The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)

Large-Scale Heterogeneous Feature Embedding

Xiao Huang, Qingquan Song, Fan Yang, Xia Hu
Department of Computer Science and Engineering, Texas A&M University
{xhuang, song_3134, nacoyang, xiahu}@tamu.edu

Abstract

Feature embedding aims to learn a low-dimensional vector representation for each instance to preserve the information in its features. These representations can benefit various off-the-shelf learning algorithms. While embedding models for a single type of features have been well-studied, real-world instances often contain multiple types of correlated features or even information within a different modality such as networks. Existing studies such as multiview learning show that it is promising to learn unified vector representations from all sources. However, high computational costs of incorporating heterogeneous information limit the applications of existing algorithms. The number of instances and dimensions of features in practice are often large. To bridge the gap, we propose a scalable framework FeatWalk, which can model and incorporate instance similarities in terms of different types of features into a unified embedding representation. To enable the scalability, FeatWalk does not directly calculate any similarity measure, but provides an alternative way to simulate the similarity-based random walks among instances to extract the local instance proximity and preserve it in a set of instance index sequences. These sequences are homogeneous with each other. A scalable word embedding algorithm is applied to them to learn a joint embedding representation of instances. Experiments on four real-world datasets demonstrate the efficiency and effectiveness of FeatWalk.

Introduction

Feature embedding (Zhang et al. 2015b), aka representation learning (Bengio, Courville, and Vincent 2013) or dimensionality reduction (Hinton and Salakhutdinov 2006), aims to learn low-dimensional vectors for all instances, such that instances with similar original features tend to have similar vector representations. By reducing the redundancy and extracting meaningful information, it could be efficiently utilized to and enhance the performance of various real-world applications such as syntactic analysis (Chen, Zhang, and Zhang 2014), human action recognition (Guo et al. 2017), and person re-identification (Wu et al. 2018).

Beyond a single data source, real-world systems are often imbued with multiple types of correlated instance features (Guo 2013) or even data of a distinct modality such as networks (Yang et al. 2015) and images (Srivastava and Salakhutdinov 2012). Learning a representation jointly from all types of features is potentially helpful to make it more informative, since they often complement each other. For example, Facebook users have multiple types of features such as attributes in the introductions, words in posts, contents in photos, and rich friend relationships (Tang et al. 2015). These features are all highly correlated with each other (Smith, Fischer, and Yongjian 2012), i.e., the contents posted by users would reflect their status described in the introductions; in turn, the social status have a significant impact on users' words (Hogg and Terry 2000); and friends tend to share posts with similar topics (McPherson, Smith-Lovin, and Cook 2001). In addition, recent work such as multiview learning (Ding and Fu 2014; Gong et al. 2014) and attributed network embedding (Huang, Li, and Hu 2017a; Liu, Huang, and Hu 2017) have demonstrated the benefits of joint learning. Therefore, it is promising to perform feature embedding based on instance features collected from multiple aspects.

However, as it becomes increasingly easier and cheaper to collect data, existing heterogeneous feature embedding algorithms (Zhang et al. 2015c) face many challenges when applied to real-world systems. Three major ones are summarized as follows. First, the ever-growing data volume along with the complex data properties put demands on the scalability of algorithms. For example, there are 2.23 billion[1] monthly active Facebook users in the second quarter of 2018, and each user could have thousands of posts and friends. Capacities of existing algorithms such as coupled matrix factorization (Yang et al. 2015) and canonical correlation analysis (Foster, Kakade, and Zhang 2008; Yuan, Sun, and Ge 2014) are highly restricted under large-scale settings. Second, real-world instance features often are heterogeneous sources or even might be within a different modality such as networks (Yang et al. 2015). High computational costs of fusing heterogeneous information limit the applications of existing algorithms. A widely used approach is to calculate the instance proximity, i.e., similarities between instances, based on each source, and conduct joint learning based on these homogeneous instance proximities (Huang, Li, and Hu 2017b). But the computations of

[1] www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/

instance proximities would lead to high time and space complexity, not to mention the subsequent operations. Another effective way is to employ deep neural networks (Ngiam et al. 2011), which is not scalable either. Third, the data sparsity problem is significant (Zhang et al. 2015b), e.g., in social media, a large proportion of words or attributes often only contribute to a small number of users, providing insufficient information to accomplish effective embedding.

To bridge the gap, we study the problem of large-scale heterogeneous feature embedding. We focus on two most common types of data in practice, i.e., feature matrices and the topological structure (Yang et al. 2015). We target at jointly embedding multiple feature matrices and an instance relation network. It could be summarized as two research questions. (1) How to effectively utilize heterogeneous instance features and a relation network to learn a unified embedding representation? (2) How to cope with the large scale issue while maintaining the effectiveness of the joint embedding framework? Guided by these questions, we propose a scalable heterogeneous feature embedding framework - FeatWalk. Our major contributions are listed as follows.

- Formally define the problem of large-scale heterogeneous feature embedding;
- Propose an effective framework FeatWalk to incorporate multiple types of high-dimensional instance features into a joint embedding representation;
- Design an efficient algorithm that avoids computing similarity measure, and provides an alternative way to simulate the similarity-based random walks among instances to model the local instance proximity;
- Empirically validate the efficiency and effectiveness of FeatWalk on four datasets from real-world systems.

Problem Statement

Notations: We use boldface lowercase alphabets to denote vectors (e.g., $\mathbf{h}$) and boldface uppercase alphabets to denote matrices (e.g., $\mathbf{H}$). For a matrix $\mathbf{H}$, its transpose is represented as $\mathbf{H}^\top$ and its i^{th} row is denoted as $\mathbf{h}_i$. We use $\{\mathbf{X}^{(i)}\}$ and $\{x_i\}$ to represent a sequence of matrices $\mathbf{X}^{(i)}$ and scalars x_i. The main symbols are list in Table 1.

