Group Link Prediction Using Conditional Variational Autoencoder

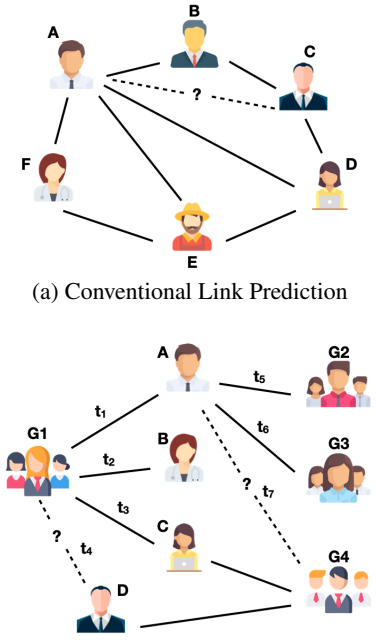
Hao Sha, Mohammad Al Hasan, George Mohler
Indiana University Purdue University Indianapolis
723 W Michigan St, SL 280
Indianapolis, Indiana 46202
haosha@iupui.edu, alhasan@iupui.edu, gmohler@iupui.edu

Abstract

Though link prediction is a well-studied problem with a large body of solutions, existing methods do not handle the case where the predicted link is between an individual and a group. This limitation prevents link prediction models from being directly applicable to many real-life prediction problems where more than two individuals are involved. Examples of such problems include: finding missing members of a group, soliciting new members to join a group, or recommending a group for a person to join. The central aspect of all of the above problems is predicting linkage between a group and an individual, a task we refer to as *group link prediction*. In this work we propose an innovative method consisting of two complementary models for solving group link prediction: group-to-individual-link (member-recommendation) and individual-to-group-link (group-recommendation). Both of the proposed models use a conditional variational auto-encoder (CVAE) for solving the respective task. For a given individual (or group), the model learns a conditional probability distribution of a link for each candidate group (or individual) link. We compare our proposed model and a collection of competing models on various real-world datasets and show the superior performance of the proposed model in group link prediction.

Introduction

Link prediction (Liben-Nowell and Kleinberg 2003) is a widely studied problem with successful applications in social networks (Liben-Nowell and Kleinberg 2003), co-authorship networks (Hasan et al. 2006), protein-protein interactions (Lei and Ruan 2012) and item recommendation (Chen, Li, and Huang 2005). As illustrated in Fig. 1a, given the current state of a network (say, a friendship network), the conventional link prediction task looks at a pair of disconnected nodes (such as A and C in Fig. 1a) and predicts if a link will form between them at a future time. Numerous machine learning models have been proposed for solving this task; for a comprehensive listing of the models, see the following surveys (Lü and Zhou 2011; Al Hasan and Zaki 2011).



(a) Conventional Link Prediction

(b) Group Link Prediction

Figure 1: Conventional link prediction and group link prediction in social networks. (a) In link prediction, links are between a pair of individuals. (b) In group link prediction, links are between a group and an individual. The goal is to predict future links given the current state of the network.

In many real-world networks, patterns of link formation are not exclusively limited to two nodes. For instance, in a co-authorship network, more than two people may co-author an article. Likewise, in a network of online groups, where members sharing a group are connected by edges, the addition of a new individual to a group creates links between the individual and all of the existing members of that group. For accurately predicting links in such networks, one would need to consider the collective link formation between an individual and a group of individuals, a task which we refer to as "group link prediction".

Fig. 1 illustrates the difference between traditional link prediction and group link prediction. In the top graph, we

are simply interested to know whether A and C, two disconnected nodes, will be connected in the future. In the bottom graph, we have two types of nodes: individuals and groups. Observing that a group G_1 connects with individuals A, B, and C at time t_1, t_2, and t_3, we would like to know the likelihood that G_1 will link with individual D at time t_4. Conversely, knowing that individual A joins groups G_2 and G_3 at time t_5 and t_6, we would like to predict if she will join group G_4 at time t_7.

Many real-life problems related to social media can be modeled as a task of group link prediction. For instance, Facebook makes suggestions for potential users to join certain Facebook Groups (e.g. sports enthusiast group, animal lover group)—a task of predicting/recommending a potential group to a user. The same is true for LinkedIn's recommendation of potential employers to a job-seeker. The above are examples of an individual connecting to a group given a list of possible groups—a task, what we also name as **group-recommendation**. The role of group and individual can be reversed in other applications. For instance, given a group (a partial list of emails), Gmail uses auto-complete to recommend additional email recipients. In this case, a group is creating a link to an individual out of many possible choices, which we call **member-recommendation**. Other examples of such tasks are below: a jobseeker is recommended to a potential employers, based on how much she aligns with the current employees at the company; Meetup.com (an event-based social network platform) recommends a user for an event, based on who else are participating in that event.

The existing link prediction solutions (Liben-Nowell and Kleinberg 2003; Hasan et al. 2006; Chen, Li, and Huang 2005) make use of a pair-wise score $r(u, v)$ to measure the similarity between entities u and v. Often $r(u, v)$ is constructed based on some topological properties of the network, including neighbor-based scores like Common Neighbor, Jaccard Index, Adamic/Adar and Preferential Attachment, and path-based scores like Graph Distance, Katz$_\beta$, and hitting time, or scores from dot product of latent representation of u and v obtained through matrix factorization (Koren, Bell, and Volinsky 2009). Given the current state of the network, one predicts the pair (u, v) with the highest $r(u, v)$ to form a link in the future.

Link prediction can also be posed as a binary classification problem with the labels indicating whether two entities are connected (Hasan et al. 2006). In this approach, the features may be node embeddings obtained through static methods like DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), Node2Vec (Grover and Leskovec 2016), and LINE (Tang et al. 2015); dynamic methods like TNE (Zhu, Steeg, and Galstyan 2016), DynamicTriad (Zhou et al. 2018), CTDNE (Nguyen et al. 2018), and HTNE (Zuo et al. 2018); attribute methods like Graph2Gauss (Bojchevski and Günnemann 2018), Neural-Brane (Dave et al. 2018), and DANE (Li et al. 2017); graph neural network (GNN) methods like Graph Convolutional Networks (GCN) (Kipf and Welling 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017a), and Deep Graph Infomax (DGI) (Veličković et al. 2019)

One possible adaptation of existing methods for solving group link prediction is to aggregate the pair-wise similarity scores $r(u, v)$ of traditional link prediction over a group, through functions like sum, mean, max, and min, to obtain the corresponding score $r(g, v)$ between a group g and an individual v. For instance, we can use the maximum $r(u, v)$ for $u \in g$ as $r(g, v)$. Similarly, for adapting the classification-based approach, we can aggregate (e.g. sum, max-pool, etc) the embedding vectors of all group members to represent the entire group.

Group link prediction can also be cast as link prediction on heterogeneous networks (Shi et al. 2017). With individuals and groups as nodes, we can construct a bipartite heterogeneous network as illustrated in Fig. 1b. In this context, without aggregation, $r(g, v)$ can be directly obtained from measures such as random walk with restart (Lao and Cohen 2010) and meta-path similarity (Sun and Han 2012). Moreover, nodal representations can be learned for link prediction, via methods like meta-path guided random walks (Dong, Chawla, and Swami 2017) and matrix factorization (Shi et al. 2019).

While potentially easy to implement, these adaptations are not specifically designed for group link prediction and, as we will show in experiments to follow, are not consistently effective for group link prediction. In this work, we propose a Conditional Variational Auto-encoder (**CVAE**) (Sohn, Lee, and Yan 2015) based model specially designed for solving both the **member-recommendation** task and the **group-recommendation** task. In addition, we provide a variant of the CVAE model - Conditional Variational Auto-encoder with History (**CVAEH**) to incorporate the temporal characteristics, where the historical links are considered. The contributions of this work are three-fold:

1. We reframe a special case of link prediction on heterogeneous networks that considers the links between an individual and a group, which we call "group link prediction".
2. We propose a CVAE-based model to solve group link prediction. We also introduce a second CVAE model (named CVAEH) that considers the temporal effect by incorporating the historical links.
3. We examine the group link prediction problem in five real-world datasets and show the superiority of our CVAE/CVAEH models in comparison with various competing methods.

The rest of the paper is organized as follows. In Method, we give a detailed description of our model. In Related Works, we review other works related to group link prediction. In Experimental Results, we provide details on the datasets and the various baseline models used for comparison. We then present several group link prediction experiments and the results. Finally, we summarize our work in Conclusions.

Method

Problem Description

First we describe the terminology. Given a set of n individuals $G = \{v_1, v_2, \cdots, v_n\}$, a subset of them form a group $g_i \subseteq G$ to participate in an event e_i at time t_i, i.e. $e_i = (t_i, g_i)$. We denote a sequence of historical

events up to time t_i as $\mathcal{H}(t_i) = \{e_1, e_2, \cdots, e_{i-1}\} = \{(t_1, g_1), (t_2, g_2), \cdots, (t_{i-1}, g_{i-1})\}$. Note, throughout the paper, v denotes an individual and g denotes a group. We use the subscription "ob" to indicate that the subject is observed, and "unob" to indicate the opposite.

