Heterogeneous Network Embedding via Deep Architectures

Shiyu Chang\(^1\), Wei Han\(^2\), Jiliang Tang\(^3\), Guo-Jun Qi\(^4\), Charu C. Aggarwal\(^5\), Thomas S. Huang\(^1\)

\(^1\)Beckman Institute, University of Illinois at Urbana-Champaign, IL 61801.
\(^2\)Computer Science and Engineering, Arizona State University, Tempe, AZ 85281.
\(^3\)University of Central Florida, Orlando, FL, 32816.
\(^4\)IBM T.J. Watson Research Center, NY, 10598.
{chang87, weihan3, t-huang1}@illinois.edu, jiliang.tang@asu.edu, guojun.qi@ucf.edu charu@us.ibm.com

ABSTRACT
Data embedding is used in many machine learning applications to create low-dimensional feature representations, which preserves the structure of data points in their original space. In this paper, we examine the scenario of a heterogeneous network with nodes and content of various types. Such networks are notoriously difficult to mine because of the bewildering combination of heterogeneous contents and structures. The creation of a multidimensional embedding of such data opens the door to the use of a wide variety of off-the-shelf mining techniques for multidimensional data. Despite the importance of this problem, limited efforts have been made on embedding a network of scalable, dynamic and heterogeneous data. In such cases, both the content and linkage structure provide important cues for creating a unified feature representation of the underlying network. In this paper, we design a deep embedding algorithm for networked data. A highly nonlinear multi-layered embedding function is used to capture the complex interactions between the heterogeneous data in a network. Our goal is to create a multi-resolution deep embedding function, that reflects both the local and global network structures, and makes the resulting embedding useful for a variety of data mining tasks. In particular, we demonstrate that the rich content and linkage information in a heterogeneous network can be captured by such an approach, so that similarities among cross-modal data can be measured directly in a common embedding space. Once this goal has been achieved, a wide variety of data mining problems can be solved by applying off-the-shelf algorithms designed for handling vector representations. Our experiments on real-world network datasets show the effectiveness and scalability of the proposed algorithm as compared to the state-of-the-art embedding methods.

Categories and Subject Descriptors
H.2.8 [Database Management]: Database Applications--Data Mining.

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’15, August 11–14, 2015, Sydney, NSW, Australia.
Copyright 2015 ACM 978-1-4503-3664-2/15/08 ...$15.00.
http://dx.doi.org/10.1145/2783258.2783296.

Keywords
Heterogeneous embedding, network embedding, feature learning, cross-domain knowledge propagation, deep learning, dimensionality reduction.

1. INTRODUCTION
Vectorized data representations frequently arise in many data mining applications. They are easier to handle since each data can be viewed as a point residing in an Euclidean space [24, 30]. Thus, similarities between different data points can be directly measured by an appropriate metric to solve traditional tasks such as classification [37], clustering [21, 43, 45] and retrieval [10]. As shown in [3], learning good representations is one of the fundamental problems in data mining and Web search, and it often has a stronger impact on performance than designing a more sophisticated model.

Unfortunately, many networked data sources (e.g. Facebook, YouTube, Flickr and Twitter) cannot be naturally represented as vectorized inputs. A combination of graphs and relational data is commonly used to represent these social networks and social media data. Current research has focused on either pre-defined feature vectors [42] or sophisticated graph-based algorithms [48] to solve the underlying tasks. A significant amount of research has been accomplished on various topics, such as collective classification [37], community detection [44], link prediction [47], social recommendation [35], targeted advertising [22], and so on. The development of a unified representation for networked data in embedded vector form is of great importance to encode both content and links. The basic assumption is that, once the vectorized representation is obtained, the network mining tasks can be readily solved by off-the-shelf machine learning algorithms.

Nevertheless, feature learning of networked data is not a trivial task because the data possesses many unique characteristics, such as its size, dynamic nature, noise and heterogeneity. First, the amount of multimedia data on social Websites has increased exponentially. The estimated number of photos on Facebook has reached 100 billion by mid-2011 and it has been continuously increasing by 350 million new uploaded photos each day\(^1\). Given the varied backgrounds of the users, social media tends to be noisy. In addition, researchers have noticed that spammers generate more data than legitimate users [4], which makes network mining even more difficult. Furthermore, social media data contains diverse and heterogeneous information. For instance, different model types are reported when an event takes place. As a Google search example of

\(^1\)http://www.businessinsider.com/facebook-350-million-photos-each-day-2013-9
The key advantages of the proposed HNE framework are the idea of a network-preserved embedding is illustrated in figure 2. In feature space such that they are consistent with network links. The work's inter-connections to design a proper loss function that en-

especially when the linkage information plays the key role in re-

than a shallow embedding solution to capture the network links, 

through the proposed framework. This makes it more powerful 

both the local and global linkage structures of underlying networks 

iteratively solving an optimization problem with respect to feature 

videos). Our model coordinates two mutually reinforcing parts by 

[12, 18] are aggregated to handle vectorized data (e.g., text docu-

works (DNN) [1, 14] and Convolutional Neural Networks (CNN) 

to a unified latent space so that objects from different 

complex heterogeneous networks that impose tremendous chal-

To address the aforementioned challenges, we present a novel 

idea on network representation learning, termed Heterogeneous 

Network Embedding (HNE), which jointly considers both the content 

as well as the relational information. HNE maps different hetero-

geneous objects into a unified latent space so that objects from dif-

ficult to scale to large graphs. A nonlinear embedding model proposed by Yuan et. al. [46], used Restricted Boltzmann Machines (RBMs) for cross-model link analysis. However, it did not utilize all of the social information from the raw data (required a feature vectorization step), which resulted in suboptimal solutions. Moreover, work in computer vision and speech [18] has shown that layer-wise RBM training is inefficient for large-scale settings when compared to DNNs and CNNs.