Let $\{\mathbf{X}^{(i)}\}$, for $i = 1, \ldots, I$, be a set of correlated feature matrices of N instances from I different views. Let the last one $\mathbf{X}^{(I)} = \mathbf{G}$ be a weighted adjacency matrix that describes the relations among the N instances. To have physical meanings, we assume that elements of all matrices in $\{\mathbf{X}^{(i)}\}$ are non-negative. A specific example would be the products on Amazon. They have descriptions from multiple sources such as product information and customer reviews, which complement each other and could be used to construct $\{\mathbf{X}^{(i)}\}$. Customer purchase records (Linden, Smith, and Zada 2005) are also available and could be used to build $\mathbf{G}$.

Given these assumptions, we formally define the large-scale heterogeneous feature embedding problem as follows. *Given a large number of instances, associated with a set of instance feature matrices* $\{\mathbf{X}^{(i)}\}$ *and an instance relation network* $\mathbf{G}$*, we aim to learn a low-dimensional representation* $\mathbf{h}_i$ *for each instance* i*, such that all the meaningful information in* $\{\mathbf{X}^{(i)}\}$ *and* $\mathbf{G}$ *could be well preserved in* $\mathbf{H}$*. The amount of learned meaningful information in* $\mathbf{H}$ *could be evaluated based on the performance of* $\mathbf{H}$ *in real-world applications such as classification and clustering.*

Table 1: Main symbols and their definitions in the paper.

Notation	Definition
N	total number of instances
$\mathbf{X}^{(i)} \in \mathbb{R}_+^{N \times M^{(i)}}$	the i^{th} instance feature matrix
$M^{(i)}$	number of categories in $\mathbf{X}^{(i)}$, $i = 1, \ldots, I$
$\mathbf{G} \in \mathbb{R}_+^{N \times N}$	a weighted adjacency matrix, $\mathbf{G} = \mathbf{X}^{(I)}$
$\mathbf{S}^{(i)} \in \mathbb{R}^{N \times N}$	instance similarity matrix based on $\mathbf{X}^{(i)}$
M, $\mathbf{X}$, $\mathbf{S}$	$M = M^{(1)}$, $\mathbf{X} = \mathbf{X}^{(1)}$, $\mathbf{S} = \mathbf{S}^{(1)}$
d	dimension of embedding representations
$\mathbf{H} \in \mathbb{R}^{N \times d}$	final low-dimensional representation
$\mathcal{Q}^{(i)}$	sequences of indices learned from $\mathbf{X}^{(i)}$
$W^{(i)}$	total number of sequences in $\mathcal{Q}^{(i)}$

Heterogeneous feature embedding can be roughly separated into two categories, i.e., multiview and multimodal feature embedding. The former aims to learn a unified representation of instances from multiple feature matrices observed from different aspects (Li et al. 2015). The latter focuses on multiple sources with distinct modalities such as networks, images, and audio (Ngiam et al. 2011).

Different from these previous studies, in addition to the multiple feature matrices, our work takes the instance relation network into consideration, which is a common and crucial type of information in real-world systems. The topological structure has a different modality than feature matrices. We also target at developing a scalable framework to make it practical. It is distinct from attributed network embedding (Huang et al. 2018) since the latter focuses on embedding a single network and a single feature matrix.

Large-scale Feature Embedding

To jointly embed the heterogeneous information in $\{\mathbf{X}^{(i)}\}$, we propose an efficient framework - *FeatWalk*. It achieves scalability by avoiding the computation of instance similarities, and provides an alternative way to simulate similarity-based random walks among instances. Figure 1 illustrates its main idea. Although $\{\mathbf{X}^{(i)}\}$ are heterogeneous, proximities between instances defined by rows of each $\mathbf{X}^{(i)}$ are homogeneous. FeatWalk first learns the instance proximities defined by all $\{\mathbf{X}^{(i)}\}$ via *Feature Walks*, and then jointly incorporates them into a unified embedding representation $\mathbf{H}$. To model the instance proximity in $\mathbf{X}$, an intuitive solution is to construct a new graph $\mathbf{S}$ with instance similarities as edge weights, and then perform random walks on $\mathbf{S}$ to learn a set of sequences $\mathcal{Q}^{(1)}$. However, $\mathbf{S}$ is often dense because of the common feature categories. As N keeps increasing, it would become too expensive to be manipulated. We propose a distributed algorithm - *Feature Walks*, which could obtain the same results as the intuitive solution but avoid the computation of $\mathbf{S}$. The learned $\mathcal{Q}^{(1)}$ consists of instance indices that record the walking trajectories such as $[1, 6, 4, 2, 5]$. $\mathcal{Q}^{(1)}$ pre-

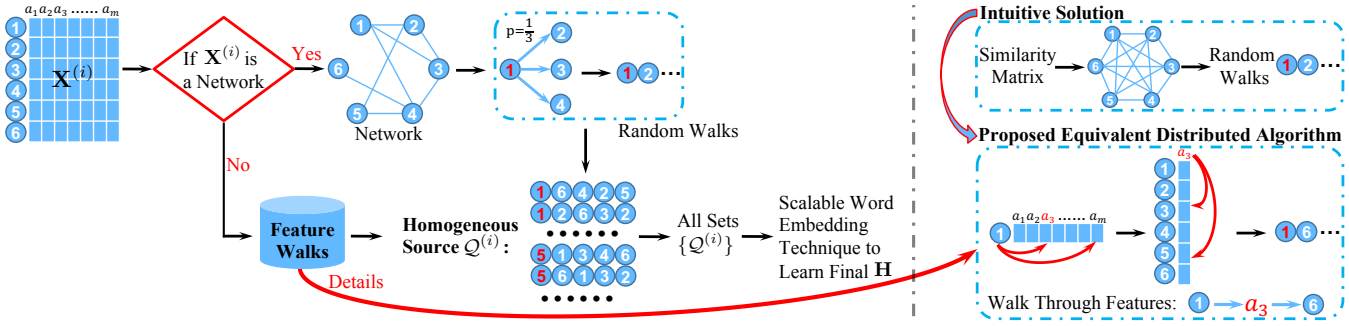


Figure 1: To avoid the computation of similarity matrices, FeatWalk performs equivalent random walks through features.

serves the information in $\mathbf{X}$. If $\mathbf{X}^{(i)}$ is a network, e.g., $\mathbf{G}$, we conduct random walks on this network directly to learn the sequences $\mathcal{Q}^{(i)}$. Finally, all $\{\mathcal{Q}^{(i)}\}$ are homogeneous. By considering the instance indices as words and sequences as sentences, a scalable word embedding technique is applied to $\{\mathcal{Q}^{(i)}\}$ to learn a joint representation $\mathbf{H}$.

