasks R1 and R2, with respect to the sampled paths λ_i and λ_j, respectively. Here, d is the number of dimensions of node embeddings, $\odot$ is the element-wise product of matrices, and M_{ab} is the element in the a-th row and the b-th column of matrix M.

To summarize, the cross-view loss function is:

$$\begin{aligned} L_{\text{cross}} =& \frac{1}{|\Lambda_i|}\sum_{\lambda_i\in\Lambda_i}(L_{i\to j}(\lambda_i) + L_{i\to j\to i}(\lambda_i)) \\ &+ \frac{1}{|\Lambda_j|}\sum_{\lambda_j\in\Lambda_j}(L_{j\to i}(\lambda_j) + L_{j\to i\to j}(\lambda_j))\,, \end{aligned} \quad (15)$$

[1]Inner product is commonly utilized to measure the similarity of two vectors [13], [41]. The inner product value of two vectors is low when they are similar.

where Λ_i and Λ_j are the set of random paths sampled from views ϕ_i and ϕ_j, respectively. The loss functions $L_{i\to j}(\cdot)$, $L_{j\to i}(\cdot)$, $L_{i\to j\to i}(\cdot)$, and $L_{j\to i\to j}(\cdot)$ are defined in Equations (11)∼(14), respectively.

C. Model Optimization

We minimize the overall loss function L_{overall} of our TransN framework in Equation (2) by the stochastic optimization algorithm Adam [18] and the back-propagation algorithm [36].

Specifically, the loss function of our framework TransN is optimized by Algorithm 1, where ∇ is the symbol for gradient. In each iteration of Algorithm 1, we first minimize the single-view loss function L_{single} in Equation (3), where the parameters Θ_{single} to be optimized are the view-specific node embeddings for each view. Then, for each view-pair, we minimize the cross-view loss function L_{cross} in Equation (15), where the parameters Θ_{cross} to be optimized are the embeddings of nodes shared between the pair of views, and the trainable parameters W and b in our proposed translators.

In each iteration of Algorithm 1, we conduct both the single-view and the cross-view algorithms. Therefore, for each pair of views, although our cross-view algorithm only updates the embeddings of their common nodes, the updated embeddings of the common nodes will further be utilized to update the embeddings of their non-common nodes in the single-view algorithm of the next iteration. Therefore, for each pair of views, the information of their non-common nodes is transferred indirectly through their common nodes, which is not limited to their own view.

It is noteworthy that the importance of each view is related to the specific downstream graph mining tasks [34]. Since we aim to produce network embeddings for general graph mining tasks, it is reasonable to assume that the importances of views are equal. Therefore, in TransN, the final embedding of each node is the average of its view-specific embeddings.

D. Complexity Analysis

In this section, we prove the time complexity of Algorithm 1 in the following Theorem 1.

Theorem 1. *Given a network with z views and z' view-pairs, the time cost of Algorithm 1 is:*

$$\mathcal{O}(\delta T\rho(z+z') + dT\rho(z\log_2(\mu) + z'\mathcal{H}\rho)), \tag{16}$$

where δ is the average degree of the network, T is the number of sampled paths for each view in our single-view algorithm, and T is also the number of sampled pairs of paths for each view-pair in our cross-view algorithm. In addition, ρ is the length of each sampled path, d is the number of dimensions of node embeddings, μ is the maximal number of nodes in each view, and $\mathcal{H}$ is the number of self-attention and feed-forward layers in a translator.

Proof. In this proof, we prove the time costs of the single-view and cross-view algorithms for each iteration, respectively.

(1) For each view, the single-view algorithm first samples T paths with length ρ by our proposed random walk method. Particularly, according to Equations (6) and (7), each step in our random walk costs $\mathcal{O}(\delta)$ time to determine, where δ is the average degree of the network. Therefore, it costs $\mathcal{O}(\delta T\rho)$ time to sample paths from each view.

Then, the single-view algorithm optimizes the loss function in Equation (3). Note that, Equation (3) is the same as the skip-gram loss function in Word2Vec model [27], whose optimization time cost is proved to be [26] $\mathcal{O}(c\cdot(d+d\cdot\log_2(\mu)))$ for each sampled node in our single-view algorithm. Here, c is the number of context nodes for each sampled node, which is fixed to be 2 as discussed in Section III-A. Since we sample $T\rho$ nodes for each view, the time cost of the single-view algorithm is $\mathcal{O}(T\rho\cdot(d+d\cdot\log_2(\mu)))$ for each view.

