Heterogeneous Network Embedding via Deep Architectures

Shiyu Chang[1], Wei Han[1], Jiliang Tang[2],
Guo-Jun Qi[3], Charu C. Aggarwal[4], 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.

KDD '15, August 10-13, 2015, Sydney, NSW, Australia.

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

Figure 1: Illustration of the heterogeneity of different data sources describing the same topic "MH 17".

"Malaysia Airlines MH 17" illustrates in Figure 1, relevant results include not only text documents but also images and videos.

Moreover, social media data does not exist in isolation, but in combination with various data types [28]. These interactions can be formed either explicitly or implicitly with the linkages between them. Images and texts co-occurring within the same Web page provide explicit linkages between them, whereas text-to-text linkages are formed by the hyper-links between different Web documents. On the other hand, interactive activities by users can be viewed as implicit feedback, which connect different social media components. If users describe multiple images with similar tags, it is reasonable to assume that a semantic relationship exists between such images. It is evident that such huge amounts of data results in complex heterogeneous networks that impose tremendous challenges on learning uniform representations.

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 heterogeneous objects into a unified latent space so that objects from different spaces can be directly compared.

Unlike to traditional linear embedding models [29, 44, 50], the proposed method decomposes the feature learning process into multiple nonlinear layers of a deep architecture. Both Deep Neural Networks (DNN) [1, 14] and Convolutional Neural Networks (CNN) [12, 18] are aggregated to handle vectorized data (e.g., text documents) or tensor-based multimedia objects (e.g., color images and videos). Our model coordinates two mutually reinforcing parts by iteratively solving an optimization problem with respect to feature learning and objective minimization. The deep architecture models both the local and global linkage structures of underlying networks through the proposed framework. This makes it more powerful than a shallow embedding solution to capture the network links, especially when the linkage information plays the key role in revealing semantic correlations between different objects due to the homophily property. Along this line of research, we utilize the network's inter-connections to design a proper loss function that enforces the similarities between different objects in the embedded feature space such that they are consistent with network links. The idea of a network-preserved embedding is illustrated in figure 2. The key advantages of the proposed HNE framework are the following:

- *Robust*: HNE explores global consistency between different heterogeneous objects to learn unified feature representations guided by network structures.
- *Unsupervised*: HNE is unsupervised and task independent, which makes it suitable for many network orientated data mining applications.
- *Out-of-sample*: HNE is able to handle the out-of-sample problem. This addresses the challenge associated with a dynamic network in which new nodes may be added over time.

2. RELATED WORK

2.1 Network Embedding

A branch of latent feature embeddings is motivated by applications such as collaborative filtering and link prediction in networks that model the relations between entities from latent attributes [16]. These models often transfer the problem as learning an embedding of the entities, which corresponds algebraically to a matrix factorization problem of observed relationships. Zhu *et. al.* [50] proposed a joint factorization approach on both the linkage adjacency matrix and document-term frequency for Web page categorization. Similar concepts also include [29, 44]. In addition, Shaw *et.al.* [38] proposed a structure preserving embedding framwork that embeds graphs to a low-dimensional Euclidean space with global topological properties preserved. Moreover, DeepWalk [31] learns latent representations of vertices in a network from truncated random walks. However, these models focus only on single relations that do not adapt to heterogeneous settings and most of them are hardly to generate to other unseen samples.

A natural extension of these methods to heterogeneous settings is by stacking multiple relational matrices together, and then applying a conventional tensor factorization [26, 29, 39]. The disadvantage of such multi-relational embeddings is the inherent sharing of parameters between different terms, which does not 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 confluence 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

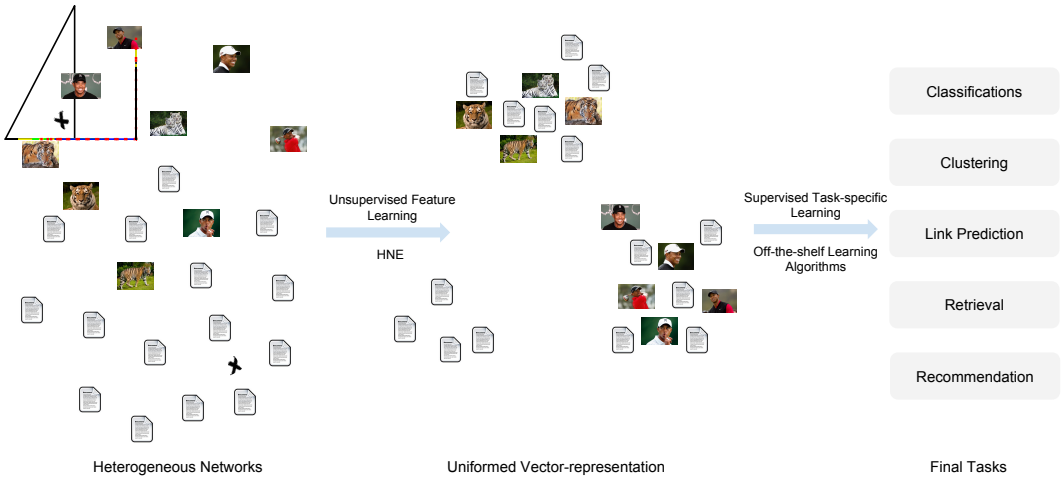


Figure 2: The flowchart of the proposed Heterogeneous Network Embedding (HNE) framework.

