Is a Single Vector Enough? Exploring Node Polysemy for Network Embedding

Ninghao Liu,[1] Qiaoyu Tan,[1] Yuening Li,[1] Hongxia Yang,[2] Jingren Zhou,[2] Xia Hu[1]
[1]Department of Computer Science and Engineering, Texas A&M University, TX, USA
[2]Alibaba Group, Hangzhou, China
{nhliu43,qytan,liyuening,xiahu}@tamu.edu,{yang.yhx,jingren.zhou}@alibaba-inc.com

ABSTRACT

Networks have been widely used as the data structure for abstracting real-world systems as well as organizing the relations among entities. Network embedding models are powerful tools in mapping nodes in a network into continuous vector-space representations in order to facilitate subsequent tasks such as classification and link prediction. Existing network embedding models comprehensively integrate all information of each node, such as links and attributes, towards a single embedding vector to represent the node's general role in the network. However, a real-world entity could be *multifaceted*, where it connects to different neighborhoods due to different motives or self-characteristics that are not necessarily correlated. For example, in a movie recommender system, a user may love comedies or horror movies simultaneously, but it is not likely that these two types of movies are mutually close in the embedding space, nor the user embedding vector could be sufficiently close to them at the same time. In this paper, we propose a *polysemous embedding* approach for modeling multiple facets of nodes, as motivated by the phenomenon of word polysemy in language modeling. Each facet of a node is mapped to an embedding vector, while we also maintain an association degree between each pair of node and facet. The proposed method is adaptive to various existing embedding models, without significantly complicating the optimization process. We also discuss how to engage embedding vectors of different facets for inference tasks including classification and link prediction. Experiments on real-world datasets help comprehensively evaluate the performance of the proposed method.

KEYWORDS

Network Embedding; Recommender Systems; Disentangled Representation Learning; Graph Mining

ACM Reference Format:
Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, Xia Hu. 2019. Is a Single Vector Enough? Exploring Node Polysemy for Network Embedding. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.1145/3292500.3330967

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '19, August 4–8, 2019, Anchorage, AK, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330967

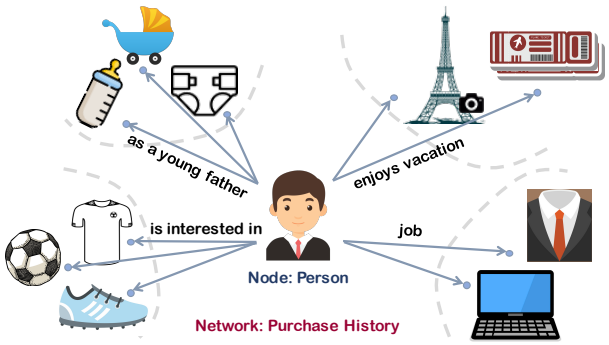

Figure 1: An example of multiple facets within a node. The network is built from online purchase history, where customers and items are nodes, and purchase actions are links.

1 INTRODUCTION

Networks are ubiquitous data structures utilized for modeling information systems [1], such as social networks, recommender systems, biological networks and knowledge graphs. In these systems, real-world entities such as users, items, molecules and knowledge concepts are abstracted as nodes in networks, while the relations between entities are modeled as links between them. The recent advances in network embedding propose to represent each node as a low-dimensional vector, by considering the node's neighborhood and feature information [5, 14, 26, 30–32]. Similar nodes are mapped close to each other in the embedding space. Node embedding has been shown to be an effective representation scheme that facilitates downstream network analysis tasks such as classification, clustering and link prediction [1, 7].

It has been shown that, in many real-world applications, entities may possess disparate characteristics or aspects [25, 28]. In other words, a node in the network can be seen as a capsule containing different aspects. Different links stretching out from an entity to their neighbors could in fact result from the expression of its different aspects. In some scenarios, it could be problematic to fuse these disparate aspects into a single vector-space representation for a node. For example, in an online shopping website in Figure 1 where customers and items are the nodes, a customer may have bought items of disparate genres. If we represent each customer with only a single vector, then the embedding vectors of the customer and items have to be simultaneously close to each other. It could be hard to achieve this, because other customers have different interests and it may mess up the distribution of embedding vectors.

In this paper, we call such a phenomenon as *node polysemy*, where each node could have multiple facets, by making an analogy to a similar property possessed by words in natural language (e.g.,

"bank" can refer to either financial institutions or land near a river, depending on different contexts) [10, 28]. In this setting, each node has multiple facets, while each facet of a node owns an embedding vector. In this work, we want to develop a polysemous network embedding method in order to discover multiple facets of nodes and learn representations for them.

The challenges of developing polysemous network embedding models are three-fold. The first is how to determine the facets of nodes, as well as to flexibly update embeddings of different facets in the vector space. For each data sample (e.g., links or random walks), we need to determine which facet of each node is likely to be activated, so as to update the corresponding embedding of that facet in the training process. The second challenge is how to maintain the correlation among embedding vectors of different facets. Although we split a node into multiple facets, different facets may not be completely uncorrelated to each other. Some information will be lost if we simply model each facet independently. Third, when considering node polysemy, how to make the modeling process adaptive to the existing well-established base models such as Deepwalk [26], LINE [31], PTE [30] and GCN [14]. Besides, the computational complexity of the new model will inevitably increase when different facets of nodes are considered. Therefore, an efficient optimization algorithm is also needed, especially when considering its compatibility to negative sampling.

Specifically, in this paper, we propose a polysemous network embedding method in order to take into account multiple facets of nodes in network data. Each facet of a node is represented using an embedding vector. The correlation between different node facets can be preserved in our method. The developed modeling strategy is flexible to be applied to various base embedding models without making radical changes to their base formulation. We first show how to revise Deepwalk to tackle node polysemy, and then extend our discussion to more base embedding models and more complex application scenarios. The optimization process for training the polysemous embedding models is specially designed to guarantee its efficiency and implementation feasibility, especially when applying negative sampling. Finally, to evaluate whether considering multiple aspects of nodes could actually benefit learning node representations and downstream data mining tasks, we conduct experiments with different tasks to compare the performance between polysemous embedding models and their corresponding single-vector base models. The contribution of this paper is summarized as follows:

- We propose a novel polysemous network embedding method to incorporate different facets of nodes into representation learning. Each aspect of a node owns an embedding vector, and the relations among embedding vectors of different aspects are also considered.
- We specifically formulate the problem to enable the resultant optimization algorithm to be efficient and feasible for implementation. Also the proposed method is flexible to be adapted to various existing transductive embedding models.
- We conduct intensive experiments considering different downstream tasks and application scenarios, providing insights on how and when we benefit from modeling node polysemy in the network embedding.