2.2 Deep Learning

In recent years, machine learning research has seen a marked 

switch from hand-crafted [49] features to those that are learned 

from raw data, mainly due to the success of deep learning. Deep 

learning models have become increasingly important in speech recognition, object recognition/detection, and more recently in natural language processing. Deep learning techniques are universal function approximators. However, historically, there has been very limited success in training deep networks with more than two hidden layers. The difficulty lies in the high dimensional parameter space and highly non-convex objective function, where gradient-based methods are trapped by local minima.

Recent advances in deep learning have benefited from a con-

fluence of factors, such as the availability of large-scale datasets, computational resources, and advances in both unsupervised and supervised training algorithms. Unsupervised deep learning, often referred to as "pre-training," provides robust initialization and regularization with the help of unlabeled data, which is copiously available. For example, Hinton and Salakhutdinov [14] first employed layer-wise initialization of deep neural networks with the use of RBMs. A similar approach is weight initialized with auto-encoders, as proposed by Bengio et. al. [1]. More recently, supervised learning with multilayer neural networks has become possible. The method of dropout [15] has shown particular promise. A seven-layer convolutional network developed by Krizhevsky et. al. [18] achieved state-of-the-art performance on the ImageNet Large
A heterogeneous network [41] is defined as a network with multiple types of objects and/or multiple types of links. As a mathematical abstraction, we define an undirected graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is a set of vertices and $E$ is a set of edges. An edge $e_{ij}, \; \forall i, j \in \{1, \cdots, n\}$ belongs to the set $E$ if and only if an undirected link exists between nodes $i$ and $j$. Moreover, the graph $G$ is also associated with an object type mapping function $f_o : \mathcal{V} \to \mathcal{O}$ and a link type mapping function $f_e : \mathcal{E} \to \mathcal{R}$, where $\mathcal{O}$ and $\mathcal{R}$ represent the object and relation sets, respectively. Each node $v_i \in \mathcal{V}$ belongs to one particular object type as $f_o(v_i) \in \mathcal{O}$. Similarly, each link $e_{ij} \in \mathcal{E}$ is categorized in different relations as $f_e(e_{ij}) \in \mathcal{R}$. It is worth mentioning that the linkage type of an edge automatically defines the node types of its end points. The heterogeneity of a network is reflected by the size of the sets $\mathcal{O}$ and $\mathcal{R}$, respectively. In the case of $|\mathcal{O}| = |\mathcal{R}| = 1$, the network is homogeneous; otherwise, it is heterogeneous. An example of a heterogeneous network is illustrated in the left-hand side of Figure 2, which contains two object and three link types. For further ease in understanding, we will assume object types of image (I) and text (T). The link relationships $\mathcal{R}$ correspond to image-to-image (red dotted line), text-to-text (green dashed line) and image-to-text (blue solid line), which are denoted by $R_{II}$, $R_{TT}$ and $R_{IT}$, respectively. Therefore, in this case, we have $|\mathcal{O}| = 2$ and $|\mathcal{R}| = 3$. While this simplified abstraction in the text and image domain is both semantically and notationally convenient for further discussion in this paper, this assumption is without loss of generality because the ideas are easily generalizable to any number of types.

Thus, any vertex $v_i \in \mathcal{V}$ can be categorized into two disjoint subsets $\mathcal{V}_I$ and $\mathcal{V}_T$ corresponding to the text image domains, respectively. Therefore, we have $\mathcal{V}_I \cup \mathcal{V}_T = \mathcal{V}$ and $\mathcal{V}_I \cap \mathcal{V}_T = \phi$. Similarly, the edge set $\mathcal{E}$ can be partitioned into three disjoint subsets, which are denoted by $E_{II}$, $E_{TT}$ and $E_{IT}$, respectively. Furthermore, each node is summarized by unique content information. In particular, images are given as a squared tensor format as $X_i \in \mathbb{R}^{d_I \times d_I}$ for every $v_i \in \mathcal{V}_I$, while texts are represented by a $d_T$-dimensional feature vector as $z_j \in \mathbb{R}^{d_T}$ for all $v_j \in \mathcal{V}_T$. For example, the content representations could be a raw pixel format in RGB color space for images, or it could be the Term Frequency - Inverse Document Frequency (TF-IDF) [32] scores of a text document. We represent linkage relationship as a symmetric adjacency matrix $A \in \mathbb{R}^{n \times n}$, which the $(i, j)$th entry of $A$ equals one if $e_{ij} \in \mathcal{E}$; otherwise $A_{ij} = -1$ (for model simplicity).
across networks. Essentially, the embedding process encodes both heterogeneous content and linkage information to a multidimensional representation for each object. A linkage-guided deep learning framework is then proposed to jointly model the latent space embedding with feature learning simultaneously. This can be solved efficiently by using back propagation techniques. Finally, we discuss the straightforward extension of the HNE algorithm to the general case of more than two object types.

