Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Scalable Multiplex Network Embedding

Hongming Zhang[1], **Liwei Qiu**[2], **Lingling Yi**[2], **Yangqiu Song**[1]
[1]Department of CSE, The Hong Kong University of Science and Technology
[2]Tencent Technology (SZ) Co., Ltd., China
hzhangal@cse.ust.hk, deolcaqiu@tencent.com, chrisyi@tencent.com, yqsong@cse.ust.hk

Abstract

Network embedding has been proven to be helpful for many real-world problems. In this paper, we present a scalable multiplex network embedding model to represent information of multi-type relations into a unified embedding space. To combine information of different types of relations while maintaining their distinctive properties, for each node, we propose one high-dimensional common embedding and a lower-dimensional additional embedding for each type of relation. Then multiple relations can be learned jointly based on a unified network embedding model. We conduct experiments on two tasks: link prediction and node classification using six different multiplex networks. On both tasks, our model achieved better or comparable performance compared to current state-of-the-art models with less memory use.

1 Introduction

Network or graph embedding, which uses dense vectors to represent nodes, has been well studied for many years [Archdeacon, 1996; Chung, 1997; Goyal and Ferrara, 2017]. Recently, inspired by recurrent neural networks [Goodfellow *et al.*, 2016], network embedding has been re-investigated and developed based on random walks over graphs and stochastic gradient descent optimization, which can be applied to very large graphs, such as real social networks [Perozzi *et al.*, 2014; Tang *et al.*, 2015; Grover and Leskovec, 2016]. The embedding vector of nodes can be used to encode some of the topological structure of the network, and then can be used as features for downstream models. Network embedding has been proven to be helpful for network analysis tasks including link prediction, node classification, and community detection.

Multiplexity property is a common feature of networks, especially for social networks. In the social network analysis, multiplexity refers to multifaceted relationships between two people [Verbrugge, 1979]. If we generalize this idea to all kinds of network, by "multiplex network," we mean a group of networks which contains multiple kinds of relations, and each kind of the relations can create a layer of the network. Take the social network as an example. In a social network such as Facebook, users often have different kinds of interactions with each other like friendship relation, forwarding articles to each other, conversation, money transferring, etc. Each of them will create a layer of the network among all users. If we consider them as a united one, we will get a huge multiplex network. While each layer of the network can only represent one kind of interactions among users, to better understand the whole multiplex network, it is better to integrate different types of information from these networks together without sacrificing their distinctive properties.

Considering the fact that a network could be huge, in this paper, we present a scalable multiplex network embedding model to efficiently store and learn information of multi-type relations into a unified embedding space. For each node, we propose one high-dimensional common embedding vector, which is shared across all layers of the multiplex network, and we use that to build a bridge among different layers of the network. For each node, to learn its distinct property on different layers with small memory occupation, we also propose a low-dimensional additional vector for each type of relation. To align these embeddings with different dimensions together, we introduce a global transformation matrix for each layer of the network. Following DeepWalk [Perozzi *et al.*, 2014], we use stochastic gradient descent to optimize our model. Since the global transformation matrix will be updated much more times than node vectors will be, we add an additional regularization to constrain the size of the matrix.

Our contributions can be summarized as follows.

- We formally define the problem of multiplex network embedding to handle the multiplexity property of networks.
- We propose a scalable multiplex network embedding model to represent information from multi-type relations into a unified embedding space.
- We evaluate our embedding algorithm on two network analysis tasks: link prediction and node classification. On both tasks, our model achieved better or comparable performance compared to current state-of-the-art models with less memory use.

The source code of our model is available at: https://github.com/HKUST-KnowComp/MNE.

2 Related Work

2.1 Multiplex Network Analysis

Traditionally in social science, multiplexity has been used to characterize the multiple facets of social exchange relationships among users [Verbrugge, 1979]. The idea of multiplexity can be generalized to all kinds of network. In data mining community, people sometimes also use the term "multi-relational network" to represent the multi-type relations in social networks and verified that considering such relations in the social network can help data mining tasks such as community mining or link prediction [Cai *et al.*, 2005; Chen *et al.*, 2016]. In addition, in bioinformatics community, it has been shown that by integrating multiple networks, the node representation can be improved for genes functional analysis [Cho *et al.*, 2016].

Nowadays, graph mining methods on the multiplex network are mainly targeting on specific tasks. For link prediction, traditional methods [Newman, 2001; Adamic and Adar, 2003] can be applied to multiplex network without considering the information of the typed edges. For community detection, cross-layer centrality measurement [Bródka *et al.*, 2012] is proposed to capture the centrality of nodes in a multiplex network. Besides that, multilayered local clustering coefficient (MLCC) and cross-layer clustering coefficient (CLCC) [Kazienko *et al.*, 2010; Lytras *et al.*, 2010] are also proposed to describe cluster coefficient of a node in a multiplex network.