2 POLYSEMOUS NETWORK EMBEDDING

In this section, we introduce the core idea of polysemous network embedding, by using Deepwalk as the base model. Then, we design an optimization algorithm to train the polysemous embedding model. We will also discuss how to estimate the facet of a node in different contexts. Finally, we introduce how to combine embedding vectors of different facets towards downstream tasks such as classification and link prediction.

2.1 Polysemous Deepwalk

In the setting of polysemous embedding, for the Deepwalk model, each node v_i is associated with a target embedding matrix $\mathbf{U}_i \in \mathbb{R}^{K_i \times D}$ and a context embedding matrix $\mathbf{H}_i \in \mathbb{R}^{K_i \times D}$. Here K_i is the number of embedding vectors possessed by node v_i considering its different possible facets. The traditional Deepwalk model could be seen as a special case where $K_i = 1$. The embedding vector for facet k of node v_i is denoted as $\mathbf{U}_i^k \in \mathbb{R}^D$ or $\mathbf{H}_i^k \in \mathbb{R}^D$, and D is the dimension of each embedding vector. Different nodes may be associated with different number of embedding vectors, depending on the diversity of their characteristics in the network. In this work, for illustration purposes, we simply let all nodes have the same number of embedding vectors, so that $K_i = K$ and K is a predefined constant integer. In practice, the value of K could be estimated from data. For examples, K can be approximately set as the number of latent interest categories in recommender systems, or be estimated as the number of major topics in an academic network. We will discuss in later sections about how to assign facets to nodes with different probabilities.

Deepwalk utilizes the skip-gram model [22] that is trained using the maximum likelihood estimation, where we try to find the model parameters that maximize the likelihood of obtained *observations*. Specifically, let θ be the parameters to be optimized and O be the set of all observations, the objective function to be maximized is:

$$\begin{aligned}\mathcal{L}_{DW}(\theta) &= \sum_{o \in O} \log\, p(o|\theta) = \sum_{o \in O} \log\, p\Big((\mathcal{N}(v_i), v_i)|\theta\Big) \\ &= \sum_{o \in O} \sum_{v_j \in \mathcal{N}(v_i)} \log\, p(v_j|v_i),\end{aligned} \tag{1}$$

where each observation $o \in O$ is defined as a tuple, $o = (\mathcal{N}(v_i), v_i)$, consisting of a central node v_i and its context $\mathcal{N}(v_i)$. Each node within the context is denoted as v_j so that $v_j \in \mathcal{N}(v_i)$. The model parameters, i.e., the embedding vectors of nodes, are used in computing $p(v_j|v_i)$ which is the conditional probability of having v_j as one of the contexts given v_i.

However, in our settings where each node owns multiple facets, the activated facet of a node varies under different contexts. Also, the facet of a given context is determined by the combination of all facets of the nodes within the context. Suppose the distribution of node facets are known in advance, and we treat them as prior knowledge denoted as $\mathcal{P}$. Taking the additional information into account, the objective is reformulated as:

$$\begin{aligned}\mathcal{L}_{PolyDW}(\theta) &= \sum_{o \in O} \log\, p(o|\mathcal{P}, \theta) \\ &= \sum_{o \in O} \log\, [\sum_{s(o)} p(o|s(o), \mathcal{P}, \theta) \cdot p(s(o)|\mathcal{P}, \theta)].\end{aligned} \tag{2}$$

Figure 2: The overall training procedure of Polysemous Deepwalk. Each color refers to one facet contained in the input network. Each histogram visualizes a probability distribution. Note that the above network is only a toy example for illustration purposes, where numerical values do not reflect real results.

Here $s(o)$ is defined as one case of activated facets of all nodes within o, so $s(o) = \{s(v|o) \mid v \in v_i \cup \mathcal{N}(v_i)\}$ where $s(v|o)$ is the activated facet of node v in the context of o. In a given observation o, suppose the activated facet of v_i is k_i and the facet of each $v_j \in \mathcal{N}(v_i)$ is k_j, the conditional probability $p(o|s(o), \mathcal{P}, \theta)$ is thus defined as:

$$p(o|s(o), \mathcal{P}, \theta) = \prod_{v_j \in \mathcal{N}(v_i)} p(v_j|v_i, s(o)), \tag{3}$$

and each product factor is calculated as

$$p(v_j|v_i, s(o)) = \frac{\exp(< \mathbf{H}_j^{k_j}, \mathbf{U}_i^{k_i} >)}{\sum_{v,k} \exp(< \mathbf{H}_v^{k}, \mathbf{U}_i^{k_i} >)}, \tag{4}$$

which is similar to the softmax function in traditional skip-gram models, except that general node embeddings are replaced by node facet embeddings. Here "$< \cdot, \cdot >$" denotes the inner product of two vectors. The denominator acts as normalization over all possible nodes and facets. For readability, we omit $\mathcal{P}$ after the expansion in Equation 3, as it is no longer applied in later steps.