Therefore, assume that the single-view algorithm samples T paths with length ρ from each view of a network with z views, where each view has at most μ nodes. For each iteration of Algorithm 1, the time cost of the single-view algorithm is:

$$\mathcal{O}(z(\delta T\rho + T\rho(d + d\log_2\mu)) = \mathcal{O}(zT\rho(\delta + d\log_2(\mu))),$$

where δ is the average degree of the network, and d is the number of dimensions of node embeddings.

(2) For each view-pair, the cross-view algorithm first samples T pairs of paths with length ρ, which costs $\mathcal{O}(\delta T\rho)$ time as proved in part (1) of this proof.

Then, the cross-view algorithm optimizes the loss function in Equation (15). According to Section III-B, for each sampled path, the cross-view algorithm needs to minimize two translation losses and two reconstruction losses. Note that a reconstruction task is essentially the combination of two translation tasks. Therefore, for each sampled path, the optimization time

Algorithm 1 Optimization Algorithm of TransN

Input: A heterogeneous network $G = \{V, E, C_V, C_E\}$, the number T of sampled paths in each view, learning rates γ_{single} and γ_{cross}, the set of hyper-parameters Φ, and the number of iterations K

Output: The embedding $\vec{n}$ for each node $n \in V$.

Notation: Θ_{single} : the parameters to optimize in the single-view loss function L_{single} (in Equation (3)).
Θ_{cross} : the parameters to optimize in the cross-view loss function L_{cross} (in Equation (15)).

```
1: Generate views and view-pairs of the network $G$
2: for $iter = 1...K$ do
3:     for each view $\phi_i$ do
4:         $\{\lambda_1, \lambda_2, ...\lambda_T\} \leftarrow$ paths sampled from view $\phi_i$
5:         for each sampled path $\lambda$ do
6:             Compute $L_{\text{single}}$ according to Section III-A
7:             $\Theta_{\text{single}} \leftarrow \Theta_{\text{single}} - \gamma_{\text{single}} \cdot \nabla_{\Theta_{\text{single}}} L_{\text{single}}$
8:     for each view-pair $\eta_{i,j}$ do
9:         $\{\langle\lambda_i, \lambda_j\rangle_1, ...\langle\lambda_i, \lambda_j\rangle_T\} \leftarrow$ paths sampled from $\eta_{i,j}$
10:        for each sampled pairs of paths $\langle\lambda_i, \lambda_j\rangle$ do
11:            Compute $L_{\text{cross}}$ according to Section III-B
12:            $\Theta_{\text{cross}} \leftarrow \Theta_{\text{cross}} - \gamma_{\text{cross}} \cdot \nabla_{\Theta_{\text{cross}}} L_{\text{cross}}$
13: for each node $n \in V$ do
14:     $\vec{n} \leftarrow$ the average of node $n$'s view-specific embeddings
```

TABLE II: Statistic of Heterogeneous Network Datasets in Our Experiments

Dataset	#Nodes	#Edges	Node Types (#Nodes of Each Type)	#Labeled Nodes	Edge Types (#Edges of Each Type)
AMiner	4,774	17,795	Author(2,161), Paper(2,555), Venue(58)	Papers(2,555)	AA(3,836), AP(6,072), PP(5,332), PV(2,555)
BLOG	63,166	1,983,003	User(57,753), Keywords(5,413)	Users(57,753)	UU(1,409,112), UK(329,941), KK(243,950)
App-Daily	192,416	666,145	Applet(147,968), User(16,527), Keyword(27,921)	Applets(5,375)	AU(299,630), AK(366,515)
App-Weekly	418,374	3,843,931	Applet(155,194), User(233,396), Keyword(29,784)	Applets(5,375)	AU(3,427,677), AK(416,254)

cost of the cross-view algorithm is $\mathcal{O}(2\mathbb{T} + 2 \cdot 2\mathbb{T}) = \mathcal{O}(\mathbb{T})$, where $\mathbb{T}$ is the optimization time cost of a translation task.

Note that the time cost of optimizing a self-attention layer is proved to be $\mathcal{O}(\rho^2 \cdot d)$ [44] for each sampled path. In addition, the time cost of optimizing a feed-forward layer by the back-propagation algorithm is the same as the time cost of computing function $\mathcal{F}$ in Equation (9) [36], which is clearly $\mathcal{O}(\rho^2 \cdot d)$ for each sampled path, where ρ is the length of the sampled paths, and d is the number of dimensions of node embeddings. Therefore, for each sampled path, the optimization time cost of the cross-view algorithm is $\mathcal{O}(\mathbb{T}) = \mathcal{O}(\mathcal{H}\rho^2 d)$, where $\mathcal{H}$ is the number of self-attention layers and feed-forward layers in a translator.