In this paper, we propose a general solution for multiplex network analysis from the perspective of network embedding.

2.2 Network Embedding

Network embedding (or sometimes called graph embedding, historically also graph drawing) has been well studied for many years [Archdeacon, 1996; Chung, 1997; Goyal and Ferrara, 2017]. It is also related to manifold learning [Huo *et al.*, 2007] while manifold learning was usually applied to dimensionality reduction for high-dimensional data. Network embedding focuses on generating the vector representation of nodes for real networks or graphs to facilitate further analysis of networks. Traditional approaches to network embedding usually involve time-consuming computational components, such as eigenvalue decomposition to analyze the spectral property of graphs [Spielman, 2011]. However, given modern very large-scale online social networks, such approaches may be less efficient and sometimes infeasible to generate the node representations.

Recently, inspired by the development of recurrent neural networks [Goodfellow *et al.*, 2016], particularly the efficient word embedding method, word2vec [Mikolov *et al.*, 2013b; Mikolov *et al.*, 2013a], many network embedding approaches have been proposed to handle large-scale social networks. For example, DeepWalk [Perozzi *et al.*, 2014] proposes to perform random walk on the network to generate sequences of nodes and then perform skip-gram algorithm (a technique used in word2vec) on those sequences to achieve the embedding. On top of DeepWalk, Node2Vec [Grover and Leskovec, 2016] adds two parameters to control the random walk process and make it biased random walk. Some other embedding models focus on specific kinds of structures in the network. For example, LINE [Tang *et al.*, 2015] tries to use the embedding to approximate the first-order and second-order proximities of the network.

All the models above have been proven useful on single network analysis, but they did not consider multiplexity. Recently, an embedding approach is proposed to target the multiplexity property [Liu *et al.*, 2017]. They proposed three methods to learn one overall embedding from the multiplex network. Unlike their approach, we propose one high-dimensional common embedding and several lower-dimensional additional embeddings for each type of relations. A recent work [Li *et al.*, 2018] called itself multi-layer network embedding. Essentially, it is a heterogeneous information network since in the definition in their paper, different layers have different type of nodes.

3 Multiplex Network Embedding

3.1 Problem Definition

We first introduce our problem and notations. Given a multiplex network, our goal is to learn a representation for each node. Suppose $\mathcal{N}$ is the set of nodes and there are M different relation types, represented as $\mathcal{G}_1, ..., \mathcal{G}_M$. For each relation based network $\mathcal{G}_i = (\mathcal{N}_i, \mathcal{E}_i)$, we have $\mathcal{E}_i \subseteq \mathcal{N}_i \times \mathcal{N}_i$ and $\mathcal{N}_i \subseteq \mathcal{N}$.

For each node $n \in \mathcal{N}$, it has one common embedding vector, which is shared across all the relation types and denoted as $\mathbf{b}_n \in \mathbb{R}^d$. We use that common embedding as a bridge to transfer information across different relation types. To capture the distinct property of each sub-network, for each node, we also propose a series of low-dimensional vectors $\mathbf{u}_n^i \in \mathbb{R}^s$ for each relation type i. Here, d is the dimension of common embedding and s is the dimension of typed relation vector. To avoid our model becoming too large, we set $s \ll d \ll |\mathcal{N}|$, where $|\mathcal{N}|$ is the number of all nodes. To align $\mathbf{b}_n$ and $\mathbf{u}_n^i$, we introduce a transformation matrix $\mathbf{X}^i \in \mathbb{R}^{s \times d}$. Then we use a new vector $\mathbf{v}_n^i$ to represent the embedding vector of n for relation type i:

$$\mathbf{v}_n^i = \mathbf{b}_n + w^i \cdot \mathbf{X}^{iT} \mathbf{u}_n^i, \tag{1}$$

where we use a weight w_i to denote the overall importance of relation type i.

3.2 Optimization

For each relation type i of the multiplex network, we conduct random walk to generate a sequence of nodes and then perform Skip-gram algorithm used in word2vec [Mikolov *et al.*, 2013a] over the sequences to learn the embeddings. As shown in Eq. (1), a node's final embedding is composed of three parts: common embedding $\mathbf{b}_n$, relation-based embedding $\mathbf{u}_n^i$, and transformation matrix $\mathbf{X}^i$. We need to learn these parameters at the same time.