Tackling the Heterogeneity Among Features

$\{\mathbf{X}^{(i)}\}$ are collected from different aspects. They are not only mutually dependent on and complement each other (Smith, Fischer, and Yongjian 2012; Guo et al. 2017), but also heterogeneous with each other. To incorporate the heterogeneous information, a commonly used approach (Zhang et al. 2015b) is to calculate the instance proximity based on each source respectively, and learn $\mathbf{H}$ from all instance proximities jointly. It could effectively tackle the heterogeneity since instance proximities are homogeneous.

Definition 1. (Instance Proximity) *It refers to the similarities between instances defined by the features of instances, i.e., the rows of each* $\mathbf{X}^{(i)}$.

However, the computation and manipulations of similarity matrices increase exponentially as N increases. Thus, existing algorithms such as coupled spectral embedding (Huang, Li, and Hu 2017b) and similarity-based deep models (Huang, Loy, and Tang 2016; Wu et al. 2018) cannot be directly applied under large-scale settings.

FeatWalk follows this effective approach and copes with the scalability issue by sampling local instance proximity in a distributed manner. We illustrate its basic idea with the first type of features $\mathbf{X}$ and the last one $\mathbf{G}$. It is straightforward to extend to the scenarios with multiple $\{\mathbf{X}^{(i)}\}$.

Intuitive Solution As shown in Figure 1, to model the instance proximity defined by $\mathbf{X}$, an intuitive solution is to compute its similarity matrix to construct a new graph $\mathbf{S}$, and perform truncated random walks on $\mathbf{S}$ to learn and preserve the instance proximity into a set of sequences $\mathcal{Q}^{(1)}$.

Details are introduced as follows. The weight of edge between instances i and j in $\mathbf{S}$ is defined as the similarity between their instance features. The probability of walking from instance i to j is determined by the edge weight, i.e.,

$$P(i \to j) = \frac{s_{ij}}{\sum_{n=1}^{N} s_{in}}. \tag{1}$$

Each walk has the same length L. Each sequence in $\mathcal{Q}^{(1)}$ consists of instance indices that record the walking trajectories. It could capture the local instance proximity, because as the number of learned sequences keeps increasing, the probability of index j follows index i in $\mathcal{Q}^{(1)}$ would approach to $P(i \to j)$. Thus, learning an embedding representation based on the indices' co-occurrence probabilities is equivalent to the one based on the instances' linking probabilities.

However, this intuitive solution has several problems. First, as N increases, the size of $\mathbf{S}$ would increase exponentially. The calculation, storage, and manipulations of $\mathbf{S}$ would become expensive. Second, $\mathbf{S}$ is often quite dense, which makes the random walks on $\mathbf{S}$ inefficient. For example, when creating feature matrix for Twitter users based on their tweets, the commonly used words such as "good" and "think" would make $\mathbf{S}$ close to a clique. Then the time complexity of sampling a neighbor from $\mathbf{s}_i$ would become $\mathcal{O}(N)$ (Devroye 1986), which is expensive. Therefore, we propose a distributed algorithm - *Feature Walks*, which solves these problems by avoiding the computation of $\mathbf{S}$.

Feature Walks Since the computation and operations of $\mathbf{S}$ are expensive, we design an alternative way to simulate the similarity-based random walks on $\mathbf{S}$, with details as follows.

I. We normalize the feature matrix $\mathbf{X}$. We use ℓ_2 norm to normalize each row of $\mathbf{X}$ and get $\bar{\mathbf{X}}$. Since it is hard for random walks to simulate the probabilities with small values, we remove the small elements in $\bar{\mathbf{X}}$, i.e., elements smaller than βMean($\bar{\mathbf{X}}$), and get $\hat{\mathbf{X}}$. Mean($\bar{\mathbf{X}}$) denotes the mean value of all elements in $\bar{\mathbf{X}}$ and β is a threshold value. We use ℓ_1 norm to normalize each row of $\hat{\mathbf{X}}$ and get a new matrix $\mathbf{Y}$, i.e.,

$$y_{im} = \frac{\hat{x}_{im}}{\sum_{p=1}^{M} \hat{x}_{ip}}. \tag{2}$$

II. Given an initial instance i, we randomly select a feature category based on the normalized feature of instance i, i.e., $\hat{\mathbf{x}}_i$. Let a_m denote the m^{th} feature category, then the probability of selecting a_m is defined as follows,

$$P(i \to a_m) = y_{im}. \tag{3}$$

III. Given that a_m is selected, we randomly select an instance based on the m^{th} column of $\mathbf{Y}$. The probability of selecting instance j defined as follows,

$$P(a_m \to j) = \frac{y_{jm}}{\sum_{n=1}^{N} y_{nm}}. \tag{4}$$

In such a way, we accomplish the walk from instance i to j. **IV**. The length of each walk is set as L. To make sure local proximities of all instances could be sampled, we select each instance as the initial index in turns. However, the number of random walks using instance i as an initial index, i.e., w_i, is not fixed. We assign it based on the complexity of corresponding instance, which is defined as follows,

$$w_i = \frac{\text{nnz}(\mathbf{x}_i)W^{(1)}}{\sum_{n=1}^{N}\text{nnz}(\mathbf{x}_n)}, \tag{5}$$

where $\text{nnz}(\cdot)$ denotes the number of non-zero elements, and $W^{(1)}$ is the total number of walks assigned for modeling $\mathbf{X}$. We design the function in (5) based on two assumptions. First, instances with more features tend to have more edges, and they would require more random walks to simulate its linking probabilities. For example, if in $\mathbf{S}$, instances 4 and 6 have four and one edges respectively. We need to walk from instance 4 at least four times and from instance 6 at least one time to capture their relationships with all neighbors. Second, in a network, instances with more edges tend to be more important (Narayanan, Belkin, and Niyogi 2006). To make the local proximity of important instances well-preserved, we sample more sequences for them. Thus, we assign the numbers of walks $\{w_i\}$ based on the complexity.