Therefore, assume that the cross-view algorithm samples T pairs of paths with length ρ from each view of a network with z' view-pairs. For each iteration of Algorithm 1, the time cost of the cross-view algorithm is

$$\mathcal{O}(z' \cdot (\delta T\rho + T\mathcal{H}\rho^2 d)),$$

where $\mathcal{H}$ is the number of self-attention and feed-forward layers in a translator, and d is the number of dimensions of node embeddings.

To conclude, given a network with z views and z' view-pairs, the time cost of Algorithm 1 is:

$$\begin{aligned}&\overbrace{\mathcal{O}(zT\rho(\delta + d\log_2(\mu)))}^{\text{Single-View Algorithm}} + \overbrace{\mathcal{O}(z'(\delta T\rho + T\mathcal{H}\rho^2 d))}^{\text{Cross-View Algorithm}}\\ =&\mathcal{O}(\delta T\rho(z + z') + dT\rho(z\log_2(\mu) + z'\mathcal{H}\rho)). \quad (17)\end{aligned}$$

Theorem 1 is proved. □

IV. Experiments

In this section, we conduct extensive experiments to evaluate our TransN framework. Specifically, we first introduce the settings of our experiments. Then, we compare TransN framework with the state-of-the-art networks embedding methods on two widely considered network mining tasks: node classification and link prediction. Finally, we conduct an ablation study on our TransN framework to investigate whether each of its components helps to generate better network representations.

A. Experiment Setup

1) Datasets: We evaluate our TransN framework on four real-world heterogeneous networks, whose statistics are listed in Table II.

- **AMiner** [42], [43] is an academic network consisting of 2,161 authors and 2,555 papers published in 58 computer science venues, where each paper has a label indicating its research topic. The network has four types of edges, which represents: co-authorship (author-to-author), authorship (author-to-paper), citation (paper-to-paper), and publication (paper-to-venue), respectively. All edges in this network have unit weights.
- **BLOG** [47] is a social network which contains 57,753 users and 5,413 keywords posted in their blogs, where each user has a label representing her major interest field. There are three types of edges in this network, which are: friendship (user-to-user), keyword-usage (user-to-keyword), and keyword-relevance (keyword-to-keyword). All edges in this network have unit weights.
- **App-Daily*** is a network built upon the usage and query logs in a single day from a real-world mobile applet store, which contains 147,968 applets, 16,527 users, and 27,921 query keywords. In this network, an edge $\langle u, x\rangle$ exists between a user u and an applet x as long as user u uses applet x, where the weight of edge $\langle u, x\rangle$ is the time that user u spends on applet x in this day. In addition, the weight of an edge $\langle x, k\rangle$ between an applet x and a query keyword k is the number of times that some users download applet x through the query result pages of keyword k in this day. Particularly, there are 5,375 applets labeled with their categories.
- **App-Weekly*** is a network built upon the usage and query logs in a week from a real-world mobile applet store, which consists of 155,194 applets, 233,396 users, and 29,784 query keywords. The meanings of the edges in this network are the same as those in App-Daily, and there are 5,375 applets labeled with their categories.

2) Compared Methods: We compare TransN with seven state-of-the-art network embedding methods as follows.

- **LINE** [41] is a network embedding method on homogeneous networks. We use LINE which considers the second order proximity between nodes.
- **Node2Vec** [13] is a homogeneous network embedding method, which aims to capture the local structure information in a network by exploiting biased random walks.
- **Metapath2Vec** [8] is a path-based heterogeneous network embedding method, which improves Node2Vec by constraining random walks with user-defined meta-paths.
- **HIN2VEC** [10] is a path-based embedding method for heterogeneous networks. Compared with Metapath2Vec, it requires user to specify meta-paths with fixed lengths, instead of a particular meta-path for each network.
- **MVE** [34] is a multi-view heterogeneous network embedding method. Since all other methods are unsupervised, for a fair comparison, we use the unsupervised variant [34] of MVE which assigns equal weights for views when fusing view-specific embeddings.
- **R-GCN** [37] is an embedding method which utilizes the graph neural network [50] to learn the embeddings for entities and relations on knowledge graphs.
- **SimplE** [17] is a recent embedding method on knowledge graphs, which is an enhancement of the Canonical

*The App-Daily and App-Weekly are provided by Tencent Corporation.