It is cumbersome to directly apply gradient descent to optimize the objective function in Equation 2 ~ Equation 4, due to the summation term inside the logarithm function. Also, it becomes unclear of how to incorporate negative sampling [22] to approximate the normalization term for more efficient computation. Therefore, we further derive the objective function as below:

$$\begin{aligned}
\mathcal{L}_{PolyDW}(\theta) &= \sum_{o \in O} \log \Big[\sum_{s(o)} p(o|s(o), \mathcal{P}, \theta) \cdot p(s(o)|\mathcal{P}, \theta) \Big] \\
&\geq \sum_{o \in O} \sum_{s(o)} p(s(o)|\mathcal{P}, \theta) \cdot \log\ p(o|s(o), \mathcal{P}, \theta) \\
&= \sum_{o \in O} \sum_{s(o)} p(s(o)|\mathcal{P}) \cdot \Big[\sum_{v_j \in \mathcal{N}(v_i)} \log\ p(v_j|v_i, s(o)) \Big] \\
&= \mathcal{L}^*_{PolyDW}(\theta)
\end{aligned} \tag{5}$$

Algorithm 1: Polysemous Deepwalk

Input: Input network $\mathcal{G}$, number of nodes N, facet sampling rate R.
Output: Embedding matrix $\mathbf{U}$, context embedding matrix $\mathbf{H}$.

```
Initialize U and H ;
Estimate facet distribution p(v_i) for each node, so that P = {p(v_i)},
  1 ≤ i ≤ N ;
Obtain observations O via random walks and context window sliding ;
while not converged do
    for o ∈ O do
        Obtain a target-context tuple o = (N(v_i), v_i) ;
        for 1 ≤ r ≤ R do
            Sample a facet k_i = s(v_i|o) ~ p(v_i|o), defined in Eq. 7 ;
            Sample a facet k_j = s(v_j|o) ~ p(v_j|o), defined in Eq. 7,
              for each context node v_j ∈ N(v_i) ;
            Obtain negative samples ;
            Update U_i^{k_i}, H_j^{k_j} for v_j ∈ N(v_i), as well as facet
              embeddings of negative samples, using stochastic
              gradient descent ;
```

The intuition behind the above transformation is that, instead of maximizing the original objective function, we turn to maximize its lower bound [13], denoted as $\mathcal{L}^*_{PolyDW}(\theta)$, by applying Jensen's inequality. The final form of the objective function is similar to that of the skip-gram model, except the external summation over $s(o)$. As a result, we could adopt the same negative sampling strategy as in the traditional skip-gram model to approximate the normalization term in $p(v_j|v_i, s(o))$. Therefore, one major advantage of the proposed polysemous embedding model is that the training process can be easily implemented through minimal modification to the existing learning frameworks, by adding an additional sampling step of assigning activated facets to nodes in each observation o.

Specifically, the summation over $\sum_{s(o)}$, following the distribution of $p(s(o)|\mathcal{P})$, is implemented through *facet sampling* separately for each node in o. The overall optimization algorithm is specified

in Algorithm 1. After initialization and node-facet distribution estimation (will be introduced later), a number of random walks are sampled just as in traditional Deepwalk (Line 3). Then, for each observation o, several rounds of facet sampling are conducted on each node within o (the loop in Line 7). In each round, each node has one facet activated (Line 8 $\sim$ 10), so that the embedding vector corresponding to that facet will be updated using SGD (Line 11). The major additional computation cost, compared with traditional Deepwalk, comes from the increased size of training data in O by a factor of sampling rate R.

The overall process of polysemous Deepwalk is illustrated in Figure 2, where "Objective Optimization" has been introduced as above, while the other steps related with facet distribution and facet assignment will be discussed in detail in the next subsection.

2.2 Node-Facet Assignment

We now introduce how the prior knowledge $\mathcal{P}$ can be obtained, and how to determine the facet of nodes given a specific observation o. For now, we limit the discussion to undirected homogeneous plain networks, and provide one method to obtain the global distribution of node facets by only leveraging the network adjacency matrix. For networks with attribute information or heterogeneous information networks, we may resort to other strategies and we leave it to future work. Let $\mathbf{A}$ be the symmetric adjacency matrix of the network, we perform community discovery on the network [15][36]:

$$\min_{\mathbf{P}\geq 0} \|\mathbf{A} - \mathbf{P}\cdot\mathbf{P}^T\|_F^2 + \alpha\|\mathbf{P}\|_F^2, \tag{6}$$

where $\mathbf{P}$ can be solved using gradient search algorithms. The probability $p(k|v)$, that node v is assigned with the k-th facet, can be calculated as $p(k|v) = \frac{\mathbf{P}_{v,k}}{\sum_{c=1,\ldots,K}\mathbf{P}_{v,c}}$. We define the facet distribution of a node as $\mathbf{p}(v) = [p(1|v), \ldots, p(K|v)]$. We treat $\mathbf{p}(v_i), 1 \leq i \leq N$ as the prior knowledge $\mathcal{P}$, as it encodes the global understanding of the network states. Applying attribute information could achieve better estimation, but it is beyond the discussion in this work.

After obtaining the prior knowledge $\mathcal{P}$, we are able to estimate the overall facet of each observation o as well as the facet of nodes within o. Given an observation $o = (\mathcal{N}(v_i), v_i)$, a straightforward way to define its facet distribution $\mathbf{p}(o)$ is by averaging the facet distributions of nodes inside the observation, i.e., $\mathbf{p}(o) = \Big(\mathbf{p}(v_i) + \sum_{v_j\in\mathcal{N}(v_i)}\mathbf{p}(v_j)\Big)/(|\mathcal{N}(v_i)|+1)$. Considering that the activated facet of a node depends on the specific context where it is observed, given o, the node's facet $s(v|o)$ is sampled according to the distribution $\mathbf{p}(v|o)$, which is defined heuristically as below

$$\mathbf{p}(v|o) = \min(\mathbf{p}(v), \mathbf{p}(o)), \tag{7}$$

where we include a $\min(\cdot,\cdot)$ operator because it is undesirable to assign a node v_i with facet k if $p(k|v_i) \approx 0$, even when the k-th entry in $\mathbf{p}(o)$ is large. To make it a valid probability distribution, $\mathbf{p}(v|o)$ is further normalized to sum to 1.

Till now, we have introduced the whole training process of polysemous Deepwalk as shown in Figure 2. Given the input network, we first estimate node facet distributions as prior knowledge $\mathcal{P}$. Then, random walks are performed to construct node-context observations O. After that, within each walk sample o, each node is assigned with an activated facet. Finally, node embeddings of correspondent facets are updated through optimization.

2.3 Joint Engagement of Multiple Embeddings for Inference

We are able to obtain multiple embedding vectors for each node after training the polysemous model. Then the question arises that, during inference, how to collectively consider different embedding vectors for subsequent tasks. Here we discuss two major network analysis tasks including classification and link prediction.