Group link prediction is to predict the temporal association between an individual v and a group g in an event e at time t. It can be decomposed into two sub-problems - member-recommendation and group-recommendation. In essence, they differ in what is observed and what is to predict. If we observe a group of individuals in an event, and want to predict an individual to join them, then we have a member-recommendation problem; if we observe an individual, and want to predict a group for the individual to join an event with, then we have a group-recommendation problem. To facilitate ground truth based evaluation, we formalize them as the following:

Definition 0.1. Member-recommendation Given an event $e_i = (t_i, g_i)$ and the history $\mathcal{H}(t_i)$, we randomly hold out a member $v_{unob,i} \in g_i$ from g_i as *unobserved*, and denote the rest of the group $g_{ob,i} = g_i \setminus \{v_{unob,i}\}$ as *observed*. The member-recommendation task is to recommend an individual $v \in G$ for $g_{ob,i}$ such that $v = v_{unob,i}$, provided that $g_{ob,i}$ and $\mathcal{H}(t_i)$ are known.

Definition 0.2. Group-recommendation Given an event $e_i = (t_i, g_i)$ and the history $\mathcal{H}(t_i)$, we randomly select an individual $v_{ob,i} \in g_i$ as *observed*, and denote the rest of the group $g_{unob,i} = g_i \setminus \{v_{ob,i}\}$ as *unobserved*. The group-recommendation task is to recommend a group g for $v_{ob,i}$ such that $g = g_{unob,i}$, provided that $v_{ob,i}$ and $\mathcal{H}(t_i)$ are known.

Remark. For group-recommendation, in general, we need to rank every group g for a given individual $v_{ob,i}$, which is intractable as the number of possible g is 2^n. Therefore, we simplify group-recommendation to picking the positive group $g_{pos} = g_{unob,i}$ which $v_{ob,i}$ actually joins at time t_i, out of a set of m negative groups $\{g_{neg,i} | i = 1, 2, \cdots, m\}$, which $v_{ob,i}$ does not join.

Remark. Although our proposed models can be easily adapted to the more general task of recommending one group for another (or given a partial group, predicting the rest of the group), we leave this exciting direction to future work.

Preliminaries

Variational Autoencoder A variational autoencoder (VAE) (Kingma and Welling 2014) is an unsupervised neural network model for embedding high-dimensional data into a latent space. Unlike a traditional autoencoder, this is accomplished by approximating the true distribution $P_{real}(\mathbf{x})$ for the data $\mathbf{x}$ through a variational Bayes approach. VAEs have the added advantage of being capable of generating artificial data that resembles the real data. The generative process starts by sampling a latent variable $\mathbf{z}$ from a prior Gaussian distribution $P(\mathbf{z})$. The VAE then generates a data point $\mathbf{x}$ conditioning on $\mathbf{z}$, using a generative distribution $P_\theta(\mathbf{x}|\mathbf{z})$, where θ are the parameters of the generative model. Usually $P(\mathbf{z})$ is assumed to be standard Gaussian and a neural network is used to model $P_\theta(\mathbf{x}|\mathbf{z})$. To obtain the parameters θ, one can maximize the log-likelihood $logP_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim P(\mathbf{z})}[logP_\theta(\mathbf{x}|\mathbf{z})P(\mathbf{z})]$. In practice, directly maximizing this log-likelihood is inefficient, as for most $\mathbf{z}$, $P_\theta(\mathbf{x}|\mathbf{z})$ will be nearly zero, and contribute almost nothing to the estimate of $P_\theta(\mathbf{x})$. Instead, it would be more efficient to sample from the posterior distribution $P(\mathbf{z}|\mathbf{x})$. However, in general, the posterior distribution inference is intractable. To alleviate the difficulty, one can approximate $P(\mathbf{z}|\mathbf{x})$ by a proposal distribution $Q_\phi(\mathbf{z}|\mathbf{x})$, where ϕ denotes the model parameters. Furthermore, (Kingma and Welling 2014) proposed to maximize a variational lower bound of the log-likelihood as the following:

$$\begin{aligned} logP_\theta(\mathbf{x}) &= \mathop{\mathbb{E}}_{Q_\phi(\mathbf{z}|\mathbf{x})}[log\frac{P_\theta(\mathbf{x},\mathbf{z})}{Q_\phi(\mathbf{z}|\mathbf{x})}] + KL(Q_\phi(\mathbf{z}|\mathbf{x})||P(\mathbf{z}|\mathbf{x})) \\ &\geq \mathop{\mathbb{E}}_{Q_\phi(\mathbf{z}|\mathbf{x})}[logP_\theta(\mathbf{x}|\mathbf{z})] - KL(Q_\phi(\mathbf{z}|\mathbf{x})||P(\mathbf{z})) \\ &= L_{VAE}(\mathbf{x};\phi,\theta), \end{aligned} \tag{1}$$

where KL denotes the Kullback–Leibler divergence. In Eq. 1, $Q_\phi(\mathbf{z}|\mathbf{x})$ is essentially an encoder, mapping a data point $\mathbf{x}$ to $\mathbf{z}$ in the latent space. The generative model $P_\theta(\mathbf{x}|\mathbf{z})$, on the other hand, acts as a decoder, converting the sampled latent vector $\mathbf{z}$ back to the data space. In practise, $Q_\phi(\mathbf{z}|\mathbf{x})$ is a neural network that maps a data point to two vectors - mean μ and the diagonal elements of the standard deviation σ. The reparameterization trick is used to sample $\mathbf{z}$ from $\mu + \sigma \odot \mathbf{z_0}$, where $\mathbf{z_0} \sim \mathcal{N}(0, I)$. The lower bound $L_{VAE}(\mathbf{x};\phi,\theta)$ can thus be optimized using stochastic gradient ascent w.r.t ϕ and θ. The optimal θ should render $P_\theta(\mathbf{x})$ resembling the true data distribution $P_{real}(\mathbf{x})$.

Conditional Variational Autoencoder A conditional variational auto-encoder (CVAE) (Sohn, Lee, and Yan 2015) approximates the conditional probability distribution $P_{real}(\mathbf{x}|\mathbf{y})$, where $\mathbf{x}$ is the data and $\mathbf{y}$ is the observed features. For instance, if $\mathbf{x}$ is an MNIST handwritten digit image and $\mathbf{y}$ encodes a number, then CVAE would generate an image of that number. In contrast, VAE would generate images of any number between 0 and 9. Like the VAE model, a variational lower bound defined as the following can be maximized for training:

$$\begin{aligned} logP_\theta(\mathbf{x}|\mathbf{y}) \geq & \mathop{\mathbb{E}}_{Q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[logP_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})] \\ & - KL(Q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})||P(\mathbf{z}|\mathbf{y})) \\ & = L_{CVAE}(\mathbf{x},\mathbf{y};\phi,\theta), \end{aligned} \tag{2}$$

where $Q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$, $P_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})$, and $P(\mathbf{z}|\mathbf{y})$ represent the encoder, decoder, and conditional prior networks, respectively. Note, in our implementation, we do not explicitly construct a conditional prior network. Instead, we adopt the reparameterization trick to model the conditional prior $P(\mathbf{z}|\mathbf{y})$. Also note, unlike the existing applications of CVAE focusing on image generation/reconstruction (Sohn, Lee, and Yan 2015; Ivanov, Figurnov, and Vetrov 2019), we use CVAE to learn a conditional probability for the unobserved group member(s) given the observed one(s).