Theoretical Analysis of Feature Walks We now prove that the output of *Feature Walks* is equivalent to the output of random walks on $\mathbf{S}$. Let $\mathbf{D}$ be a diagonal matrix with the reciprocal of the sum of each column of $\mathbf{Y}$ on the diagonal.

Theorem 1. *The probability of walking from instance i to j via* Feature Walks *is equal to the one via random walks on the similarity graph $\mathbf{S}$, with the definition as follows.*

$$\mathbf{S} = \mathbf{Y}\mathbf{D}\mathbf{Y}^{\mathrm{T}}. \tag{6}$$

Proof. In *Feature Walks*, the process of walking from instance i to feature category a_m is independent of the process of walking from a_m to j. Thus, the probability of walking from i to j is defined as follows.

$$\begin{aligned} P(i \to j) &= \sum_{m=1}^{M} P(i \to a_m)P(a_m \to j), \\ &= \sum_{m=1}^{M} \frac{y_{im}y_{jm}}{\sum_{n=1}^{N} y_{nm}}. \end{aligned} \tag{7}$$

It should be noted that $P(i \to j) = P(j \to i)$. On the other hand, for the random walks on $\mathbf{S}$, we have,

$$s_{ij} = [y_{i1}, \ldots, y_{iM}] \circ [d_{11}, \ldots, d_{MM}]\mathbf{y}_j^{\top} = \sum_{m=1}^{M} \frac{y_{im}y_{jm}}{d_{mm}}, \tag{8}$$

where notation $\circ$ denotes the element-wise multiplication and degree $d_{mm} = 1/\sum_{n=1}^{N} y_{nm}$. Thus, we have $P(i \to j)$ equals the probability of walking from i to j in the random walks on $\mathbf{S}$, with $\sum_{i=1}^{N} s_{ij} = 1$. □

Instance Proximity in Network Given the relation network $\mathbf{G}$, we model its instance proximity via conducting random walk on it directly. It is because instances with stronger relationships tend to be more similar. Homophily hypothesis (McPherson, Smith-Lovin, and Cook 2001) and social influence (Zhang et al. 2015a) have demonstrated that instances with similar features tend to have similar network structures, and the latter would also have a significant impact on the former ones.

Similar to *Feature Walks*, the length of each walk is set as L, and the total number of walks assigned to model $\mathbf{G}$ is set as $W^{(I)}$. The number of walks using i as the initial index, i.e., $\hat{w}_i$, is defined as follows,

$$\hat{w}_i = \frac{\text{nnz}(\mathbf{g}_i)W^{(I)}}{\sum_{n=1}^{N}\text{nnz}(\mathbf{g}_n)}. \tag{9}$$

Joint Learning with Multiple Feature Matrices

All the sequences in all the sets $\{\mathcal{Q}^{(i)}\}$ are homogeneous with each other. Thus, we could put them together to jointly perform the instance proximity learning. Let W be the total number of sequences that we could sample. To balance the contributions of $\mathbf{X}$ and $\mathbf{G}$, $W^{(1)}$ and $W^{(I)}$ are defined as,

$$W^{(1)} = \alpha W \quad \text{and} \quad W^{(I)} = (1-\alpha)W. \tag{10}$$

By applying a scalable word embedding method (Mikolov et al. 2013), we could learn a joint embedding representation $\mathbf{H}$ from $\{\mathcal{Q}^{(i)}\}$. Word embedding (Pennington, Socher, and Manning 2014) aims to map each word in a set of sentences into a low-dimensional vector, so that words with similar semantic meaning would have similar vector representations. Since massive amounts of documents are available in practice, many scalable word embedding algorithms such as word2vec (Mikolov et al. 2013) and GloVe (Pennington, Socher, and Manning 2014) have been proposed. We could take advantage of these efficient algorithms to learn joint low-dimensional representations of instances from $\{\mathcal{Q}^{(i)}\}$, by considering the instances in $\{\mathcal{Q}^{(i)}\}$ as words in sentences.

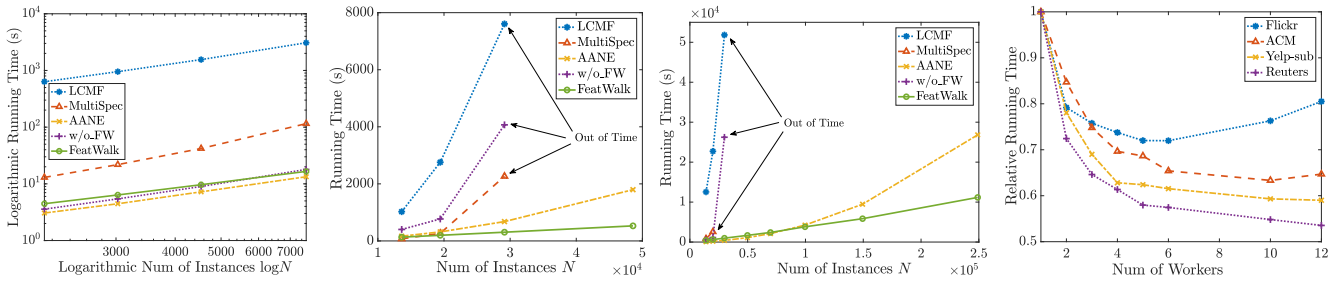
It is straightforward to extend FeatWalk to multiple instance features $\{\mathbf{X}^{(i)}\}$. We could perform *Feature Walks* on each $\mathbf{X}^{(i)}$ and learn multiple sets of homogeneous sentences. Then a joint $\mathbf{H}$ could be learned from these sentences.

