Relation Learning on Social Networks with Multi-Modal Graph Edge Variational Autoencoders

Carl Yang*, Jieyu Zhang*, Haonan Wang*, Sha Li*, Myungwan Kim#, Matt Walker#, Yiou Xiao#, Jiawei Han*

*University of Illinois, Urbana Champaign, 201 N. Goodwin Ave, Urbana, IL 61801, USA

#LinkedIn Co., 599 N. Mathilda Ave, Sunnyvale, CA 94085, USA

*{jiyang3, jieyuz2, haonan3, shal2, hanj}@illinois.edu, #{mukim, mtwalker, yixiao}@linkedin.com

ABSTRACT

While node semantics have been extensively explored in social networks, little research attention has been paid to profile edge semantics, *i.e.*, social relations. Ideal edge semantics should not only show that two users are connected, but also why they know each other and what they share in common. However, relations in social networks are often hard to profile, due to noisy multi-modal signals and limited user-generated ground-truth labels.

In this work, we aim to develop a unified and principled framework that can profile user relations as edge semantics in social networks by integrating multi-modal signals in the presence of noisy and incomplete data. Our framework is also flexible towards limited or missing supervision. Specifically, we assume a latent distribution of multiple relations underlying each user link, and learn them with multi-modal graph edge variational autoencoders. We encode the network data with a graph convolutional network, and decode arbitrary signals with multiple reconstruction networks. Extensive experiments and case studies on two public DBLP author networks and two internal LinkedIn member networks demonstrate the superior effectiveness and efficiency of our proposed model.

KEYWORDS

relation learning, social networks, graph variational autoencoder

ACM Reference Format:

Carl Yang, Jieyu Zhang, Haonan Wang, Sha Li, Myungwan Kim, Matt Walker, Yiou Xiao, Jiawei Han. 2020. Relation Learning on Social Networks with Multi-Modal Graph Edge Variational Autoencoders. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM'20), February 3–7, 2020, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3336191.3371829

1 INTRODUCTION

On social networks, while nodes are explicitly associated with rich contents (*e.g.*, attributes, diffusions), the *semantics* of each link is often implicit. Without such semantics, we cannot truly understand the interaction between users. In this work, we propose and study the problem of *relation learning on social networks*. The goal is to learn the relation semantics underlying each existing link in the social network, which naturally improves the *targeting* of various downstream services, such as friend suggestion, attribute profiling, user clustering, influence maximization and recommendation.

Unlike relation prediction or extraction among entities [2, 9, 13, 20, 22, 29, 38, 40, 41, 55], relation learning on social networks is hard due to the anonymous nature of users, lack of large-scale free-text as context, and very limited labeled data [43]. Moreover, information on social networks is *multi-modal*, *noisy* and *incomplete* [46, 47], leading to various useful but low-quality signals, which are challenging for a unified model to properly *regulate* and *integrate*.

Figure 1 gives an example of a toy social network. As shown in (a), we assume the existence of some latent relation(s) for each link in the network. For example, Tom and Maria are colleagues, whereas Jack and Michael are schoolmates. Furthermore, to better reflect reality, we model each link with a *relation distribution*. For example, the relationship between Tom and Emily is built up by 80% relatives and 20% schoolmates, *i.e.*, they are from the same family, which makes the relative relation dominate their link, but they also go to the same school, thus forming a weaker schoolmate relation. We also allow a link to carry an unknown relation, modeling the uncertainty of relation strength.

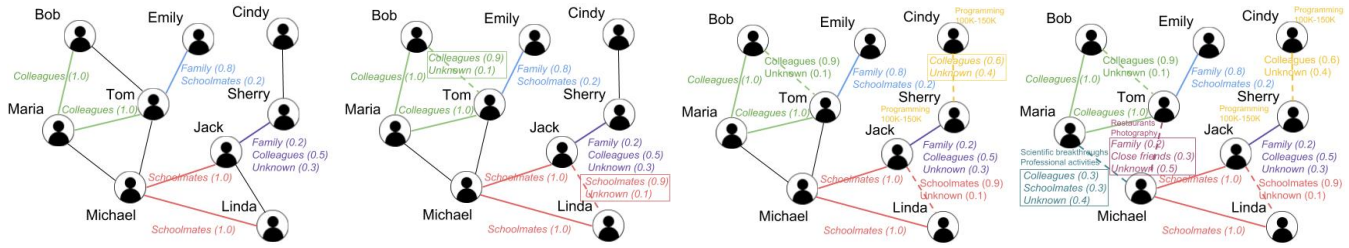
This example also demonstrates three types of signals that are helpful in relation inference.

- **Network proximity.** As illustrated in Figure 1 (b), the network structure is highly useful for inferring unknown relations. If we are confident that Tom and Maria as well as Maria and Bob are colleagues, we can easily deduce that Tom and Bob are also colleagues. Similar situations exist for other pairs like Jack and Linda, who are likely schoolmates.
- **User attribute.** As the *homophily* phenomenon [24, 49] suggests, user attributes can be highly indicative of their relations. As shown in Figure 1 (c), if Cindy and Sherry share similar skills (programming) and salary level (100K-150K), their relation is more likely to be colleagues (*e.g.*, 60%) than others (*e.g.*, 40%).
- **Information diffusion.** As shown in Figure 1 (d), users on social networks often interact in different ways, where links become biased information routes. For example, Maria often shares Michael's posts about scientific breakthroughs or professional activities, while Tom likes to comment on Michael's posts on restaurants and photography. Intuitively, it is more likely for Maria and Michael to be colleagues or schoolmates, and Tom and Michael to be relatives or close friends.

WSDM '20, February 3–7, 2020, Houston, TX, USA

ACM ISBN 978-1-4503-6822-3/20/02…$15.00

https://doi.org/10.1145/3336191.3371829



(a) Latent relations (b) Leveraging network proximities (c) Leveraging user attributes (d) Leveraging information diffusions

Figure 1: A running toy example of a LinkedIn social network of 9 users and 11 links.

Note that in real-world social networks, each of the three types of information can be highly noisy and incomplete. Moreover, high-quality training data is highly limited, if any. This requires a model for relation learning on social networks to be: (1) *powerful* to fully leverage and coherently integrate the multi-modal signals; (2) *robust* to produce reliable results when certain data are missing or inaccurate; (3) *flexible* to operate with limited or no supervision.

In the face of such challenges, we develop ReLearn, a unified multi-modal graph edge variational autoencoder framework. Essentially, our model belongs to the class of unsupervised representation learning models using autoencoders, which has been shown effective for various machine learning tasks [18, 28, 36]. On top of it, we design a Gaussian mixture variational autoencoder to encode link semantics, with the mixture weights representing the distribution over relation types *local* to the link. We further assume *global* relation prototype variables for the latent relations, which are instantiated as a Gaussian distribution in our model. Variational inference with two-step Monte Carlo sampling is designed to infer both the global Gaussian parameters and local relation distributions.