For classification, for each node, our strategy is to combine multiple vectors $\{\mathbf{U}_i^k\}_{k=1,\ldots,K}$ into a joint vector $\tilde{\mathbf{U}}_i$. Specifically, some options include:

$$\tilde{\mathbf{U}}_i = \mathbf{U}_i^1 \oplus \mathbf{U}_i^2 \oplus \ldots \oplus \mathbf{U}_i^K, \tag{8}$$

where we directly concatenate embeddings of different facets, or

$$\tilde{\mathbf{U}}_i = \Big(p(1|v_i)\cdot\mathbf{U}_i^1\Big) \oplus \Big(p(2|v_i)\cdot\mathbf{U}_i^2\Big) \oplus \ldots \oplus \Big(p(K|v_i)\cdot\mathbf{U}_i^K\Big), \tag{9}$$

where we first scale each embedding vector with the probability of belonging to the corresponding facet and then concatenate these scaled vectors. Here $\oplus$ denotes the concatenation operation. The resultant embedding vectors $\{\tilde{\mathbf{U}}_i\}_{i=1,\ldots,N}$ can be directly used in node classification. We adopt Equation 9 in experiments.

For link prediction or network reconstruction tasks, a higher similarity score between the representations of two nodes indicates greater possibility that a link exist between the nodes. We can define the similarity between two nodes v_i and v_j as:

$$similarity(v_i, v_j) = \sum_{k=1}^{K}\sum_{k'=1}^{K} p(k|v_i)\,p(k'|v_j) < \mathbf{U}_i^k, \mathbf{U}_j^{k'} >, \tag{10}$$

where different facet pairs of embedding vectors contribute to the overall similarity computation, weighted by the probability that the node belongs to the corresponding facet.

2.4 Discussion

The proposed work may be related to disentangled representation learning [9], since different representation dimensions are sensitive to varying factors behind the data. Moreover, it helps improving the interpretability of representation learning [3], because representation dimensions are separated according to node facets that could be associated with concrete meanings or characteristics of real-world networked objects.

3 MODELS EXTENDED BY POLYSEMOUS EMBEDDING

The methodology of polysemous embedding can also be applied to extend other fundamental single-embedding models. In this section, we elaborate two scenarios as examples. First, we show how polysemous embedding can be used in heterogeneous networks where links exist between nodes of different types. Second, we show how polysemous embedding can be combined with graph neural networks, where the embedding look-up tables in Deepwalk is replaced by feedforward modules.

3.1 Polysemous PTE for Heterogeneous Networks

We show how to consider node polysemy when modeling heterogeneous networks. We choose PTE [30] as the base model to be

extended, as it is a fundamental model for tackling the problem. To simplify the illustration, we only consider bipartite networks during model development. The discussion can be applied for more complex networks, since the objective function of PTE could be extended as the sum of several terms considering multiple bipartite or homogeneous networks.

Given a network containing two types of nodes $\mathcal{V}_A$ and $\mathcal{V}_B$, as well as a set of links denoted as $\mathcal{E}$, each observation o is defined as a link $o = (u_j, v_i) \in \mathcal{E}$ where $v_i \in \mathcal{V}_A, u_j \in \mathcal{V}_B$. The set of all observations is thus defined as $\mathcal{O} = \mathcal{E}$. Similar to the derivation in Section 2.1, the objective function is formulated as:

$$\begin{aligned}\mathcal{L}_{polyPTE}(\theta) &= \sum_{o\in\mathcal{E}} \log\,[\sum_{s(o)} p(o|s(o),\mathcal{P},\theta))\cdot p(s(o)|\mathcal{P},\theta)] \\ &\geq \sum_{o\in\mathcal{E}}\sum_{s(o)} p(s(o)|\mathcal{P},\theta)\cdot \log\ p(o|s(o),\mathcal{P},\theta) \\ &= \sum_{o\in\mathcal{E}}\sum_{s(o)} p(s(o)|\mathcal{P})\cdot \log\ p(u_j|v_i,s(o)),\end{aligned} \tag{11}$$

where

$$p(u_j|v_i,s(o)) = \frac{\exp(<\mathbf{H}_j^{k_j}, \mathbf{U}_i^{k_i}>)}{\sum_{v,k}\exp(<\mathbf{H}_v^{k}, \mathbf{U}_i^{k_i}>)}. \tag{12}$$

According to the original PTE model, the matrix $\mathbf{U}$ contains the embedding vectors of nodes in $\mathcal{V}_A$, and $\mathbf{H}$ contains the embedding vectors of nodes in $\mathcal{V}_B$. The resultant lower bound objective is denoted as $\mathcal{L}^*_{polyPTE}$, which is to be optimized.

3.1.1 Node Facet Assignment. The strategy of determining heterogeneous node facet distribution is similar to that of the previous section. Given the asymmetric adjacency matrix $\mathbf{A}$, we first solve

$$\min_{\mathbf{P}\geq 0,\mathbf{Q}\geq 0} \|\mathbf{A} - \mathbf{P}\cdot\mathbf{Q}^T\|_F^2 + \alpha(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2), \tag{13}$$

where we set $\alpha = 0.05$ in experiments. The resultant factor matrix $\mathbf{P}$ contains the association intensity between $\mathcal{V}_A$ and facets, while $\mathbf{Q}$ contains the facet association information for nodes in $\mathcal{V}_B$. Then, we normalize $\mathbf{P}$ and $\mathbf{Q}$ to obtain the probabilities of belonging to different facets for each node $v_i \in \mathcal{V}_A$ and $u_j \in \mathcal{V}_B$ [36]. If we denote the facet distribution as $\mathbf{p}(v_i)$ and $\mathbf{p}(u_j)$ respectively for the two types of nodes, then the facet distribution of an observation (i.e., an edge) is computed as $\mathbf{p}(o) = \big(\mathbf{p}(v_i) + \mathbf{p}(u_j)\big)/2$. Different from polysemous Deepwalk, here we simply let $\mathbf{p}(v|o) = \mathbf{p}(o)$ be applied for node-facet sampling, because the window size in PTE equals to 1 which is much smaller than Deepwalk.