For any node n in a random walk path relation type i, assuming it appears in position j, we will take $n_{j-c}, ..., n_{j+c}$ as its neighbors, where c is half of the window size. Thus, given a sequence of nodes, our objective is to minimize the following negative log-likelihood:

$$-\log P_{\theta^i}(n_{j-c}, ..., n_{j-1}, n_{j+1}, ..., n_{j+c} \mid n_j), \tag{2}$$

which can be further factorized as $-\prod_{k=j-c}^{j-1}\log P_{\theta^i}(n_k|n_j)\cdot\prod_{k=j+1}^{j+c}\log P(n_k|n_j)$ given the conditional independence assumption. Here we use P_{θ^i} to denote the parameterization for relation type i. For each $P_{\theta^i}(n_k|n_j)$, a softmax function is used to define the probability:

$$P_{\theta^i}(n_k \mid n_j) = \frac{\exp({\mathbf{v}'_{n_k}}^T \cdot \mathbf{v}^i_{n_j})}{\sum_n \exp({\mathbf{v}'_n}^T \cdot \mathbf{v}^i_{n_j})}, \quad (3)$$

where $\mathbf{v}^i_{n_j}$ represents the input word embedding of user n_j, and $\mathbf{v}'_{n_k}$, and $\mathbf{v}'_n$ represent the parameters of context vectors shared by all relation types, which are only used during the training process. To speed up the training process, following word2vec [Mikolov *et al.*, 2013a], we use negative sampling to approximate the objective function as:

$$E = -\log\sigma({\mathbf{v}'_{n_k}}^T \cdot \mathbf{v}^i_{n_j}) - \sum_{n\in\mathcal{N}^i_{n_j}} \log\sigma(-{\mathbf{v}'_n}^T \cdot \mathbf{v}^i_{n_j}), \quad (4)$$

where $\sigma(x) = 1/(1+\exp(-x))$ is the logistic function and $\mathcal{N}^i_{n_j}$ is the randomly sampled negative context node set for n_j.

Assuming the sequence is generated in relation type i, according to Eq. (1), in the new objective function, we can replace $\mathbf{v}^i_n$ with $\mathbf{b}_n + {\mathbf{X}^i}^T\mathbf{u}^i_n$. After that we employ stochastic gradient descent (SGD) to minimize the objective function. We update the context parameter vectors, node embedding vectors, and transformation matrix as follows:

$$\mathbf{b}_{n_j} := \mathbf{b}_{n_j} - \eta\cdot\frac{\partial E}{\partial \mathbf{b}_{n_j}} = \mathbf{b}_{n_j} - \eta\cdot e(n_j,n_k)\cdot\mathbf{v}'_{n_k}, \quad (5)$$

$$\mathbf{v}'_{n_k} := \mathbf{v}'_{n_k} - \eta\cdot\frac{\partial E}{\partial \mathbf{v}'_{n_k}} = \mathbf{v}'_{n_k} - \eta\cdot e(n_j,n_k)\cdot\mathbf{v}^i_{n_j}, \quad (6)$$

$$\mathbf{u}^i_{n_j} := \mathbf{u}^i_{n_j} - \eta\cdot\frac{\partial E}{\partial \mathbf{u}^i_{n_j}} = \mathbf{u}^i_{n_j} - \eta\cdot e(n_j,n_k)\cdot\mathbf{X}^i\mathbf{v}'_{n_k}, \quad (7)$$

$$\mathbf{X}^i := \mathbf{X}^i - \eta\cdot\frac{\partial E}{\partial \mathbf{X}^i} = \mathbf{X}^i - \eta\cdot e(n_j,n_k)\cdot\mathbf{u}^i_{n_j}{\mathbf{v}'_{n_k}}^T, \quad (8)$$

where $e(n_j,n_k) = \sigma({\mathbf{v}'_{n_k}}^T\cdot\mathbf{v}^i_{n_j}) - t_k$ and t_k=1 if n_k is the neighbor of n_j and t_k=0 if n_k is in the negative sampling set.

We initialize all the $\mathbf{v}^i$ with zero vector and $\mathbf{X}^i$ with zero matrix. For context parameter vector $\mathbf{v}'_n$ for all the users, we randomly initialize them. Since $\mathbf{X}^i$'s can be updated many times when optimizing the objective function, to avoid its norm becoming too huge, we add the following constraints:

$$||\mathbf{X}^i|| \le r, \quad (9)$$

where r is a controlling parameter for the Frobenius norm of matrix $||\mathbf{X}^i||$. In our experiment, we also involve an extra combination layer and since this layer has no actual meaning, we will only update common embedding during the training on that layer.