To compute graph edge representations on large-scale social networks, we combine graph convolutional networks (GCN) [17] with fully-connected feedforward networks (FNN) for our encoder, and enable batch-wise training with fixed-size neighborhood sampling. To fully leverage and integrate multi-modal signals, we attach multiple decoders to the GCN-based encoder, which can be flexibly trained with any combination of available signals. Finally, the framework can be trained with varying amount of labeled data by using the labels as priors in the objective function.

We conduct extensive experiments on four real-world large-scale social networks, *i.e.*, two public DBLP author networks and two internal LinkedIn member networks[1]. Through the comparison with various state-of-the-art baselines, we observe consistent significant improvements of 8%-28% over the best baselines. The generative nature of ReLearn further enables interpretable case studies that provide insights into the learned relations.

2 RELATED WORK AND PRELIMINARIES

Social network analysis. Some works on social network analysis have looked into the latent relations underlying uniform social links. Among them, [5, 11, 32] aim to jointly learn user attributes and relations, by assuming the relations to be mutually exclusive and determined by user attributes, whereas [19, 27, 51, 52] attempt to detect groups constructed by homogeneous relations. While both groups of methods implicitly learn the relation semantics, their assumptions about relations are restricted and unrealistic, since relations are not necessarily mutually exclusive and are not only learnable among groups. Moreover, their methods also do not integrate various signals as we consider in this work. [33] leverages text context to encode relation semantics in node embeddings. In comparison, we directly learn edge representations and text is only one of the signals we consider.

Relation learning in other contexts. The problem of relation learning has been intensively studied in knowledge graph completion and relation extraction. Some existing works rely more on the reasoning over existing knowledge graphs with typed links [2, 9, 29, 40, 41], while others leverage more on the modeling of textual contexts with weak supervision [13, 20, 22, 38, 55]. However, on social networks, nodes are untyped as well as links, and they are often anonymous without textual contexts. On the other hand, noisy signals like link structures, user attributes and information diffusions widely exist, which urges us to develop novel models for relation learning on social networks.

Network embedding After the great success of DeepWalk [25], network embedding has attracted much research attention in recent years. We mainly compare with those on content-rich networks. For example, models like TADW [45], PTE [30], Planetoid [53], paper2vec [8], STNE [21], AutoPath [44] and NEP [48] have been designed to improve network embedding by incorporating node contents like types, attributes and texts. Moreover, the convolution based models like GCN [17], GAT [34], GraphSage [10], CANE [33], DiffPool [54], JK-Net [42], FastGCN [6] and DGI [35] naturally take the input of both node features and links. However, most of them cannot be trained in an unsupervised fashion, and none of them can easily incorporate additional signals like information diffusions on networks.

Moreover, a few recent works on diffusion prediction also computes network embedding by modeling the diffusions as DAGs or trees, such as CDSK [3], DCB [1], EmbIC [4], TopoLSTM [39] and inf2vec [7]. In this way, they combine the signals of diffusions and network links. However, they often only care about local network embedding that captures the diffusion structures rather than all links on the network, and they do not integrate node contents.

Variational autoencoders Variational autoencoders (VAEs) [15, 26] combine Bayesian inference with the flexibility of neural networks for robust representation learning. By applying the reparameterization trick, VAE allows the use of standard backpropagation

[1]DBLP source: https://dblp.uni-trier.de/; LinkedIn source: https://www.linkedin.com/

to optimize continuous stochastic variables. In its simplest form, VAE can be viewed as a one-layer latent variable model:

$$p(x, z) = p(z)p(x|z) \tag{1}$$

where x is an observed variable and z is a hidden variable. Using variational inference, the goal is to maximize the evidence lower bound (ELBO):

$$\begin{aligned}\mathcal{L}\left(p_\theta, q_\phi\right) &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x, z) - \log q_\phi(z|x)\right] \\ &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - KL\left(q_\phi(z|x)\|p(z)\right).\end{aligned} \tag{2}$$

We refer readers to [15] for the derivation of this lower bound. Both $q_\phi(z|x)$ and $p_\theta(x|z)$ are parameterized by neural networks. They are referred to as the *encoder network* and the *decoder network*, respectively. The first term in the ELBO is a reconstruction loss that encourages the decoded x to be close to the observed x. The second term is a regularization term where the posterior distribution of z is pulled towards the prior, which is often a simple distribution.

To extend the use of VAE to discrete variables, [12, 23] introduced the Gumbel-Softmax distribution which is a continuous approximation of categorical variables. Given a categorical variable z and its class probabilities $\pi_1, \ldots, \pi_k$, we can sample from this distribution by first sampling k times from the Gumbel(0,1) distribution. The *argmax* operation in the original Gumbel-Max trick is replaced by a softmax operation to ensure the differentiability of the function

$$z_i = \frac{\exp\left(\left(\log\left(\pi_i\right) + g_i\right)/\tau\right)}{\sum_{j=1}^{k}\exp\left(\left(\log\left(\pi_j\right) + g_j\right)/\tau\right)}, \quad \text{for } i = 1, \ldots, k. \tag{3}$$

3 RELEARN

3.1 Problem Definition

Input. As we have discussed in Section 1, we aim to jointly consider multiple signals on social networks that are indicative of relation semantics. We use a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{D}\}$ to model all data we consider in this work. $\mathcal{V} = \{v_i\}_{i=1}^{N}$ is the set of nodes (users). $\mathcal{E} = \{e_{ij}\}_{i,j=1}^{N}$ is the set of edges (links), where $e_{ij} = 1$ denotes an existing link between v_i and v_j, and $e_{ij} = 0$ otherwise. We consider undirected links in this work, while the model can be easily generalized for directed links. $\mathcal{A}$ is the set of node features (user attributes) associated with $\mathcal{V}$, where each $a_i \in \mathcal{A}$ is a fixed-sized vector of dimension L associated with v_i. The exact features encoded in $\mathcal{A}$ is dataset-dependent and we refer the reader to Section 4 for details. $\mathcal{D} = \{d_s\}_{s=1}^{M}$ is the set of *diffusion induced networks* generated from the information diffusions over the network, which we formally define as follows.

Definition 1. *Diffusion Induced Network. A network $d_s = \{\mathcal{V}_s, \mathcal{E}_s, C_s\}$ is a diffusion induced network generated by a piece of information ξ_s that flows on the whole network $\mathcal{N} = \{\mathcal{V}, \mathcal{E}\}$, if $\mathcal{V}_s \subset \mathcal{V}$ is the set of nodes affected by ξ_s, $\mathcal{E}_s \subset \mathcal{E}$ is the set of edges among $\mathcal{V}_s$, and C_s is the contents associated with ξ_s.*

Taking $\mathcal{G}$ as input, our goal is to compute the following output of edge representations $\mathcal{H}$, which in an ideal case should encode the underlying relation semantics we aim to learn from $\mathcal{G}$.

Output. We aim to output $\mathcal{H} = \{h_{ij}\}_{i,j=1}^{N}$ as a set of edge representations. Each $h_{ij} \in \mathcal{H}$ is a fixed-sized vector learned for edge e_{ij}.