3.1.2 Engage Multiple Embeddings for Inference. The way of jointly considering a node's multiple embedding vectors, for downstream tasks, is similar to what is introduced in Section 2.3. The only difference is that the measurement of similarity between two different types of nodes is adjusted as:

$$similarity(v_i,u_j) = \sum_{k=1}^{K}\sum_{k'=1}^{K} p(k|v_i)\,p(k'|u_j) < \mathbf{U}_i^k, \mathbf{H}_j^{k'} > . \tag{14}$$

3.2 Polysemous Embedding with GCN

The idea of polysemous embedding can be realized with models of other architectures. In this subsection, we consider GCN [14] as an example and show how it could be extended as PolyGCN to consider node polysemy. Different from PolyDeepwalk and PolyPTE that keep embedding look-up tables to store embedding vectors, PolyGCN uses a feedforward network module to generate embedding of each node by gathering information from neighborhoods.

The core step of embedding generation in GCN is to aggregate information from a node's local neighborhood. Let $\mathbf{u}_d(i)$ denote the intermediate representation of node v_i on layer of depth d, then the forward propagation is

$$\mathbf{u}_d(i) \leftarrow \sigma\Big(\mathbf{W}_d\cdot \text{MEAN}(\mathbf{u}_{d-1}(i)\cup\{\mathbf{u}_{d-1}(j), \forall v_j\in\mathcal{N}(v_i)\})\Big), \tag{15}$$

where $\mathbf{W}_d$ is the weight matrix on layer d, MEAN refers to the aggregator based on the pre-processed adjacency matrix, and $\mathcal{N}(v_i)$ is the local neighborhood of v_i. The final embedding output for v_i is defined as $\mathbf{U}_i = \mathbf{u}_{d_{max}}(i)$, where d_{max} denotes the depth of the final layer. After forward propagation, GCN can be trained in an unsupervised manner similar to Deepwalk or PTE.

We here propose a feasible approach to incorporate node polysemy into GCN-based models, where each facet of embeddings corresponds to one GCN model, while the internal architecture of each GCN is unchanged. Specifically, within a certain facet k, we limit the local neighborhood of the node as other nodes with the same facet k activated, so that

$$\mathbf{u}_d^k(i) \leftarrow \sigma\Big(\mathbf{W}_d^k\cdot \text{MEAN}(\mathbf{u}_{d-1}^k(i)\cup\{\mathbf{u}_{d-1}^k(j), \forall v_j\in\mathcal{N}^k(v_i)\})\Big), \tag{16}$$

where $\mathcal{N}^k(v_i)$ denotes the local neighborhood under facet k. The final embedding output for facet k of v_i is $\mathbf{U}_i^k = \mathbf{u}_{d_{max}}^k(i)$. The main question to be answered is how to construct the local neighborhood for a specific facet k of node v. Here we still consider bipartite networks as the scenario, so for each facet there are two GCNs respectively for generating embeddings for each type of nodes. Following the similar strategy as in Section 3.1, the observed links are decomposed into interaction results of different facet pairs. We further relax the problem so that embeddings of different facets are mutually independent. Specifically, the original observation matrix $\mathbf{A} = \sum_k \mathbf{A}^k$, where $\mathbf{A}^k \geq 0$ and $\mathbf{A}^k(i,j)$ is proportional to $\mathbf{P}(i,k)\cdot\mathbf{Q}(j,k)$. $v_j \in \mathcal{N}^k(v_i)$ if both v_i and v_j connect to the same node of the other type under facet k. For training each GCN pair, we adopt the similar unsupervised training strategies as proposed in [6], where nodes are encouraged to have similar embeddings of facet k if they are connected according to the corresponding observation matrix.

4 EXPERIMENTS

We try to answer several questions through experiments. First, how effective is the proposed polysemous embedding method compared with single-vector embedding counterparts? Second, is considering node polysemy beneficial for network embedding when evaluated on different downstream tasks? Third, how will polysemous embedding models react to the changes of hyperparameters?

4.1 Experimental Settings

We first introduce the applied datasets as below. The detailed information is shown in Table 1.

- **BlogCatalog**: A social network dataset built based on connections between online bloggers. The original dataset contains both link and attribute information, while we only keep the links in our experiments. Each node is also associated with a label determined from some predefined interest categories of bloggers.
- **Flickr**: A network dataset constructed from interactions between users on a multimedia hosting website. Each link corresponds to a following relation between users. The groups that users joined are regarded as labels.
- **MovieLens**: A movie rating dataset widely used for evaluating collaborative filtering models. In our experiments, we treat it as a heterogeneous network where links exist between users and items (i.e., movies). We transform rating scores into implicit data, so that each entry is either 0 or 1 indicating whether the user rated the movie [8]. In this way, conducting recommendation on this dataset can also be regarded as performing link prediction.
- **Pinterest**: An implicit feedback data originally constructed for image recommendation. Users and images are regarded as nodes. Each link is a pin on an image initiated by a user. Each user has at least 20 links. Similarly, the link prediction task can be seen as recommending images to users.

The networks in BlogCatalog and Flickr are homogeneous, and we use them in classification and link prediction. The networks in MovieLens and Pinterest are heterogeneous, and we will apply them in link prediction tasks which can also be seen as doing recommendation. The baseline methods for comparison are as below.

- **Deepwalk** [26], **PTE** [30], **GCN** [38]: Some commonly used network embedding models that map each node to a single vector. We include them as baseline methods to analyze whether their polysemous embedding counterparts achieve better performance. Here GCN is only applied in heterogeneous link prediction.
- **NMF** [15, 16]: A traditional model that learns latent factors from data. We include NMF as one of the baseline methods because we applied it in estimating the global facets contained in data. The number of latent factors is the same as the number of facets in the proposed polysemous model.
- **MSSG** [23]: A multi-sense word embedding model original developed for natural language processing. Senses of words determined by clustering their average context representations. Each word has only one embedding vector when it is regarded as context. The model is adapted for network analysis tasks.
- **MNE** [37]: A model that factorizes the network proximity matrix into several group of embedding matrices, and adds a diversity constraint to force different matrices focus on different aspects of nodes. We only use MNE for node classification since the way of performing link prediction is not discussed in the paper.