We summarize our algorithm in Algorithm 1. As our model is a random walk based solution, assuming that we have M relation types of sub-networks and $|\mathcal{N}|$ nodes, the time complexity of our model is $O(M|\mathcal{N}|)$. The memory complexity of our model is $O((d+s*M)|\mathcal{N}|)$.

Algorithm 1 Multiplex Network Embedding

INPUT: Multiplex network which is composed of sub networks $\mathcal{G}_1,\ldots,\mathcal{G}_M$, common embedding dimension d, additional embedding dimension s, learning rate η, size of context window c, number of negative sampling k, transformation matrix size regularization parameter r.

Initialize $\mathbf{b}_n$, $\mathbf{v}^i_n$ and $\mathbf{X}^i$ for different nodes n's and relations i's.
for Each relation network $\mathcal{G}_i$ **do**
  for Each path P generated by random walk **do**
    for $n_j \in P$ **do**
      for $n_c \in P[j-c, j+c]$ **do**
        $\mathcal{N}_{n_j} = NegativeSampling(\mathcal{N}, m)$
        $\mathbf{v}^i_{n_j} = \mathbf{b}_{n_j} + {\mathbf{X}^i}^T\mathbf{u}^i_{n_j}$
        $\mathcal{N}_{sample} = \mathcal{N}_{n_j} \cup n_c$
        for $n_k \in \mathcal{N}_{sample}$ **do**
          $e(n_j,n_k) = \sigma({\mathbf{v}'_{n_k}}^T\cdot\mathbf{v}^i_{n_j}) - t_k$
          Update $\mathbf{b}_{n_j}$, $\mathbf{v}'_{n_k}$, $\mathbf{u}^i_{n_j}$, and $\mathbf{X}^i$ based on Eqs. (5), 6), (7), and (8) respectively.
        end for
        if $\left\|\mathbf{X}^i\right\| > r$ **then**
          $\mathbf{X}^i := \frac{r}{\|\mathbf{X}^i\|}\cdot\mathbf{X}^i$
        end if
      end for
    end for
  end for
end for

OUTPUT: Common embedding $\mathbf{b}_n$, additional embeddings $\mathbf{u}^i_n$, and transformation matrix $\mathbf{X}^i$ for different nodes n's and relations i's.

4 Experiments

4.1 Datasets

We use open multiplex networks from Manlio De Domenico project[1] as our experimental datasets. We select four multiplex networks from different network types. The selection criteria are that they cannot be too small and they cannot be too sparse, which means that they should have at least one edge per node per relation type. The details of these datasets are as follows:

Vickers [Vickers and Chan, 1981]: Data collected by asking 29 seventh grade students in a school in Victoria, Australia, for whom they asked three questions and each of them creates a relation type of the network.

CKM [Coleman *et al.*, 1957]: Data collected by asking physicians in four towns in Illinois, Bloomington, Quincy, and Galesburg. They asked three questions and each of them creates a relation type of the network.

LAZEGA [Lazega, 2001]: This multiplex social network consists of three kinds of relationships (Co-work, Friendship, and Advice) between partners and associates of a corporate partnership.

C.ELEGANS [Chen *et al.*, 2006]: This multiplex network consists of layers corresponding to synaptic junctions:

[1] http://deim.urv.cat/~manlio.dedomenico/data.php

Dataset (Network Type)	#layers	#nodes	#edges
Vickers (Social)	3	29	740
CKM (Social)	3	246	1,551
LAZEGA (Social)	3	71	2,223
C.ELEGANS (Neurinal)	3	279	5,863
Twitter (Online Social)	2	456,626	15,183,974
Private (Online Social)	17	40,000	1,380,470

Table 1: Statistics of datasets.

electric ("ElectrJ"), chemical monadic ("MonoSyn"), and polyadic ("PolySyn") among a group of neurons.

Since the sizes of the above multiplex networks are relatively small, to test the scalability and performance of our model, we also conduct experiments on two real large-scale online social media networks: **Twitter** and **Private**. The detailed information of these large scale social networks are as follows:

Twitter: The Twitter data used in our experiments is the higgs-twitter dataset[2]. This dataset was used to analyze the spreading of news [De Domenico *et al.*, 2013]. It crawled all tweets about a scientific term "Higgs boson discovery" in one week. We select the largest two sub-networks (following and retweet) to build the multiplex network.