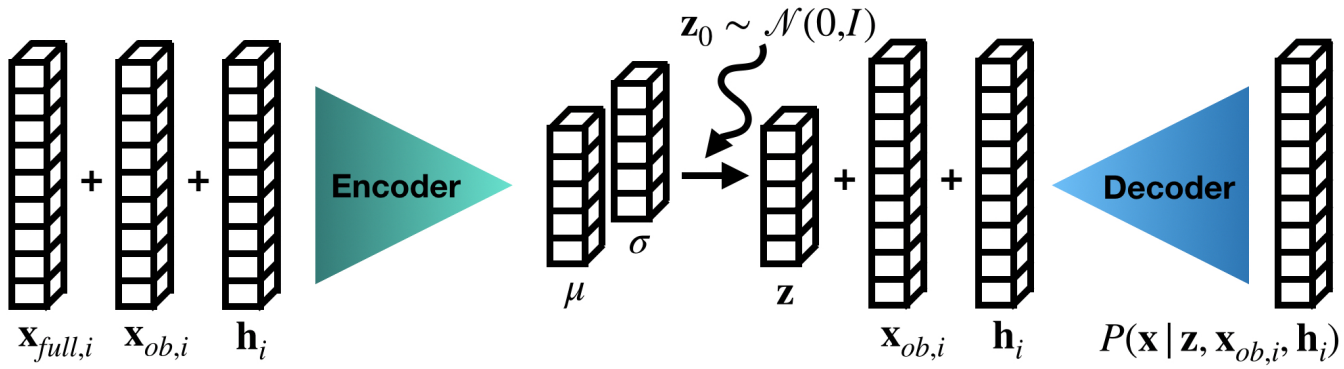


Figure 2: Network architecture. $\mathbf{x}_{full,i}$ and $\mathbf{x}_{ob,i}$ are encoding vectors representing the entire group and observed member(s), which are concatenated as the input to the encoder (green). $\mathbf{h}_i$ contains the historical event counts, which is also concatenated with the input for the CVAEH model. Mean μ and the diagonal elements of standard deviation σ are encoder outputs, with which we sample the latent vector $\mathbf{z}$. $\mathbf{z}$ is concatenated with $\mathbf{x}_{ob,i}$ and $\mathbf{h}_i$ (CVAEH only) and fed to the decoder (blue). The output $P(\mathbf{x}|\mathbf{z}, \mathbf{x}_{ob,i}, \mathbf{h}_i)$ is the conditional probability indicating the likelihood for $\mathbf{x}$ to join $\mathbf{x}_{ob,i}$.

Member-recommendation

In this section, we provide detailed description of our CVAE model and its variant—the CVAEH model, in the context of member-recommendation. We will explain how to adapt them for group-recommendation in the next section. The network architectures of these two models are illustrated in Fig. 2.

CVAE In Fig. 2, $\mathbf{x}_{full,i}$ is an n-dimensional many-hot vector that encodes the entire group g_i, whereas $\mathbf{x}_{ob,i}$ is another n-dimensional many-hot vector that only encodes the observed members $g_{ob,i}$. Here we should ignore $\mathbf{h}_i$, as it is only intended for the CVAEH model. The input of the encoder is the concatenation of $\mathbf{x}_{full,i}$ and $\mathbf{x}_{ob,i}$, whilst the outputs are the Gaussian mean μ and standard deviation σ (the diagonal elements). We then sample a latent vector $\mathbf{z}$ using the reparameterization trick (Kingma and Welling 2014), i.e. $\mathbf{z} = \mu + \sigma \odot \mathbf{z_0}$, where $\mathbf{z_0} \sim \mathcal{N}(0, I)$. The process up to now is equivalent to $Q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$ and $P(\mathbf{z}|\mathbf{y})$ in Eq. 2. Next, $\mathbf{z}$ and $\mathbf{x}_{ob,i}$ are concatenated and fed to the decoder. The output of the decoder is a conditional probability $P(v|g_{ob,i}) = P(\mathbf{x}|\mathbf{z}, \mathbf{x}_{ob,i})$, where $\mathbf{x}$ is the one-hot encoding of $v \in G$. **Now we can recommend the individual v with the highest $P(v|g_{ob,i})$ for the observed group $g_{ob,i}$.**

CVAEH is similar to CVAE, except that it introduces an additional vector $\mathbf{h_i}$ whose v's entry represents the number of events that an individual v has attended during $[t_{i-m}, t_i)$, with t_{i-m} being the time m events prior to the current time t_i. $\mathbf{h_i}$ is thus a representation of the event history $\mathcal{H}(t_i)$. As shown in Fig. 2, $\mathbf{x}_{full,i}$, $\mathbf{x}_{ob,i}$ and $\mathbf{h}_i$ are concatenated at the encoder; while $\mathbf{z}$, $\mathbf{x}_{ob,i}$ and $\mathbf{h}_i$ are concatenated at the decoder. Accordingly, the output of the decoder is a conditional probability $P(v|g_{ob,i}, \mathcal{H}(t_i)) = P(\mathbf{x}|\mathbf{z}, \mathbf{x}_{ob,i}, \mathbf{h}_i)$, where $\mathbf{x}$ is again the one-hot encoding of an individual $v \in G$. By introducing $\mathbf{h}_i$, the CVAEH model is able to take advantage of the temporal correlation in the data. As will be shown in our experiments, for the datasets where samples are time-correlated, the CVAEH model is generally better than the CVAE model in group link prediction.

Loss function To train CVAE and CVAEH, we maximize the variational lower bound defined in Eq. 2. More concretely, we minimize the following loss function using mini-batch gradient descent:

$$\begin{aligned}\mathcal{L}(\phi, \theta) = \sum_{v=1}^{n} [&-x_{full,v} log(P_\theta(x_v|\mathbf{z}, \mathbf{x}_{ob}, \mathbf{h})) \\ &- (1 - x_{full,v}) log(1 - P_\theta(x_v|\mathbf{z}, \mathbf{x}_{ob}, \mathbf{h}))] \\ &+ \frac{1}{2} \sum_{v=1}^{n} (exp(\sigma_v) + \mu_v^2 - 1 - log(\sigma_v))\end{aligned} \quad (3)$$

where x_v, $x_{full,v}$, σ_v and μ_v are the v'th element of $\mathbf{x}$, $\mathbf{x}_{full}$, σ and μ, respectively. Note, $\mathbf{h}$ here is only for the CVAEH model. Although not shown explicitly, μ, σ, and $\mathbf{z}$ are dependent on ϕ. The second summation on the right is the closed form of the KL-divergence in Eq. 2, as the result of the reparameterization trick (Kingma and Welling 2014).

Learning process is outlined in Alg. 1. Note $\mathbf{h}_i$ is only intended for the CVAEH model. The algorithm starts by initializing the encoder and decoder parameters ϕ and θ. We then sample a batch of data (line 3), and for each data we go through the encode-decode process (line $5-9$), estimate the loss $\mathcal{L}(\phi, \theta)$ in Eq. 3 (line 10), and obtain the gradients w.r.t ϕ and θ (line 11). At the end of each batch, we calculate the mean gradients and update ϕ and θ (line $13-14$). This process (line $3-14$) repeats until the model converges.

Group-recommendation

The architecture shown in Fig. 2 remains the same for group-recommendation. However, $\mathbf{x}_{ob,i}$ is now a one-hot encoded vector of the observed individual $v_{ob,i}$, and $\mathbf{x}$ is a many-hot encoded vector of a group. Consequently, the output of the decoder gives a conditional probability $P(g|v_{ob,i}) = P(\mathbf{x}|\mathbf{z}, \mathbf{x}_{ob,i})$, for $g \subseteq G \backslash \{v_{ob,i}\}$. **We thus recommend the group g with the highest $P(g|v_{ob,i})$ for the observed individual $v_{ob,i}$.** The learning process is also the same except that the contents of $\mathbf{x}_{ob,i}$, and $\mathbf{x}$ have changed.

Algorithm 1: CVAE/CVAEH learning process

Require: learning rate lr, batch size m
randomly initialize ϕ, θ;
while *not converge* **do**
  sample batch of $\mathbf{x}_{full,i}$, $\mathbf{x}_{ob,i}$, $\mathbf{h}_i$;
  for $i \leftarrow 1$ **to** m **do**
    $\mathbf{c}_i \leftarrow concat(\mathbf{x}_{full,i}, \mathbf{x}_{ob,i}, \mathbf{h}_i)$;
    $\mu, \sigma \leftarrow encoder(\mathbf{c}_i)$;
    $\mathbf{z} \leftarrow \mu + \sigma \odot \mathbf{z_0}$ where $\mathbf{z_0} \sim \mathcal{N}(0, I)$;
    $\mathbf{c}'_i \leftarrow concat(\mathbf{z}, \mathbf{x}_{ob,i}, \mathbf{h}_i)$;
    $P_\theta(\mathbf{x}|\mathbf{z}, \mathbf{x}_{ob}, \mathbf{h}_i) \leftarrow decoder(\mathbf{c}'_i)$;
    calculate $\mathcal{L}(\phi, \theta)$ using Eq. 3;
    $\partial\mathcal{L}_{\phi,i} \leftarrow \frac{\partial\mathcal{L}(\phi,\theta)}{\partial\phi}$ and $\partial\mathcal{L}_{\theta,i} \leftarrow \frac{\partial\mathcal{L}(\phi,\theta)}{\partial\theta}$;
  end
  $\phi \leftarrow \phi - lr \times \frac{1}{m}\sum_{i=1}^{m} \partial\mathcal{L}_{\phi,i}$;
  $\theta \leftarrow \theta - lr \times \frac{1}{m}\sum_{i=1}^{m} \partial\mathcal{L}_{\theta,i}$;
end

Related Works

Recommender systems Traditionally, recommender systems are mostly based on collaborative filtering (Su and Khoshgoftaar 2009), especially matrix factorization (MF) (Koren, Bell, and Volinsky 2009). To reduce cold start, many works (Feng and Wang 2012; Zhang, Wang, and Feng 2013) leverage additional information to improve performance. For example, (Zhang, Wang, and Feng 2013) adopts an extended MF approach that incorporates location features and social features. Since we do not use additional features, our approach is different, nevertheless we compare our methodology with a matrix factorization baseline in the experiments. There exists a few works on "group recommendation" (Liu et al. 2012; Recio-Garcia et al. 2009), which refers to recommending *items* for a group of users, which is not the same as our group link prediction task, where we recommend a *user* for a group or vice-versa.