4.1 Latent Embedding in Networks

The main goal of the heterogeneous embedding task is to learn mapping functions to project data from different modalities to a common space so that similarities between objects can be directly measured. Assume that the raw content $X_i$ associated with an image node can be transformed to a $d_I$-dimensional vector representation as $\tilde{x}_i$. The conversion of the raw input data into this $d_I$-dimensional vector representation will be described in the following subsection. A naive approach to do so is by stacking each column of an image as a vector or through feature machines [7, 23]. It is worth pointing out that, the values of $d_I$ and $d_T$ need not be the same, because images and text are defined in terms of completely different sets of features.

We transform two type of samples to a uniform latent space with the use of two linear transformation matrices, denoted by $U \in \mathbb{R}^{d_I \times r}$ and $V \in \mathbb{R}^{d_T \times r}$, for the image and text domains, respectively. The transformed samples are denoted by $\tilde{x}$ and $\tilde{z}$ for images and text documents, respectively, where we have:

$$\tilde{x} = U^T x, \text{ and } \tilde{z} = V^T z.$$  

(1)

Even though the image and documents may be represented in spaces of different dimensionality, the transformation matrices $U$ and $V$ map them into a common $r$-dimensional space. The similarity between two data points with the same object type can be presented as an inner product in the projected space as follows:

$$s(x_i, x_j) = \tilde{x}_i^T \tilde{x}_j = (U^T x_i)^T (U^T x_j) = x_i^T M_{II} x_j,$$

$$s(z_i, z_j) = \tilde{z}_i^T \tilde{z}_j = (V^T z_i)^T (V^T z_j) = z_i^T M_{TT} z_j,$$

(2)

Note that the embedding into a common space also enables similarity computation between two objects of different types, such as text and images, as follows:

$$s(x_i, z_j) = \tilde{x}_i^T \tilde{z}_j = (U^T x_i)^T (V^T z_j) = x_i^T M_{IT} z_j$$

$$= \tilde{z}_j^T \tilde{x}_i = (V^T z_j)^T (U^T x_i) = z_j^T M_{TI} x_i.$$  

(3)

Here, $M_{II} \in \mathbb{R}^{d_I \times d_I}$ and $M_{TT} \in \mathbb{R}^{d_T \times d_T}$ are positive semidefinite matrices while $M_{IT} \in \mathbb{R}^{d_I \times d_T}$. The latent embedding is closely related to similarity and metric learning that has been widely studied in the literature [20]. It suggests that the correlations across networks is closely related to similarity and metric learning that has been widely studied in the literature [20]. It suggests that the correlations

$$L(x_i, x_j) = \log (1 + \exp (-A_{ij} d(x_i, x_j))),$$

(6)

Note that to infer $d$ we need not know the respective entry value of $A$, or require heterogeneous nodes to be in-sample [34]. This means that the approach has the generalization ability to embed samples from unseen nodes. Consider

$$d(x_i, x_j) = s(x_i, x_j) - t_{II},$$

(5)

for all $v_i, v_j \in V_I$, where $t_{ij}$ is a relational bias value. Then, the loss function can be formulated as follows:

$$L(x_i, x_j) = \log (1 + \exp (-A_{ij} d(x_i, x_j))),$$

(6)

where $N_{II}$, $N_{IT}$ and $N_{TT}$ are the numbers of the three types of links in the network. Furthermore, $\lambda_1$, $\lambda_2$ and $\lambda_3$ are the three balancing parameters, in which the first two control the emphasis among three types of links and the last one is used balance the so-called bias-variance trade-off [13]. Moreover, in equation (7), $\| \cdot \|_F$ denotes the Frobenius norm. The bias terms in the loss function can be either treated as learning variables or set to fixed values. For simplicity, we set these bias terms to constants.

The aforementioned objective function can be efficiently solved with coordinate descent methods, which solve for each individual variable while keeping the others fixed:

Solving $U$:

Fixing parameter $V$, the objective function (7) can be reduced as follows:

$$\min_U \frac{1}{N_{II}} \sum_{v_i, v_j \in V_I} L(x_i, x_j) + \frac{\lambda_1}{N_{II}} \sum_{v_i, v_j \in V_R} L(z_i, z_j)$$

$$+ \frac{\lambda_2}{N_{IT}} \sum_{v_i \in V_I, v_j \in V_T} L(x_i, z_j) + \lambda_3 (\|U\|_F^2 + \|V\|_F^2),$$

(7)