We especially care about the representations of existing links (*i.e.*, $e_{ij} = 1$), so as to further understand their underlying relation semantics and make relation predictions through generic classification or clustering algorithms. The representations of non-existing links (*i.e.*, $e_{ij} = 0$) might also be useful for tasks like typed link prediction but is not the focus of this work.

We now formally define the relation learning problem as follows.

Definition 2. *Relation Learning on Social Networks. Given a social network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{D}\}$, learn the edge representation $\mathcal{H}$ by integrating the multiple signals from $\mathcal{E}$, $\mathcal{A}$ and $\mathcal{D}$, which captures the relations underlying $\mathcal{E}$.*

3.2 Model

In this work, we propose ReLearn, a unified model of multi-modal graph edge variational autoencoder. It follows a novel design of a single-encoder-multi-decoder framework, so as to coherently model the multi-modal signals on social networks, and flexibly operate when any of the signals are missing. A robust Gaussian mixture model with global Gaussian distributions and local mixture weights is injected to regulate the latent edge embedding space and capture the underlying relation semantics.

3.2.1 Gaussian Mixture Variational Autoencoder. Motivated by recent success of autoencoders, our idea is to find latent relations that inherently generate the observed various signals on social networks. Following this insight, we believe that the edge representation $\mathcal{H}$, as the *codec* computed via encoding and decoding the observed signals through the autoencoder framework, should reflect the underlying relations and follow a certain relation-specific distribution in the embedding space.

Particularly, we assume $\mathcal{H}$ can be further decomposed into the combination of a relation factor $\mathcal{Z}$ and an embedding factor $\mathcal{W}$:

$$h_{ij} = \sum_{k=1}^{K} z_{ijk} w_{ijk}, \tag{4}$$

where for each pair of nodes v_i and v_j, w_{ij} follows the same set of K independent global multivariate Gaussian distributions, *i.e.*,

$$\forall k = 1, \ldots, K: \; w_{ijk} \sim \mathcal{N}(\mu_k, \sigma_k^2), \tag{5}$$

and z_{ij} follows a local multinomial distribution, *i.e.*,

$$z_{ij} \sim Mul(K, \pi_{ij}), \; \pi_{ij} = (\pi_{ij1}, \ldots, \pi_{ijK}). \tag{6}$$

The idea behind this design is intuitive: We assume there are K possible latent relations, which is directly modeled by the local relation factor $z_{ij} \in \mathbb{R}^K$. The multinomial distribution is chosen to respect the fact that multiple relations can co-exist on the same link. The edge representation h_{ij} is then a weighted summation over the global embedding factor $w_{ij} \in \mathbb{R}^{K \times P}$. We use a multivariate Gaussian to model the edge semantics as a probability distribution instead of a deterministic value so that the uncertainty in the data due to noisy and inaccurate signals can be captured by its variance.

Note that, for any pair of nodes v_i and v_j, w_{ij} follows the same K global Gaussian components, which are fixed across all edges, while the mixture assignment is inferred on each edge. Such a design helps us largely reduce the number of parameters to be learned for $\mathcal{H}$ and alleviate the problem of data sparsity.

To learn the edge embedding $\mathcal{H}$, we assume that all observable signals on social networks are independently generated given $\mathcal{H}$,

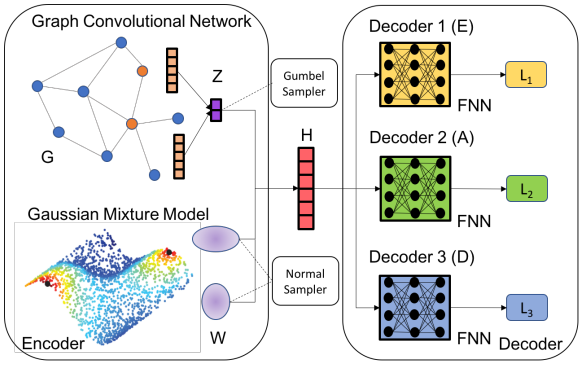


Figure 2: The multi-modal graph edge variational autoencoder architecture of ReLearn.

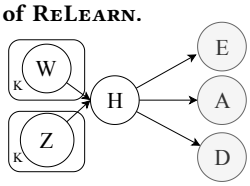


Figure 3: Plate diagram for our variational autoencoder. W is the embedding factor, Z is the relation factor (mixture weight for the Gaussian random variables), H is the edge embedding, E indicates edge existence, A encodes edge attributes and D encodes diffusion information. All random variables are defined separately for each edge.

as reflected in Figure 3. Consider a particular observed signal $\mathcal{X}$ to learn (*e.g.*, if we consider user attribute, then $\mathcal{X} = \mathcal{A}$), we can derive the corresponding evidence lower bound objective (ELBO):

$$\begin{aligned}&\mathcal{L}(p_\theta, q_\phi)\\ =&\mathbb{E}_{q_\phi(Z,W,H|X)}[\log p_\theta(Z,W,H,X) - \log q_\phi(Z,W,H|X)]\\ =&\mathbb{E}_{q_\phi(Z,W,H|X)}[\log p_\theta(X|H)]\\ &- KL[(q_\phi(W)\|p(W)] - KL[q_\phi(Z|X)\|p(Z)].\end{aligned} \tag{7}$$

In the equation, the first term is the reconstruction loss on $\mathcal{X}$, which allows the model to extract useful patterns from observed network signals that are indicative of relation semantics. The second and third terms regularize the latent variables towards the priors. When no prior knowledge is available, the unit Gaussian distribution and uniform multinomial distributions can be applied to regularize $\mathcal{W}$ and $\mathcal{Z}$, respectively. However, when labeled relations are available during training, we can use a smoothed one-hot multinomial distribution per labeled node pair as the prior to effectively inject supervision, *i.e.*,

$$p(z_{ij} = k) = \frac{\mathbb{I}(k = z^*_{ij}) + \eta}{1 + K\eta}, \tag{8}$$

where z^*_{ij} is the ground-truth relation label on e_{ij} and η is a smoothing parameter. In this way, our model can flexibly leverage any amount of supervision, and even work under no supervision.

3.2.2 Graph Edge Encoder. The goal of our encoder network is to output the local relation factor $\mathcal{Z}$, which is combined with the global embedding factor $\mathcal{W}$ to generate the edge embedding $\mathcal{H}$.

GCN [17] has been widely used to compute latent representations from node feature and network structure [16, 50]. To consider multiple signals for edge representations, we design a graph edge encoder based on GCN. Specifically, we have

$$\mathcal{U}^{(l+1)} = \text{ReLU}(\tilde{D}^{-\frac{1}{2}}\tilde{E}\tilde{D}^{-\frac{1}{2}}\mathcal{U}^{(l)}W_g^{(l)}), \tag{9}$$

which is a standard GCN layer. In our setting, $\mathcal{U}^{(0)} = \mathcal{A}$, $\tilde{E} = E + I_N$, $\tilde{D}_{ii} = \sum_j \tilde{E}_{ij}$, and W_g are the learnable GCN parameters. E is the 0-1 edge existence matrix. For the sake of scalability, we implement batch-wise training for GCN via fixed-sized neighborhood sampling [10].