Heterogeneous networks The proposed group link prediction problem can be posed as link prediction on (bipartite) heterogeneous networks (Shi et al. 2017) with nodes of individuals and groups. Node proximity on heterogeneous networks can be evaluated via path-based similarity measures, such as random walk with restart (Lao and Cohen 2010; Pham et al. 2015) and meta-path similarity (Sun and Han 2012). Furthermore, nodal representations can be learned from meta-path guided random walks (Dong, Chawla, and Swami 2017). In addition, matrix factorization can be adopted for heterogenous networks (Shi et al. 2019). We compare our models with two state-of-the-art models - metapath2vec (Dong, Chawla, and Swami 2017) and HERec (Shi et al. 2019).

Dynamic graph embedding As many real-world networks evolve over time, many dynamic graph embedding models (Zhou et al. 2018; Zhu, Steeg, and Galstyan 2016; Nguyen et al. 2018; Zuo et al. 2018) are proposed to learn time-respecting representations. For example, (Nguyen et al. 2018) leverages temporal random walks to learn node embeddings in continuous-time dynamic networks. (Zuo et al. 2018) combines the Hawkes process (Hawkes 1971) and the attention mechanism (Bahdanau, Cho, and Bengio 2015) to learn temporal embeddings via neighborhood formation. We compare our models (also dynamic) with these two models as well as a Long Short-Term Memory (LSTM) based model (Hochreiter and Schmidhuber 1997).

Graph neural networks Recent years have witnessed the surge of graph neural networks (GNN) (Hamilton, Ying, and Leskovec 2017b). Many GNN models, such as graph convolutional networks (GCN) (Kipf and Welling 2017) and GraphSAGE (Hamilton, Ying, and Leskovec 2017a), generate nodal representations by aggregating local neighborhood features or attributes. Alternatively, (Veličković et al. 2019) proposes a unsupervised node embedding method by maximizing mutual information. Furthermore, (Ying et al. 2018) extends GNN-based representation learning to web-scale bipartite networks of billions of nodes, by combining random walks and GCN. We compare our models with a state-of-the-art GNN model.

Data set	N	$\lvert V\rvert$	$\lvert E\rvert$
Enron	31145	1946	47164
HT09	3703	113	9317
SFHH	3331	403	120507
Meetup NYC	11136	25458	340751
Meetup CA	15717	36799	407584

Table 1: Data set properties. N denotes the number of events (groups). $|V|$ and $|E|$ are the number of nodes and number of edges for each network, respectively.

Experimental Results

In this section, we present experimental results to validate the effectiveness of our proposed group link prediction models for solving member and group recommendation tasks on five real-world datasets. We also compare the performance of our models with several baseline models.

Data Description

We use five real-life datasets. Their statistics is provided in Table 1. For data pre-processing, we organized the datasets into sequences of time-group pairs $\{(t_i, g_i)\}$, sorted in time ascending order. We then split each dataset into training (80%), validation (10%), and test (10%) sets. Any group member that appears in the validation or test split, but not in the training split, are removed from the corresponding data-split. Also, a group consists of less than three group members are also removed from the group-list. Let M_{train}, M_{valid}, and M_{test} denote the set of individuals in the training, validation, and test splits, respectively. The processed datasets then satisfy (1) $M_{valid} \subseteq M_{train}$, (2) $M_{test} \subseteq M_{train}$, and (3) $G = M_{train}$. More discussion of each datasets are below:

Enron Email This dataset (Klimt and Yang 2004) contains emails collected by the Federal Energy Regulatory Commission (FERC) during the investigation of the Enron Corporation. It contains $\sim 600,000$ emails from the year of 1999 to 2002, involving more than three thousands email

addresses, most of which belong to the employees of the company. We organize the emails in time ascending order and extract the sending and receiving addresses owned by the employees (i.e. addresses ending with "@enron.com"). For member-recommendation, we only keep the emails with $3 \sim 10$ people, as the emails that include too many people are mostly messages broadcasted to everyone without distinguishing the groups. After the processing described earlier, we obtain a sequence of 31145 samples involving 1949 email addresses (Table 1). For group-recommendation, we keep the people who have involved in at least 1000 emails so that we can obtain enough negative samples. This also helps identify the major players behind the Enron scandal as discussed later in the case study. This treatment renders us a sequence of 15801 emails involving 114 email addresses.

Dynamic Contact Networks These two datasets contain spatial temporal contact information of attendees during the ACM Hypertext 2009 conference (**HT09**) (Isella et al. 2011) and the 2009 SFHH conference (**SFHH**) (G'enois and Barrat 2018). The attendees of the conferences voluntarily wore radio badges that monitored their face-to-face proximity. If a group of individuals engaged in a conversation, the timestamp and their IDs would be recorded. During the course of the conferences, such events were collected, rendering a sequence of (time, attendee-list) samples. **HT09** dataset has 3703 conversational groups and 113 attendees, and **SFHH** has 3331 conversations groups involving 403 attendees (Table 1).

Event-based Social Networks These datasets (Pham et al. 2015) were crawled from an event-based social network platform - Meetup.com. This platform allows its users to create/join groups and announce events for the group members. For instance, a user in the sports-enthusiastic group may invite other group members to a football watch-party. In this work, we adopt the **Meetup-NYC** dataset and the **Meetup-CA** dataset containing events taking place in New York City and California, respectively. Note although a user can be part of multiple groups and join various events, she cannot join the events created by a group of which she is not a member. For member-recommendation, we only keep the events with $3 \sim 20$ participants due to performance consideration. This renders a sequence of 11136 events involving 25458 users (Meetup-NYC) and a sequence of 15717 events involving 36799 users (Meetup-CA) (Table 1). Also notice that for a user who is only in one group, it would be trivial to recommend an event for her, as she would join the events exclusively of this group. Therefore, to effectively evaluate our model, for group-recommendation, we only keep users who are member of at least 10 groups, which leads to a sequence of 4500 events including 2286 users (Meetup-NYC) and a sequence of 2392 events including 1649 users (Meetup-CA).

Baseline Methods

Graph topology-based Link Prediction We convert our data into an undirected graph where nodes are event participants (e.g. email addresses, conference attendees, and so on) and edges are formed between them if they are in an event together. We define some graph-based score functions $r(u, v)$ to measure the similarity between nodes u and v. A larger $r(u, v)$ indicates a higher likelihood for linking the nodes. Here we examine an array of such score functions (Liben-Nowell and Kleinberg 2003): **Common Neighbors** (CN), **Jaccard Index** (JI), **Adamic/Adar** (AA), **Preferential Attachment** (PA), and **Katz**$_\beta$. Since these scores are evaluated in a pair-wise manner, we resort to some aggregation methods to obtain the scores between a group g and an individual v. Formally, we have $r_F(g, v) = F(r(u, v)), \forall u \in g$, for $F \in \{sum, max, min, mean\}$.

Homogeneous Network Embedding-based Link Prediction We adopt a random walk based model - **Node2Vec** (Grover and Leskovec 2016) and a recent GNN based model - **Deep Graph Infomax (DGI)** (Veličković et al. 2019), to learn node representations on homogeneous networks consisting of individuals. We aggregate individual-individual cosine-similarities to obtain group-individual similarities.

Heterogeneous Network Embedding-based Link Prediction The proposed group link prediction can be cast as link prediction on heterogeneous networks with mixed nodes of individuals and groups. We adopt **metapath2vec** (Dong, Chawla, and Swami 2017) and **HERec** (Shi et al. 2019) for the task. metapath2vec utilizes meta-path based random walks and skip-gram (Mikolov et al. 2013), to generate nodal representations. We apply this method with "individual-group-individual" meta-path and use dot product as similarity measure. HERec leverages fusion functions and matrix factorization to integrate multiple embeddings w.r.t different meta-paths. We use the recommendation ratings given by the method for link prediction.

Dynamic Network Embedding-based Link Prediction We can construct dynamic networks of individuals utilizing event timestamps. In specific, we use **CTDNE** (Nguyen et al. 2018) and **HTNE** (Zuo et al. 2018) to learn time-respecting nodal representations. CTDNE uses random walks in time-ascending order to learn temporal embeddings; while HTNE combines the Hawkes process (Hawkes 1971) and the attention mechanism (Bahdanau, Cho, and Bengio 2015) to learn embeddings via neighborhood formation. With the embeddings at each time step, we use an aggregation approach as in the static case, to predict the links between individuals and groups.