Complexity Analysis

Let $\mathcal{N}_{\mathbf{X}}^{(i)}$ be the numbers of nonzero entries in $\mathbf{X}^{(i)}$. Let $\mathcal{T}$ denote the number of operations required to obtain $\mathbf{H}$ based on the W learned sentences. In the tasks of sampling from a discrete probability distribution, we use the alias method (Devroye 1986). The time complexity of the setup of alias method is $\mathcal{O}(\mathcal{N}_{\mathbf{X}}^{(i)})$ and each sampling takes $\mathcal{O}(1)$ time. Then, the time complexity of modeling $\mathbf{X}^{(i)}$ is $\mathcal{O}(\mathcal{N}_{\mathbf{X}}^{(i)} + W^{(i)}L)$. Thus, the time complexity of generating all sequences is linear with the numbers of nonzero entries in $\{\mathbf{X}^{(i)}\}$. The total time complexity of FeatWalk is $\mathcal{O}(\sum_i \mathcal{N}_{\mathbf{X}}^{(i)} + WL + \mathcal{T})$. It should be noted that the processes of learning any two sequences are independent of each other. We could implement them in parallel to further accelerate FeatWalk.

Experiments

We now empirically validate the efficiency and effectiveness of FeatWalk. There are three major questions we aim to answer. (1) How efficient is FeatWalk in performing hetero-



(a) Flickr (b) ACM (c) Yelp (d) Number of workers

Figure 2: Running time of FeatWalk and different heterogeneous feature embedding methods on Flickr, ACM, and Yelp.

geneous feature embedding compared with the state-of-the-art embedding methods? (2) How effective is the representations learned by FeatWalk compared with other embedding methods in applications such as classification? (3) What are the impacts of parameters α, the window size, L, the number of sentences per instance W/N, and d on FeatWalk?

Four Real-world Datasets

The four real-world datasets that we employed in the experiments are all publicly available. The first dataset contains two feature matrices. Each of the last three datasets contains one feature matrix and one network.

Reuters (Amini, Usunier, and Goutte 2009): 18,758 documents from Reuters are used as instances. They are originally written in English. We employ their Italian translations as $\mathbf{X}$, with $M = 11{,}452$, and the Spanish translations as $\mathbf{X}^{(2)}$, with $M^{(2)} = 9{,}243$, via the bag-of-words model. To make the embedding task more challenging, the original top 10% frequent features have been removed. Each document is from one of the six populous classes.

Flickr (Huang, Li, and Hu 2017b): 7,564 Flickr users are employed as instances. They share photos online, and attach many related tags to their photos. These tags reflect the interests of instances. We use them as $\mathbf{X}$, with $M = 12{,}047$. Instances also follow each other and form a network $\mathbf{G}$ naturally, with 239,365 undirected edges in total. The nine groups that instances have joined are employed as their labels.

ACM (Tang et al. 2008): 48,579 papers published in ACM are utilized as instances. We employ paper abstracts as $\mathbf{X}$, with $M = 10{,}000$, and construct an undirected $\mathbf{G}$ based on the citation links, with 288,374 edges in total. All instances are from nine areas such as Artificial Intelligence (AAAI, IJCAI, etc.), Data Mining (KDD, ICDM, etc.), and Machine Learning (ICML, COLT, etc.), which serve as the labels.

Yelp (Yelp 2017): 249,012 Yelp users are used as instances. They have written reviews for different businesses. We set the reviews as $\mathbf{X}$, with $M = 20{,}000$. Their friend relationships are employed to construct $\mathbf{G}$, with 1,779,803 edges in total. All businesses are categorized into eleven classed such as Nightlife and Services. Categories of the businesses that an instance has reviewed are set as his/her labels.

Baseline Methods

To study the performance of FeatWalk, we compare it with three categories of baselines. First, to investigate the impact of each type of features, we include three single feature embedding methods, i.e., NMF, Spectral, FeatWalk_X. Second, to study the impact of the networks, we include two network embedding methods, i.e., DeepWalk and LINE. Third, to analyze the efficiency and effectiveness of FeatWalk, we include three state-of-the-art heterogeneous feature embedding methods, i.e., LCMF, MultiSpec, and AANE, and a variation of FeatWalk named w/o_FW. No deep models are included since they are not scalable (Tu et al. 2018).

- *NMF* (Pedregosa et al. 2011): It is a scalable version of non-negative matrix factorization, optimized by the hierarchical alternating least squares algorithms. It reduces the dimension of $\mathbf{X}$ (or $\mathbf{X}^{(2)}$) to learn $\mathbf{H}$.
- *Spectral* (von Luxburg 2007): It calculates the cosine similarities between vectors $\mathbf{x}_i$ (or $\mathbf{x}_i^{(2)}$) to construct a new graph, and applies spectral embedding on it to learn $\mathbf{H}$.
- *FeatWalk_X*: It learns the embedding representation $\mathbf{H}$ only from the feature matrix $\mathbf{X}$ (or $\mathbf{X}^{(2)}$) via FeatWalk.
- *DeepWalk* (Perozzi, Al-Rfou, and Skiena 2014): It performs random walks on $\mathbf{G}$ and learns sequences of instance indices. word2vec is applied to them to learn $\mathbf{H}$. It is a variant of FeatWalk that only uses $\mathbf{G}$.
- *LINE* (Tang et al. 2015): It learns $\mathbf{H}$ from $\mathbf{G}$ by jointly modeling its first and second order instance proximity.
- *LCMF* (Zhu et al. 2007): It conducts classical coupled matrix factorization between $\mathbf{X}$ and $\mathbf{G}$ to learn $\mathbf{H}$ jointly.
- *MultiSpec* (Kumar, Rai, and Daume 2011): It computes all instance similarity matrices and jointly models them via coupled spectral embedding.
- *AANE* (Huang, Li, and Hu 2017a): It is the state-of-the-art embedding method for networks with node attributes.
- *w/o_FW*: FeatWalk without using *Feature Walks*. Instead, it calculates $\mathbf{S}$ directly, and performs random walks on $\mathbf{S}$.