For a pair of nodes v_i and v_j ($i < j$), we concatenate their node features to form an edge feature $\mathbf{y}_{ij} \in \mathcal{Y}$

$$\mathbf{y}_{ij} = [\mathbf{u}_i, \mathbf{u}_j], \tag{10}$$

where $\mathbf{u}_i, \mathbf{u}_j \in \mathcal{U}$ are the node features of v_i and v_j, respectively. In this work, we do not differentiate the head and tail nodes for an edge, since we only consider undirected links in the social networks.

Finally, we add a feed-forward neural network (FNN) with ReLU activations that takes edge features to compute the relation factors as $Z = f_r(Y)$. Altogether, the parameters ϕ to be learned in the encoder network is $\{\phi_g, \phi_r, \phi_w\}$, where ϕ_g is the set of parameters in GCN, ϕ_r is the set of parameters in FNN, and ϕ_w is the set of parameters in the K global relation-specific Gaussian distributions. Detailed configurations of the GCN and FNN are described in Sec. 4.

3.2.3 Multi-Modal Decoder. Figure 2 illustrates our particular design of multi-modal graph edge variational autoencoder that jointly models the network proximities $\mathcal{E}$, user attributes $\mathcal{A}$ and information diffusions $\mathcal{D}$ on social networks, while various other possibly useful signals can be easily plugged in with flexibility upon availability.

In this work, the decoder network consists of three decoders, each of which models the generation process of a particular observed signal given the edge representation $\mathcal{H}$.

(1) A network proximity decoder, which models $p_\theta(\mathcal{E}|\mathcal{H})$.
(2) A user attribute decoder, which models $p_\theta(\mathcal{A}|\mathcal{H})$.
(3) An information diffusion decoder, which models $p_\theta(\mathcal{D}|\mathcal{H})$.

In Eq. 7, we used $\mathcal{X}$ as a placeholder for any possible signal on $\mathcal{G}$. By plugging in all three decoders, we have our final ELBO.

$$\begin{aligned}&\mathcal{L}(p_\theta, q_\phi) = \mathbb{E}_{q_\phi(Z,W,H|G)}[\lambda_1 \log p_\theta(E|H) + \lambda_2 \log p_\theta(A|H) + \lambda_3\\ &\log p_\theta(D|H)] - KL[(q_\phi(W)\|p(W)] - KL[q_\phi(Z|G)\|p(Z)],\end{aligned} \tag{11}$$

where λ_i's are the weighting parameters with $\sum_{i=1}^3 \lambda_i = 1$.

Each of the three decoders are implemented as simple FNNs. Decoder 1 tries to reconstruct links on the network with the following cross-entropy loss on $\mathcal{E}$:

$$\begin{aligned}L_1 =&\mathbb{E}_{q_\phi(Z,W,H|E)}[\log p_\theta(E|H)] = \sum_{i,j} \mathbb{E}_{h\sim q_\phi} \log p_\theta(e_{ij}|h_{ij})\\ =&\sum_{i,j}\{e_{ij} \log \varsigma(f_{d1}(h_{ij})) + (1 - e_{ij}) \log[1 - \varsigma(f_{d1}(h_{ij}))]\},\end{aligned} \tag{12}$$

where $\varsigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function and f_{d1} is the FNN of decoder 1. During training, we sample positive and negative pairs of nodes, where positive samples are from node pairs with observed links (*i.e.*, $e_{ij} = 1$) on $\mathcal{G}$, and for each positive pair, we randomly corrupt one end of the link to get negative samples.

Decoder 2 tries to recover the edge attributes, which are the concatenations of node (user) attributes on the two ends (*i.e.*, $a_{ij} =$

$[a_i, a_j]$). It computes an ℓ_2 loss on $\mathcal{A}$ (constant terms omitted):

$$L_2 = \mathbb{E}_{q_\phi(Z,W,H|A)}[\log p_\theta(A|H)] = \sum_{i,j} \mathbb{E}_{h\sim q_\phi} \log p_\theta(a_{ij}|h_{ij}) = \sum_{i,j} \mathbb{E}_{h\sim q_\phi} \|a_{ij} - f_{d2}(h_{ij})\|_2^2, \tag{13}$$

where f_{d2} is the FNN of decoder 2. Since h_{ij} is the generated from the two-step Monte Carlo sampling, variance has been pushed to the encoder parameters $\mathcal{Z}$ and $\mathcal{W}$.

Decoder 3 tries to recover diffusion contents on links covered by the corresponding diffusions, by computing a similar ℓ_2 loss on $\mathcal{D}$:

$$L_3 = \mathbb{E}_{q_\phi(Z,W,H|D)}[\log p_\theta(D|H)] = \sum_{i,j} \mathbb{E}_{h\sim q_\phi} \log p_\theta(c_{ij}|h_{ij}) = \sum_{i,j} \mathbb{E}_{h\sim q_\phi} \|c_{ij} - f_{d3}(h_{ij})\|_2^2, \tag{14}$$

where f_{d3} is the FNN of decoder 3. For each diffusion induced network d, we sample pairs of nodes that are covered by links in $\mathcal{E}_d$ (where $e_{ij}^d = 1$), and c_{ij} is set to C_d. During training. we firstly sample a diffusion induced network d_s from $\mathcal{D}$, and then only sample positive pairs of nodes *w.r.t.* $\mathcal{E}_s$ and make decoder 3 learn to reconstruct the diffusion contents C_s and diffusion structures $\mathcal{E}_s$ simultaneously.

For the KL-divergence terms:

$$KL(q_\phi(W)\|p(W)) = \sum_{i,j}\sum_{k=1}^{K} KL(q_\phi(w_k)\|\mathcal{N}(0,I)) = \sum_{i,j}\sum_{k=1}^{K} \frac{1}{2}\{\|\sigma_k\|_2^2 + \|\mu_k\|_2^2 - \kappa_H - \log\det(\mathrm{diag}(\sigma_k^2))\}. \tag{15}$$

The unit Gaussian is used as the prior for all Gaussian models in $\mathcal{W}$. κ_H is the dimension of the edge representation $\mathcal{H}$.

For edges with no relation labels, we set the prior $p(Z)$ to be the uniform distribution. When relation labels are available, we set $p(Z)$ to the one-hot distribution and apply Laplace smoothing with parameter η to avoid the magnitude explosion of KL-divergence:

$$KL(q_\phi(Z|E,A,D)\|p(Z)) = \sum_{i,j,\text{unsup}} \{\sum_{k=1}^{K} z_{ijk} \log z_{ijk}\} + \sum_{i,j,\text{sup}} \{\sum_{k=1}^{K} z_{ijk} \log \frac{z_{ijk}}{\mathbb{I}(k = z_{ij}^*) + \eta}\}, \tag{16}$$

where *unsup* and *sup* denote the unsupervised and supervised node pairs respectively, and $z_{ij}^* = k$ means e_{ij} is labeled with the k-th relation. Under this setting, the model is trained in a semi-supervised learning fashion, and we only consider single label supervision in this work.