Matrix Factorization Methods Our datasets can also be converted to matrices $\mathbf{X} \in \mathbb{R}^{|G|} \times \mathbb{R}^{N}$, with the rows representing individuals and the columns representing groups (or equivalently events). If an individual v was part of a group g, then the entry $X_{v,g}$ would be one, otherwise zero. This setup resembles the user-item rating matrix widely adopted in the context of recommender system (Koren, Bell, and Volinsky 2009). By matrix factorization (MF), an individual v is associated with a vector $\mathbf{p}_v \in \mathbb{R}^d$ and a group g is associated with a vector $\mathbf{q}_g \in \mathbb{R}^d$ in some d-dimensional latent space, such that $X_{v,g} = \mathbf{p}_v^T \mathbf{q}_g$. To factorize $\mathbf{X}$, we adopt two methods - non-negative matrix factorization (**NMF**) (Cichocki and Phan 2009; Févotte and Idier 2011) and singular value decomposition (**SVD**) (Halko, Martinsson, and Tropp 2011). We can thus define the score function as $r(g, v) = \mathbf{p}_v^T \mathbf{q}_g$.

Neural Network Methods Aforementioned link prediction techniques (graph topology-based, embedding-based or MF-

based) rank the candidates according to certain score functions $r(g, v)$. In contrast, our proposed model and the following Neural Network based models (except set2vec BPR) can provide a probability distribution for the links. In specific, we propose the following baseline Neural Network models for comparison:

One-hot MLP The first model combines Multi-layer Perceptron (MLP) with one-hot encoding (one-hot MLP). The input of the model is the one-hot encoded vector of the observed participant(s), $g_{ob,i}$ in member-recommendation or $v_{ob,i}$ in group-recommendation, and the output is a probability distribution for the unobserved participant(s), $v_{unob,i}$ in member-recommendation or $g_{unob,i}$ in group-recommendation. This is essentially a multivariate logistic regression, where whether an individual would join an event is determined by a Binomial distribution given by the output probabilities. To learn the model, the cross entropy loss is minimized using mini-batch gradient descent.

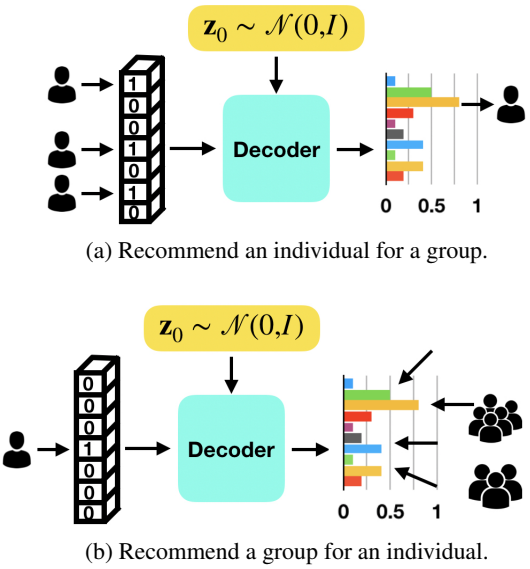
Set2vec MLP Set2vec is proposed by (Vinyals, Bengio, and Kudlur 2015) for encoding a set of elements. The set encoding is essentially a weighted sum of the individual encodings of the set members. We use set2vec to encode a group as a single vector $\mathbf{x}_s = \sum_{i=1}^{k} \alpha_i \mathbf{x}_i$, where the weights $\alpha_1, \alpha_2, ..., \alpha_k$ and the individual encoding vectors $\{\mathbf{x}_i, i = 1, 2, ..., k\}$ are learned by coupling set2vec with a downstream MLP classifier. The output of the classifier is a probability distribution, like in the one-hot MLP model.

Set2vec BPR We can also replace the cross entropy loss with the Bayesian Personalized Ranking (BPR) loss (Rendle et al. 2009), leading to the set2vec-BPR model. BPR is proposed by (Rendle et al. 2009) as a ranking framework for item recommendation. We use BPR to learn the embedding $\mathbf{x}_v$ for any individual v and the embedding $\mathbf{x}_g$ for any group g. We can then define the score function as $r(g, v) = \mathbf{x_g}^T \mathbf{x_v}$, which measures the similarity between the two embeddings.

LSTM Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) is efficient for modeling sequence related problems. We compare our model with an LSTM-based model which takes in a sequence of embedding vectors $\{\mathbf{x}_{t_i}, \mathbf{x}_{t_{i+1}}, \cdots, \mathbf{x}_{t_{i+j}}\}$ of the observed member-lists at time-steps $\{t_i, t_{i+1}, \cdots, t_{i+j}\}$, and outputs a conditional probability $P(v|\mathbf{x}_{t_i}, \mathbf{x}_{t_{i+1}}, ..., \mathbf{x}_{t_{i+j}}), v \in G$ conditioning on all the observations in the window $[t_i, t_{i+j})$. However, this model only predicts an individual rather than a group, therefore we only apply it to member-recommendation. Specifically, we use the output probability to rank the candidates at time t_{i+j} given the current observed members and the members of the previous events.

Experimental Setup

Model details For the proposed **CVAE/CVAEH**, the encoder and the decoder consist of two fully-connected (FC) layers, with the hidden dimension (hd) tuned from $\{128, 256, 512, 1024\}$. We also tune the dimension of $\mathbf{z}$, zd in $\{32, 64, 128, 256\}$. In addition, for CVAEH, we use different window sizes, ts in $\{2, 4, 8\}$ for calculating $\mathbf{h}_i$. To learn the model, we minimize the loss in Eq. 3 using mini-batch gradient descent, with the batch size $= 64$ and the learning rate, lr in $\{0.0001, 0.001, 0.01\}$.



(a) Recommend an individual for a group.

(b) Recommend a group for an individual.

Figure 3: Prediction Illustration. The one-hot encoding of the observed group member(s) is fed to the decoder. $\mathbf{z}_0$ is sampled multiple times, generating a multiplicity of outputs, which are averaged to obtain the final conditional probability. (a) For member-recommendation, we predict the one most probable member. (b) For group-recommendation, we find a group that best aligns with the probability distribution.

Prediction process In Fig. 3a, we illustrate the prediction process for **member-recommendation**. For each g_{ob}, we sample $\mathbf{z_0} \sim \mathcal{N}(0, I)$ m times, generating a series of conditional probabilities $P_i(v|g_{ob}), i = 1, 2, ..., m$. The final conditional probability used for prediction is the sample mean of these m conditional probabilities, $P(v|g_{ob}) = \frac{1}{m}\sum_{i=1}^{m} P_i(v|g_{ob}), v \in G$. We then rank the individuals v by $P(v|g_{ob})$.

In Fig. 3b, we illustrate the prediction process for **group-recommendation**. We take out an individual v_{ob} from an event and regard the rest of the member-list as unobserved g_{unob}. Similarly, for each v_{ob}, we sample $\mathbf{z_0} \sim \mathcal{N}(0, I)$ m times, and calculate the sample mean of these m conditional probabilities, $P(g|v_{ob}) = \frac{1}{m}\sum_{i=1}^{m} P_i(g|v_{ob}), g \subseteq G \setminus \{v_{ob}\}$. However, we do not scan over all the possible g as there are exponential number of possible groups. Instead, we denote the true g_{unob} as a positive sample, $g_{pos} = g_{unob}$, and randomly sample a set of 200 negative samples $S_{neg} = \{g_{neg,i}|v_{ob} \notin g_{neg,i}, i = 1, 2, ..., 200\}$ from the training set. We then rank $g \in \{g_{pos}\} \cup S_{neg}$ by the average probability $\frac{\sum_{v \in g} P(v|v_{ob})}{|g|}$. Note, here the numerator prefers a group that contains individuals with high $P(v|v_{ob})$; while the denominator penalizes a group that includes too many irrelevant individuals. Overall, this score measures to what extent a group g aligns with the conditional probability $P(g|v_{ob})$.

Performance metric As performance metrics, we use hit rate@m (Hit@m) and mean reciprocal rank@m (MRR@m).

For each query, if the target is among the top-m recommendations, we have a hit and record the reciprocal of the rank at which the target is retrieved. When averaged across queries, we obtain Hit@m and MRR@m, with the former indicating how good the model is at finding the target and the later indicating how high the model ranks the target.

For **member-recommendation**, during validation and testing, for each sample, we hold out the last member $v_{i,k}$ from a group g_i of size k, rendering $v_{unob} = v_{i,k}$ and $g_{ob} = \{v_{i,1}, \cdots, v_{i,k-1}\}$. If the true v_{unob} is among the top m predictions, we then register a hit for the Hit@m score. We perform parameter tuning over the validation set and select the optimal parameters based on the mean Hit@10 rate. The results discussed in the next section are obtained using the optimal hyper-parameters as, Enron-CVAE: $zd = 256, hd = 512, lr = 0.0001$; HT09-CVAEH: $zd = 128, hd = 1024, ts = 2, lr = 0.0001$; SFHH-CVAEH: $zd = 128, hd = 1024, ts = 2, lr = 0.0001$; Meetup-NYC-CVAE: $zd = 32, hd = 1024, lr = 0.001$; Meetup-CA-CVAE: $zd = 256, hd = 128, lr = 0.001$.