4.2 Node Classification

Node classification is a commonly applied task for evaluating network embedding outcomes. In this experiment, we compare the performance of the proposed polysemous models compared with baseline models. We use Deepwalk as the base single-embedding model. The proposed model is named as PolyDeepwalk. The BlogCatalog and Flickr datasets are applied in this experiment.

	$\lvert\mathcal{V}_A\rvert$	$\lvert\mathcal{V}_B\rvert$	$\lvert\mathcal{E}\rvert$
BlogCatalog	5,196	–	171,743
Flickr	7,575	–	239,738
MovieLens	6,040	3,952	1,000,209
Pinterest	55,187	9,916	1,500,809

Table 1: Statistics of datasets.

Unless specifically stated in each task, the default model hyperparameters are set as follows. For BlogCatalog dataset, the number of walks generated per node is 110, the length of walk is 11, the context size is 8, the number of facets is 6 considered for PolyDeepwalk, the embedding of dimension is 210 for each node in Deepwalk and $210/6 = 35$ for each node's facet in PolyDeepwalk, the number of negative samples is 5 for Deepwalk and 10 for PolyDeepwalk. For Flickr dataset, the parameter settings are similar, except that the number of facets is 5 for PolyDeepwalk, the embedding dimension is 180 for each node in Deepwalk and $180/5 = 36$ for each node's facet in PolyDeepwalk, and the number of negative samples is 15 for PolyDeepwalk. It is worth noting that the dimension of polysemous embedding is divided by the number of facets, so that after concatenation, the resultant embedding has the same length as the single-embedding counterpart. We use Support Vector Machine as the classification model, where 80% of data samples are used as training data. The performance comparison is illustrated in Figure 3 and Figure 4. Note that we do not plot the performance of NMF as its F1 scores are much lower than other models. Some work includes node attributes as additional information source when using NMF related models for node classification [11, 17], but this is beyond our experiment setting. From the figures, some observations are summarized as below:

- In general, PolyDeepwalk achieves better performances than other polysemous embedding models. Also, by varying the dimensionality, we can observe that polysemous embedding models are less likely to be affected, while the classification performance of Deepwalk gradually decreases as dimensionality increases.
- Deepwalk and PolyDeepwalk have different responses to the change of context size. The former achieves better performances as context size increases, while the latter shows the opposite trend. A possible reason is that, as the context window enlarges in PolyDeepwalk, random walks from each node are more likely to reach other network regions of different facets. As a result, the facet distributions are over smoothed, so it becomes more difficult to decide which facet a context window belongs to. It could be one of the challenges to be tackled in future work.

Besides making comparisons to baseline models, we also analyze the sensitivity of PolyDeepwalk to some of the key parameters when modeling node polysemy. The experiment results are shown in Figure 5. Some observations are made as below:

- Increasing the number of facets has positive influence on the classification performance of the proposed model. It is also worth noting that we keep the total embedding dimension (i.e., $K \times D_{\text{PolyDeepwalk}}$) to be fixed, where $D_{\text{PolyDeepwalk}}$ actually decreases as K increases, so that the downstream classification task will not be affected by the feature size.

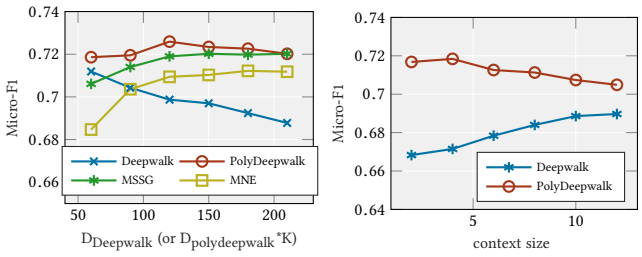

Figure 3: Node classification evaluation on BlogCatalog dataset.

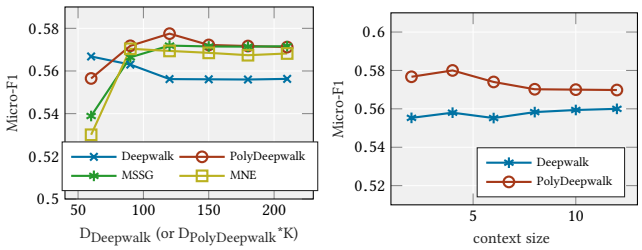

Figure 4: Node classification evaluation on Flickr dataset.

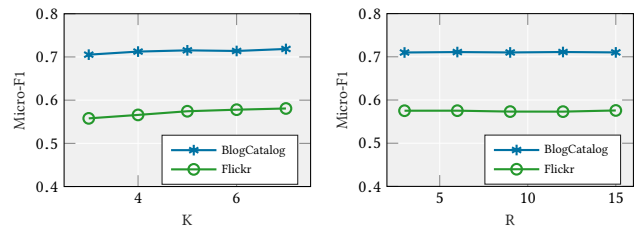

Figure 5: Parameter analysis for node classification.

- In this part of experiment, changing the facet sampling rate for each observation (i.e., context window) does not significantly perturb the results. The possible reason is that many random walks have been generated from each node, so the facet sampling is also implicitly performed for each random walk. The training samples are already adequate even when we reduce the facet sampling rate for each context window.

4.3 Link Prediction in Homogeneous Networks

Besides node classification, we also include link prediction as one of the tasks for evaluating the embedding results. We randomly select one link for each node as test data, and use the remaining network as training data. The default settings of hyperparameters are similar to those applied in the last experiment. The main difference lies on the embedding dimensionality, where we let the embedding dimension to be $D_{\text{Deepwalk}} = D_{\text{PolyDeepwalk}}$. In other words, the embedding space of PolyDeepwalk is the same as that of Deepwalk. The default dimension value, unless otherwise stated, is chosen as 150. The performance of link prediction is summarized in Table 2, from which we could observe that:

- There is no significant advantage for PolyDeepwalk on BlogCatalog dataset compared with Deepwalk. Although PolyDeepwalk achieves better performance in terms of AUC score, Deepwalk is more advantageous when evaluated using *HR@k* and k is small. However, it implicitly indicates that Deepwalk is gradually surpassed by PolyDeepwalk when k increases, which means PolyDeepwalk could be better at recovering links that do not match the major connection patterns of a node.
- PolyDeepwalk performs better than NMF, which indicates that the effectiveness of polysemous embedding is not purely inherited from the matrix factorization step which provides prior knowledge of node facet distribution.