TABLE III: Results of the Node Classification Task

	AMiner		BLOG		App-Daily		App-Weekly	
Method	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**
LINE	0.7216	0.7683	0.2086	0.4373	0.1261	0.2564	0.1238	0.2310
Node2Vec	0.7056	0.7861	0.2312	0.4502	0.1277	0.2424	0.1209	0.2341
Metapath2Vec	0.7869	0.8086	0.2763	0.4680	0.1875	0.3636	0.1757	0.3235
HIN2VEC	0.7998	0.8672	0.3069	0.4774	0.1731	0.3333	0.1472	0.3235
MVE	0.7603	0.8578	0.2590	0.4538	0.1567	0.2727	0.1288	0.2924
R-GCN	0.8325	0.8939	0.2860	0.4633	0.1833	0.3429	0.1637	0.2737
SimplE	0.7927	0.8097	0.3036	0.4648	0.1648	0.3011	0.1292	0.2986
TransN	**0.8465**	**0.9176**	**0.3230**	**0.4840**	**0.3713**	**0.5758**	**0.3016**	**0.4706**

TABLE IV: AUC Scores of the Link Prediction Task

Method	**AMiner**	**BLOG**	**App-Daily**	**App-Weekly**
LINE	0.7221	0.5819	0.7421	0.7520
Node2Vec	0.7434	0.5732	0.7339	0.7707
Metapath2Vec	0.8323	0.6059	0.8227	0.8552
HIN2VEC	0.8016	0.6123	0.8311	0.7880
MVE	0.7967	0.5820	0.7491	0.7822
R-GCN	0.8605	0.6389	0.7933	0.7867
SimplE	0.8425	0.6121	0.8205	0.8246
TransN	**0.8835**	**0.7551**	**0.8467**	**0.8668**

Polyadic decomposition [15] for tensors.

It is noteworthy that, the homogeneous network embedding methods do not take the type information of nodes and edges into their consideration. Therefore, the node and edge types of the networks input into methods LINE [41] and Node2Vec [13] are removed in our experiments.

In addition, the knowledge graph embedding methods R-GCN [37] and SimplE [17] consider the types of nodes and edges in their learning process, which can be directly applied to heterogeneous networks. Since methods R-GCN and SimplE do not utilize the weight of edges, the networks input into these methods have unit edge weights in our experiments.

3) Parameter Settings: For all methods, the number of dimensions of node embeddings is set as 128, and the initial learning rate is set as 0.025 as recommended in [34], [41]. The other parameters are the same with those suggested in their original papers [8], [10], [13], [17], [34], [37], [41]. Particularly, for method Metapath2Vec, we adopt the recommended meta-path "APVPA" [8] on network AMiner, and use meta-path "UTU" on network BLOG, as well as meta-path "UAKAU" on networks App-Daily and App-Weekly.

For our proposed framework TransN, we set the length of each random walk to be 80 as recommended in [13], and we set the number of sampled paths starting from each node n as $\max(\min(\tau_n, 32), 10)$, where τ_n is node n's degree. We set the number of encoders as $\mathcal{H}$=6 as suggested in [44].

B. Quantitative Results

In this section, we compare the methods on two widely-used network mining tasks [13], [34]: (supervised) node classification and (unsupervised) link prediction, where the experimental results are analyzed in the following two subsections.

1) Node Classification.: For each experimental network, we first learn the node embeddings by each method. Then, we randomly select 90% of the labeled nodes for training, where the remaining nodes are for testing. Next, we use the embeddings and labels of nodes in the training set to train a Logistics regression classifier [28] with default parameters in the scikit-learn package [32]. Finally, we use the classifier to predict the labels of nodes in the testing set, whose performance is evaluated by the commonly-used micro-F1 and macro-F1 scores [13], [41]. For each method, we repeat the prediction for ten times and report the average F1 scores.

Table III shows the results of the node classification task. Generally, the embedding methods which considers the types of node and edges (i.e., Metapath2Vec, HIN2VEC, MVE, TransN, R-GCN, and SimplE) perform better than the homogeneous ones (i.e., LINE and Node2Vec). Moreover, our proposed framework TransN outperforms all the compared methods on both Macro-F1 and Micro-F1 scores, which indicates that TransN can generate more useful network embeddings than the competitors in the node classification task.