For **group-recommendation**, we similarly hold out the last member of each sample g_i in the validation set and the test set, which leads to $v_{ob} = v_{i,k}$, $g_{pos} = g_{unob} = \{v_{i,1}, v_{i,2}, \cdots, v_{i,k-1}\}$, and $S_{neg} = \{g_{neg,i} | v_{i,k} \notin g_{neg,i}, i = 1, 2, ..., 200\}$. If g_{pos} is among the top m highest ranking groups in $\{g_{pos}\} \cup S_{neg}$, we register a hit for the Hit@m rate. We perform hyper-parameter tuning over the validation set using the mean Hit@10 rate, and obtain the optimal hyper-parameters as, Enron-CVAE: $zd = 256, hd = 1024, lr = 0.01$; HT09-CVAEH: $zd = 32, hd = 128, ts = 2, lr = 0.0001$; SFHH-CVAEH: $zd = 64, hd = 1024, ts = 2, lr = 0.0001$; Meetup-NYC-CVAE: $zd = 64, hd = 1024, lr = 0.01$; Meetup-CA-CVAE: $zd = 32, hd = 256, lr = 0.01$.

Setup for baseline methods For fair comparison, we also perform hyper-parameter tuning for the baseline methods on the same validation set. For the embedding-based methods, we test different embedding dimensions in $\{64, 128, 256, 512\}$. For the neural network models, we also tune the hidden dimension in $\{128, 256, 512, 1024\}$, learning rate in $\{0.0001, 0.001, 0.01\}$ and the batch size in $\{32, 64\}$. For set2vec MLP and LSTM, we test time-step in $\{5, 10, 15\}$ and $\{2, 4, 8, 16, 32\}$, respectively. To convert $r(u, v)$ to $r(g, v)$, we try aggregation in $\{sum, mean, max, min\}$. For the hyper-parameters not mentioned here, we adopt the same values as suggested in the original papers publishing the methods. We use Hit@10 rate for determining the optimal hyper-parameters. The optimal models are compared with CVAE/CVAEH on the same test set.

Results

Member recommendation In Table 2 and 3, we show the hit rates and MRR scores respectively, for the member recommendation task, using the various models over the five datasets. The methods are arranged in row clusters based on their approaches. Our proposed methods, CVAE and CVAEH are in the last row cluster. For every method, the results are obtained using the optimal hyper-parameters. For the graph topology-based and the embedding-based methods, the results are shown for the best aggregation function among sum, min, max, and mean. For easy comparison, we highlight the highest and the second highest scores.

Hit rates can indicate the models' ability to include the target in the top candidate list. From Table 2, we can see that our models perform significantly better than the competing methods in hit rates. Specifically, for the HT09 and SFHH datasets, our CVAEH model's performance is around 20% better than the best competing method in both Hit@10 and Hit@20 metrics. For other datasets, our models' performance is generally 4% to 8% better over the best of the competing methods. Only for the Hit@20 rate for the Meetup-NYC and the Meetup-CA datasets, our method came in a close second and fourth position, out of the 17 competing methods.

MRR scores can indicate how high the models rank the target in the candidate list. From Table 3, we can see that for all datasets, the highest MRR scores are achieved by either CVAE or CVAEH. In particular, for the HT09 and SFHH datasets, our CVAEH model outperforms the competing models by $\sim 14\%$ and $\sim 32\%$, respectively. For the other datasets, our method is better than the best competing method by at least $\sim 10\%$, except for Meetup-CA ($\sim 1\%$ to 2%).

Combining the two metrics, we can see that our method not only predicts the target members more accurately, but also ranks them higher than the competing methods. Moreover, our methods show consistently great performance over the different datasets, as opposed to that, for example, the Katz β method varies from comparable to ours in Meetup-NYC and Meetup-CA to significantly poorer in HT09 and SFHH. This is likely because our model does not explicitly rely on the network topology. Our method is also better than the dynamic network embedding methods (CTDNE and HTNE) and heterogeneous network embedding methods (metapath2vec and HERec), as our method combines the advantages of both approaches, incorporating the dynamic and heterogeneous characteristics of the data.

Group recommendation In Table 4 and 5, we show the hit rates and MRR scores for the group recommendation task, using different methods over the five datasets. Identical to the member recommendation, we show the results of the best hyper-parameters and the best aggregation function (for the graph topology-based and the embedding-based methods).

The results suggest that our CVAEH model outperforms the other models for the HT09 dataset, in both hit rates and MRR scores, leading the runner up by $\sim 50\%$ and $\sim 30\%$, respectively. For the SFHH dataset, CVAEH achieves the highest hit rates and the second highest MRR scores. For the Enron dataset, our CVAE model outperforms the competing baselines except for Hit@20 in a close second position. For the Meetup-NYC and Meetup-CA datasets, our methods score the second or third highest hit rates and MRR. Notice that the static methods like metapath2vec and Jaccard Index perform very well in the Meetup-NYC and Meetup-CA datasets. This is not surprising, as in these datasets, who joins an event with whom is very static, given that the

Model	Enron		HT09		SFHH		Meetup-NYC		Meetup-CA	
	H@10	H@20	H@10	H@20	H@10	H@20	H@10	H@20	H@10	H@20
CN	0.349	0.433	0.075	0.108	0.027	0.186	0.309	0.393	0.359	0.461
JI	0.412	0.522	0.057	0.100	0.129	0.302	0.282	0.360	0.320	0.432
AA	0.366	0.453	0.067	0.111	0.033	0.189	0.317	0.384	0.367	**0.474**
PA	0.036	0.046	0.075	0.111	0.039	0.401	0.015	0.047	0.004	0.011
Katz β	0.446	0.561	0.124	0.148	0.165	0.314	**0.318**	**0.409**	**0.387**	**0.483**
Node2Vec	0.188	0.345	0.140	0.232	0.000	0.000	0.070	0.178	0.076	0.206
CTDNE	0.390	0.526	0.154	0.240	0.275	0.290	0.170	0.274	0.193	0.291
HTNE	0.130	0.222	0.412	0.496	0.168	0.222	0.041	0.063	0.071	0.132
metapath2vec	0.277	0.442	0.350	0.588	0.027	0.153	0.107	0.231	0.150	0.275
HERec	0.036	0.074	0.412	0.507	0.192	0.216	0.004	0.010	0.015	0.031
NMF	0.257	0.350	0.466	0.561	0.308	0.356	0.112	0.187	0.124	0.228
SVD	0.278	0.393	0.555	0.677	0.177	0.254	0.126	0.208	0.159	0.266
one-hot MLP	0.462	0.536	0.553	0.712	0.371	0.440	0.326	0.392	0.339	0.428
set2vec MLP	0.434	0.514	0.561	0.617	0.207	0.296	0.261	0.336	0.312	0.381
set2vec BPR	0.374	0.494	0.501	0.590	0.069	0.111	0.173	0.258	0.214	0.316
LSTM	0.438	0.522	0.503	0.621	0.269	0.323	0.253	0.315	0.301	0.385
DGI	0.275	0.368	0.051	0.213	0.051	0.141	0.193	0.258	0.200	0.281
CVAE	**0.521**	**0.601**	**0.615**	**0.728**	**0.428**	**0.500**	**0.341**	**0.406**	**0.400**	0.455
CVAEH	**0.485**	**0.578**	**0.709**	**0.806**	**0.509**	**0.560**	0.275	0.345	0.267	0.331

Table 2: Member recommendation hit rates. The highest and the second highest hit rates are bold.