The gradient is given by the following:

$$\frac{\partial \ell}{\partial U} = \frac{1}{N_{II}} \sum_{v_i, v_j \in V_I} -A_{ij} (x_i^T x_j + x_i^T x_j) U + 2\lambda_1 U$$

$$+ \frac{\lambda_2}{N_{IT}} \sum_{v_i \in V_I, v_j \in V_T} -A_{ij} x_i^T V$$

$$+ \lambda_3 \lambda_3 (\|U\|_F^2 + \|V\|_F^2).$$

(9)

Solving $V$:

Similarly, the variable $V$ can be handled as follows:

$$\min_V \frac{\lambda_1}{N_{TT}} \sum_{v_i, v_j \in V_T} L(x_i, x_j) + \lambda_3 (\|V\|_F^2)$$

$$+ \frac{\lambda_2}{N_{IT}} \sum_{v_i \in V_I, v_j \in V_T} L(x_i, z_j) + 2\lambda_3 U$$

(10)

Taking the derivative with respect to $V$, we obtain:

$$\frac{\partial \ell}{\partial V} = \frac{\lambda_1}{N_{TT}} \sum_{v_i, v_j \in V_T} -A_{ij} (z_i^T z_j + z_i^T z_j) V$$

$$+ \frac{\lambda_2}{N_{IT}} \sum_{v_i \in V_I, v_j \in V_T} -A_{ij} z_i^T U$$

$$+ \lambda_3 \lambda_3 (\|U\|_F^2 + \|V\|_F^2).$$

(11)
So far, we have shown that our loss function integrates network structures that map different heterogeneous components into a unified latent space. However, such embedding functions are still linear, which might lack the power to model complex network connections. In the following, we will present how Equation (7) fits into the deep learning framework.

4.2 The Deep Architecture

In the previous section, we broken things down into two steps: 1) manually construct a feature representation, 2) embed different modalities into a common space. In this section we tightly integrate these two steps into a deep learning framework by learning the feature representation and embedding together.

\[
\min_{W, D_T, D_T} \frac{1}{|T|} \sum_{\mathbf{v}, \mathbf{v} \in V_T} L(p_{D_T}(\mathbf{X}_i), p_{D_T}(\mathbf{X}_j)) + \lambda_1 (\|U\|_F^2 + \|V\|_F^2) \\
+ \lambda_2 \sum_{\mathbf{v}, \mathbf{v} \in V_T} L(q_{D_T}(\mathbf{z}_i), q_{D_T}(\mathbf{z}_j)) \\
+ \lambda_3 \sum_{\mathbf{v}, \mathbf{v} \in V_T} L(p_{D_T}(\mathbf{X}_i), q_{D_T}(\mathbf{z}_j)),
\]

(12)

Here, \( p(\cdot) \) and \( q(\cdot) \) are two nonlinear functions parameterized by \( D_T \) and \( D_T \), and \( D_T \) and \( D_T \) are two sets of parameters associated with the deep image and text networks, respectively. Specifically, we utilize the CNNs structure as building blocks to learn image features while fully connected (FC) layers are used to extract discriminative representations for pre-processed texts. The feature learning and information embedding are mutually reinforced by our approach.

The image module exploits spatially local correlations by enforcing a local connectivity between neurons from adjacent layers. The parameters on each layer are referred to as filters. The architecture confines the learned filters to reflect the spatial local patterns of images. In addition, each sparse filter is replicated across the entire visual field, which share the same parameters (both weights \( W \) and bias \( b \)). The output of each filter is usually termed as “feature map”, and conceptually, a feature map is obtained by convolving an input image with a linear filter, adding a bias term and then applying a non-linear function. We denote the \( k \)-th feature map at a given layer (a given depth) as \( h^k \), which is determined by the corresponding weights \( W^k \) and bias \( b^k \). Then, the feature map is obtained as follows:

\[
h^k = \max\{0, \langle W^k \ast \mathcal{M} \rangle + b^k\},
\]

(13)

Here, \( \ast \) denotes the convolution operation and \( \mathcal{M} \) is an input from the previous layer of the deep image module. The definition of convolution of a filter \( g \) with a 2D signal \( f \) is as follows:

\[
o[m, n] = f[m, n] \ast g[m, n] \\
= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f[u, v]g[u - m, v - n].
\]

Moreover, the \( \max\{0, \cdot\} \) operator, called the rectified linear unit, provides the non-linearity. To form rich representations of a given dataset, each layer is composed of multiple feature maps so that each filter \( W^k \) forms a three-dimensional tensor for every combination of source feature map, vertical and horizontal size. A graphical illustration of the image module is provided in figure 3, which contains five convolution layers and two FC layers. Each input image \( \mathbf{X} \in \mathbb{R}^{d_1 \times d_2 \times 3} \) is represented as a 4096-dimensional vector through a series of nonlinear operations in both the training and testing phases. Once the set of parameters \( D_T \) is fixed, the feature of each individual input images is deterministic.