Scale Visual Recognition Challenge [36], one of the most challenging tasks in computer vision. Later on, features from one of network's intermediate layers proved to be a superior feature representation for other vision tasks, such as object detection [11].

There has been a growing interest in leveraging data from multiple modalities within the context of deep learning architectures. Unsupervised deep learning methods such as auto-encoders [25] and RBMs [40] are deployed to perform feature learning by joint reconstruction of audio and video with a partially shared network, and successfully applied to several multimodal tasks. Alternatively, there has also been effort on learning the joint representation with multiple tasks [6]. For image and text, a particular useful scenario is zero-shot learning of image classification on unseen labels achieved by incorporating the semantically meaningful embedding space to image labels [9, 27].

To the best of our knowledge, there have been few previous attempts to make use of the linkage structure in a network for representation learning. In [19] a conditional temporary RBM was used to model the dynamics of network links, and the main task was to predict future links with historical data. The most similar work to ours is that of [46]. A content RBM was trained on multimedia link data. We differ from their work in several respects. We use a completely supervised learning scheme in an unsupervised setting; our method can scale up to large scale datasets; we perform multi-task learning to fuse the information from different modalities.

3. PRELIMINARIES

In this section, we introduce our notation as well as the mathematical definitions we will use to describe heterogeneous networks.

3.1 Notation

Throughout this paper, all vectors are column vectors and are denoted by bold lower case letters (e.g., $\boldsymbol{x}$ and $\boldsymbol{y}$), while matrices are represented by bold upper case letters (e.g., $\boldsymbol{X}$ and $\boldsymbol{M}$). The i^{th} row of a matrix $\boldsymbol{X}$ is given by $\boldsymbol{X}_{i\cdot}$, while $\boldsymbol{X}_{\cdot j}$ represents the j^{th} column. We use calligraphic letters to represent sets (e.g., $\mathcal{V}$ and $\mathcal{E}$). The notation $|\cdot|$ denotes the cardinality of a given set. The empty set is denoted by ϕ.

3.2 Heterogeneous Networks

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 $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is a set of vertices and $\mathcal{E}$ is a set of edges. An edge $e_{ij},\ \forall i, j \in \{1, \cdots, n\}$ belongs to the set $\mathcal{E}$ 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 $f_v : \mathcal{V} \rightarrow \mathcal{O}$ and a link type mapping function $f_e : \mathcal{E} \rightarrow \mathcal{R}$, 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 $f_v(v_i) \in \mathcal{O}$. Similarly, each link $e_{ij} \in \mathcal{E}$ is categorized in different relations as $f_v(e_{ij}) \in \mathcal{R}$. 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 $\mathcal{E}_{II}$, $\mathcal{E}_{TT}$ and $\mathcal{E}_{IT}$, respectively. Furthermore, each node is summarized by unique content information. In particular, images are given as a squared tensor format as $\boldsymbol{X}_i \in \mathbb{R}^{d_I \times d_I \times 3}$ for every $v_i \in \mathcal{V}_I$, while texts are represented by a d_T-dimensional feature vector as $\boldsymbol{z}_j \in \mathbb{R}^{d_T}$ for all $\boldsymbol{v}_j \in \mathcal{V}_T$. 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 $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, which the (i, j)th entry of $\boldsymbol{A}$ equals one if $e_{ij} \in \mathcal{E}$; otherwise $\boldsymbol{A}_{ij} = -1$ (for model simplicity).