Private: Our Private data is based on an online social network, where users can have friendship relations with each other and can send articles to their friends. From all of our users, we randomly chose 40,000 users who have annotated themselves affiliated with a university and we recorded all the article forward activities within that group in one month. After applying topic modeling on the content of the articles, we grouped all the articles by topics. Thus, we got 16 topics and each one of them creates a relation type of the multiplex network. If we add the base friendship relation, we have in total 17 relation types.

The statistics of these datasets are listed in Table 1.

4.2 Baseline Methods

We will first compare our model with following state-of-the-art embedding-based baseline methods.

- **DeepWalk**: DeepWalk [Perozzi *et al.*, 2014] first applies random walk on the network, treats the path as a sentence, and then uses Skip-gram algorithm to train the embeddings.
- **LINE**: LINE [Tang *et al.*, 2015] adds direct link fitting term to the DeepWalk cost function, and adds second hop friends into one hop friends to incorporate higher-order relations.
- **Node2Vec**: Node2Vec [Grover and Leskovec, 2016] adds a pair of parameters to control the random walk process and makes it better for certain types of nodes, such as hubs or tail users.
- **Principled Multilayer Network Embedding**: PMNE [Liu *et al.*, 2017] proposed three different models to merge multiplex network together to generate one overall embedding for each of the node, we will compare our model with all of their three models. We denote their network aggregation, results aggregation, and Co-analysis model as PMNE (n), PMNE (r), and PMNE (c) respectively.

Besides the above embedding methods, we will also compare our embedding model with the state-of-the-art network structure-based methods for the link prediction task.

- **Common neighbor (CN)**: The CN metric is one of the most widespread measurements used in link prediction tasks due to its simplicity [Newman, 2001]. For each pair of nodes, the more common neighbors it has, the more likely it will have one edge.
- **Jaccard Coefficient (JC)**: For a pair of nodes, JC normalized the number of common neighbors with the number of their total neighbors.
- **Adamic/Adar (AA)**: AA [Adamic and Adar, 2003] is similar to JC, but unlike JC, AA gives more weight to the nodes with fewer neighbors and compared with other structure-based methods, it achieved state-of-the-art performance on a series of networks [Liben-Nowell and Kleinberg, 2007].

4.3 Experimental Settings

Evaluation Metrics

For link prediction, following the commonly used evaluation criteria in similar tasks, we use ROC-AUC [Hanley and McNeil, 1982] as the evaluation criteria in our experiment. A higher value represents better performance and an ideal model that ranks all the positive samples above negative samples will achieve AUC value of 1. For Node Classification, we employ the ℓ_2-regularized logistic regression on learned embedding to train classifiers and we evaluate all the embedding models based on the prediction accuracy.

Model Parameters

To be fair, following their original papers, for all the embedding based methods, we set their embedding dimensions and the dimension of common embedding in our model to be 200. For all the random walk based methods, we set the width of the window to be ten and select five negative samples. As LINE has two kinds of embedding (the so-called first-order proximity and second-order proximity), we set the embedding dimension for both embeddings to be 100 and concatenate them together. For Node2Vec, we empirically use the best hyper-parameter for training, which is $p = 2$ and $q = 0.5$. For the three PMNE models, we will use the hyper-parameters given by their original paper. For our algorithm multiplex network embedding (MNE), we simply set the dimension of additional vectors to be 10.

4.4 Link Prediction

As shown in Table 2, we evaluate AUC values of different models on all of the datasets with five-fold cross-validation setting. Following other embedding methods, for each pair of nodes, we calculate the cosine similarity of their embedding. The larger the similarity the more likely there exists a link between them. As the multiplex network has more than one

[2]https://snap.stanford.edu/data/higgs-twitter.html

Model	Vickers	CKM	LAZEGA	C.ELEGANS	Twitter	Private
DeepWalk	0.821 (0.030)	0.781 (0.008)	0.780 (0.007)	0.821 (0.006)	0.502 (0.002)	0.621 (0.007)
LINE	0.676 (0.011)	0.637 (0.012)	0.695 (0.006)	0.732 (0.006)	0.519 (0.003)	0.512 (0.010)
Node2Vec	0.821 (0.030)	0.781 (0.008)	0.780 (0.007)	0.820 (0.006)	0.504 (0.003)	0.644 (0.010)
PMNE (n)	0.810 (0.032)	**0.917** (0.008)	0.792 (0.009)	0.843 (0.003)	0.446 (0.004)	0.629 (0.006)
PMNE (r)	0.844 (0.025)	0.904 (0.008)	0.813 (0.007)	0.835 (0.007)	0.446 (0.001)	0.659 (0.005)
PMNE (c)	0.837 (0.029)	0.847 (0.016)	0.797 (0.011)	0.824 (0.009)	0.449 (0.002)	0.506 (0.004)
Common Neighbor (CN)	0.799 (0.011)	0.877 (0.006)	0.809 (0.007)	0.869 (0.002)	0.592 (0.002)	0.691 (0.002)
Jaccard Coeficient (JC)	0.778 (0.007)	0.873 (0.006)	0.826 (0.007)	0.833 (0.001)	0.520 (0.002)	0.573 (0.004)
Adamic/Adar (AA)	0.803 (0.019)	0.875 (0.013)	0.814 (0.008)	0.881 (0.001)	0.592 (0.002)	0.691 (0.003)
MNE	**0.871** (0.014)	0.900 (0.010)	**0.839** (0.013)	**0.910** (0.006)	**0.622** (0.003)	**0.723** (0.002)