For macro-F1 scores, TransN exceeds all the competitors by at least 98% and 72% on networks App-Daily and App-Weekly, while only achieves a gain of around 5.2% on network BLOG. There are two reasons for this difference. Firstly, the single-view algorithm in TransN has novel designs to capture the information residing in the weights of edges. Therefore, TransN has more advantages on weighted networks (e.g., App-Daily and App-Weekly) than unit-weighted networks (e.g., BLOG). Secondly, the density of network BLOG is above 20 times higher than those of App-Daily and App-Weekly. On such a dense network, the information is sufficient for the compared methods to produce promising embeddings. However, on sparse networks such as App-Daily and App-Weekly, TransN performs significantly better than the competitors.

2) Link Prediction.: We first randomly remove 40% edges from each experimental network, and randomly select the same number of nonadjacent node pairs. Then, we consider the link prediction task as a binary classification problem, where the end-nodes of the removed edges are positive examples, and the selected pairs of nonadjacent nodes are negative examples. Next, on the remaining sub-networks, we learn the node embeddings by each method. For each pair of nodes, we model the likelihood of an edge existing between them by the inner-product of their embeddings. Finally, we use the likelihoods as prediction scores and measure the performance of each method by the AUC metric [9].

Table IV shows the results of the link prediction task. Generally, TransN outperforms all the competitors on our experimental networks, which indicates that TransN can better capture and infer node relations than the compared methods.

Specifically, for AUC scores, TransN exceeds all the compared methods by at least 18.1% on network BLOG, while only achieves gains of 1.9% and 1.3% over the competitors on

TABLE V: Results of the Ablation Study on TransN

	AMiner		**BLOG**		**App-Daily**		**App-Weekly**	
Method	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**	**Macro-F1**	**Micro-F1**
TransN-Without-Cross-View	0.7415	0.8573	0.3021	0.4694	0.1197	0.1818	0.1310	0.2647
TransN-With-Simple-Walk	0.7725	0.8776	0.3194	0.4715	0.2945	0.3697	0.2237	0.3994
TransN-With-Simple-Translator	0.7761	0.8690	0.3159	0.4752	0.2591	0.3636	0.2235	0.3588
TransN-Without-Translation-Tasks	0.7778	0.8706	0.3200	0.4769	0.2402	0.4061	0.2277	0.4176
TransN-Without-Reconstruction-Tasks	0.7490	0.8549	0.3072	0.4770	0.2476	0.3939	0.2360	0.3706
TransN	**0.8465**	**0.9176**	**0.3230**	**0.4840**	**0.3713**	**0.5758**	**0.3016**	**0.4706**

networks App-Daily and App-Weekly, respectively. The reason for this difference is that TransN exploits information from multiple views to predict an unobserved edge, whereas such information is more effective when the relations in different views are more correlated. Assuming that we remove an edge between two users on network BLOG, TransN is very likely to recover this edge when these two users post many common keywords in their blogs. In this case, TransN predicts an edge by transferring information from the "user-to-keyword" view to the "user-to-user" view, which is helpful since similar users usually post common keywords. However, on networks App-Daily and App-Weekly, it is less effective to predict edges by transferring information between views. The reason is that the relations "user-to-applet" and "keyword-to-applet" are weakly correlated, since a user's usage of an applet scarcely relates to whether the applet is searched by a keyword. Therefore, compared with the competitors, TransN has more advantages on networks whose views are more correlated (e.g. BLOG).

C. *Ablation Study on TransN*

In this section, we conduct experiments to verify the effectiveness of different parts in TransN. Specifically, we remove five critical components from our framework TransN separately, and evaluate the performance of each degenerated method on the node classification task, where the experiment settings are the same with those in Section IV-B. The degenerated methods are as follows.

- **TransN-Without-Cross-View** is TransN without the cross-view algorithm, which is essentially Algorithm 1 without Lines 8$\sim$12.
- **TransN-With-Simple-Walk** is TransN which uses simple random walks as the input of its single-view algorithm. The starting node of each simple random walk is randomly selected, and simple random walks neglect the weights of edges.
- **TransN-With-Simple-Translator** is TransN which uses a single feed-forward layer [11] to replace each translator in our cross-view algorithm.
- **TransN-Without-Translation-Tasks** is TransN without translation tasks, which only considers reconstruction tasks and losses in the cross-view algorithm.
- **TransN-Without-Reconstruction-Tasks** is TransN without reconstruction tasks, which only considers translation tasks and losses in the cross-view algorithm.

The results of the ablation study on TransN are shown in Table V. Generally, TransN out-performs all of the degenerated methods, which indicates that each component in TransN has some contributions to the overall performance and thus is indispensable.