3.2.4 Training. Training our model involves the learning of all parameters in the encoder network $q_\phi(Z, W, H|G)$ and decoder network $p_\theta(G|Z, W, H)$. As our multi-modal decoders jointly integrate multiple observed signals on social networks, $p_\theta(G|Z, W, H)$ can be further decomposed into

$$p_\theta(G|Z,W,H) = p_{\theta_1}(E|H)p_{\theta_2}(A|H)p_{\theta_3}(D|H), \tag{17}$$

The equation holds because we assume the variable dependence structure in Figure 3, which allows us to learn the whole decoder network p_θ by iteratively optimizing each of the three decoders *w.r.t.* their corresponding losses. During the iterative training process, each decoder is jointly trained with the same encoder q_ϕ, which allows the model to effectively integrate the multiple observed signals, capture the underlying relation semantics and regularize it with proper prior knowledge.

The training of each encoder-decoder combination generally follows that of variational inference for variational autoencoders. We design an efficient variational inference algorithm with two-step Monte Carlo sampling and reparameterization tricks. It allows joint learning of $\mathcal{W}$ and $\mathcal{Z}$, together with other non-stochastic parameters in the encoder and decoder networks through principled Bayesian inference. Except for the particular reconstruction losses, the algorithm works in the exact same way for all three decoders.

Algorithm 1 ReLearn Training

1: **procedure** Training ▷ Input
2: $\mathcal{G}$: the social network; B: batch size; T: number of batches.
3: **for** $t = 1 : T$ **do**
4: **for** $\mathcal{X}$ in $\{\mathcal{E}, \mathcal{A}, \mathcal{D}\}$ **do**
5: Sample B pairs of nodes with observed signals of $\mathcal{X}$.
6: Use the encoder network to compute $q_\phi(Z|G)$.
7: **for** $k = 1 : K$ **do**
8: Draw B random variables $\epsilon_k \sim \mathcal{N}(0, I)$.
9: Compute $\hat{W}_k = \mu_k + \sigma_k \epsilon_k$.
10: Draw B random variables $G_k \sim$ Gumbel(0, 1).
11: Compute $\hat{Z}_k = \frac{\exp((\log(Z_k)+G_k)/\tau)}{\sum_{k'=1}^{K}\exp((\log(Z_{k'})+G_{k'})/\tau)}$.
12: **end for**
13: Compute $H = \sum_{k=1}^{K} \hat{Z}_k \odot \hat{W}_k$.
14: Use the decoder network to compute $p_\theta(X|H)$.
15: Compute the ELBO with q_ϕ and p_θ.
16: Update $\{\phi, \theta\}$ with gradient backpropagation.
17: **end for**
18: **end for**
19: **end procedure**

Without loss of generality, in Algorithm 1, we again use $\mathcal{X}$ to refer to any of the three signals to describe our training process. In Line 8-9 and 10-11, we apply the reparameterization trick to $\mathcal{W}$ and $\mathcal{Z}$ by drawing random samples from the standard Normal distribution and Gumbel distribution [12, 23], respectively, which allows us to push the randomness to the continuous variables ϵ and discrete variables G, and directly optimize the encoder parameters ϕ through standard backpropagation.

As shown in Algorithm 1, besides the sampling process which takes $O(1)$ time for each batch, the whole training process of ReLearn can be done through standard stochastic gradient backpropagation, which allows us to fully leverage well-developed optimization software like mini-batch adam [14] and hardware like GPU. Due to the inductive nature of ReLearn, we do not need to enumerate every pair of nodes in the network. Therefore, the overall computational complexity of training is $O(TBK)$, which are all constant numbers irrelevant of the network size. In other words, the actual training time of ReLearn depends more on the quality and consistency of the network signals than the size of the network.

In our experiments, we observe that the training of ReLearn often converges with $TB = \rho|V|$ with $\rho \in [1, 10]$, which gives a rough computational complexity of $O(|V|)$, where $|V|$ is the number of nodes. This often leads to much less training time than most baselines on the same networks.

4 EXPERIMENTS

4.1 Experimental Settings

4.1.1 Datasets. We use two public DBLP author networks and two internal LinkedIn member networks for our experiments.

In the DBLP networks, nodes are authors and links are co-authorships. Node attributes are generated from publications and information diffusions are generated from citations. Particularly, user attributes are computed by averaging the word embedding[2] of keywords and titles in their publications, which are 300-dim. Information diffusions are generated by firstly selecting papers with $10 - 100$ citations, and construct author subnetworks by including authors who cite the corresponding papers and their links. Diffusion contents are then the paper embedding of the cited paper, which are also 300-dim. We use the ground-truth relation labels of advisor-advisee and colleague relations from [37]. A subnetwork DBLP-Sub is generated by including all pairs of authors with ground-truth relation labels and their direct co-authors. DBLP-All is the whole network with all authors and links on DBLP.

In the LinkedIn networks, nodes are members (users) and links are bi-directional member connections. We generate two relatively small and complete networks of members in Bay Area, US and Australia. Node attributes are generated based on the anonymous user profiles, including features like skills, locations, languages and so on. Numerical features like longitudes and latitudes are directly adopted, whereas categorical features like skills and languages are firstly converted into bag-of-skill and bag-of-language vectors, and then further reduced to smaller dimensions via incremental PCA[3]. The final dimension of user attributes is 466.

Ideally, information diffusions should be generated based on public posts, such as popular articles shared by users. However, due to privacy concern, we could not get that data in this work. Alternatively, we use users' following of influential individuals to model the influence propagation. This following relation is one-directional and different from connections, which we believe to be indicative to users' personal interests. Particularly, we randomly choose influential individuals with $10 - 100$ followers and generate diffusion induced networks by including the followees and their own connections. Diffusion contents are generated by embedding the textual descriptions of the influential individuals from their profile, by averaging the word embedding in the same way as we do for papers on DBLP. The diffusion content vectors are 300-dim.

To generate the ground-truth relation labels, if two connected members attend the same school in the same time, we label their relation as schoolmate, and the same is done for colleague. Note that, we exclude the education and working experience for generating node attributes, because they are highly correlated with the ground-truth relation we use for evaluation. However, this does not weaken the utility of our model, since this reliable generation of schoolmate and colleague relations can only cover a small portion of all observable connections ($< 0.3\%$). Moreover, ReLearn can be used to learn many other relations that cannot be easily verified or even defined (*e.g.*, relatives, townsmen, close friends), in an unsupervised way.

