Learning Content-rich Diffusion Network Embedding

1Guangyuan Wang, 2Zongyi Wang
1,2CS department, University of Illinois at Urbana-Champaign, IL 61801
{gwang10, zwang195}@illinois.edu

Abstract – Information networks are ubiquitous in the real world, while embedding, as a kind of network representation, has received attention from many researchers because of its effectiveness in preserving the semantics of the network and its broad application including classification, link prediction etc. Previously, many methods have been proposed to learn network embedding from non-attributed and static networks. Networks are treated simply as nodes and links. However, information is not merely encoded in the structure, nodes itself may have their intrinsic attributes and little research have been done to incorporate this information. Furthermore, outside information will also diffuse on the network over the time and for a specific time we can have a different diffusion structure. In this paper, we will be introducing the idea of diffusion network, and we propose two models for embedding learning that aim to efficiently capture the rich content of nodes as well as that of the diffusion. We then evaluate the quality of our embedding by conducting node classification experiments, the result of which shows that our method for generating embedding outperforms other baselines.

1 INTRODUCTION

There has been intensive research studying the structure of a network, either homogenous or heterogeneous. Many papers propose the encode the information of a network in an embedding. Network embedding has proven to be successful in capturing the semantics of an information network. By learning the embedding representation we can perform tasks such as node clustering/classification[17,18,19], link prediction[15] etc.

Earlier studies on network embedding construct the complete node affinity graph based on node feature similarity, and then compute the eigenvector of the affinity matrix as node embeddings [2, 3, 4]. Later, the Skipgram model from word2vec[5] has proven to be successful for word embedding[5]. This has inspired quite a few new node embedding techniques, such as Deepwalk [6], Node2vec [7], Planetoid [8], LINE[9], GraRep [10], SDNE [11], etc. By sampling truncated random walks (e.g., Deepwalk, Node2vec, Plainetoid) or measuring proximity preserving objectives (e.g., LINE, GraRep), the embeddings they compute can preserve network structures in an implicit and coarse way.

Deep learning also provides a new way to generate network embeddings, as it naturally provides a way of modelling sophisticated non-linear relationships, which is powerful especially in a complex network setting. This approach is adopted by SDNE[11], DNGR model[12] and SINE[13].

It is clear that all these existing works study the networks in a static and non-attributed setting. However, in real-world situations, a network often contains much more information than the link structure. First of all, nodes in an information network have its own properties. Take DBLP network as an example, authors may have nationality and affiliations, which can potentially be useful for co-authorship prediction tasks. Another example is LinkedIn data, where users have education background and work places. Second, outside information may also have an effect upon the network, which we call diffusion. For example, a group of author in the DBLP network may cite the same paper at different time, and they would form a distinct diffusion structure, while the content of this paper, which we refer to as the diffusion content, is important to know when doing predictions on those authors. The highlight of our work is to take node content and diffusion into account while generating our embedding.

The main challenge for us can be summarized as: first, to develop a method to leverage diffusion...
structure. A diffusion structure can be a clique, where each node is equivalent with respect to the diffusion, or it can also be a path, indicating strong bias during the process of propagation. Knowing the diffusion structure is important for knowing the diffusion information property. Second, to develop a method to incorporate the contents. As previously demonstrated, either the node content and the diffusion content will be essential for prediction tasks.

To leverage diffusion structure, our current approach is to generate pairs to preserve proximity within the structure. In order to incorporate content into our embedding learning, we use an autoencoder to encode the content, and jointly train the content embedding with node pair embeddings. There are a few limitations in our current approach, which we will briefly talk about later.

In the following chapter, we first introduce some related work and some important definitions, then we propose two methods that incorporate content and diffusion for generating embedding. Finally, we show our experiment results, comparing it to three other baselines and discuss our future improvement.

2 RELATED WORK

2.1 Heterogenous/Homogenous Network embedding

There has already been a great number of researches done on both homogenous information network as well as heterogeneous ones. The definition of the two networks can be found in the next section.

Popular works of homogenous network embeddings includes Deepwalk[6], where the idea of random walk and the skip-gram model in word2vec[5] is joined to produce the embedding representation. LINE[9] models the first order proximity and second order proximity of nodes for embedding training. Based on these ideas, a group of heterogenous network embedding techniques is also developed. An example will be metapath2vec[22], which is a random walk approach based on the given metapath schema.