Experimental Settings

To analyze the effectiveness of FeatWalk and baselines, we apply their learned embedding representations $\mathbf{H}$ to perform classification, following the commonly-used way of

Table 2: Classification performance of FeatWalk and all baselines on Flickr, ACM, and Yelp-sub in terms of micro-average.

	Flickr			ACM			Yelp-sub			Yelp			
Training	25%	50%	100%	25%	50%	100%	25%	50%	100%	10%	25%	50%	100%
# Instances	3,026	4,538	7,564	19,432	29,147	48,579	19,921	29,881	49,802	69,723	99,605	149,407	249,012
NMF	0.629	0.718	0.773	0.653	0.660	0.664	0.680	0.686	0.688	0.678	0.692	0.694	0.689
Spectral	0.771	0.813	0.846	0.688	0.700	N.A.	0.683	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
FeatWalk_X	0.803	0.841	0.868	0.676	0.675	0.667	**0.701**	0.710	0.714	0.706	0.691	0.703	0.699
DeepWalk	0.373	0.465	0.535	0.576	0.630	0.684	0.310	0.318	0.350	0.324	0.345	0.366	0.368
LINE	0.332	0.421	0.516	0.549	0.624	0.693	0.243	0.264	0.294	0.295	0.313	0.336	0.354
LCMF	0.676	0.725	0.749	0.690	0.706	N.A.	0.680	0.686	N.A.	N.A.	N.A.	N.A.	N.A.
MultiSpec	0.720	0.800	0.859	0.709	0.719	N.A.	0.667	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.
AANE	0.811	0.854	0.885	0.701	0.715	0.722	0.694	0.703	0.711	0.698	**0.709**	**0.711**	**0.714**
FeatWalk	**0.831**	**0.865**	**0.893**	**0.722**	**0.738**	**0.751**	0.700	**0.710**	**0.717**	**0.708**	0.691	0.704	0.701

validating feature embedding methods (Xia et al. 2010; Zhang et al. 2015b). The classification task aims to classify a new instance into one or multiple categories, based on its embedding representation and the trained classifier. The performance is measured by two standard metrics, i.e., micro-average and macro-average (Huang, Li, and Hu 2017b).

We apply 5-fold cross-validation on all datasets, i.e., randomly select $\frac{4}{5}$ of all instances as a training group and the remaining as a test group. If the dataset has a network, then the edges between the training and test groups would be kept. We concatenate feature matrices in two groups to create new instance feature matrices. To evaluate an embedding method, we apply it to the new instance feature matrices to learn $\mathbf{H}$, which contains vector representations for instances in the training group $\mathbf{H}_{\text{train}}$ and instances in the test group $\mathbf{H}_{\text{test}}$. To perform classification, we build an SVM (Pedregosa et al. 2011) classifier based on $\mathbf{H}_{\text{train}}$ and corresponding labels. Then we apply the learned classifier to predict the labels of instances in the test group based on $\mathbf{H}_{\text{test}}$.

We use the original papers' default settings to determine the parameters of baselines. FeatWalk_X and w/o_FW use the same parameters as FeatWalk. If it is not specified, d is set as 100 and 100% of the instances in the training group are used. We performed ten test runs and used the arithmetic average as the final experimental results. We ran the experiments on a Dell OptiPlex 9030 i7-16GB desktop.

Efficiency of FeatWalk

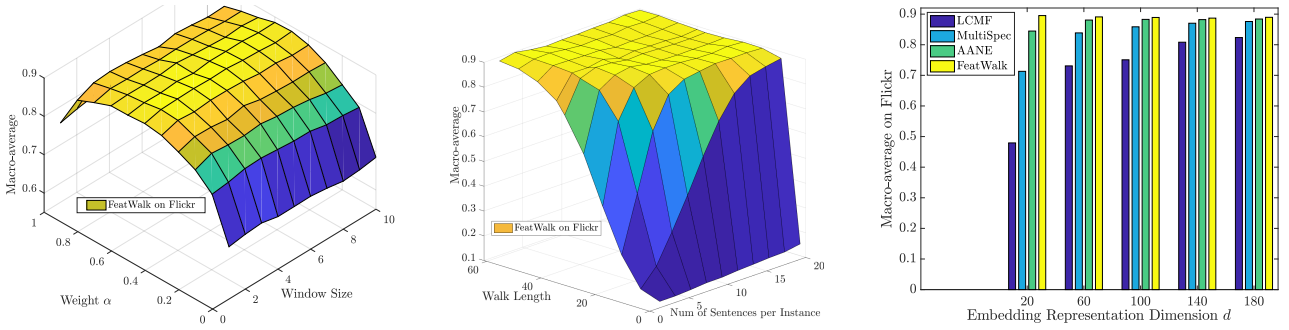
To investigate the first question proposed at the beginning of this section, we compare FeatWalk with the three state-of-the-art heterogeneous feature embedding methods and w/o_FW. The running time of all methods as a function of the number of instances N on Flickr, ACM, and Yelp is shown in Figure 2. The result on Reuters is similar, so we omit it.

From the results in Figure 2, we have three major observations. First, the running time of FeatWalk is almost linear to N, which demonstrates its scalability. For example, in Figure 2c, the slope of the green curve (FeatWalk) remains invariable as N increases. As N keeps increasing, LCMF, MultiSpec, and w/o_FW run out of time since their running time increase exponentially. Second, the distributed sampling algorithm *Feature Walks* has significantly accelerated FeatWalk and made it scalable. When N is small, as shown in Figure 2a, w/o_FW has almost the same running time as FeatWalk, since the manipulations of similarity matrix $\mathbf{S}$ are cheap at this time. However, when N keeps increasing, as shown in Figure 2b, the running time of w/o_FW increases exponentially until it runs out of time. FeatWalk has significantly less running time than w/o_FW on both ACM and Yelp. Third, FeatWalk always has the least running time when N is large. When N is small, AANE might have less running time than FeatWalk. FeatWalk has almost the same running time as AANE on Flickr, and is always faster than AANE on ACM. On Yelp, FeatWalk needs more running time than AANE when $N < 82{,}000$, but it becomes faster than AANE when $N \geq 82{,}000$ since it is almost linear to N. It should be noted that AANE is designed for embedding a single feature matrix with a single network, while FeatWalk is general to multiple ones, e.g., FeatWalk could be applied to Reuters, while AANE can not.