Table 2: Link prediction based on similarities between two nodes. All the numbers are the averaged AUC score based on five-fold cross validation. The standard deviations are reported in the parentheses.

relation types, we will compute AUC for each relation type first and take the average of all the relation types as the final result. For the models designed for single layer network, we will train a separate embedding for each relation type of the network and use that to predict links on the corresponding relation type, which means that they do not have information from other relation types of the network. One thing worth mentioning is that, in the two large online social network, the following/friendship network is the base network for other relation types. For example, in our Private data, one can only send articles to their friends. To make the evaluation more similar to our real task, for the AUC evaluation of these two datasets, we will randomly select the same amount of edges from the base network, which do not appear in that relation type to be the negative examples and we will not evaluate the base network. Based on the experiment results, we have the following interesting observations:

(1) For nearly all the networks, jointly considering different relation types of the network is helpful. The models that take the whole network into consideration outperform all the models designed for the single network (Deepwalk, LINE, Node2Vec). This is consistent with our assumption that the information from any single network is not enough and information in different relation types can be the complement to each other.

(2) The performance of baseline methods varies a lot based on the topological structure of different networks. For example, if the network is a dense one like LAZEGA or C.ELEGANS, the performance of traditional simple methods are good enough, sometimes even better than the embedding based approach. However, when the network is relatively sparse like Vickers or CKM, embedding based methods have better performance.

(3) The main difference between our model and PMNE is that PMNE generates one overall embedding for a node and our model generates a set of embeddings to capture the distinct information about each relation type. The experimental results show that such information could be important, especially on our Private dataset, where difference among different relation types of the network is more obvious.

To summarize, on six multiplex networks, our model can stably outperform or achieve comparable performance with all the baseline methods. Our understanding is that the common embedding in our model merges information from different relation types, and the additional vectors keep the distinct property of each relation type of the network.

4.5 Parameter Sensitivity

In our algorithm, there are three parameters: the dimension of low-dimensional addition vectors s, the weight of additional vectors w^i, and the transformation matrix norm limitation r. To clearly show the influence of these parameters, we repeat our link prediction experiment with different parameter settings. For the small multiplex datasets, we take the average of them to see the trend.

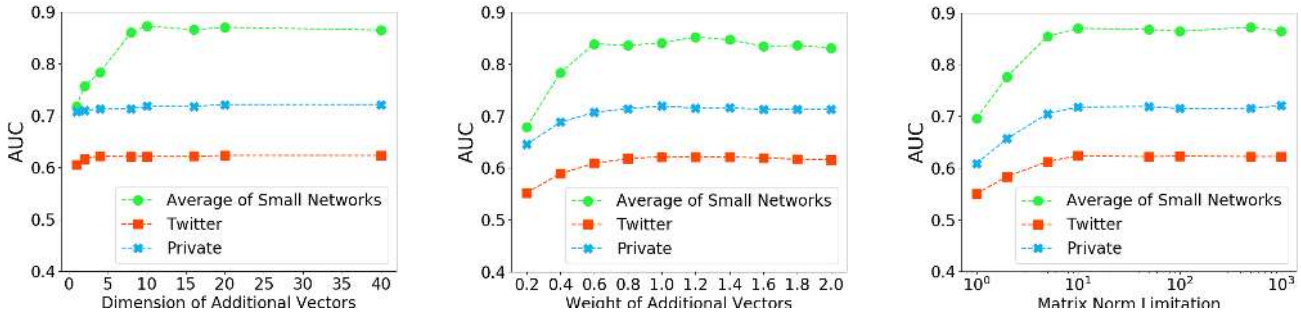
As shown in Figure 1(a), the performance of our model is improved when we increase the dimension of the additional vector. When the dimension reaches 10, the performance reaches the top, which proves that with the help of common embedding, we can use a very low dimension (about one-tenth of the base dimension) vector to represent the property of each relation type of the multiplex network.