4. HETEROGENEOUS NETWORK EMBEDDING

In this section, we present our HNE framework mathematically by first introducing a novel loss function to measure correlations

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 $\boldsymbol{X_i}$ associated with an image node can be transformed to a d'_I-dimensional vector representation as $\boldsymbol{x_i}$. 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 $\boldsymbol{U} \in \mathbb{R}^{d'_I \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{d_T \times r}$, for the image and text domains, respectively. The transformed samples are denoted by $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{z}}$ for images and text documents, respectively, where we have:

$$\tilde{\boldsymbol{x}} = \boldsymbol{U}^T \boldsymbol{x}, \text{ and } \tilde{\boldsymbol{z}} = \boldsymbol{V}^T \boldsymbol{z}. \tag{1}$$

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:

$$\begin{aligned} s(\boldsymbol{x_i}, \boldsymbol{x_j}) &= \tilde{\boldsymbol{x}_i}^T \tilde{\boldsymbol{x}_j} = (\boldsymbol{U}^T \boldsymbol{x}_i)^T \boldsymbol{U}^T \boldsymbol{x}_j = \boldsymbol{x_i}^T \boldsymbol{M}_{II} \boldsymbol{x_j}, \\ s(\boldsymbol{z_i}, \boldsymbol{z_j}) &= \tilde{\boldsymbol{z}_i}^T \tilde{\boldsymbol{z}_j} = (\boldsymbol{V}^T \boldsymbol{z}_i)^T \boldsymbol{V}^T \boldsymbol{z}_j = \boldsymbol{z_i}^T \boldsymbol{M}_{TT} \boldsymbol{z_j}, \end{aligned} \tag{2}$$

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:

$$\begin{aligned} s(\boldsymbol{x_i}, \boldsymbol{z_j}) &= \tilde{\boldsymbol{x}_i}^T \tilde{\boldsymbol{z}_j} = (\boldsymbol{U}^T \boldsymbol{x}_i)^T \boldsymbol{V}^T \boldsymbol{z}_j = \boldsymbol{x}_i^T \boldsymbol{M}_{IT} \boldsymbol{z}_j \\ &= \tilde{\boldsymbol{z}_j}^T \tilde{\boldsymbol{x}_i} = (\boldsymbol{V}^T \boldsymbol{z}_j)^T \boldsymbol{U}^T \boldsymbol{x}_i = \boldsymbol{z}_j^T \boldsymbol{M}_{IT}^T \boldsymbol{x}_i, \end{aligned} \tag{3}$$

Here, $\boldsymbol{M}_{II} \in \mathbb{R}^{d'_I \times d'_I}$ and $\boldsymbol{M}_{TT} \in \mathbb{R}^{d_T \times d_T}$ are positive semi-definite matrices while $\boldsymbol{M}_{IT} \in \mathbb{R}^{d'_I \times d_T}$. 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 between two nodes in a network can be either parameterized by the projection matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ or through a bilinear function defined by the matrices $\boldsymbol{M}_{II}$, $\boldsymbol{M}_{TT}$ and $\boldsymbol{M}_{IT}$. This provides the flexibility to model the heterogeneous relationship in an application-specific way.

The heterogeneous objects interact with each other either explicitly and implicitly. These interactive pieces of information are represented as heterogeneous linkages in networks. The assumption is that if two objects are connected, the similarity measure between them should reflect this fact by providing a larger value compared to the ones that are isolated. Consider a pair of images denoted as $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. To encode the link information, we design a pairwise decision function $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as follows:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) \begin{cases} > 0 \text{ if } \boldsymbol{A}_{ij} = 1, \\ < 0 \text{ otherwise.} \end{cases} \tag{4}$$

Note that to infer d we need not know the respective entry value of $\boldsymbol{A}$, 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

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = s(\boldsymbol{x}_i, \boldsymbol{x}_j) - t_{II}, \tag{5}$$

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

$$L(\boldsymbol{x}_i, \boldsymbol{x}_j) = \log\left(1 + \exp\left(-\boldsymbol{A}_{i,j} d(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)\right), \tag{6}$$

which can be seen as a binary logistic regression guided by network linkages. The loss function of text-text and image-text are similar to equation (6) by simply replacing $s(\cdot)$ with that of the corresponding modality. Similarly, the bias terms, denoted by t_{TT} and t_{IT}, can be set to the corresponding ones. It leads to our objective functions in the form of:

$$\begin{aligned} \min_{\boldsymbol{U},\boldsymbol{V}} \; & \frac{1}{N_{II}} \sum_{v_i, v_j \in \mathcal{V}_I} L(\boldsymbol{x}_i, \boldsymbol{x}_j) + \frac{\lambda_1}{N_{TT}} \sum_{v_i, v_j \in \mathcal{V}_T} L(\boldsymbol{z}_i, \boldsymbol{z}_j) \\ & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} L(\boldsymbol{x}_i, \boldsymbol{z}_j) + \lambda_3 (\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2), \end{aligned} \tag{7}$$

where N_{II}, N_{TT} and N_{IT} are the numbers of the three types of links in the network. Furthermore, λ_1, λ_2 and λ_3 are the three balancing parameters, in which the first two control the emphasis among three types of linkages 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 $\boldsymbol{U}$:

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

$$\begin{aligned} \min_{\boldsymbol{U}} \; & \frac{1}{N_{II}} \sum_{v_i, v_j \in \mathcal{V}_I} \log\left(1 + e^{-\boldsymbol{A}_{i,j} \boldsymbol{x}_i^T \boldsymbol{U}\boldsymbol{U}^T \boldsymbol{x}_j}\right) + \lambda_3 \|\boldsymbol{U}\|_F^2 \\ & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} \log\left(1 + e^{-\boldsymbol{A}_{i,j} \boldsymbol{x}_i^T \boldsymbol{U}\boldsymbol{V}^T \boldsymbol{z}_j}\right) \end{aligned} \tag{8}$$

The gradient is given by the following:

$$\begin{aligned} \frac{\partial(\cdot)}{\partial \boldsymbol{U}} = & \frac{1}{N_{II}} \sum_{v_i, v_j \in \mathcal{V}_I} \frac{-\boldsymbol{A}_{i,j}(\boldsymbol{x}_j \boldsymbol{x}_i^T + \boldsymbol{x}_i \boldsymbol{x}_j^T)\boldsymbol{U}}{1 + e^{\boldsymbol{A}_{i,j} \boldsymbol{x}_i^T \boldsymbol{U}\boldsymbol{U}^T \boldsymbol{x}_j}} + 2\lambda_3 \boldsymbol{U} \\ & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} \frac{-\boldsymbol{A}_{i,j} \boldsymbol{x}_i \boldsymbol{z}_j^T \boldsymbol{V}}{1 + e^{\boldsymbol{A}_{i,j} \boldsymbol{x}_i^T \boldsymbol{U}\boldsymbol{V}^T \boldsymbol{z}_j}}. \end{aligned} \tag{9}$$

Solving $\boldsymbol{V}$:

Similarly, the variable $\boldsymbol{V}$ can be handled as follows:

$$\begin{aligned} \min_{\boldsymbol{V}} \; & \frac{\lambda_1}{N_{TT}} \sum_{v_i, v_j \in \mathcal{V}_T} \log\left(1 + e^{-\boldsymbol{A}_{i,j} \boldsymbol{z}_i^T \boldsymbol{V}\boldsymbol{V}^T \boldsymbol{z}_j}\right) + \lambda_3 \|\boldsymbol{V}\|_F^2 \\ & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} \log\left(1 + e^{-\boldsymbol{A}_{i,j} \boldsymbol{x}_i^T \boldsymbol{U}\boldsymbol{V}^T \boldsymbol{z}_j}\right). \end{aligned} \tag{10}$$

Taking the derivative with respect to $\boldsymbol{V}$, we obtain:

$$\begin{aligned} \frac{\partial(\cdot)}{\partial \boldsymbol{V}} = & \frac{\lambda_1}{N_{TT}} \sum_{v_i, v_j \in \mathcal{V}_T} \frac{-\boldsymbol{A}_{i,j}(\boldsymbol{z}_j \boldsymbol{z}_i^T + \boldsymbol{z}_i \boldsymbol{z}_j^T)\boldsymbol{V}}{1 + e^{\boldsymbol{A}_{i,j} \boldsymbol{z}_i^T \boldsymbol{V}\boldsymbol{V}^T \boldsymbol{z}_j}} + 2\lambda_3 \boldsymbol{V} \\ & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} \frac{-\boldsymbol{A}_{i,j} \boldsymbol{z}_j \boldsymbol{x}_i^T \boldsymbol{U}}{1 + e^{\boldsymbol{A}_{i,j} \boldsymbol{x}_i^T \boldsymbol{U}\boldsymbol{V}^T \boldsymbol{z}_j}}. \end{aligned} \tag{11}$$

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 *i*ntegrate these two steps into a deep learning framework by learning the feature representation and embedding together.

$$\begin{aligned}\min_{\boldsymbol{U},\boldsymbol{V},\mathcal{D}_I,\mathcal{D}_T} & \frac{1}{N_{II}} \sum_{v_i,v_j \in \mathcal{V}_I} L(p_{\mathcal{D}_I}(\boldsymbol{X}_i), p_{\mathcal{D}_I}(\boldsymbol{X}_j)) + \lambda_3(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2) \\ & + \frac{\lambda_1}{N_{TT}} \sum_{v_i,v_j \in \mathcal{V}_T} L(q_{\mathcal{D}_T}(\boldsymbol{z}_i), q_{\mathcal{D}_T}(\boldsymbol{z}_j)) \\ & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} L(p_{\mathcal{D}_I}(\boldsymbol{X}_i), q_{\mathcal{D}_T}(\boldsymbol{z}_j)),\end{aligned} \tag{12}$$