Table 3: LCMF and AANE can not be applied to Reuters.

# Instances	7,503 (25%)		11,255 (50%)		18,758 (100%)	
	$\mathbf{X}^{(1)}$	$\mathbf{X}^{(2)}$	$\mathbf{X}^{(1)}$	$\mathbf{X}^{(2)}$	$\mathbf{X}^{(1)}$	$\mathbf{X}^{(2)}$
NMF	0.658	0.673	0.684	0.689	0.695	0.695
Spectral	0.679	0.689	0.694	0.704	0.710	0.714
FeatWalk_X	0.698	0.700	0.727	0.728	0.744	0.745
MultiSpec	0.707		0.727		0.740	
FeatWalk	**0.720**		**0.748**		**0.771**	

The running time of FeatWalk can be further reduced by using the multi-thread implementation. Figure 2d shows the relative running time of FeatWalk as a function of the number of workers c on all datasets. From the results, we have three observations. First, FeatWalk becomes more efficient as c increases. Second, as c increases from 1 to 2, the running time decreases less than 50%. It is because the multi-thread implementation only accelerates the learning of sequences $\{\mathcal{Q}^{(k)}\}$, not the word embedding process. Third, the running time of FeatWalk on Flickr keeps increasing when $c \leq 5$. It is because $N = 7{,}564$ is relatively small and it takes extra time for the communications between the coordinator and workers. It should be noted that word2vec can be replaced by other more efficient algorithms to make FeatWalk faster.



(a) α and window size (b) Walk length L and W/N (c) Embedding representation dimension d

Figure 3: The impacts of parameters α, window size, walk length L, number of sentences per instance W/N, and d on FeatWalk.

Effectiveness of FeatWalk

To answer the second proposed question, we compare the classification performance of all methods on the four datasets, which are list in Tables 2 and 3. Since Spectral, LCMF, and MultiSpec are not capable of embedding the dataset Yelp, we randomly sample 20% of it and construct a new dataset named Yelp-sub. DeepWalk, LINE, LCMF, and AANE could not be applied to the dataset Reuters since it has no network.

From the results in Tables 2 and 3, we have three major findings. First, FeatWalk_X performs better than single feature embedding methods, i.e., NMF and Spectral. By taking advantage of the extra information ($\mathbf{G}$ or $\mathbf{X}^{(2)}$), FeatWalk further improves the performance. For example, on Reuters, FeatWalk achieves 3.6% of improvements than FeatWalk_X and 10.9% of improvements than NMF. Second, by incorporating the heterogeneous information, FeatWalk outperforms all network embedding methods, i.e., DeepWalk and LINE. For example, FeatWalk achieves 9.8% of improvements than DeepWalk on ACM. Third, on all datasets except Yelp, FeatWalk achieves better performance than all heterogeneous feature embedding methods, i.e., LCMF, MultiSpec, and AANE. For example, on Flickr, FeatWalk achieves a gain of 0.9% over AANE. It demonstrates that the way that FeatWalk has used to incorporate the heterogeneous information is effective.

To study the performance of all methods w.r.t. different training set percentages, i.e., the percentage of instances in the training group (among the four folds in the cross-validation) that have been used, we vary it as $\{25\%, 50\%, 100\%\}$. The results on the four datasets are shown in Tables 2 and 3. From the results, we observe that the aforementioned findings hold consistently as the training set percentage increases. FeatWalk undeviatingly outperforms all baselines on all datasets except Yelp. For example, on Reuters, when the training set percentage is 50%, FeatWalk achieves 6.3% of improvements than Spectral and 2.9% of improvements than MultiSpec. On Yelp, the performance of FeatWalk decreases when the training set percentage increases from 10% to 25%. It is because the sequence sets reach the memory limit and an online version of word2vec is used to learn $\mathbf{H}$.

Parameter Analysis

We now study the third proposed question. Performance of FeatWalk on Flickr as a function of the first instance feature matrix weight α and window size, a function of L and W/N, and a function of d are shown in Figures 3a, 3b and 3c. Results on other datasets are similar, so we omit them.

First, we vary α from 0 to 1 and the window size from 1 to 10. When $\alpha = 0$, only $\mathbf{G}$ is used to learn $\mathbf{H}$. When $\alpha = 1$, only $\mathbf{X}$ is used to learn $\mathbf{H}$. The window size denotes the maximum distance that is used to define context words in word2vec. From the results in Figure 3a, we observe that FeatWalk achieves the best performance when $\alpha = 0.62$, i.e., when the contributions of $\mathbf{X}$ and $\mathbf{G}$ are balanced. When α is fixed, the performance of FeatWalk keeps stable as the window size increases from 1 to 10. Second, we vary the walk length L as from 2 to 60 and the number of sentences per instance W/N from 2 to 20. From the results in Figure 3b, we find that the performance of FeatWalk keeps increasing as the product of L and W/N increases, and keeps stable when the product is sufficiently large. Third, we vary d as $\{20, 60, 100, 140, 180\}$. From the results in Figure 3c, we observe that, as d increases from 20 to 180, the performance of FeatWalk keeps stable and is always better than all baselines.

Conclusion And Future Work

We investigate the problem of heterogeneous feature embedding and propose a scalable framework - FeatWalk. It could encode multiple instance feature matrices and even network relations into unified instance vector representations. Without calculating any similarity measure among instances, we design an alternative way to simulate the similarity-based random walks among instances, which samples the local instance similarities and preserves them in walking trajectories. Along with the trajectories learned via random walks on the network relations, we apply a scalable word embedding algorithm to learn the joint representations of instances from these trajectories. Experiments on the four real-word datasets validate the scalability and effectiveness of FeatWalk. Our future work is to explore semi-supervised frameworks to incorporate label information and dynamic algorithms to cope with streaming environments.