[2] http://nlp.stanford.edu/data/glove.840B.300d.zip

[3] https://scikit-learn.org/stable/auto_examples/decomposition/plot_incremental_pca.html

Dataset	#Nodes	#Links	#Diff.	Rel.(%)
DBLP-Sub	23,418	282,146	100,859	0.4341
DBLP-All	1,476,370	4,109,102	410,822	0.0196
LinkedIn-Bay	1,481,521	67,819,313	45,686	0.2239
LinkedIn-Aus	6,598,127	328,005,877	129,510	0.1592

Table 1: Statistics of the four datasets we use. #Diff. is the number of information diffusions, and Rel.(%) is the coverage of labeled relations over all observable links.

4.1.2 Compared algorithms. Since the problem setting of ReLearn is quite different from relation learning on knowledge graphs, we find a comprehensive list of baselines from the state-of-the-art on network inference and embedding. However, none of the existing models can combine all signals as we consider in this work. Besides existing baselines, we also compare multiple variants of ReLearn to provide in-depth understanding over the utilities of different model components.

- **GraphSage** [10]: One of the strongest and most efficient variants of the popular GCN model that integrates node attributes and link structures for learning network embeddings.
- **STNE** [21]: The state-of-the-art unsupervised text-rich network embedding algorithm based on self-translation of sequences of text embeddings into sequences of node embeddings.
- **PTE** [30]: Extension of the popular network embedding algorithm LINE [31] into text-rich network embedding. We also enable supervision for PTE by constructing multiple bipartite graphs connected by links with different relation labels.
- **Planetoid** [53]: Extension of the popular network embedding algorithm DeepWalk [25] into text-rich network embedding. We also enable supervision for Planetoid through pair-wise sampling for relation prediction.
- **TopoLSTM** [39]: One of the state-of-the-art diffusion prediction model with network embedding. Embedding of edges not covered by any diffusion is computed as the average of the embedding of all neighboring edges.
- **Inf2vec** [7]: Another State-of-the-art diffusion prediction model with network embedding. The same process for TopoLSTM is done for edges not covered by any diffusion.
- ReLearn w/o diff: To study the ability of ReLearn in integrating multiple signals, we decompose the model by removing each decoder. As an example, we show the performance of ReLearn *without* decoder 3 (the information diffusion decoder). We find that with the additional attribute decoder, this model variant still performs better than the base model of GVAE [16].
- ReLearn w/o vae: To study the effectiveness of our novel Gaussian mixture VAE in capturing the latent relations, we remove VAE and directly use the output of the graph edge encoder as the edge representation and input of the multi-modal decoders.
- ReLearn w/o sup: The unsupervised training version of ReLearn by using the uniform multinomial distribution as the prior for the mixture weights Z for all edges.
- Relearn: Our full ReLearn model[4].

The implementations of all existing baselines are provided by their original authors and the parameters are either set as the default values or tuned to the best via standard five-fold cross validation.

[4] Code available at: https://github.com/yangji9181/RELEARN

As for ReLearn, for the encoder network, we use a two-layer GCN, with embedding sizes 200 and 100. We set the number of sampled neighbors to 30. After that, we use a single-layer FNN of size 3 with ReLU activations. The edge embedding size and dimension of Gaussian mixtures are set to 100. For the decoder network, we use three 2-layer FNNs with ReLU activations for the three signals, with sizes 200 and 300. For link reconstruction, we set the positive-negative sampling ratio to 1. The weights of three decoders are simply set to the same. The number of latent relations are set to 2 for all datasets. For training, we set the batch size to 1024 and learning rate set to 0.001 on all datasets. For DBLP datasets, we set the number of batches to 500, and for LinkedIn datasets, we set the number of batches to 5000.

4.1.3 Evaluation metrics. The node embeddings learned by all compared algorithms are concatenated into edge embeddings and then fed into MLPs with the same structure, which is then trained and tested on the same splits of labeled relations. Standard classification accuracy is computed based on the prediction of the MLPs using the network embedding generated by different algorithms.

4.2 Performance Comparison with Baselines

We quantitatively evaluate ReLearn against all baselines on the task of relation learning. Table 2 shows the classification accuracy evaluated for all compared algorithms. The results all passed the significant t-tests with p-value 0.01.

As we can see in Table 2, ReLearn constantly outperforms all baselines by significant margins on all datasets, while the compared algorithms have varying performances. Taking a closer look at the results on different datasets, we observe that the task of learning the schoolmate and colleague relations on LinkedIn is much harder than the adviser-advisee and colleague relations on DBLP. This is probably because the social contents and links are often more noisy and complex than those in the publication networks. ReLearn excels on both of the LinkedIn networks, outperforming the best baseline by 17.9% and 28.5%, respectively. Such significant improvements strongly indicate the power of ReLearn in capturing complex noisy signals on social networks for high-quality relation learning. Moreover, the full ReLearn model also consistently outperforms all other ReLearn variants, which further corroborates the effectiveness of ReLearn in integrating multi-modal network signals and limited supervision.

4.3 In-depth Model Analysis

To comprehensively evaluate the performance of ReLearn in comparison with the baselines, we design a series of in-depth analysis, by varying the amount of training data, as well as adding noise and sparsity to the network signals.

Efficiency towards limited training data. One major challenge of relation learning on social networks is the lack of high-quality relation labels. Therefore, an ideal model should be efficient in leveraging limited training data. To study such efficiency of ReLearn, we conduct experiments on all datasets with varying amounts of training data. Particularly, for each of the 4:1 splitting of training and testing data, we use 10% - 100% of the 80% training data to train ReLearn and all compared algorithms, and evaluate on the 20% testing data. The results on DBLP-All and LinkedIn-Bay are presented in Figure 4.

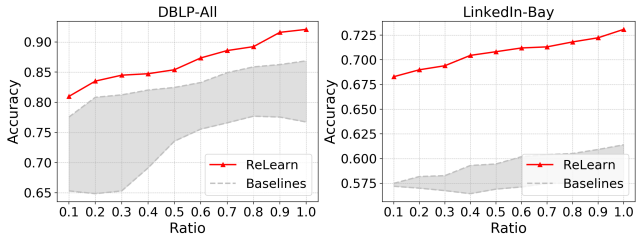


Figure 4: Varying amounts of training data.

Robustness towards attribute noise. On real-world social networks, user attributes are often highly noisy, since users might fill in various free-style contents and even random contents. Therefore, an ideal model for relation learning should be robust towards attribute noise. To study such robustness of ReLearn, we conduct experiments on all datasets by adding different amounts of random noise onto the user attributes. Particularly, since all models take the normalized numerical embedding of attributes as input, we add the unit multivariate Gaussian noise scaled by 0.1-0.5 to the attribute vector of each user. The modified input for all compared algorithms (including ReLearn) is the same. The results on DBLP-All and LinkedIn-Bay are presented in Figure 5.

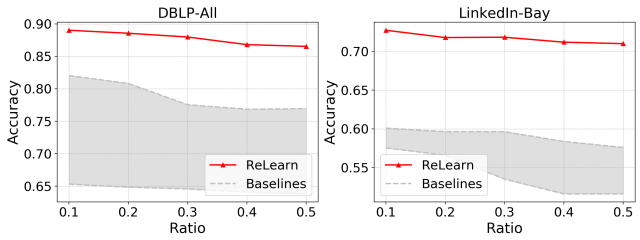


Figure 5: Varying amounts of attribute noise.