In contrast, since text documents are unstructured that do not contain a spatial information, fully connected layers are commonly used to extract application oriented feature on top of TF-IDF inputs. The feature transformation is expressed as follows:

\[
q_{D_T}(\mathbf{z}) = \max\{0, W_T \mathbf{z} + b_T\}
\]

(15)

This is performed through a single fully connected layer, where \( W_T \in \mathbb{R}^{r \times d_T} \) and \( b_T \in \mathbb{R}^r \) indicates the number of neurons in a given layer. Similarly, rich representations can be learned by stacking multiple fully connected layers with different number of neurons (\( r \) can be set to different values in different layers) to construct the deep text architecture for word documents.

Since the linear heterogeneous embedding in section 4.1 can be viewed as transforming inputs to a common space, we can achieve this by cascading an extra linear embedding layer to each deep module. Define

\[
\tilde{p}_{D'_T}(\mathbf{X}) = U^T p_{D_T}(\mathbf{X}), \quad \tilde{q}_{D'_T}(\mathbf{z}) = V^T q_{D_T}(\mathbf{z}),
\]

(16)

where \( D'_T = D_T \cup \{U\} \) and \( D'_T = D_T \cup \{V\} \). Then, the objective function in equation (12) is equivalent to the following:

\[
\min_{D'_T, D'_T} \frac{1}{|T|} \sum_{\mathbf{v}, \mathbf{v} \in V_T} L'(p_{D'_T}(\mathbf{X}_i), \tilde{p}_{D'_T}(\mathbf{X}_j)) \\
+ \frac{\lambda_1}{|T|} \sum_{\mathbf{v}, \mathbf{v} \in V_T} L'(\tilde{q}_{D'_T}(\mathbf{z}_i), \tilde{q}_{D'_T}(\mathbf{z}_j)) \\
+ \frac{\lambda_2}{|T|} \sum_{\mathbf{v}, \mathbf{v} \in V_T} L'(\tilde{p}_{D'_T}(\mathbf{X}_i), \tilde{q}_{D'_T}(\mathbf{z}_j)),
\]

(17)

The problem of over-fitting can be effectively prevented by using dropout [15] instead of \( L_2 \) regularizations. The new loss term \( L'('.,.\) \) is defined as

\[
L'(\mathbf{a}, \mathbf{b}) = \log \left(1 + \exp\left(-\mathbf{a}^T \mathbf{b}\right)\right).
\]

(18)

for any vector \( \mathbf{a}, \mathbf{b} \) with a same dimensionality. For simplicity, we refer to both deep image and text modules as a series of nonlinear feature transformations with an additional linear common space embedding.

To perform end-to-end HNE learning, we connect the deep image and text modules accordingly to the image-text, text-text, and image-text losses in equation (17). As an example, we illustrate the text-text module in figure 4, and the other two can be extended in a similar manner. Figure 4 contains two text modules that comprise the pairwise text-text module. The illustrated deep text-text module contains two FC layers followed by a linear embedding layer. A pair of text documents are fed from the left and computed in a left-to-right direction. The outputs from the embedding layer are the vectorized representation of corresponding objects in the common latent space. These are further channeled to a prediction layer to calculate the loss using equation (17). To make the text-text module symmetric (feeding the same objects from the top or the bottom pass of the text-text modules will lead to a same latent representation), we need to tighten these parameters. In figure 4, if two neurons have the same color, they share the same weight and bias.
4.3 Optimization

The objective function in equation (17) can be efficiently minimized by stochastic gradient descent (SGD) by sampling mini-batches from the training set. The advantage of training stochastically is that each mini-batch can be loaded onto GPU and computed in a parallel scheme if needed. Popular open-source deep learning packages using GPU-based implementations include Cuda-convnet [18], Caffe [17] and Theano [2], etc.. For each input image pair, the gradient of $D_I$ is given as follows:

$$
\frac{\partial(-)}{\partial D_I} = \frac{\partial(-)}{\partial \hat{p}_{D_I}(X_i)} \frac{\partial \hat{p}_{D_I}(X_i)}{\partial D_I} + \frac{\partial(-)}{\partial \hat{p}_{D_I}(X_j)} \frac{\partial \hat{p}_{D_I}(X_j)}{\partial D_I},
$$

where $c_{ij} = \frac{-A_{ij}}{1 + e^{A_{ij} \hat{p}_{D_I}(X_i) \hat{p}_{D_I}(X_j)}}$. It is worth mentioning that the summation from both $X_i$ and $X_j$ parts is because we tie the parameters of each image module within the image-image subnetwork (symmetric to pairwise inputs). Moreover, the gradient $\frac{\partial \hat{p}_{D_I}(X_i)}{\partial D_I}$ and $\frac{\partial \hat{p}_{D_I}(X_j)}{\partial D_I}$ are dependent only on the structure of the deep neural network. In other words, once the deep architecture has been fixed, their gradients are automatically defined. Furthermore, for each input text pair, the gradient is similar but changing the input and network parameters in equation (19) to that of the corresponding text case.