All these methods try to capture the structural information in a network, and they do a good job at capturing the higher-order structure, either with higher-order proximity or random walk. However, most of them do not try to incorporate the node attributes features. TADW[16] is one of the few that examine the node content. However, it is still evaluating a static network.

2.2 HIN2vec

The goal of HIN2Vec[24] is to learn a meaningful representation for each node in the network and the meta-path by encoding the rich information embedded in the meta-path and the whole network structure. The framework consists of two phrases: (1) Training data preparation based on random walk and negative sampling. (2) Representation learning.

For the training data preparation, the meta-path is pre-specified by the user and then based on the random walk result, training data are generated based on the provided meta-path. For example: random walk result is P1, P2, A1, P3, A1, then assume the meta-path given P-P-A, then we can generate the following relation <P1, A1, P-P-A> where P1 is the starting point of the meta-path P-P-A and A1 is the end point of the same meta-path. The negative data can be generated by replacing the starting node or the end randomly with the node of the same type.

The origin model of HIN2Vec is to predict a particular relation given the two input node. That is, given a relation <x,y,r> where x, and y are two nodes in the HIN, and r is the meta-path that connect the two relations, we want to predict whether <x,y,r> is true or not.

We adapt the idea of HIN2Vec for incorporating diffusion content into our framework. A general analogy is that r is our diffusion content, while (x, y) is one pair of nodes in the diffusion structure.

2.3 Neural Tensor Network

Neural Tensor Network[20] is developed for reasoning over relationship between two objects by learning vector representations. The highlight of this work is that instead of a linear neural network layer, a bilinear tensor layer is used to that directly relate the two vector representations across multiple dimensions. Comparing to standard neural networks, where relation of two inputs can only be modeled implicitly through the nonlinearity, Neural Tensor Network can relate the two inputs multiplicatively. Each slice of the tensor can be seen as being responsible for one type of relation between the object pairs.

This idea can be naturally adapted by out embedding learning framework. We use Neural Tensor Network to model correlation between embeddings of one pair of nodes in the diffusion structure.

3 PRELIMINARY
A series of definitions needs to be made before we move on to introducing our model.

3.1 (Static) Heterogenous/Homogenous Information Network

We adopt the definition from [22]: Heterogenous Information Network (HIN) is a directed graph $G = (V; E; \phi; \psi)$, where $V$ is the set of nodes; $E \subseteq V \times V$ is the set of edges in $V$. $\phi: V \to A$ and $\psi: E \to R$ are type mapping functions for nodes and edges, respectively. Here each node $v \in V$ is mapped to one particular node type in $A$, i.e., $\phi(v) \in A$, and each link $e \in E$ belongs to a particular edge type in $R$, i.e., $\psi(e) \in R$. When $|A| > 1$ or $|R| > 1$, the network is called a heterogeneous information network; otherwise, it is called a homogeneous information network.

3.2 Diffusion Structure/Diffusion Content

The diffusion structure for information $i$ is a set of tuples $d_i = \{(v_k, t_k)\}$, indicating that at time $t_k \in T$, user node $v_k \in V$ obtain information $i$.

Diffusion content: for each $d \in D$, there is a content vector $c$ associated with $d$ ($c$ can represent text or images).

3.3 Diffusion network

Diffusion network $DN = (V, E, D)$ is constructed by attaching the set of diffusion structures $D = \{d_i\}$ for all information $i \in I$ to a originally static network $G' = (V, E)$.

3.4 Content-rich Diffusion Network

A Content-rich Diffusion Network, a.k.a $CDN = (V, E, D, A, C)$ is constructed by attaching node and diffusion content representations to a Diffusion Network $DN = (V, E, D)$. $A = \{a\}$ is the set of node content vectors, where each node $v \in V$ has a vector $a$ associated with it.

3.5 Content-rich Diffusion sub-network

Given a content-rich diffusion network, by incorporating each diffusion structure $d_i$, a content-rich diffusion sub-network is formed if there is an edge between $(u, v)$ and in diffusion structure $d_i$, $t_u > t_v$. Denoting information is diffused from $u \to v$. Following such construction, a content-rich diffusion sub-network is formed. Figure 1.