Specifically, the result of TransN-With-Simple-Walk proves the effectiveness of our control strategy of the random walks on heterogeneous networks (see Equation (4) in our single-view algorithm). In addition, the result of TransN-With-Simple-Translator demonstrates the necessity of using our proposed translator rather than a simple feed-forward layer in our cross-view algorithm. Moreover, TransN-Without-Translation-Tasks and TransN-Without-Reconstruction-Tasks achieve similar performances on all networks, which indicates that the translation and reconstruction tasks are equally important for learning embeddings in our framework TransN.

Particularly, TransN-Without-Cross-View has the worst performance on all networks in the experiment, which reveals that our cross-view algorithm is crucial to the performance of our framework TransN. As discussed in Section III-B, the information learned inside each single view could be biased and inaccurate. Therefore, TransN cannot perform well without our cross-view algorithm which jointly considers the information across multiple views.

D. *Case Study*

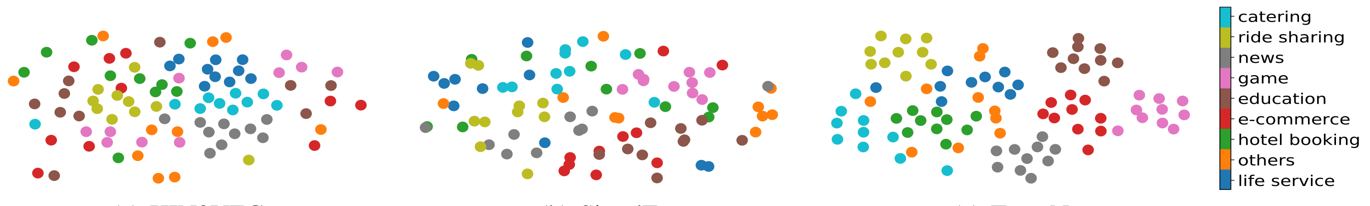
From network App-Daily, we randomly select ten applets for each of the nine category (e.g., catering, ride sharing, and life service), and visualize the embeddings of the selected 90 applets in Figure 6, where the embeddings are learned by HIN2VEC [10], SimplE [17], and TransN, respectively. Each point in Figure 6 corresponds to the embedding of an applet from App-Daily, whose color represents the applet's category.

According to Figure 6, for applets with different categories, their embeddings learned by TransN are more separated from each other compared with those learned by HIN2VEC and SimplE, which demonstrates that TransN is more effective than HIN2VEC and SimplE in the node classification task. Note that the embeddings of applets in "others" category are distant to each other for all three methods, since the "others" category is too general and could contain applets which are completely unrelated to each other.

Moreover, as shown in Figure 6(c), the embeddings learned by TransN reveal the relations between categories of applets. For instance, the embeddings of applets in category "hotel booking" are closer to those in category "catering" than to those in category "game", which indicates that the "hotel booking" applets have stronger commonality with the "catering" applets than the "game" applets.

V. Related Works

Our work is related to the existing approaches for learning network embeddings, which can be generally divided into three categories: (1) methods on homogeneous networks, (2) methods on heterogeneous networks, and (3) methods on knowledge graphs.



(a) HIN2VEC (b) SimplE (c) TransN

Fig. 6: 2D t-SNE [25] projections of the embeddings of 90 applets from App-Daily, ten each from 9 categories.

Methods on Homogeneous Networks. Most of the previous studies focus on homogeneous networks. Early methods such as LLE [35] can only preserve the relations between adjacent nodes, and neglect relations between nonadjacent nodes [41]. To address this problem, many methods are proposed to preserve the higher-order proximity [4] in a network. Specifically, a few of them [30] are factorization-based [12], which represent networks by adjacency matrices and obtain the node embeddings by matrix factorization, while most of them [13], [19], [33], [41], [45] utilize neural networks to preserve the node proximities. Specifically, DeepWalk [33] and Node2Vec [13] aim to learn the relations between nodes on sampled paths of a network by utilizing the skip-gram model [27]. Besides, SDNE [45] utilizes deep auto-encoders [11] to preserve the first and the second order of proximities [4] on a network, while DNGR [5] exploits the random walks together with auto-encoders, which aims to preserve the higher-order of proximity on a network. In addition, GCN [19] is a semi-supervised method which learns the network embeddings by iteratively aggregating neighbors' embeddings for each node, and LINE [41] is designed to preserve both the first and the second order proximity on a network by minimizing the KL divergence [11] between the distribution of actual node relations and the one of the learned node embeddings.