As shown in Figure 1(b), when we increase the weight of the additional vector, the performance of our model will arise. But after reaching to 1, the performance will start to go down slightly. This is because in our embedding training phase, we fixed the weights as one. It seems tuning this weights using cross-validation does not further improve link prediction.

As shown in Figure 1(c), if we set the regularization parameter of the transformation matrix to be too small, we may restrict the learning process of our model. When the regularization parameter is larger than 10, the performance becomes stable and the influence of our restriction is not significant.

4.6 Node Classification

In this section, we conduct the experiment on node classification task to evaluate the quality of embedding captured by our model. As the CKM social network is the only multiplex network that provides the ground truth about node labels, we use it as the experimental dataset. Considering the size of datasets being small, we conduct 2-fold cross-validation experiment on this task. The dataset labels all the researchers with their original companies. We use this feature as labels



(a) additional vector dimension s (b) weight of additional vectors w^i (c) matrix norm limitation r

Figure 1: Performance of our model on different experimental settings. For the experiment on s, we set $w^i = 1$ and $r = 1000$. For the experiment on w^i, we set $s = 10$ and $r = 1000$. For the experiment on r, we set $s = 10$ and $w^i = 1$.

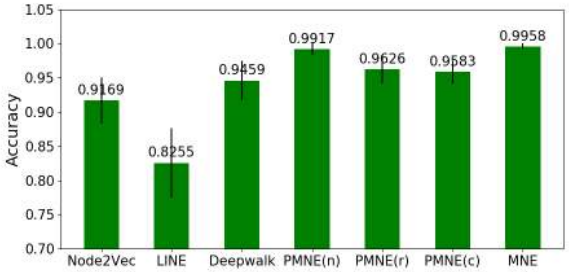


Figure 2: Node classification result.

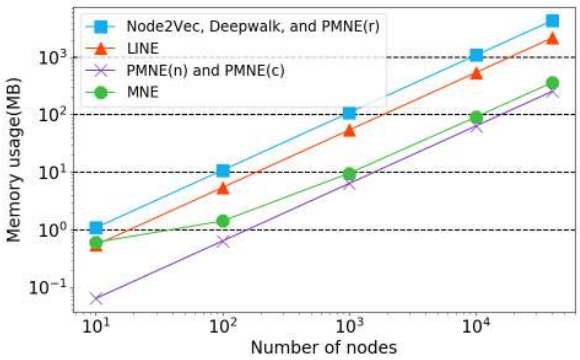


Figure 3: Scalability of memory use.

to train the classifier and evaluate our embedding. For Deepwalk, LINE, and Node2Vec, as they are designed for single network, we first train a separate embedding for each relation type and average them as the final embedding for each node. For our model, we set the weight w^i of all the types to be 1.

The result is shown in Figure 2. Our model can achieve the best performance among all the embedding models. This proves that through simple average operation, our model can produce the high-quality overall representation of nodes.

4.7 Scalability of Our Model

In this section, we investigate the scalability of our model with our large-scale online social network. As we all know, in real life, the social network could be huge. For example, there are about 1 billion active users on our private social network. Every day, more than 500 million relational edges are created. If we learn embedding for all of the sub-networks, the training and storage of these models could be a big challenge. Hence we propose the structure of small additional vector and transformation matrix, trying to reduce the overall model size without sacrificing the overall performance.

As shown in Figure 3, the memory used by our model is almost linear to the network size. Compared with Node2Vec and LINE, our model occupy one-tenth of the memory space without sacrificing performance. The reason is obvious. As we showed in section 3, the memory complexity of our model is $O((d + s * M)|\mathcal{N}|)$ and the memory complexity of Deepwalk and LINE is $O((d * M)|\mathcal{N}|)$, where s is designed to be much smaller than d. For the PMNE(n) and PMNE(c), as they merge the multiplex network into one single network, their model is the smallest but their model loses distinct information about relational edges, which is proved important in the previous experiment.

5 Conclusion and Future Work

In this paper, we present a new scalable embedding model for multiplex networks. We tested our model on two tasks: link prediction and node classification. The experimental results show that our model can effectively capture the overall representation of nodes without sacrificing the distinct property of every single relation types. We also conduct scalability experiments using our social network to show that when the network is large, we only need one-tenth of the old model size to achieve better performance. In the future, we will explore the idea of multiplex network for tasks in other areas.