In addition, we also analyze the sensitivity of PolyDeepwalk by varying the values of certain hyperparameters, including dimensionality, the number of facets (K), and the facet sampling rate (R) for each context window. The results are shown in Figure 6. Some phenomenons are observed as below:

- Network embedding models are sensitive to embedding dimension in link prediction tasks, as we can observe a clear growth of AUC score when embedding dimension increases.
- Increasing the number of facets K appropriately could boost link prediction performance. However, increasing K will also augment the embedding storage and inference computation cost. We should better keep an appropriate K value, in order to balance the performance and efficiency.
- Similar to what we have discussed in node classification, increasing the facet sampling rate does not bring much benefits. Therefore, when the number of walks per node is large enough, we can keep the face sampling rate to be low in order to reduce computation costs.

4.4 Link Prediction in Heterogeneous Networks

In some application scenarios such as recommender systems, each link stretches across two types of nodes. Here we construct two bipartite networks from two recommendation datasets: MovieLens and Pinterest. Models such as PTE and unsupervised GCN can handle such scenario, and we extend them as PolyPTE and PolyGCN to be evaluated. For each user node, we hold out its latest interaction as the test set and use the remaining data for training.

The default values of hyperparameters, unless otherwise stated, are as follows. For both datasets, the number of facets K is 5, the embedding dimension is set as 30, and the facet sampling rate R for each observation (i.e., link) equals K^2. The negative sampling rate is set as 30 for PolyPTE. Different from the last experiment where $D_{\text{PolyDeepwalk}} = D_{\text{Deepwalk}}$, the embedding dimension in this experiment is set as D/K. Performances of heterogeneous link prediction are shown in Table 3 under different metrics, where we can observe that:

- Polysemous embedding models achieves better performances than their single-embedding counterparts. It thus suggests that considering node polysemy is beneficial for modeling the association relation between different types of node entities.
- Polysemous embedding models in general perform better than NMF, which demonstrates that the effectiveness of polysemous models comes beyond the prior knowledge obtained from NMF.
- PolyGCN is not as good as PolyPTE in performance boosting. A possible reason is that we do not consider inter-facet relations in

	BlogCatalog					Flickr				
	HR@10	HR@50	HR@100	HR@200	AUC	HR@10	HR@50	HR@100	HR@200	AUC
PolyDeepwalk	0.234	0.493	0.615	0.737	0.957	0.210	0.395	0.487	0.590	0.928
Deepwalk	0.253	0.512	0.639	0.764	0.950	0.195	0.348	0.430	0.530	0.912
NMF	0.041	0.132	0.196	0.280	0.801	0.048	0.148	0.226	0.335	0.852
MSSG	0.102	0.272	0.376	0.505	0.910	0.071	0.152	0.207	0.268	0.811

Table 2: Performance of Homogeneous Link Prediction

	MovieLens					Pinterest				
	HR@10	HR@50	HR@100	HR@200	AUC	HR@10	HR@50	HR@100	HR@200	AUC
PolyPTE	0.067	0.210	0.334	0.491	0.892	0.062	0.204	0.318	0.460	0.919
PTE	0.040	0.143	0.227	0.336	0.832	0.007	0.030	0.052	0.088	0.703
PolyGCN	0.050	0.158	0.250	0.389	0.863	0.036	0.102	0.159	0.236	0.801
GCN	0.039	0.130	0.204	0.313	0.813	0.028	0.059	0.087	0.130	0.721
NMF	0.051	0.165	0.265	0.404	0.865	0.005	0.021	0.040	0.075	0.654
MSSG	0.060	0.181	0.288	0.427	0.868	0.039	0.128	0.199	0.304	0.838

Table 3: Performance of Heterogeneous Link Prediction

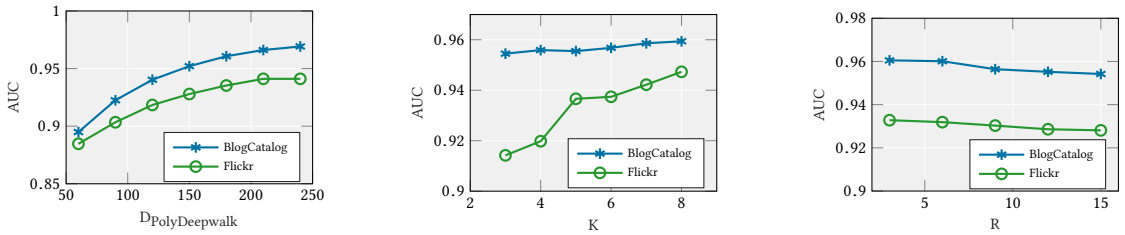

Figure 6: Parameter analysis for link prediction on homogeneous networks.

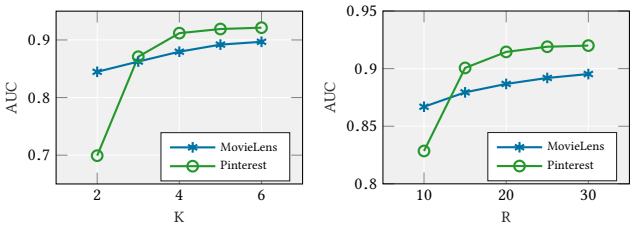

Figure 7: Parameter analysis for PolyPTE.

PolyGCN. However, GCN-based models still have much room for improvement, especially when attribute information is available.

Similar to previous experiment tasks, here we also analyze the sensitivity of PolyPTE to model hyperparameters, as shown in Figure 7 for MovieLens and Pinterest. Besides some similar observations as in the last experiment, PolyPTE is sensitive to the facet sampling rate, especially when its value is low. A possible reason for such contrasting observations between PolyDeepwalk and PolyPTE could be the different forms of observation samples $o \in O$. PolyDeepwalk samples plenty of random walks from each node (i.e., $o = \{\mathcal{N}(v), v\}$), while PolyPTE uses edge based sampling strategy (i.e., $o = (v_i, v_j)$). Therefore, if the facet sampling rate is low, PolyPTE could not fully consider all the possible correlation patterns between different facets of users and items. It will thus negatively affect the embedding performance.