Here, $p(\cdot)$ and $q(\cdot)$ are two nonlinear functions parameterized by $\mathcal{D}_I$ and $\mathcal{D}_T$. $\mathcal{D}_I$ and $\mathcal{D}_T$ are two sets of parameters associated with the deep image and text networks, respectively. Specifically, we utilizes 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 $\boldsymbol{W}_I^k$ and bias b_I^k). 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 $\boldsymbol{h}^k$, which is determined by the corresponding weights $\boldsymbol{W}^k$ and bias b^k. Then, the feature map is obtained as follows:

$$\boldsymbol{h}^k = \max\{0, (\boldsymbol{W}^k * \boldsymbol{M}) + b^k\}, \tag{13}$$

Here, $*$ denotes the convolution operation and $\boldsymbol{M}$ 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:

$$\begin{aligned} o[m,n] &= f[m,n] * g[m,n] \\ &= \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f[u,v] g[u-m, v-n]. \end{aligned} \tag{14}$$

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 $\boldsymbol{W}^k$ 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 $\boldsymbol{X} \in \mathbb{R}^{d_I \times d_i \times 3}$ 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 $\mathcal{D}_I$ 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

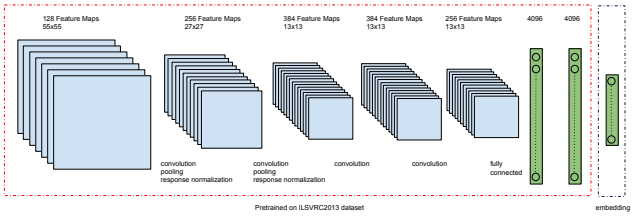


Figure 3: An example of the deep image module which consists of five convolution layers and two fully connected layers.

used to extract application orientated feature on top of TF-IDF inputs. The feature transformation is expressed as follows:

$$q_{\mathcal{D}_T}(\boldsymbol{z}) = \max\{0, \boldsymbol{W}_T \boldsymbol{z} + b_T\} \tag{15}$$

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

$$\tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}) = \boldsymbol{U}^T p_{\mathcal{D}_I}(\boldsymbol{X}), \text{ and } \tilde{q}_{\mathcal{D}'_T}(\boldsymbol{z}) = \boldsymbol{V}^T p_{\mathcal{D}_T}(\boldsymbol{z}), \tag{16}$$

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

$$\begin{aligned}\min_{\mathcal{D}'_I,\mathcal{D}'_T} & \frac{1}{N_{II}} \sum_{v_i,v_j \in \mathcal{V}_I} L'(\tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_i), \tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_j)) \\ & + \frac{\lambda_1}{N_{TT}} \sum_{v_i,v_j \in \mathcal{V}_T} L'(\tilde{q}_{\mathcal{D}'_T}(\boldsymbol{z}_i), \tilde{q}_{\mathcal{D}'_T}(\boldsymbol{z}_j)) \\ & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} L'(\tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_i), \tilde{q}_{\mathcal{D}'_T}(\boldsymbol{z}_j)),\end{aligned} \tag{17}$$

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

$$L'(\boldsymbol{a}, \boldsymbol{b}) = \log\left(1 + \exp\left(-\boldsymbol{A}_{i,j} \boldsymbol{a}^T \boldsymbol{b}\right)\right), \tag{18}$$

for any vector $\boldsymbol{a}, \boldsymbol{b}$ 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-image, 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.

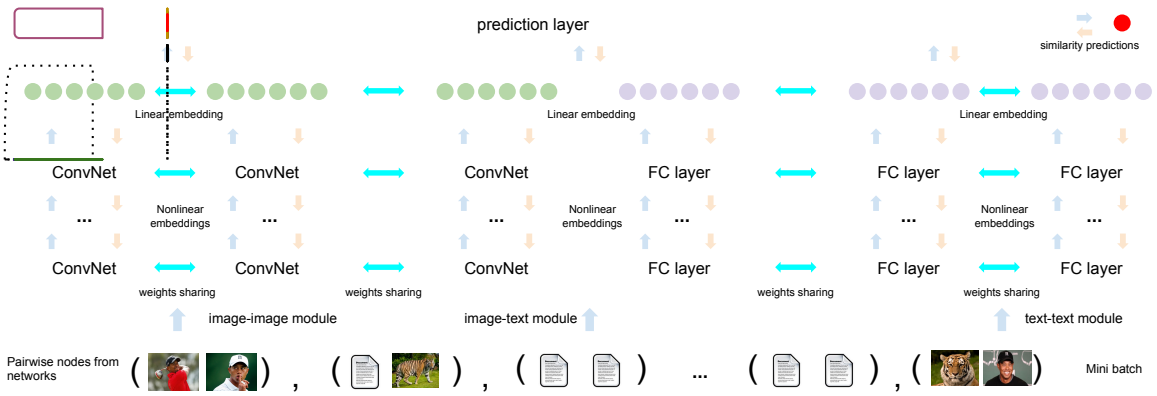


Figure 5: The overall architecture of HNE . The same color indicates the shared weights. The arrows are directions of forward feeding and back propagation.