Acknowledgments

This paper was supported by HKUST-WeChat WHAT Lab, China 973 Fundamental R&D Program (No. 2014CB340304), and the Early Career Scheme (ECS, No. 26206717) from Research Grants Council in Hong Kong.

References

[Adamic and Adar, 2003] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.

[Archdeacon, 1996] Dan Archdeacon. Topological graph theory. *A survey. Congressus Numerantium*, 115(5-54):18, 1996.

[Bródka *et al.*, 2012] Piotr Bródka, Krzysztof Skibicki, Przemyslaw Kazienko, and Katarzyna Musial. A degree centrality in multi-layered social network. *CoRR*, abs/1210.5184, 2012.

[Cai *et al.*, 2005] Deng Cai, Zheng Shao, Xiaofei He, Xifeng Yan, and Jiawei Han. Community mining from multi-relational networks. In *PKDD*, pages 445–452, 2005.

[Chen *et al.*, 2006] Beth L Chen, David H Hall, and Dmitri B Chklovskii. Wiring optimization can relate neuronal structure and function. *Proceedings of the National Academy of Sciences of the United States of America*, 103(12):4723–4728, 2006.

[Chen *et al.*, 2016] Chen Chen, Hanghang Tong, Lei Xie, Lei Ying, and Qing He. FASCINATE: fast cross-layer dependency inference on multi-layered networks. In *KDD*, pages 765–774, 2016.

[Cho *et al.*, 2016] Hyunghoon Cho, Bonnie Berger, and Jian Peng. Compact Integration of Multi-Network Topology for Functional Analysis of Genes. *Cell Systems*, 3(6):540–548.e5, December 2016.

[Chung, 1997] Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[Coleman *et al.*, 1957] James Coleman, Elihu Katz, and Herbert Menzel. The diffusion of an innovation among physicians. *Sociometry*, 20(4):253–270, 1957.

[De Domenico *et al.*, 2013] Manlio De Domenico, Antonio Lima, Paul Mougel, and Mirco Musolesi. The anatomy of a scientific rumor. *Scientific reports*, 3:2980, 2013.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[Goyal and Ferrara, 2017] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *CoRR*, abs/1705.02801, 2017.

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

[Hanley and McNeil, 1982] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[Huo *et al.*, 2007] Xiaoming Huo, Xuelei (Sherry) Ni, and Andrew K. Smith. A survey of manifold-based learning methods. In T. W. Liao and E. Triantaphyllou, editors, *Recent Advances in Data Mining of Enterprise Data*. World Scientific, 2007.

[Kazienko *et al.*, 2010] Przemyslaw Kazienko, Piotr Bródka, and Katarzyna Musial. Individual neighbourhood exploration in complex multi-layered social network. In *IEEE/WIC/ACM, Proceedings*, pages 5–8, 2010.

[Lazega, 2001] Emmanuel Lazega. *The collegial phenomenon: The social mechanisms of cooperation among peers in a corporate law partnership*. Oxford University Press on Demand, 2001.

[Li *et al.*, 2018] Jundong Li, Chen Chen, Hanghang Tong, and Huan Liu. Multi-layered network embedding. *SDM18*, 2018.

[Liben-Nowell and Kleinberg, 2007] David Liben-Nowell and Jon M. Kleinberg. The link-prediction problem for social networks. *JASIST*, 58(7):1019–1031, 2007.

[Liu *et al.*, 2017] Weiyi Liu, Pin-Yu Chen, Sailung Yeung, Toyotaro Suzumura, and Lingli Chen. Principled multilayer network embedding. In *ICDM Workshops*, pages 134–141, 2017.

[Lytras *et al.*, 2010] Miltiadis D. Lytras, Patricia Ordóñez de Pablos, Adrian Ziderman, Alan Roulstone, Hermann A. Maurer, and Jonathan B. Imber, editors. *WSKS, Proceedings*, volume 111 of *Communications in Computer and Information Science*. Springer, 2010.

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

[Mikolov *et al.*, 2013b] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

[Newman, 2001] Mark EJ Newman. Clustering and preferential attachment in growing networks. *Physical review E*, 64(2):025102, 2001.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *KDD*, pages 701–710, 2014.

[Spielman, 2011] Daniel Spielman. Spectral graph theory. In *Combinatorial Scientific Computing*, chapter 16. Chapman and Hall/CRC Press, 2011.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.

[Verbrugge, 1979] Lois M Verbrugge. Multiplexity in adult friendships. *Social Forces*, 57(4):1286–1309, 1979.

[Vickers and Chan, 1981] M Vickers and S Chan. Representing classroom social structure. *Victoria Institute of Secondary Education, Melbourne*, 1981.