We can see that image-image inputs only contribute to learning discriminative representations for image modules. On the other hand, the cross-model inputs will affect the learning specific to both image and text. Their gradients are shown respectively as follows:

$$
\frac{\partial(-)}{\partial D_I} = \frac{-A_{ij} \hat{p}_{D_I}(X_i) \hat{p}_{D_I}(X_j)}{1 + e^{A_{ij} \hat{p}_{D_I}(X_i) \hat{p}_{D_I}(X_j)}} \frac{\partial \hat{p}_{D_I}(X_i)}{\partial D_I},
$$

and

$$
\frac{\partial(-)}{\partial D_T} = \frac{-A_{ij} \hat{p}_{D_T}(X_i) \hat{p}_{D_T}(X_j)}{1 + e^{A_{ij} \hat{p}_{D_T}(X_i) \hat{p}_{D_T}(X_j)}} \frac{\partial \hat{p}_{D_T}(X_i)}{\partial D_T}.
$$

4.4 Discussion

The trained deep neural network assigns different types of data to some points in a unified space so that similarities can be directly compared. So far, we have shown the proposed embedding scheme for heterogeneous networks with only two object types: text and images. While these two types are a natural representative in many real settings, it is conceivable to expect more than two types of inputs. The proposed methods can be easily extended to handle multiple input types by considering an individual deep module for each type of data. Then, the objective function in equation (17) will consider all possible pairs of input types. If there are $|\mathcal{C}|$ input types, the new objective will contain $|\mathcal{C}| + (|\mathcal{C}|^2)$ object types.

Because deep learning is highly nonlinear and non-convex, globally optimal convergence is not assured. The initialization of parameters is crucial to the final performance. The literature has shown that well-designed pre-training can significantly improve final performances even when the final task is different from the pre-training task. It is worth mentioning that the proposed embedding method is unsupervised and can be used as a pre-training step for any further fine-tuning. In other words, if we want to classify network.
nodes, we can either obtain final features from the embedding layer and apply off-the-shelf machine learning algorithms or we can replace the prediction layer to a soft-max layer, and then fine-tune the entire deep network to a task-specific one.

5. EXPERIMENTAL RESULTS

In this section, we evaluate our proposed algorithm on several real-world datasets for both homogeneous and heterogeneous settings. The experimental results show evidence of significant improvement over many conventional baselines.

5.1 Datasets and Experiment Settings

We use two publicly available datasets from real-world social sites. The first one is BlogCatalog which is used in [42] to select features in linked social media data. The second one is a heterogeneous dataset, which is referred to as NUS-WIDE [5]. This dataset contains both images and text. All experiment results are averaged over five different runs. The detailed descriptions and statistics for both datasets are provided below.

- **BlogCatalog** [42]: It is a social blogging site where registered users are able to categorize their blogs under predefined classes. Such categorizations are used to define class labels, whereas “following” behaviors are used to construct linkages between users. The TF-IDF features are extracted from blogs as a vector representation of each individual user. Thus, blog users are represented as different nodes of the constructed networks associated with content features. It is worth mentioning that, the user blogging networks is undirected, where the co-following and co-followed relationships are the same. Some detailed statistics are summarized in Table 1.

- **NUS-WIDE** [5]: The dataset was originally collected by the Lab for Media Search in the National University of Singapore in the year 2009. The dataset includes 269,648 unique images with associate tags from Flickr. The total number of tags is 5,018. Additionally, there are 81 ground-truth attribute labels on each image and tag. Since the original dataset injected many “noise” samples that did not originally belong to any of the 81 concepts, these samples were removed. Moreover, we used the most frequent 1,000 tags as text documents and extracted their TF-IDF features. We further removed those image-text pairs that did not contain any considered words. Finally, we randomly sampled 53,844 and 36,352 image-text pairs for training and testing, respectively. We constructed a heterogeneous network as the input of our proposed framework by treating images and text as separate nodes. In total, the training network contained 107,688 nodes while the testing network had 72,704. The semantic linkages between two nodes are initially constructed if they share at least one concept. We then random sample at most 30 links per node to construct the sparse matrix $A$. It is worth mentioning that we only evaluate our framework in an out-of-sample manner. In other words, we ensure the training information absolutely does not appear in any of the testing cases.

5.2 Network Reconstructions

Before proceeding to evaluate the proposed method in the task of classification, clustering or retrieval, we will first provide a basic and intuitive evaluation of the quality of network linkage reconstruction to validate our assumptions. Since the goal of the proposed formulation in equation (17) is that a good latent embedding brings objects with links closer while it pushes objects without linkage structures further, the ideal performance of the learned model can reach perfect network linkage reconstructions using equation (4). We first visualize the network linkage structure of the BlogCatalog dataset by randomly selecting 500 nodes and plotting their connectivities in figure 6. The color of each node indicates its class. As we can see, the social “following” relationships tend to connect users with similar attributes, at least from a relative point of view. On the other hand, they are noisy from an absolute point of view, in which 59.89% of links in the entire dataset connect to nodes with different classes.

We apply the proposed algorithm to learn an embedding function while monitoring the link reconstruction accuracy as shown in figure 7. The stochastic learning is conducted by randomly selecting 128 pairs of nodes to use as a mini-batch. The horizontal axis indicates the index of the epoch. And each epoch contains 500 mini-batches. On average, each mini-batch can be trained in less than 0.15 seconds on a single Nvidia Tesla K40 GPU. In figure 7, the reconstruction performance on each mini-batch is recorded, and the line indicates the median filtered values. As more samples have been viewed by the deep HNE learner, it is able to correctly reconstruct more than 80% of the pairwise connections as compared to the initial number of 55%. Similarities propagate through sparse links across the whole network to obtain a global consistency.