Methods on Heterogeneous Networks. Many methods [6], [8], [10], [21]–[24], [29], [34] are proposed to learn embeddings on heterogeneous networks. Specifically, the path-based methods D2AGE [23] and IPE [22] aim to produce embeddings particularly for the proximity search task. As for general network mining tasks, Metapath2Vec [8] and HIN2Vec [10] aim to capture the proximity between nodes on sampled paths of a network, where both of them exploit user-specified meta-paths [39] to control the random walks. Besides, HNE [6] is a multi-view method which integrates the information from various views by projecting their embeddings into a common space. In addition, several methods [21], [24], [34] aim to combine view-specific embeddings into a consistent embedding shared by all views with (semi-)supervised learning. Recently, DMNE [29] is proposed to learn a unified node embedding for each node by co-regularizing the embeddings of different types of nodes through minimizing their proposed disagreement loss.

Methods on Knowledge Graphs. A knowledge graph (KG) [31], [46] is a multi-relational graph whose nodes correspond to entities and edges represents fact triplets. Here, a fact triplet (h, r, t) means that entity h has relation r with entity t. Specifically, many methods are proposed to learn embeddings on KGs, where most of them aim to minimize some scoring functions $f(h, r, t)$ for each triplet (h, r, t). For instance, TransE [3] aims to minimize the scoring function $f(h, r, t) = \|\vec{h}+\vec{r}-\vec{t}\|_2$, where $\vec{h}$, $\vec{r}$, and $\vec{t}$ are the embeddings of entity h, relation r, and entity t, respectively. Later, TransH [48], TransR [20], and TransD [16] are proposed to overcome the flaws of TransE in dealing with one-to-many, many-to-one, and many-to-many relations. Recently, SimplE [17] is proposed to learn fully expressive embeddings on KGs, and RotatE [40] is proposed to capture various relation patterns such as symmetry, inversion, and composition in their trained embeddings. Particularly, method R-GCN [37] does not learn KG embeddings by minimizing a scoring function. Instead, R-GCN proposes a graph neural network [50] to learn KG embeddings in an end-to-end framework.

The major differences between the embedding methods on KGs and those on heterogeneous networks are as follows.

- The KG embedding methods [3], [17], [40] can produce embeddings for both entities (nodes) and relations (edges), while the embedding methods [8], [10], [34] on heterogeneous networks only focus on learning node embeddings.
- The KG embeddings [31] focus on preserving the type information of *relationship* between entities (nodes), while the heterogeneous network embeddings [8], [10], [34] aim to preserve the *proximity* between nodes. Therefore, the KG embedding methods [17], [37] often neglect the weights of edges, while in heterogeneous network embedding methods [8], [34], larger edge weights usually means higher proximity between edges' end-nodes.

VI. Conclusion

In this paper, we propose TransN, a framework to learn network embeddings on heterogeneous networks. In TransN, we propose a single-view algorithm to learn the node embeddings inside each single view, which preserves the node proximities through exploiting the information in paths sampled by our proposed biased correlated random walks. Moreover, we propose a cross-view algorithm to transfer information across views by translating and reconstructing embeddings of common nodes in different views. We evaluate TransN on real-world heterogeneous networks, where the experimental results show that our framework TransN is more effective than the compared methods on various network mining tasks.

References

[1] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[2] E. A. Bender, L. B. Richmond *et al.*, "Correlated random walks," *The Annals of Probability*, vol. 12, no. 1, pp. 274–278, 1984.

[3] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[4] H. Cai, V. W. Zheng, and K. Chang, "A comprehensive survey of graph embedding: problems, techniques and applications," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[5] S. Cao, W. Lu, and Q. Xu, “Deep neural networks for learning graph representations.” in *AAAI*, 2016, pp. 1145–1152.

[6] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, “Heterogeneous network embedding via deep architectures,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 119–128.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[8] Y. Dong, N. V. Chawla, and A. Swami, “metapath2vec: Scalable representation learning for heterogeneous networks,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 135–144.

[9] T. Fawcett, “An introduction to roc analysis,” *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[10] T.-y. Fu, W.-C. Lee, and Z. Lei, “Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning,” in *Proceedings of the 26th ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1797–1806.

[11] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[12] P. Goyal and E. Ferrara, “Graph embedding techniques, applications, and performance: A survey,” *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[13] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[14] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T.-Y. Liu, and W.-Y. Ma, “Dual learning for machine translation,” in *Advances in Neural Information Processing Systems*, 2016, pp. 820–828.