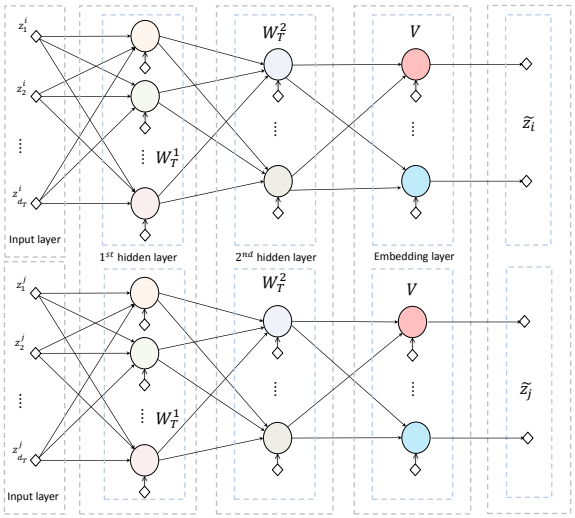


Figure 4: An example of the deep text-text module by concatenating a pair of text modules. Same coloring indicates shared weights.

The overall architecture of learning such a heterogeneous embedding function from a given network is visualized in figure 5. Three modules are shown in the figure, corresponding to image-image, image-text and text-text from left to right. These are connected to the prediction layer. Pairwise training samples are formed as mini-batches feeding from the bottom to the top. Once the value of the loss has been obtained, the gradients of each parameter in the deep network are calculated using backpropagation techniques.

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 $\mathcal{D}'_I$ is given as follows:

$$\begin{aligned}\frac{\partial(\cdot)}{\partial \mathcal{D}'_I} &= \frac{\partial(\cdot)}{\partial \tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_i)}\frac{\partial \tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_i)}{\partial \mathcal{D}'_I} + \frac{\partial(\cdot)}{\partial \tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_j)}\frac{\partial \tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_j)}{\partial \mathcal{D}'_I} \\ &= c_{ij}\cdot\left(\tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_i)\frac{\partial \tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_j)}{\partial \mathcal{D}'_I} + \tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_j)\frac{\partial \tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_i)}{\partial \mathcal{D}'_I}\right),\end{aligned} \tag{19}$$

where $c_{ij} = \frac{-\boldsymbol{A}_{i,j}}{1+e^{\boldsymbol{A}_{i,j}\tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_i)^T\tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_i)}}$. It is worth mentioning that the summation from both $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ parts is because we tie the parameters of each image module within the image-image sub-network (symmetric to pairwise inputs). Moreover, the gradient $\frac{\partial \tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_i)}{\partial \mathcal{D}'_I}$ and $\frac{\partial \tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_j)}{\partial \mathcal{D}'_I}$ 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:

$$\frac{\partial(\cdot)}{\partial \mathcal{D}'_I} = \frac{-\boldsymbol{A}_{i,j}\tilde{p}_{\mathcal{D}'_T}(\boldsymbol{z}_j)}{1+e^{\boldsymbol{A}_{i,j}\tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_i)^T\tilde{p}_{\mathcal{D}'_T}(\boldsymbol{z}_j)}}\cdot\frac{\partial \tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_i)}{\partial \mathcal{D}'_I}, \tag{20}$$

and

$$\frac{\partial(\cdot)}{\partial \mathcal{D}'_T} = \frac{-\boldsymbol{A}_{i,j}\tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_i)}{1+e^{\boldsymbol{A}_{i,j}\tilde{p}_{\mathcal{D}'_I}(\boldsymbol{X}_i)^T\tilde{p}_{\mathcal{D}'_T}(\boldsymbol{z}_j)}}\cdot\frac{\partial \tilde{p}_{\mathcal{D}'_I}(\boldsymbol{z}_i)}{\partial \mathcal{D}'_T}. \tag{21}$$

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 $|\mathcal{O}|$ input types, the new objective will contain $|\mathcal{O}| + \binom{|\mathcal{O}|}{2}$ object types.

Because deep learning is highly nonlinear and non-convex, globally optimal convergence is not assured. The initialization of parameters are 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 than the pre-training task. It is worth mentioning that the proposed embedding method is unsupervised and can be used as pre-training step for any further fine-tuning. In other words, if we want to classify network

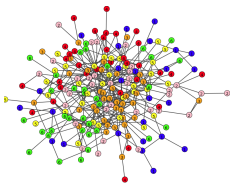


Figure 6: Linkage structures between 500 randomly selected nodes in the BlogCatalog dataset. The node color indicates the label of each node.

