Clustering and Ranking in Heterogeneous Information Networks via Gamma-Poisson Model

Junxiang Chen [*] Wei Dai [*] Yizhou Sun [†] Jennifer Dy [*]

Abstract

Clustering and ranking have been successfully applied independently to homogeneous information networks, containing only one type of objects. However, real-world information networks are oftentimes heterogeneous, containing multiple types of objects and links. Recent research has shown that clustering and ranking can actually mutually enhance each other, and several techniques have been developed to integrate clustering and ranking together on a heterogeneous information network. To the best of our knowledge, however, all of such techniques assume the network follows a certain schema. In this paper, we propose a probabilistic generative model that simultaneously achieves clustering and ranking on a heterogeneous network that can follow arbitrary schema, where the edges from different types are sampled from a Poisson distribution with the parameters determined by the ranking scores of the nodes in each cluster. A variational Bayesian inference method is proposed to learn these parameters, which can be used to output ranking and clusters simultaneously. Our method is evaluated on both synthetic and real-world networks extracted from the DBLP and YELP data. Experimental results show that our method outperforms the state-of-the-art baselines.

1 Introduction

Information networks are oftentimes used to represent objects and their interactions in real-world systems, where each object is represented by a vertex and the relationship between two objects is represented by an edge. Usually networks are assumed involving only one type of vertices and one type of edges between vertices, called *homogeneous information networks*. A friendship network is a typical example, in which each individual is represented by a vertex and friendship is represented by edges between vertices.

In real-world settings, however, there might exist different types of relationships between different types of objects. We call such networks *heterogeneous information networks*. One example of heterogeneous information networks is the bibliographic network constructed from the DBLP database[1], which is illustrated in Figure 1. The network contains three types of vertices: *authors(A), words(W) and venues(V)*. Edges exist between authors representing co-authorship, between authors and venues representing authors publishing papers in venues, between authors and words representing authors using words, and between words and venues representing words appearing in venues. Note that, in this example, there are multi-edges between the same pair of vertices, since there might be more than one relationships between the same pair of objects. In this paper, we do not consider self-edges that connect a vertex to itself.

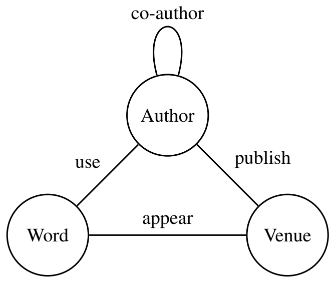


Figure 1: One example of a heterogeneous information network is the bibliographic network constructed from the DBLP database.

In information networks, an object is usually more likely to connect to a certain group of objects with particular characteristics, and therefore, objects form clusters or communities within the network. Cluster analysis [1–3] or community detection [4–8] techniques are developed to reveal such structures in networks. In addition, it is very helpful to identify the important vertices in a network. Ranking methods [9–11] are thus developed to disclose the relative importance of each vertex. Both ranking and clustering draw plenty of attention from the research communities, which are traditionally studied independently.

Combining clustering and ranking together usually achieves better results, as shown in RankClus [12] and NetClus [13], where clustering and ranking are combined together for analyzing heterogeneous networks. However, both algorithms assume that the heterogeneous network follows some particular schema. For example, RankClus assumes

[*]Electrical & Computer Engineering Department, Northeastern University, Boston, MA, USA. {jchen, wei, jdy}@ece.neu.edu

[†]College of Computer and Information Science, Northeastern University, Boston, MA, USA. yzsun@ccs.neu.edu

[1]`http://www.informatik.uni-trier.de/~ley/db/`

that the network is bi-typed, and NetClus assumes the network follows a star network schema. These two methods can not be applied to networks with more complex structures.

We overcome this limitation by developing a Gamma-Poisson generative model, called *GPNRankClus* (**G**amma-**P**oisson **N**etwork Model for **Rank**ing and **Clus**tering), which models the probability of each type of edges without assuming any schema information of the network. To achieve this, we assign each vertex a positive unbounded "ranking score" for each of the clusters. Therefore, the ranking for vertices of a given type in a given cluster can be estimated by sorting the scores associated with that cluster, and the clustering result for a given vertex is given by comparing the scores for that vertex across different clusters. Furthermore, we assume that edges in the network follow Poisson distributions that are parameterized by these scores as well as the link type. By using variational Bayesian inference, we can evaluate the posterior scores for these parameters according to the observed network. Our method is tested on both synthetic and real-world networks. Experimental results show that our method outperforms the state-of-the-art baselines.