[15] F. L. Hitchcock, “The expression of a tensor or a polyadic as a sum of products,” *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.

[16] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, “Knowledge graph embedding via dynamic mapping matrix,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 687–696.

[17] S. M. Kazemi and D. Poole, “Simple embedding for link prediction in knowledge graphs,” 2018.

[18] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[19] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.

[20] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, “Learning entity and relation embeddings for knowledge graph completion,” in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[21] Y. Liu, L. He, B. Cao, P. S. Yu, A. B. Ragin, and A. D. Leow, “Multi-view multi-graph embedding for brain network clustering analysis,” *arXiv preprint arXiv:1806.07703*, 2018.

[22] Z. Liu, V. W. Zheng, Z. Zhao, Z. Li, H. Yang, M. Wu, and J. Ying, “Interactive paths embedding for semantic proximity search on heterogeneous graphs,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1860–1869.

[23] Z. Liu, V. W. Zheng, Z. Zhao, F. Zhu, K. C.-C. Chang, M. Wu, and J. Ying, “Distance-aware dag embedding for proximity search on heterogeneous graphs,” in *Proceddings of the 32th AAAI Conference on Artificial Intelligence*, 2018.

[24] T. Ma, C. Xiao, J. Zhou, and F. Wang, “Drug similarity integration through attentive multi-view graph auto-encoders,” *arXiv preprint arXiv:1804.10850*, 2018.

[25] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[28] N. M. Nasrabadi, “Pattern recognition and machine learning,” *Journal of electronic imaging*, vol. 16, no. 4, p. 049901, 2007.

[29] J. Ni, S. Chang, X. Liu, W. Cheng, H. Chen, D. Xu, and X. Zhang, “Co-regularized deep multi-network embedding,” in *Proceedings of the World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2018, pp. 469–478.

[30] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, “Asymmetric transitivity preserving graph embedding,” in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1105–1114.

[31] H. Paulheim, “Knowledge graph refinement: A survey of approaches and evaluation methods,” *Semantic web*, vol. 8, no. 3, pp. 489–508, 2017.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, “Scikit-learn: Machine learning in python,” *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[33] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in *Proceedings of the 20th ACM SIGKDD*. ACM, 2014, pp. 701–710.

[34] M. Qu, J. Tang, J. Shang, X. Ren, M. Zhang, and J. Han, “An attention-based collaboration framework for multi-view network representation learning,” in *Proceedings of the 26th ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1767–1776.

[35] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[36] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, p. 533, 1986.

[37] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, “Modeling relational data with graph convolutional networks,” *arXiv preprint arXiv:1703.06103*, 2017.

[38] T. Shen, T. Zhou, G. Long, J. Jiang, S. Pan, and C. Zhang, “Disan: Directional self-attention network for rnn/cnn-free language understanding,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[39] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, “Pathsim: Meta path-based top-k similarity search in heterogeneous information networks,” *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 992–1003, 2011.

[40] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, “Rotate: Knowledge graph embedding by relational rotation in complex space,” *arXiv preprint arXiv:1902.10197*, 2019.

[41] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “Line: Large-scale information network embedding,” in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[42] J. Tang, J. Sun, C. Wang, and Z. Yang, “Social influence analysis in large-scale networks,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 807–816.

[43] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, “Arnetminer: extraction and mining of academic social networks,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 990–998.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[45] D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in *Proceedings of the 22nd ACM SIGKDD*. ACM, 2016, pp. 1225–1234.

[46] Q. Wang, Z. Mao, B. Wang, and L. Guo, “Knowledge graph embedding: A survey of approaches and applications,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[47] X. Wang, L. Tang, H. Gao, and H. Liu, “Discovering overlapping groups in social media,” in *Proceedings of the 10th IEEE International Conference on Data Mining*. IEEE, 2010, pp. 569–578.

[48] Z. Wang, J. Zhang, J. Feng, and Z. Chen, “Knowledge graph embedding by translating on hyperplanes,” in *Twenty-Eighth AAAI conference on artificial intelligence*, 2014.

[49] Wikipedia, “Softmax function,” 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Softmax_function

[50] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” *arXiv preprint arXiv:1901.00596*, 2019.

[51] J. Zhao, X. Xie, X. Xu, and S. Sun, “Multi-view learning overview: Recent progress and new challenges,” *Information Fusion*, vol. 38, pp. 43–54, 2017.