5.3 BlogCatalog

In this section, we evaluate the performance of the HNE framework and compare it with state-of-the-art algorithms in various tasks in the field of data mining and Web search.

5.3.1 Classification

To demonstrate the effectiveness of the representation provided by HNE, we compare our learned features with those of other
feature learning methods, while keeping the classification scheme fixed. The other baseline representations are as follows:

- **Content**: Only the content feature from the original space.
- **Links**: We treat the adjacency structures as the features.
- **Link-content**: We combine features from the previous two.
- **LUFS** [42]: Unsupervised feature selection framework for linked only social media data considering both content and links.
- **LCMF** [50]: A matrix co-factorization method that utilizes both linkage structure and content features.

To ensure a fair comparison, we used the same representation dimensionality and used the standard Nearest Neighbor (NN) classifier. In other words, the number of latent factors for LCMF is set to be the same as our output dimensionality and the first three methods are projected to a low-dimensional space using the Principal Component Analysis.

The average classification accuracies for the BlogCatalog dataset are shown in figure 8, with the output dimensionality fixed to 100. As shown, the proposed HNE method consistently outperforms to the other baselines under different training set sizes. This is because the network linkage information encodes useful insights for learning a low-dimensional embedding space by bridging linked nodes.

The rightmost bar under each setting is achieved by treating latent embedding learning as a pre-training step and fine-tuning the entire deep network by replacing the loss layer with a multi-class soft-max layer. It shows that the unsupervised latent space learning provides very good initializations for the supervised classification task using deep architectures and also shows that we can also achieve much higher accuracies.

### 5.3.2 Clustering

We also compared different feature representations under the clustering task. Compared to classification, clustering is totally unsupervised, and it heavily relies on the similarity measure between different objects. We adopted the commonly used cosine similarity. The results are reported in table 2 using both accuracy and normalized mutual information (NMI) as evaluation metrics.

The results are similar to those for the classification task. Using only links provides the worst results. This may be because, without global content information, the similarity measurements tend to be local and sensitive to noisy links. On the other hand, content similarities alone are insufficient to capture the relational knowledge. Therefore, a naive combination of the links and content provides comparable performance with other baselines. The proposed method of jointly learning the embedded space outperforms other baselines and achieves the state-of-the-art.

### 5.4 NUS-WIDE

Compared to the BlogCatalog dataset, the NUS-WIDE dataset forms a heterogeneous network that contains both images and text. We illustrate the performance of our framework for the task of classification and cross-modal retrieval in the following subsections. Note that the latter application is not possible in the homogeneous scenario of the previous dataset.

#### 5.4.1 Heterogeneous Classifications

Given the heterogeneous scenario of this dataset, we compared our proposed method to a different set of unsupervised baselines that can specifically handle multimodal data inputs:

- **CCA**: The Canonical Correlation Analysis embeds two types of input sources into a common latent space by optimizing with respect to their correlations.
- **DT** [33]: A transfer learning method is used to bridge semantic distances between image and text by latent embeddings.
- **LHNE**: The linear version of HNE solves the optimization function in equation (7).

Since our proposed method is an end-to-end learning framework, it does not require feature extraction for image inputs. We extract 4096-dimensional Cuda-convnet [18] features for all other baseline methods. The output (data in the common space) dimensionality is set to 400. Since the NUS-WIDE dataset is multi-label with unbalanced classes, we use the average precision (AP) to evaluate the classification performance for each possible label outcome. AP uses precision-recall curves for algorithmic quantification for each label. These curves are used to obtain the mean average precision (mAP). The mAP in multi-label classification domains is the standard metric which is widely used in PASCAL challenges [8] in computer vision communities. To ensure fair comparison, we use linear support vector machines (SVM) as a common classification algorithm for all algorithms. The reason of using SVM is that calculating AP requires probabilistic interpreted confidence scores, which is inconvenient to obtain from NN classifiers.

The classification results are illustrated in table 4, which contains three different settings. The “image only” setting means that we learn embedding functions from the heterogeneous training set, and then train an SVM, and test classification performance on image nodes. Under the “image + text” setting, we consider all objects in the testing network. We observe that, for all methods, categorizing text documents only is the most difficult task. This may be because of the fact that the input text is sparse compared to images. Moreover, without the deep training, the linear version of our proposed method obtains comparable results as DT which outperforms CCA. The deep architecture HNE improves the performance further under all three different settings, which demonstrates the advantage of jointly optimizing the feature learning and latent embedding with nonlinear functions.

#### 5.4.2 Multimodal Search

To further demonstrate that the learned features can be leveraged with many data mining and web search tasks, we compared our proposed method with the aforementioned baselines in the task of...
Table 3: Some cross-model retrieval results of the proposed HNE method.