Table 1: Detailed statistics of the BlogCatalog dataset.

	Statistics
Number of nodes	5196
Number of links	171,743
Number of classes	6
Content dimensionality	8189
Balanced classes	yes

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 groundtruth 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 $\boldsymbol{A}$. 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.

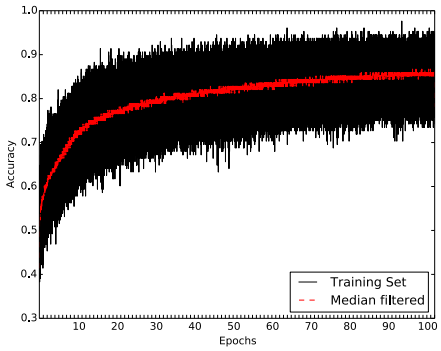


Figure 7: The Linkage reconstruction rate on the BlogCatalog.

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

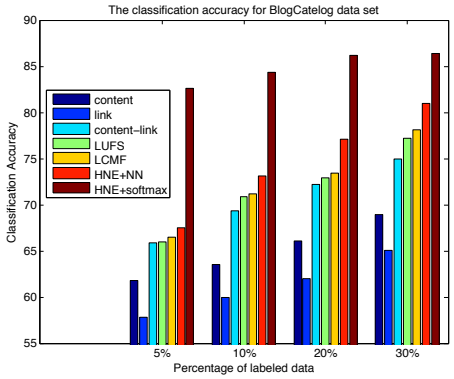


Figure 8: The classification accuracies among different methods under various size of training sets.

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 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 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 spacing 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.

Table 2: The clustering result for BlogCatalog dataset.

Methods	Accuracy	NMI
content	49.06 %	0.3192
link	40.76 %	0.2482
content-link	51.69 %	0.3457
LUFS	49.88 %	0.3221
LCMF	53.91 %	0.3678
HNE	**62.37 %**	**0.4388**

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.

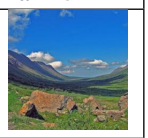
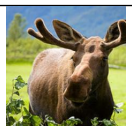
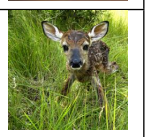
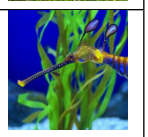
Query	rank 1	rank 2	rank 3	rank 4	rank 5
Mountain					
Sunset					
Cow					
Leaf					

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

Sample	CCA	DT	LHNE	HNE
Image only	51.96 %	52.07 %	53.16 %	**54.28 %**
Text only	51.37 %	51.88 %	51.34 %	**52.76 %**
Image + Text	52.54 %	53.22 %	53.32 %	**54.99 %**

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

Method	rank 1	rank 5	rank 10	rank 20
CCA	21.05 %	16.84 %	18.95 %	18.68 %
DT	20.53 %	25.26 %	22.63 %	22.37 %
LHNE	26.32 %	21.05 %	21.02 %	22.27 %
HNE	**36.84 %**	**29.47 %**	**27.89 %**	**26.32 %**

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 that the algorithm usually converges to a stable result in practice.

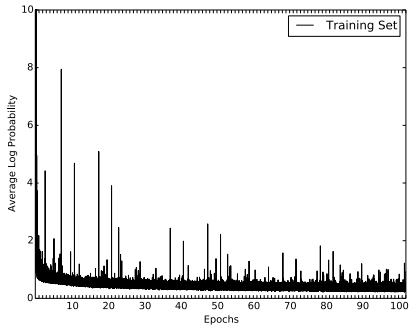


Figure 9: The objective convergence study of proposed method in the BlogCatalog dataset.

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.

1318971. This work was partially sponsored by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053. Jiliang Tang is, in part, supported by National Science Foundation under Grant No. IIS-1217466.

7. REFERENCES

[1] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. *NIPS*, 19:153, 2007.

[2] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a cpu and gpu math expression compiler. In *Proceedings of SciPy*, volume 4, page 3, 2010.

[3] S. Chang, W. Han, X. Liu, N. Xu, P. Khorrami, and T. S. Huang. Multimedia classification. *Data Classification: Algorithms and Applications*, page 337, 2014.

[4] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia. Who is tweeting on twitter: human, bot, or cyborg? In *ACSAC*, pages 21–30. ACM, 2010.

[5] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. Nus-wide: a real-world web image database from national university of singapore. In *ICIVR*, page 48, 2009.

[6] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167, 2008.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, pages 886–893. IEEE, 2005.

[8] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010.

[9] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov, et al. Devise: A deep visual-semantic embedding model. In *NIPS*, pages 2121–2129, 2013.