4 MODELS

There are two main challenges in solving this problem:

1) How to preserve the diffusion subnetwork structure.
2) How can the diffused information be utilized in the network structures.

To solve the first challenge, the network embedding techniques can be used. In network embedding, the representation of each node is learned to preserve the network structure. Most of network embedding techniques aim to learn an embedding that preserve first-order and higher-order proximity. By using the graph embedding techniques, the diffusion sub-network’s nodes proximity can be preserved.

In order to utilize the content diffused on the network, most natural way is to turn the content into vectors. Since the content vector not only contains the information of the content but also can be easily compiled together graph embedding. Additionally, an auto-encoder can be added to further extract the most valuable information.

Based on the above two intuition, our model has the following two objectives:

1) Using graph embedding to preserve the network structure.
2) Jointly train auto-encoder along with training graph embedding. Thus, the node embedding captures the content information.

The input data is prepared in the following way for the model learning:

1) For text based content, tokenize the word from the whole corpus and then using standard word2vec[5] to get the word embedding of each of the word token. For content such as sentence, the content vector is represented as the sum of the word embedding for the words appeared in the content. For image contents, the embedding of image can be extracted from the pre-trained CNN network. If multi-typed data appears together in content, then the content vector is the sum of all the vector from different type of data.

2) Diffusion sub-network structure are represented as node pairs. If there is an edge between
node \((u, v)\), then \((u, v)\) as a node pair will be generated. The pairs are generated for all the connected pairs in each of the diffusion sub-network.

3) The model input is finalized as a triplet \((u, v, c)\) where \(u, v\) represent the nodes in the diffusion sub-network, and \(c\) represented the content diffused from \(u \rightarrow v\).

In the following sections we are presenting two models that aim to achieve the objective.

### 4.1 Individual Embedding based Learning

Given the input triple \((u, v, c)\), intuitively, we can learn a model to predict what is the information diffused from \(u \rightarrow v\). However, this approach exhibits several drawbacks. First, if there are many different information diffused from \(u \rightarrow v\), then this problem becomes a multi-label problem, that is, what is the possible set of information that \(u\) diffuses to \(v\) among all information. The multi-label problem is hard to solve, and also have a relatively low predication accuracy effect the quality of the learned embedding. Second, the efficiency of the model is relatively poor due to large number of classes the model has to deal with.

Instead, a more efficient approach is adapted from Hin2Vec[24]. In this framework, the model is a decision function which output how likely a triplet \((u, v, c)\) is the true data sample, or not. Meaning whether \(c\) is the information is diffused from \(u \rightarrow v\). The our framework, the content vector \(c\) is first passed through auto-encoder to get reduced content representation \(c'\), then \(c''\) is recovered by decoder.

\[
\begin{align*}
\tilde{c} &= \text{Dropout}(c) \\
c' &= g_{en}(W_{en}\tilde{c} + b_{en}) \\
c'' &= g_{de}(W_{de}c' + b_{de}) \\
L_1 &= \|c - c''\|^2_S
\end{align*}
\]

where dropout(.) is the stochastic process that randomly set of portions of input to 0. And \(g_{en}\) is the activation function of encoder, and \(g_{de}\) is the activation function of decoder. \(c'\) is the extracted information from auto-encoder, and then passed to decision function along with current node embedding of \(u, v\).

\[
y = \sigma \left( \sum (u \odot v \odot c') \right)
\]

where \(\sigma(.)\) is the sigmoid activation function, \(\odot\) is the element wise multiplication, thus the loss is represented as

\[
L_2 = (1 - y') \log(1 - y) + y' \log(y)
\]

Where \(y'\) is the ground truth whether \((u, v, c)\) is true data sample. The loss \(L = L_1 + L_2\) consists of autoencoder reconstruction loss with the predication loss. The overview of this model is shown in figure 2.

The intuition of the model is that if node \(u\) and node \(v\) are close in the network structure then there embedding tend to be similar which means the summation of their element multiplication should be large. If the content representation is similar to node embedding \(u\) and \(v\), then the model is likely to predict 1.