Model	Enron		HT09		SFHH		Meetup-NYC		Meetup-CA	
	R@10	R@20	R@10	R@20	R@10	R@20	R@10	R@20	R@10	R@20
CN	0.186	0.192	0.022	0.024	0.026	0.032	0.161	0.167	0.208	0.215
JI	0.210	0.218	0.012	0.015	0.070	0.081	0.127	0.132	0.157	0.165
AA	0.200	0.206	0.036	0.039	0.006	0.017	0.167	0.172	0.213	0.220
PA	0.022	0.023	0.042	0.044	0.005	0.028	0.002	0.005	0.001	0.001
Katz β	0.238	0.246	0.108	0.110	0.043	0.053	0.166	0.173	**0.218**	**0.225**
Node2Vec	0.033	0.044	0.025	0.032	0.000	0.000	0.009	0.017	0.010	0.019
CTDNE	0.107	0.117	0.024	0.030	0.126	0.127	0.041	0.048	0.065	0.071
HTNE	0.032	0.039	0.085	0.0916	0.034	0.037	0.007	0.008	0.015	0.019
metapath2vec	0.068	0.079	0.082	0.100	0.007	0.016	0.020	0.028	0.041	0.049
HERec	0.009	0.012	0.091	0.101	0.026	0.032	0.001	0.001	0.023	0.026
NMF	0.102	0.108	0.410	0.416	0.144	0.147	0.037	0.039	0.026	0.028
SVD	0.114	0.121	0.214	0.222	0.045	0.050	0.027	0.029	0.032	0.034
one-hot MLP	0.276	0.281	0.456	0.461	0.208	0.211	**0.193**	**0.198**	0.210	0.215
set2vec MLP	0.277	0.283	**0.470**	**0.474**	0.096	0.102	0.137	0.142	0.171	0.176
set2vec BPR	0.098	0.106	0.172	0.179	0.010	0.013	0.047	0.053	0.056	0.063
LSTM	0.262	0.267	0.393	0.401	0.080	0.084	0.135	0.140	0.177	0.182
DGI	0.111	0.119	0.018	0.022	0.012	0.019	0.039	0.039	0.087	0.092
CVAE	**0.304**	**0.311**	0.387	0.397	**0.260**	**0.266**	**0.213**	**0.218**	**0.222**	**0.227**
CVAEH	**0.311**	**0.317**	**0.535**	**0.542**	**0.275**	**0.278**	0.154	0.159	0.148	0.152

Table 3: Member recommendation mean reciprocal ranks (MRR). The highest and the second highest MRR scores are bold.

users on meetup.com do not frequently change their interest groups, and per the platform policy, all the events are exclusively joined by the members of the same interest group. In contrast, for the other datasets where group membership is constantly changing, our models are better.

Comparison between CVAE and CVAEH Notice that in Table 2 - 5, CVAEH outperforms CVAE in HT09 and SFHH; whereas we observe the opposite in the other datasets. This is because the samples (member-list) in these two datasets are correlated in time, while those in the others are not. As the CVAEH model uses the history vector $\mathbf{h}_i$ to take account of the temporal effect, it performs better when the data are time-correlated. Conversely, if the data are time-independent, the history vector $\mathbf{h}_i$ may introduce noise, making the CVAEH model less effective than the CVAE model. In Fig. 4, we plot the correlograms for the datasets. The horizontal axis is the time difference (window size); the vertical bars are the autocorrelations; the correlation bands

Model	Enron		HT09		SFHH		Meetup-NYC		Meetup-CA	
	H@10	H@20	H@10	H@20	H@10	H@20	H@10	H@20	H@10	H@20
CN	0.318	0.422	0.054	0.084	0.009	0.030	0.123	0.200	0.159	0.358
JI	0.531	0.603	0.264	0.380	0.204	0.231	0.215	0.300	0.332	0.487
AA	0.202	0.357	0.054	0.092	0.015	0.027	0.179	0.242	0.224	0.379
PA	0.078	0.129	0.008	0.038	0.009	0.045	0.099	0.141	0.043	0.095
Katz β	0.150	0.307	0.016	0.046	0.021	0.051	0.267	0.352	0.336	0.478
Node2Vec	0.469	0.565	0.067	0.089	0.024	0.039	0.184	0.271	**0.362**	0.509
CTDNE	0.641	0.719	0.194	0.358	0.195	0.195	**0.294**	0.379	0.293	0.440
HTNE	0.698	0.775	0.318	0.412	0.006	0.021	0.222	0.314	0.345	0.517
metapath2vec	0.562	0.734	**0.515**	**0.574**	**0.497**	**0.572**	**0.410**	**0.500**	**0.418**	**0.608**
HERec	0.141	0.231	0.089	0.092	0.063	0.120	0.209	0.272	0.142	0.228
NMF	0.579	0.820	0.429	0.482	0.222	0.311	0.233	0.332	0.293	0.461
SVD	0.700	0.782	0.434	0.520	0.269	0.362	0.258	0.357	0.315	0.470
one-hot MLP	**0.751**	0.826	0.488	0.536	0.042	0.120	0.278	0.372	0.284	0.517
set2vec MLP	0.737	**0.837**	0.474	0.531	0.030	0.072	0.274	0.352	0.306	0.448
set2vec BPR	0.727	0.818	0.488	0.585	0.153	0.344	0.274	0.368	0.323	0.500
DGI	0.643	0.739	0.173	0.280	0.290	0.437	0.200	0.242	0.203	0.336
CVAE	**0.757**	**0.835**	0.501	0.571	0.066	0.093	0.287	**0.381**	0.332	**0.522**
CVAEH	0.706	0.796	**0.768**	**0.873**	**0.512**	**0.743**	0.220	0.332	0.181	0.349

Table 4: Group recommendation hit rates. The highest and the second highest hit rates are bold

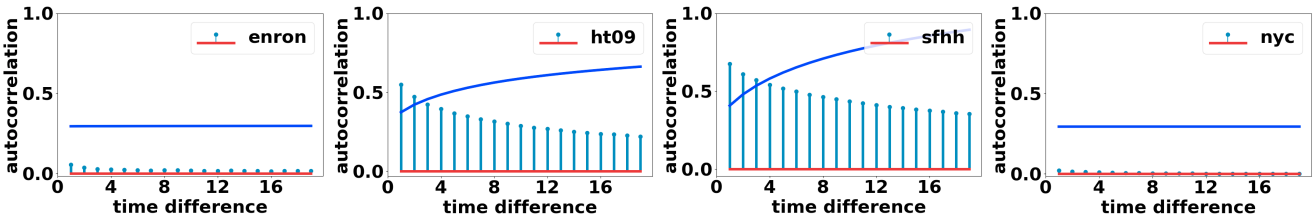
Model	Enron		HT09		SFHH		Meetup-NYC		Meetup-CA	
	R@10	R@20	R@10	R@20	R@10	R@20	R@10	R@20	R@10	R@20
CN	0.135	0.143	0.020	0.022	0.005	0.006	0.052	0.057	0.071	0.083
JI	0.264	0.269	0.114	0.122	0.066	0.069	0.132	0.138	**0.196**	**0.208**
AA	0.062	0.073	0.013	0.015	0.007	0.009	0.077	0.081	0.084	0.094
PA	0.019	0.023	0.003	0.005	0.000	0.000	0.005	0.007	0.011	0.014
Katz β	0.050	0.061	0.005	0.007	0.010	0.012	0.109	0.115	0.162	0.171
Node2Vec	0.184	0.191	0.015	0.017	0.013	0.014	0.089	0.095	0.179	0.188
CTDNE	0.320	0.326	0.069	0.081	0.115	0.115	0.160	0.166	0.125	0.135
HTNE	0.360	0.366	0.176	0.182	0.001	0.002	0.100	0.106	0.123	0.134
metapath2vec	0.298	0.310	0.347	0.351	**0.296**	**0.301**	**0.242**	**0.249**	**0.192**	**0.205**
HERec	0.069	0.075	0.024	0.025	0.025	0.029	0.124	0.130	0.062	0.069
NMF	0.317	0.330	**0.411**	**0.421**	0.184	0.188	0.145	0.152	0.120	0.131
SVD	0.430	0.438	0.382	0.387	0.164	0.178	0.146	0.155	0.141	0.149
one-hot MLP	**0.467**	**0.472**	0.392	0.395	0.017	0.022	0.153	0.160	0.103	0.120
set2vec MLP	0.457	0.464	0.393	0.397	0.010	0.013	**0.167**	**0.172**	0.115	0.125
set2vec BPR	0.419	0.425	0.379	0.386	0.111	0.127	0.154	0.160	0.143	0.155
DGI	0.403	0.409	0.027	0.034	0.095	0.096	0.125	0.128	0.077	0.086
CVAE	**0.472**	**0.478**	0.403	0.407	0.025	0.026	0.161	0.168	0.132	0.145
CVAEH	0.389	0.395	**0.537**	**0.545**	**0.258**	**0.275**	0.124	0.132	0.073	0.085

Table 5: Group recommendation mean reciprocal ranks (MRR). The highest and the second highest MRR scores are bold.

indicate the 90% confidence interval. The plots (the middle two) suggest that the samples in HT09 and SFHH are correlated in $\sim$ 2 steps; while the samples in the other datasets are time independent. In fact, the optimal performance is achieved using a window size of 2 to calculate the history vector $\mathbf{h}_i$ for CVAEH.

Case study: Enron email member composition In this experiment we investigate why particular individuals may be included in an email thread, based upon their relationship with other individuals also receiving the email. Given that an individual v_{ob} is in a group g_i, is there a way to auto-detect the key members that lead to the inclusion of v_{ob} in an email? Our modeling framework is well suited for this task. Recall that in group-recommendation, our model provides a probability $P(v|v_{ob}), v \in G$. We therefore consider the group g_i and identify a group of "influencers" $\{v|P(v|v_{ob}) \geq c, v \in g_i\}$, where c is a threshold. In particular, we apply this analysis to the Enron dataset and examine the relationship between v_{ob}'s job title and the influenecers' job title (Klimt and Yang 2004). Here we use a threshold $c = 0.1$.