<table>
<thead>
<tr>
<th>Query</th>
<th>rank 1</th>
<th>rank 2</th>
<th>rank 3</th>
<th>rank 4</th>
<th>rank 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mountain</td>
<td><img src="mountain1.jpg" alt="Image" /></td>
<td><img src="mountain2.jpg" alt="Image" /></td>
<td><img src="mountain3.jpg" alt="Image" /></td>
<td><img src="mountain4.jpg" alt="Image" /></td>
<td><img src="mountain5.jpg" alt="Image" /></td>
</tr>
<tr>
<td>Sunset</td>
<td><img src="sunset1.jpg" alt="Image" /></td>
<td><img src="sunset2.jpg" alt="Image" /></td>
<td><img src="sunset3.jpg" alt="Image" /></td>
<td><img src="sunset4.jpg" alt="Image" /></td>
<td><img src="sunset5.jpg" alt="Image" /></td>
</tr>
<tr>
<td>Cow</td>
<td><img src="cow1.jpg" alt="Image" /></td>
<td><img src="cow2.jpg" alt="Image" /></td>
<td><img src="cow3.jpg" alt="Image" /></td>
<td><img src="cow4.jpg" alt="Image" /></td>
<td><img src="cow5.jpg" alt="Image" /></td>
</tr>
<tr>
<td>Leaf</td>
<td><img src="leaf1.jpg" alt="Image" /></td>
<td><img src="leaf2.jpg" alt="Image" /></td>
<td><img src="leaf3.jpg" alt="Image" /></td>
<td><img src="leaf4.jpg" alt="Image" /></td>
<td><img src="leaf5.jpg" alt="Image" /></td>
</tr>
</tbody>
</table>

Table 4: The classification result in terms of mAP (mean average precision) for the NUS-WIDE dataset.

<table>
<thead>
<tr>
<th>Sample</th>
<th>CCA</th>
<th>DT</th>
<th>LHNE</th>
<th>HNE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image only</td>
<td>51.96%</td>
<td>52.07%</td>
<td>53.16%</td>
<td>54.28%</td>
</tr>
<tr>
<td>Text only</td>
<td>51.37%</td>
<td>51.88%</td>
<td>51.34%</td>
<td>52.76%</td>
</tr>
<tr>
<td>Image + Text</td>
<td>52.54%</td>
<td>53.22%</td>
<td>53.32%</td>
<td>54.99%</td>
</tr>
</tbody>
</table>

Table 5: The cross-modal retrieval result (p@k) for the NUS-WIDE dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>rank 1</th>
<th>rank 5</th>
<th>rank 10</th>
<th>rank 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCA</td>
<td>21.05%</td>
<td>16.84%</td>
<td>18.95%</td>
<td>18.68%</td>
</tr>
<tr>
<td>DT</td>
<td>20.53%</td>
<td>25.26%</td>
<td>22.63%</td>
<td>22.37%</td>
</tr>
<tr>
<td>LHNE</td>
<td>26.32%</td>
<td>21.05%</td>
<td>21.02%</td>
<td>22.27%</td>
</tr>
<tr>
<td>HNE</td>
<td>36.84%</td>
<td>29.47%</td>
<td>27.89%</td>
<td>26.32%</td>
</tr>
</tbody>
</table>

cross-modal retrieval. Among all 81 labels, about 75 of them appear in the TF-IDF text vector. We manually constructed 75 query vectors in the original 1000-dimensional text domain by setting the corresponding label entries to one and the remaining to zero. Using the learned embedding function, we projected these query vectors to the common latent space to retrieve all image samples in the test set using the standard Euclidean distance.

The average precision at rank \( k \) (p@k) over all queries is reported in table 5. We observe consistent results as other tasks, and the proposed method significantly outperforms other baselines. Table 3 illustrates some sample retrieval results. For the query “mountain”, the third retrieved result is incorrect. This might due to the extreme visual similarities between the other mountain images and the one with a cow. The retrieval results for the query “cow” is not as good as the others. The first five returned images contain three deer. This is because these images have multiple labels and are connected by the concept “animal”. Since our method as well as the ranking functions are totally unsupervised, these links between “deer” and “cow” objects confuse our embedding learning. We expect performance gains by using supervised ranking methods.

5.5 Convergence

In this section, we examine the convergence of the HNE algorithm. For illustrative purposes, we demonstrate the value of objective function changes for the BlogCatalog dataset in figure 9. The X-axis is same as the one used in figure 7. As shown in the graph, the objective value continuously decreases in the first 60 epochs and then stabilizes. This result shows that the algorithm usually converges to a stable result in practice.

6. CONCLUSION

In this paper, we proposed a novel embedding scheme in the field of network science. This approach transfers different objects in heterogeneous networks to unified vector representations. The proposed method jointly considers contents and topological structures in networks for creating the embedding. We use deep learning techniques to capture the complex interactions between heterogeneous components. Such a highly nonlinear multi-layered embedding architecture is robust, scalable and beneficial to many data mining and Web search applications. Furthermore, the approach has generic applicability because a robust feature representation is useful in many tasks. The experimental studies show that the proposed method significantly outperforms conventional baselines in various settings.

Acknowledgment

This work was founded in part to Shiyu Chang, Wei Han and Thomas S. Huang by the National Science Foundation under Grand No.