Robustness towards missing links. On real-world social networks, real-world friends may not necessarily have established links. Therefore, an ideal model for relation learning should be robust towards missing links. To study such robustness of ReLearn, we conduct experiments on all datasets by randomly removing existing links in the network. Particularly, we randomly remove 2%-10% of links in the whole networks. The modified input for all compared algorithms (including ReLearn) is the same. The results on DBLP-All and LinkedIn-Bay are presented in Figure 6.

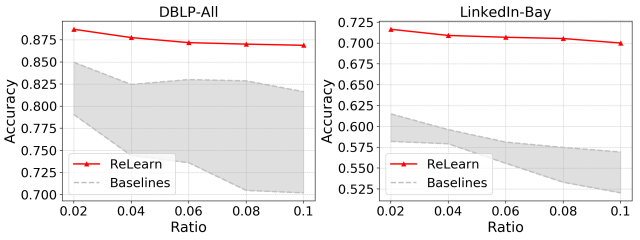


Figure 6: Varying amounts of link removal.

Algorithm	DBLP-Sub	DBLP-All	LinkedIn-Bay	LinkedIn-Aus
GraphSage	0.8596 ± 0.0201	0.8482 ± 0.0158	0.6139 ± 0.0367	0.5831 ± 0.0072
STNE	0.7577 ± 0.0425	0.7434 ± 0.0214	0.5695 ± 0.0236	0.5554 ± 0.0160
PTE	0.7265 ± 0.0018	0.6988 ± 0.0222	0.5636 ± 0.0378	0.5549 ± 0.0041
Planetoid	0.8531 ± 0.0205	0.8686 ± 0.0206	0.5608 ± 0.0301	0.5448 ± 0.0045
TopoLSTM	0.6675 ± 0.0435	0.7374 ± 0.0149	0.5874 ± 0.0257	0.5616 ± 0.0062
Inf2vec	0.6618 ± 0.0401	0.7453 ± 0.0181	0.6198 ± 0.0388	0.5848 ± 0.0068
ReLearn w/o diff	0.8890 ± 0.0031	0.8465 ± 0.0138	0.6616 ± 0.0390	0.6934 ± 0.0022
ReLearn w/o vae	0.8433 ± 0.0154	0.8376 ± 0.0060	0.6293 ± 0.0194	0.6626 ± 0.0087
ReLearn w/o sup	0.8947 ± 0.0170	0.8980 ± 0.0115	0.6771 ± 0.0211	0.7134 ± 0.0048
ReLearn	**0.9224 ± 0.0026**	**0.9208 ± 0.0042**	**0.7308 ± 0.0457**	**0.7514 ± 0.0033**

Table 2: Relation learning accuracy of compared algorithms on four real-world social networks.

Area I	Area II	Area III
Define | Create . . . Implement | Support | Succeed	Writer, Dancer, Entrepreneur . . .	. . . Benefits Negotiation, Salary Negotiation
Training, Program Development, Exercise Prescription . . .	Blogger & Youtuber . . .	Corporate Advisor | Investment Banker . . . Shareholder Representative
. . . Sponsorship Program Development, Fellowship Application	FASHION, BEAUTY, TRAVEL, LIFE . . .	Project Manager | Leader . . . Performance Manager | PA/EA
. . . Talent Management & Success Planning	Social & Environmental Justice . . .	Recruitment, Performance Management . . . Gap Management
Talent Acquisition, Recruiting, Head Hunting . . .	Chef Traditional Italian . . . Proactive	Change & Transition Management, Programme Management . . .
Recruitment . . . Development, Relationship Management	Wellness Coach-Clean Food . . . Warrior-Positive Thinker	People Management, Performance Coaching, Human Resource . . .
An Entrepreneur. A Scholar . . .	Food . . . Driven & Hungry	. . . Beautiful Web Design & Digital Media Solutions
Portfolio Building | Training . . .	. . . A Bohemian Fashion Boutique	Test Automation, Test Management, Technical Testing . . .
Learning & Development, Organisational Culture, Engagement . . .	Licensed Waterproofing Technician . . .	Intellectual Property . . .

Table 3: Decoded diffusion contents on edges generated with three different latent relations.

Remarks on runtimes. While the exact runtimes of compared algorithms are hard to determine due to different convergence rates of each train, we observe that the runtime of ReLearn is close to the more efficient baselines like PTE, Planetoid and GraphSage, and is often significantly shorter than the heavier baselines like STNE, TopoLSTM and Inf2vec.

5 CASE STUDIES

To observe how ReLearn captures the relation semantics among users with learned edge representations, we visualize the embedding space by plotting some of the labeled edges in the LinkedIn-Aus network. We employ standard PCA to reduce the embeddings from 100-dim to 2-dim for plotting. As we can see from Figure 7, edges carrying the two relations clearly form two clusters.

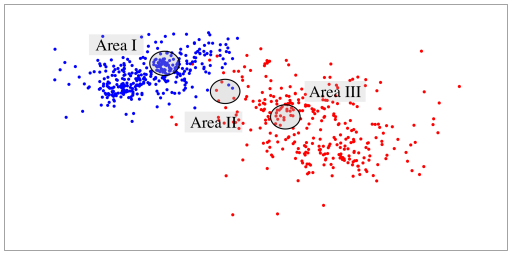


Figure 7: Visualization of edge representations on LinkedIn-Aus computed by ReLearn. Red and blue colors denote the ground-truth labels of schoolmate and colleague.

Moreover, the generative nature of ReLearn allows us to further interpret the learned latent relations, by sampling edge representations from the learned Gaussian mixture model and decoding them with the multiple learned decoders. This is especially useful in the unsupervised learning scenario, where besides the latent distributions, we also want to make sense of the learned relations.

In Table 3, as an example, we show the decoded textual feature from decoder 3 (*i.e.*, the information diffusion decoder), which provides valuable insights into the learned relations. The edge representations are generated by sampling from the Gaussian distribution of W_1, W_2 and a uniform mixture of W_1 and W_2, which roughly corresponds to the three marked areas in Figure 7.

As we can observe in Table 3, edges in Area I likely carry the schoolmate relation, with decoded contents mainly about Learning and Advising, whereas Area III clearly corresponds to colleagues, due to decoded topics like Management and Performance. Edges in Area II hold a mixture of the two relations, with more personal life oriented contents like Food, Travel, Wellness, *etc.* Although the encoder does not directly consider information diffusion, it effectively helps the decoder to capture this information during the joint training process.

Note that, in this example, we already know that the two relations we learn are schoolmates and colleagues, which we use as a verification of the utility of ReLearn. In the more realistic situations where we have no access to ground truth, the multiple decoders of ReLearn still provide meaningful interpretations over the learned relations, which are valuable for downstream services like relation-specific friendship recommendation and content routing.