5 RELATED WORK

Network embedding models have received a lot of attention recently due to its effectiveness in learning representations for nodes in network data. Existing work on network embedding can be categorized based on various criteria. Many basic models such as Deepwalk [26], LINE [31], node2vec [5] and SDNE [32] focus on analyzing plain networks, while more recent work starts tackling more complex networks such as considering attributes [11, 18], community structures [33] and node heterogeneity [2, 30]. Many methods solve the embedding problem by resorting to matrix factorization [27], either explicitly [24] or implicitly [26, 31], while recently feed-forward modules such as graph convolutional networks [6, 14] are attracting more and more attention. In addition, some challenges that have been tackled in network embedding include model scalability improvement [38], modeling dynamic networks [40], considering human-in-the-loop scenarios [12]. Finally, network embedding has been applied in applications such as recommender systems [4], fraud detection [35] and behavioral modeling [34].

Some recently proposed models, not limited to network analysis, start to utilize the structural knowledge for representation learning. Ma et al. make use of hierarchical taxonomy to regularize node representation learning [21], which does not solve the same problem as in this paper since each node is still assigned only one embedding vector. Yang et al. extends matrix factorization models to explore multiple facets of homogeneous networks towards embedding [37],

where the method may not be extended to other base models or scenarios. Some other work such as [39] and [20] try to extract hierarchical taxonomy from embedding results, which also do not solve the same problem as in this paper.

The property of multiple facets owned by network nodes can be analogized to a similar phenomenon of word polysemy in natural languages [28]. However, the modeling methods cannot be directly applied to our problem of network analytics due to various incompatibility issues [10, 19], such as the differences in language models and network models, the difference of sample formats, and the absence of the concept of documents in network data. Also, [23] does not consider polysemy in the context vectors or the correlation between different word senses. Some relevant problems have been discussed in Heterogeneous Information Networks [29], but it is hard to adapt the methodology to other scenarios.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose a polysemous embedding method for considering multiple facets of nodes in network embedding. After estimating the degrees of association between each node and its facets, each facet of a node owns a embedding vector. The proposed method is flexible as it can be used to extend various existing embedding models. During the training process, embedding vectors of different facets are updated, based on the observation where the nodes locate. We also introduce how to combine multiple embedding vectors of each node during classification and link prediction. Experiments on real-world datasets comprehensively evaluate when we benefit from considering node polysemy compared with single-vector embedding baseline methods. Some possible future work are as follow: (1) Exploring the possibility of doing node facets association and polysemous embedding simultaneously, (2) Applying polysemous network embedding models to more complicated network systems such as knowledge graphs, (3) Developing optimization algorithms to further accelerate the training procedure.

ACKNOWLEDGMENTS

The work is, in part, supported by NSF (#IIS-1657196, #IIS-1718840, #IIS-1750074). The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[1] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *arXiv preprint arXiv:1711.08752*, 2017.
[2] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*. ACM.
[3] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *arXiv preprint arXiv:1808.00033*, 2018.
[4] Mihajlo Grbovic and Haibin Cheng. Real-time personalization using embeddings for search ranking at airbnb. In *KDD*, 2018.
[5] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*. ACM, 2016.
[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.
[7] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.
[8] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, 2016.
[9] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.
[10] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *ACL*, 2012.
[11] Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *WSDM*. ACM, 2017.
[12] Xiao Huang, Qingquan Song, Jundong Li, and Xia Hu. Exploring expert cognition for attributed network embedding. In *WSDM*, 2018.
[13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
[15] Da Kuang, Chris Ding, and Haesun Park. Symmetric nonnegative matrix factorization for graph clustering. In *SDM*, 2012.
[16] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2001.
[17] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. Attributed network embedding for learning in a dynamic environment. In *CIKM*, 2017.
[18] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. Attributed social network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 2018.
[19] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. Topical word embeddings. In *AAAI*, 2015.
[20] Ninghao Liu, Xiao Huang, Jundong Li, and Xia Hu. On interpretation of network embedding via taxonomy induction. KDD, 2018.
[21] Jianxin Ma, Peng Cui, Xiao Wang, and Wenwu Zhu. Hierarchical taxonomy aware network embedding. In *KDD*, 2018.
[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.
[23] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*, 2014.
[24] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *KDD*, 2016.
[25] Adam Perer, Ido Guy, Erel Uziel, Inbal Ronen, and Michal Jacovi. Visual social network analytics for relationship discovery in the enterprise. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, 2011.
[26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*. ACM, 2014.
[27] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*. ACM, 2018.
[28] Joseph Reisinger and Raymond J Mooney. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010.
[29] Yu Shi, Qi Zhu, Fang Guo, Chao Zhang, and Jiawei Han. Easing embedding learning by comprehensive transcription of heterogeneous information networks. In *KDD*, 2018.
[30] Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*. ACM, 2015.
[31] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *WWW*, 2015.
[32] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *KDD*. ACM, 2016.
[33] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *AAAI*, 2017.
[34] Daheng Wang, Meng Jiang, Qingkai Zeng, Zachary Eberhart, and Nitesh V Chawla. Multi-type itemset embedding for learning behavior success. In *KDD*, 2018.
[35] Haibo Wang, Chuan Zhou, Jia Wu, Weizhen Dang, Xingquan Zhu, and Jilong Wang. Deep structure learning for fraud detection. In *KDD*, 2018.
[36] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *SIGIR*, 2003.
[37] Liang Yang, Yuanfang Guo, and Xiaochun Cao. Multi-facet network embedding: Beyond the general solution of detection and representation. In *AAAI*, 2018.
[38] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, 2018.
[39] Chao Zhang, Fangbo Tao, Xiusi Chen, Jiaming Shen, Meng Jiang, Brian Sadler, Michelle Vanni, and Jiawei Han. Taxogen: Unsupervised topic taxonomy construction by adaptive term embedding and clustering. In *KDD*, 2018.
[40] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. 2018.