![Figure 2. Over view of individual embedding learning model.](image)

### 4.2 Pairwise Node Embedding based Learning

Improving can be made by using addition neural network to capture the relationship between \(u, v\), thus in model we added the addition layer of neural network layer. The neural network has a bilinear layer as well as linear layer. The bilinear layer is used to capture all possible different relation between \(u, v\), and linear layer is used to capture additional information when combining \(u, v\) together. The result is a pairwise embedding containing possible relationship between \(u, v\). The auto-encoder is used in the same way as before.

\[
p = \tanh \left( u^T W_1^{[1:k]} v + W_2 \begin{bmatrix} u \\ v \end{bmatrix} + b \right)
\]

\[
y = \sigma \left( \sum (p \odot c') \right)
\]

where \(p\) represent the pairwise embedding between \(u, v\). And the decision function is now taking two vectors instead of three vectors. The loss from prediction is:

\[
L_2 = (1 - y') \log(1 - y) + y' \log(y)
\]

the loss from auto-encoder is
\[ L_1 = ||c - c'||^2 \]

therefore, the total loss is \( L = L_1 + L_2 \).

The intuition of the model is pairwise embedding will capture information co-exist in both node \( u, v \). For example, assume node \( u \) and \( v \) are author in DBLP network, then pairwise embedding may capture information that they have the same interest. And if the content is about the same area, then there is a high chance that the information is diffused from \( u \to v \). The overview of the model pipeline is shown below.

**EXPERIMENTS**

**5.1 Dataset**

We use the DBLP dataset for evaluation as well as comparison to other methods proposed by different papers. This dataset consists of 3 types of nodes, author, paper and venue. A ground truth of 4 classes of authors is given, each class corresponding to a different research field. A total of 4k authors are labeled.

**5.2 Baselines and parameter settings**

The method we compare to are PTE[21], ESIm[23], metapath2vec[22].

PTE[21] decomposes an HIN down to a collection of bipartite graphs and then learn the embedding representation.

ESIm[23] extends on the idea of PathSim[25], ESim explore the similarity embedded in the structure of a HIN with metapath guided embedding technique.

Metapath2vec[22] is a metapath guided random walk framework. This work consists of three major parts: (1) heterogeneous Skip-Gram, (2) meta-path guided random walk for generating skip-gram data, (3) heterogeneous negative sampling.

For all those three methods, we use the default parameter settings provided in the code written by its authors. Because our content-rich network is generated based on APA(author-paper-author) relation, we use APA to generate metapath instances for HIN embedding methods incorporating metapath features.

**5.3 Our model and parameter settings**

We use paper content as diffusion content, extract key phrases, use word2vec[5] to get the initial embedding (which is trained based on the entire corpus), then take this as the input to an autoencoder. The resulting vector is the paper content embedding, which we use to guide the learning of author embeddings with either model 1 or model 2.

**5.4 Node Classification**

One of the most important application of node embedding is node classification[1], which we will use as an indicator of the quality of our embedding. Embedding vectors serve as perfect feature representation. Based on the embedding vectors, the task is to classify the nodes into different classes with high accuracy. to be put into various kinds of classifiers.

We obtain embedding of DBLP authors from all these methods. Using the embedding as train features, an SVM is used for the classification task. For the labeled authors, we use 90% training data and 10% validation data for conducting the experiment.

Table 1 shows the result of our experiment.

**5.5 Node Clustering**

Another important application is node clustering. The task is to partition nodes into several clusters. After embeddings are generated for each node, a simple K-Means algorithm can be used to perform the clustering. We use F1-score, NMI(normalized mutual information) and Jaccard score as our evaluation metric. Table 2 shows the result of our node clustering experiment.

Even though we have not achieved the top score, we believe our model has great potential, since we are incorporating much more information and modeling a more complex network. Discuss about future improvements can be found in the next section.
6 CONCLUSION AND FUTURE DIRECTION

Information networks is worth studying because of its wide application, and network embedding is an essential way to represent the rich information of a network. Previously, researchers only focus on static and non-attributed networks, but our experiment shows that we can improve node classification accuracy by adapting these content and diffusion. But we believe improvement can still be made in our future work.

An important aspect would be analyzing the higher-order structure of the diffusion structure, although we exploit the content of nodes in our methods, we did not examine carefully the higher-order structure information. For example, methods like random walk or graph convolution may help capture this kind of information. We believe we would achieve better results if we combine our methods with those emphasizing capturing structure information.

REFERENCES