6 CONCLUSION

In this work, for the novel and challenging problem of relation learning on social networks, we develop ReLearn, a multi-modal graph edge variational autoencoder framework to coherently combine multiple signals on social networks towards the capturing of underlying relation semantics on user links. Moreover, the generative nature of ReLearn allows us to sample relational pairs for interpreting the learned relations, while its inductive nature enables efficient training regardless of the network sizes. Finally, the general and flexible design of ReLearn makes it readily applicable to any real-world social platforms with multi-modal network signals, where the learned node and edge embeddings can be used to improve the targeting of various downstream services.

ACKNOWLEDGEMENT

Research was sponsored in part by U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C-0099 and FA8750-19-2-1004, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, DTRA HDTRA11810026, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov).

REFERENCES

[1] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *NIPS*. 1993–2001.
[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*. 2787–2795.
[3] Simon Bourigault, Cedric Lagnier, Sylvain Lamprier, Ludovic Denoyer, and Patrick Gallinari. 2014. Learning social network embeddings for predicting information diffusion. In *WSDM*. 393–402.
[4] Simon Bourigault, Sylvain Lamprier, and Patrick Gallinari. 2016. Representation learning for information diffusion through social networks: an embedded cascade model. In *WSDM*. 573–582.
[5] Deepayan Chakrabarti, Stanislav Funiak, Jonathan Chang, and Sofus A Macskassy. 2014. Joint Inference of Multiple Label Types in Large Networks. In *ICML*.
[6] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: fast learning with graph convolutional networks via importance sampling. In *ICLR*.
[7] Shanshan Feng, Gao Cong, Arijit Khan, Xiucheng Li, Yong Liu, and Yeow Meng Chee. 2018. Inf2vec: Latent Representation Model for Social Influence Embedding. ICDE.
[8] Soumyajit Ganguly and Vikram Pudi. 2017. Paper2vec: Combining graph and text information for scientific paper representation. In *ECIR*. 383–395.
[9] Matt Gardner and Tom Mitchell. 2015. Efficient and expressive knowledge base completion using subgraph feature extraction. In *EMNLP*. 1488–1498.
[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*. 1025–1035.
[11] Jingrui He, Jaime G Carbonell, and Yan Liu. 2007. Graph-Based Semi-Supervised Learning as a Generative Model.. In *IJCAI*, Vol. 7. 2492–2497.
[12] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *ICLR*.
[13] Meng Jiang, Jingbo Shang, Taylor Cassidy, Xiang Ren, Lance M Kaplan, Timothy P Hanratty, and Jiawei Han. 2017. MetaPAD: Meta Pattern Discovery from Massive Text Corpora. In *KDD*. 877–886.
[14] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.
[15] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. *ICLR* (2014).
[16] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. In *NIPS Workshop on Bayesian Deep Learning*.
[17] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[18] Quoc V Le. 2013. Building high-level features using large scale unsupervised learning. In *ICASSP*. 8595–8598.
[19] Jure Leskovec and Julian J Mcauley. 2012. Learning to discover social circles in ego networks. In *NIPS*. 539–547.
[20] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*, Vol. 1. 2124–2133.
[21] Jie Liu, Zhicheng He, Lai Wei, and Yalou Huang. 2018. Content to node: Self-translation network embedding. In *KDD*. 1794–1802.
[22] Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. 2017. Heterogeneous Supervision for Relation Extraction: A Representation Learning Approach. In *EMNLP*.
[23] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*.
[24] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.
[25] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.
[26] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*.
[27] Yiye Ruan, David Fuhry, and Srinivasan Parthasarathy. 2013. Efficient community detection in large networks using content and links. In *WWW*. 1089–1098.
[28] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks* 61 (2015), 85–117.
[29] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*. 926–934.
[30] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*. 1165–1174.
[31] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. 1067–1077.
[32] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *KDD*. ACM, 817–826.
[33] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *ACL*, Vol. 1. 1722–1731.
[34] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.
[35] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. In *ICLR*.
[36] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*. 1096–1103.
[37] Chi Wang, Jiawei Han, Yuntao Jia, Jie Tang, Duo Zhang, Yintao Yu, and Jingyi Guo. 2010. Mining advisor-advisee relationships from research publication networks. In *KDD*. 203–212.
[38] Chenguang Wang, Yangqiu Song, Dan Roth, Chi Wang, Jiawei Han, Heng Ji, and Ming Zhang. 2015. Constrained Information-Theoretic Tripartite Graph Clustering to Identify Semantically Similar Relations.. In *IJCAI*. 3882–3889.
[39] Jia Wang, Vincent W Zheng, Zemin Liu, and Kevin Chen-Chuan Chang. 2017. Topological recurrent neural network for diffusion prediction. In *ICDM*. 475–484.
[40] Quan Wang, Bin Wang, Li Guo, and others. 2015. Knowledge Base Completion Using Embeddings and Rules. In *IJCAI*. 1859–1866.
[41] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *EMNLP*. 1591–1601.
[42] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*.
[43] Carl Yang and Kevin Chang. 2019. Relationship Profiling over Social Networks: Reverse Smoothness from Similarity to Closeness. In *SDM*. 342–350.
[44] Carl Yang, Mengxiong Liu, Frank He, Xikun Zhang, Jian Peng, and Jiawei Han. 2018. Similarity Modeling on Heterogeneous Networks via Automatic Path Discovery. In *ECML-PKDD*.
[45] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network Representation Learning with Rich Text Information.. In *IJCAI*. 2111–2117.
[46] Carl Yang, Xiaolin Shi, Luo Jie, and Jiawei Han. 2018. I Know You'll Be Back: Interpretable New User Clustering and Churn Prediction on a Mobile Social Application. In *KDD*.
[47] Carl Yang, Dai Teng, Siyang Liu, Sayantani Basu, Jieyu Zhang, Jiaming Shen, Chao Zhang, Jingbo Shang, Lance Kaplan, Timothy Harratty, and others. 2019. Cubenet: Multi-facet hierarchical heterogeneous network construction, analysis, and mining. In *KDD demo*.
[48] Carl Yang, Jieyu Zhang, and Jiawei Han. 2019. Neural Embedding Propagation on Heterogeneous Networks. In *ICDM*.
[49] Carl Yang, Lin Zhong, Li-Jia Li, and Luo Jie. 2017. Bi-directional joint inference for user links and attributes on large social graphs. In *WWW*.
[50] Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, and Pan Li. 2019. Conditional Structure Generation through Graph Variational Generative Adversarial Nets. In *NeurIPS*.
[51] Jaewon Yang, Julian McAuley, and Jure Leskovec. 2013. Community detection in networks with node attributes. In *ICDM*. 1151–1156.
[52] Tianbao Yang, Rong Jin, Yun Chi, and Shenghuo Zhu. 2009. Combining link and content for community detection: a discriminative approach. In *KDD*. 927–936.
[53] Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*.
[54] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *NIPS*. 4805–4815.
[55] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING*. 2335–2344.