[10] Y. Fu, H. Xiong, Y. Ge, Z. Yao, Y. Zheng, and Z.-H. Zhou. Exploiting geographic dependencies for real estate appraisal: A mutual perspective of ranking and clustering. In *ACM SIGKDD*, pages 1047–1056. ACM, 2014.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587. IEEE, 2014.

[12] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier networks. In *JMLR*, volume 15, pages 315–323, 2011.

[13] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[14] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[15] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

[16] R. Jenatton, N. L. Roux, A. Bordes, and G. R. Obozinski. A latent factor model for highly multi-relational data. In *NIPS*, pages 3167–3175, 2012.

[17] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014.

[18] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 25, 2012.

[19] X. Li, N. Du, H. Li, K. Li, J. Gao, and A. Zhang. A deep learning approach to link prediction in dynamic networks, 2014.

[20] Z. Li, S. Chang, F. Liang, T. S. Huang, L. Cao, and J. R. Smith. Learning locally-adaptive decision functions for person verification. In *CVPR*, pages 3610–3617. IEEE, 2013.

[21] C. Liu, K. Zhang, H. Xiong, G. Jiang, and Q. Yang. Temporal skeletonization on sequential data: Patterns, categorization, and visualization. In *ACM SIGKDD*, pages 1336–1345. ACM, 2014.

[22] D. Liu, G. Ye, C.-T. Chen, S. Yan, and S.-F. Chang. Hybrid social media network. In *ACM MM*, pages 659–668. ACM, 2012.

[23] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

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

[25] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *ICML*, pages 689–696, 2011.

[26] M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.

[27] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. *arXiv:1312.5650*, 2013.

[28] M. Ou, P. Cui, F. Wang, J. Wang, W. Zhu, and S. Yang. Comparing apples to oranges: a scalable solution with heterogeneous hashing. In *ACM SIGKDD*, pages 230–238. ACM, 2013.

[29] A. Paccanaro and G. E. Hinton. Learning distributed representations of concepts using linear relational embedding. *TKDE*, 2001.

[30] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. *EMNLP*, 12, 2014.

[31] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *KDD*, pages 701–710. ACM, 2014.

[32] G.-J. Qi, C. Aggarwal, and T. Huang. Towards semantic knowledge propagation from text corpus to web images. In *WWW*, pages 297–306. ACM, 2011.

[33] G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Transfer learning of distance metrics by cross-domain metric sampling across heterogeneous spaces. In *SDM*, pages 528–539. SIAM, 2012.

[34] D. E. Rapach and M. E. Wohar. In-sample vs. out-of-sample tests of stock return predictability in the context of data mining. *Journal of Empirical Finance*, 13(2):231–247, 2006.

[35] X. Ren, J. Liu, X. Yu, U. Khandelwal, Q. Gu, L. Wang, and J. Han. Cluscite: Effective citation recommendation by information network-based clustering. In *KDD*. ACM, 2014.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, 2014.

[37] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

[38] B. Shaw and T. Jebara. Structure preserving embedding. In *ICML*, pages 937–944. ACM, 2009.

[39] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *KDD*, pages 650–658. ACM, 2008.

[40] N. Srivastava and R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *NIPS*, pages 2222–2230, 2012.

[41] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu. Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In *KDD*. ACM, 2012.

[42] J. Tang and H. Liu. Unsupervised feature selection for linked social media data. In *KDD*, pages 904–912. ACM, 2012.

[43] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu. Learning deep representations for graph clustering. In *AAAI*, 2014.

[44] T. Yang, R. Jin, Y. Chi, and S. Zhu. Combining link and content for community detection: a discriminative approach. In *KDD*, pages 927–936. ACM, 2009.

[45] J. Yi, L. Zhang, J. Wang, R. Jin, and A. K. Jain. A single-pass algorithm for efficiently recovering sparse cluster centers of high-dimensional data. In *ICML*, pages 658–666, 2014.

[46] Z. Yuan, J. Sang, Y. Liu, and C. Xu. Latent feature learning in social media network. In *ACM MM*, pages 253–262. ACM, 2013.

[47] J. Zhang, S. Y. Philip, and Z.-H. Zhou. Meta-path based multi-network collective link prediction. In *KDD*. ACM, 2014.

[48] P. Zhao, J. Han, and Y. Sun. P-rank: a comprehensive structural similarity measure over information networks. In *CIKM*, pages 553–562. ACM, 2009.

[49] J. Zhou, F. Wang, J. Hu, and J. Ye. From micro to macro: data driven phenotyping by densification of longitudinal electronic medical records. In *ACM SIGKDD*, pages 135–144. ACM, 2014.

[50] S. Zhu, K. Yu, Y. Chi, and Y. Gong. Combining content and link for classification using matrix factorization. In *SIGIR*, pages 487–494. ACM, 2007.