Acknowledgments

This work is, in part, supported by DARPA (#N66001-17-2-4031) and NSF (#IIS-1750074 and #IIS-1718840). The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

Amini, M.; Usunier, N.; and Goutte, C. 2009. Learning from multiple partially observed views - an application to multilingual text categorization. In *NIPS*, 28–36.

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *TPAMI* 35(8):1798–1828.

Chen, W.; Zhang, Y.; and Zhang, M. 2014. Feature embedding for dependency parsing. In *COLING*, 816–826.

Devroye, L. 1986. Sample-based non-uniform random variate generation. In *Winter Simulation Conference*, 260–265.

Ding, Z., and Fu, Y. 2014. Low-Rank common subspace for multi-view learning. In *ICDM*, 110–119.

Foster, D. P.; Kakade, S. M.; and Zhang, T. 2008. Multi-view dimensionality reduction via canonical correlation analysis. *Toyota Technical Institute-Chicago*.

Gong, Y.; Ke, Q.; Isard, M.; and Lazebnik, S. 2014. A multi-view embedding space for modeling internet images, tags, and their semantics. *International Journal of Computer Vision* 106(2):210–233.

Guo, Y.; Tao, D.; Liu, W.; and Cheng, J. 2017. Multiview cauchy estimator feature embedding for depth and inertial sensor-based human action recognition. *IEEE Trans. on Systems, Man, and Cybernetics* 47(4):617–627.

Guo, Y. 2013. Convex subspace representation learning from multi-view data. In *AAAI*, 387–393.

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

Hogg, M. A., and Terry, D. I. 2000. Social identity and self-categorization processes in organizational contexts. *Academy of Management Review* 25(1):121–140.

Huang, X.; Song, Q.; Li, J.; and Hu, X. 2018. Exploring expert cognition for attributed network embedding. In *WSDM*, 270–278.

Huang, X.; Li, J.; and Hu, X. 2017a. Accelerated attributed network embedding. In *SDM*, 633–641.

Huang, X.; Li, J.; and Hu, X. 2017b. Label informed attributed network embedding. In *WSDM*, 731–739.

Huang, C.; Loy, C. C.; and Tang, X. 2016. Local similarity-aware deep feature embedding. In *NIPS*, 1262–1270.

Kumar, A.; Rai, P.; and Daume, H. 2011. Co-Regularized multi-view spectral clustering. In *NIPS*, 1413–1421.

Li, Y.; Nie, F.; Huang, H.; and Huang, J. 2015. Large-scale multi-view spectral clustering via bipartite graph. In *AAAI*, 2750–2756.

Linden, G. D.; Smith, B. R.; and Zada, N. K. 2005. Use of product viewing histories of users to identify related products. Google Patents.

Liu, N.; Huang, X.; and Hu, X. 2017. Accelerated local anomaly detection via resolving attributed networks. In *IJCAI*, 2337–2343.

McPherson, M.; Smith-Lovin, L.; and Cook, J. M. 2001. Birds of a feather: Homophily in social networks. *Annual Review of Sociology* 27(1):415–444.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

Narayanan, H.; Belkin, M.; and Niyogi, P. 2006. On the relation between low density separation, spectral clustering and graph cuts. In *NIPS*, 1025–1032.

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

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; et al. 2011. Scikit-learn: Machine learning in python. *JMLR* 12:2825–2830.

Pennington, J.; Socher, R.; and Manning, C. 2014. GloVe: Global vectors for word representation. In *EMNLP*, 1532–1543.

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

Smith, A. N.; Fischer, E.; and Yongjian, C. 2012. How does brand-related user-generated content differ across youtube, facebook, and twitter? *Journal of Interactive Marketing* 26(2):102–113.

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

Tang, J.; Zhang, J.; Yao, L.; Li, J.; Zhang, L.; and Su, Z. 2008. Arnetminer: Extraction and mining of academic social networks. In *KDD*, 990–998.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *WWW*, 1067–1077.

Tu, K.; Cui, P.; Wang, X.; Wang, F.; and Zhu, W. 2018. Structural deep embedding for hyper-networks. In *AAAI*, 426–433.

von Luxburg, U. 2007. A tutorial on spectral clustering. *Statistics and Computing* 17(4):395–416.

Wu, L.; Wang, Y.; Gao, J.; and Li, X. 2018. Deep adaptive feature embedding with local sample distributions for person re-identification. *Pattern Recognition* 73:275–288.

Xia, T.; Tao, D.; Mei, T.; and Zhang, Y. 2010. Multiview spectral embedding. *IEEE Trans. on Systems, Man, and Cybernetics* 40(6):1438–1446.

Yang, C.; Liu, Z.; Zhao, D.; Sun, M.; and Chang, E. Y. 2015. Network representation learning with rich text information. In *IJCAI*, 2111–2117.

Yelp. 2017. www.yelp.com/dataset/challenge. *Yelp Dataset Challenge*.

Yuan, Y.-H.; Sun, Q.-S.; and Ge, H.-W. 2014. Fractional-order embedding canonical correlation analysis and its applications to multi-view dimensionality reduction and recognition. *Pattern Recognition* 47(3):1411–1424.

Zhang, J.; Tang, J.; Li, J.; Liu, Y.; and Xing, C. 2015a. Who influenced you? predicting retweet via social influence locality. *TKDD* 9(3).

Zhang, L.; Zhang, Q.; Zhang, L.; Tao, D.; Huang, X.; and Du, B. 2015b. Ensemble manifold regularized sparse low-rank approximation for multiview feature embedding. *Pattern Recognition* 48(10):3102–3112.

Zhang, Q.; Zhang, L.; Du, B.; Zheng, W.; Bian, W.; and Tao, D. 2015c. MMFE: Multitask multiview feature embedding. In *ICDM*.

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