In Fig. 5, we plot a bar chart showing the composition of



(a) Enron (b) HT09 (c) SFHH (d) Meetup-nyc

Figure 4: Correlograms for the samples in the datasets. The horizontal axis is the time difference; the vertical bars (light blue) are the autocorrelations; the correlation bands (dark blue) indicate the 90% confidence interval. Note, as the correlograms for Meetup-NYC and Meetup-CA are very similar, we only show the one for Meetup-NYC.

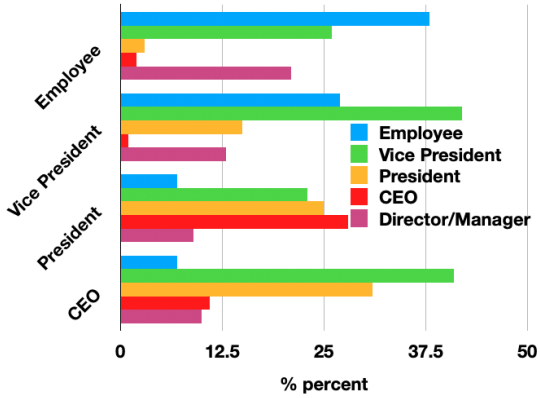


Figure 5: Case study: Ernon email member composition. The color bars indicate the percent composition of critical members ("influencers") grouped by job titles.

the influencers for v_{ob} (vertical axis) based on her job titles. The horizontal axis indicates the percent composition of influencers' job titles. Moving up the hierarchy, the blue bars shorten, indicating that regular employees are less likely to influence the higher level management people. A similar trend can be observed for the director/manager (purple bars). Conversely, the yellow bars lengthen, indicating that presidents become more and more important to the people on the top. We also see that regular employees primarily join a conversation because their coworkers or managers are in it. The green bars are long for all roles, suggesting that vice presidents are generally influential. If v_{ob} is a president, she would primarily join a group involving other management people. Lastly, we can see that a CEO is in an email mainly because the presidents or vice presidents are in it. Overall, our model reveals that the individual-influencer relationship obeys the hierarchical structure of the company.

Conclusions

In this work, we proposed a new problem - group link prediction. Unlike the traditional link prediction which predicts the link between two individuals, our task was to predict the link between an individual and a group. We decompose the problem into member-recommendation and group-recommendation tasks. To solve these two tasks, we proposed a CVAE model and a CVAEH model that takes account of the historical events. The models learned a conditional probability distribution over the unobserved members given the observed ones. We compared our model with a series of competing methods over five real-world datasets. Our CVAEH model shows superior performance for the datasets where the group organizations are correlated in time; while our CVAE model are better in the other cases.

Acknowledgments

This research is sponsored by the National Science Foundation (NSF) under Grant Numbers SCC-1737585, ATD-1737996, and IIS-1909916.

References

Al Hasan, M.; and Zaki, M. J. 2011. A survey of link prediction in social networks. In *Social network data analytics*, 243–275.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

Bojchevski, A.; and Günnemann, S. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *ICLR '18*.

Chen, H.; Li, X.; and Huang, Z. 2005. Link prediction approach to collaborative filtering. In *JCDL '05*, 141–142.

Cichocki, A.; and Phan, A. H. 2009. Fast Local Algorithms for Large Scale Nonnegative Matrix and Tensor Factorizations. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* 92-A: 708–721.

Dave, V. S.; Zhang, B.; Chen, P.-Y.; and Hasan, M. A. 2018. Neural-Brane: Neural Bayesian Personalized Ranking for Attributed Network Embedding. *Data Science and Engineering* .

Dong, Y.; Chawla, N. V.; and Swami, A. 2017. Metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *KDD '17*.

Feng, W.; and Wang, J. 2012. Incorporating Heterogeneous Information for Personalized Tag Recommendation in Social Tagging Systems. In *KDD '12*.

Févotte, C.; and Idier, J. 2011. Algorithms for Nonnegative Matrix Factorization with the -Divergence. *Neural Computation* 23(9): 2421–2456.

G'enois, M.; and Barrat, A. 2018. Can co-location be used as a proxy for face-to-face contacts? *EPJ Data Science* 7(1): 11.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD '16*.

Halko, N.; Martinsson, P. G.; and Tropp, J. A. 2011. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Rev.* 53(2): 217–288. ISSN 0036-1445.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017a. Inductive Representation Learning on Large Graphs. In *NIPS '17*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017b. Representation Learning on Graphs: Methods and Applications. *IEEE Data Engineering Bulletin* .

Hasan, M. A.; Chaoji, V.; Salem, S.; and Zaki, M. 2006. Link prediction using supervised learning. In *In Proc. of SDM 06 workshop on Link Analysis, Counterterrorism and Security*.

Hawkes, A. G. 1971. Spectra of some self-exciting and mutually exciting point processes. *Biometrika* 58(1): 83–90.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9: 1735–1780.

Isella, L.; Stehlé, J.; Barrat, A.; Cattuto, C.; Pinton, J.; and Van den Broeck, W. 2011. What's in a Crowd? Analysis of Face-to-Face Behavioral Networks. *Journal of Theoretical Biology* 271(1): 166–180.

Ivanov, O.; Figurnov, M.; and Vetrov, D. 2019. Variational Autoencoder with Arbitrary Conditioning. In *ICLR '19*.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In *ICLR '14*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Klimt, B.; and Yang, Y. 2004. The Enron Corpus: A New Dataset for Email Classification Research. *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)* 3201: 217–226.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42(8): 30–37. ISSN 0018-9162.

Lao, N.; and Cohen, W. W. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine Learning* 81: 53–67.

Lei, C.; and Ruan, J. 2012. A novel link prediction algorithm for reconstructing protein–protein interaction networks by topological similarity. *Bioinformatics* 29(3): 355–364.

Li, J.; Dani, H.; Hu, X.; Tang, J.; Chang, Y.; and Liu, H. 2017. Attributed Network Embedding for Learning in a Dynamic Environment. In *CIKM '17*.

Liben-Nowell, D.; and Kleinberg, J. 2003. The Link Prediction Problem for Social Networks. In *CIKM '03*.

Liu, X.; Tian, Y.; Ye, M.; and Lee, W.-C. 2012. Exploring Personal Impact for Group Recommendation. In *CIKM '12*.

Lü, L.; and Zhou, T. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications* 390(6): 1150–1170.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS '13*.

Nguyen, G. H.; Lee, J. B.; Rossi, R. A.; Ahmed, N. K.; Koh, E.; and Kim, S. 2018. Continuous-Time Dynamic Network Embeddings. In *WWW '18*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. DeepWalk: Online Learning of Social Representations. In *KDD '14*.

Pham, T. N.; Li, X.; Cong, G.; and Zhang, Z. 2015. A general graph-based model for recommendation in event-based social networks. In *ICDE '15*, 567–578.

Recio-Garcia, J. A.; Jimenez-Diaz, G.; Sanchez-Ruiz, A. A.; and Diaz-Agudo, B. 2009. Personality Aware Recommendations to Groups. In *RecSys '09*.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI '09*.

Shi, C.; Hu, B.; Zhao, W. X.; and Yu, P. S. 2019. Heterogeneous Information Network Embedding for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31(2): 357–370.

Shi, C.; Li, Y.; Zhang, J.; Sun, Y.; and Yu, P. S. 2017. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering* 29(1): 17–37.

Sohn, K.; Lee, H.; and Yan, X. 2015. Learning Structured Output Representation using Deep Conditional Generative Models. In *NIPS '15*.

Su, X.; and Khoshgoftaar, T. M. 2009. A Survey of Collaborative Filtering Techniques. *Adv. in Artif. Intell.* 2009.

Sun, Y.; and Han, J. 2012. *Mining Heterogeneous Information Networks: Principles and Methodologies*. Morgan Claypool Publishers.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. LINE: Large-Scale Information Network Embedding. In *WWW '15*.

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *ICLR '19*.

Vinyals, O.; Bengio, S.; and Kudlur, M. 2015. Order Matters: Sequence to sequence for sets. In *ICLR '15*.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD '18*.

Zhang, W.; Wang, J.; and Feng, W. 2013. Combining latent factor model with location features for event-based group recommendation. In *KDD*, 910–918.

Zhou, L.; Yang, Y.; Ren, X.; Wu, F.; and Zhuang, Y. 2018. Dynamic Network Embedding by Modelling Triadic Closure Process. In *AAAI*.

Zhu, L.; Steeg, G. V.; and Galstyan, A. 2016. Scalable Link Prediction in Dynamic Networks via Non-Negative Matrix Factorization. *IEEE Transactions on Knowledge and Data Engineering 2016* .

Zuo, Y.; Liu, G.; Lin, H.; Guo, J.; Hu, X.; and Wu, J. 2018. Embedding Temporal Network via Neighborhood Formation. In *KDD '18*.