The rest of the paper is organized as follows. We introduce the related work in the next section. Our model is introduced in Section 3, and the interpretation about why and when our model works is provided in Section 4. In Section 5, we present in detail the variational inference for our model. In Section 6, we illustrate the experimental results. The paper ends with a conclusion in Section 7.

2 Related Work

There has been an increasing amount of interest and work in automatically detecting clusters/communities and in ranking vertices given a network. However, most of the existing work can only perform either clustering or ranking.

There are a variety of methods that discovers communities in a network by clustering. Spectral clustering [14] is a method that can be applied to the network data by making use of the eigenvectors of the Laplacian matrix, which is derived from the adjacency matrix of the network. Affinity propagation [2] finds the cluster structure in the network by recursively updating the "responsibility" and "availability" messages that are exchanged between data points in the network. SCAN (Structural Clustering Algorithm for Networks)[1] clusters the network data by observing how the nodes share neighbors. Stochastic blockmodels [5] is a probabilistic generative model that assumes each vertex in a binary network belongs to one of the clusters. The model is further extended by introducing mixed membership [6], overlapping clusters [7] and multi-edges [8]. Most of these studies focus on clustering in homogeneous networks, but there also exists some research that extends these methods to heterogeneous networks. For example, in [3], spectral clustering is extended to cluster multi-type relational data.

The most well-known algorithms for ranking is HITS [9] and PageRank [10]. HITS assigns each vertex in the network an authority score, which estimates the value of the content of the page, and a hub score, which evaluates the value of the links of the page to other pages. PageRank assumes a random surfer model, and ranks each page according to a score that represents the average time that a surfer who randomly clicking on links will arrive on that page. Recently, some ranking algorithms are proposed to address the heterogeneity of the network. For example, PopRank [11] ranks the popularity of objects in a heterogeneous network via knowledge propagation.

Most of these methods treat ranking and clustering as independent tasks. But recent studies have shown that ranking and clustering can mutually enhance each other. For example, RankClus [12] and NetClus [13] integrate clustering and ranking together on heterogeneous networks, and achieve better performance in both tasks. In both work, each vertex was assigned two dependent vectors, one representing the probability that the vertex belongs to each cluster and the other representing the within-cluster rank in each cluster for that vertex. As mentioned in Section 1, both methods are only applicable to some specified network schema, i.e., bi-typed schema and star network schema, respectively. In this paper, we propose GPNRankClus, which performs both ranking and clustering a network simultaneously. Moreover, unlike RankClus and NetClus, our proposed model can handle heterogeneous networks for any type of schema.

3 The *GPNRankClus* Model

Let us start by defining our notations. We assume that there are M different types of vertices in the network, denoted as $\{T_m\}_{m=1}^M$. We denote each vertex of type T_m as $\{v_n^{(T_m)}\}_{n=1}^{N_{T_m}}$, where N_{T_m} denotes the number of vertices in type T_m. The total number of vertices in the network is given by $N = \sum_{m=1}^M N_{T_m}$. Multiple edges are allowed between vertices. We denote the number of edges between vertices $v_i^{(T_a)}$ and $v_j^{(T_b)}$ using $\boldsymbol{W}_{ij}^{(T_a,T_b)}$, where $T_a, T_b \in \{T_m\}_{m=1}^M, 1 \le i \le N_{T_a}, 1 \le j \le N_{T_b}$. The edges of type (T_a, T_b) can be summarized in a matrix $\boldsymbol{W}^{(T_a,T_b)} \in \mathbb{R}^{N_{T_a} \times N_{T_b}}$ such that $\boldsymbol{W}_{ij}^{(T_a,T_b)} = \boldsymbol{W}_{ji}^{(T_b,T_a)}$.

In this paper, our goal is to simultaneously solve clustering and ranking problem in heterogeneous networks; i.e., given the vertices and edges of different types, we want to simultaneously achieve the following goals:

1. Clustering: We want to find a partition of the vertices, denoted by K non-overlapping sets $\{C_k\}_{k=1}^K$, such that $\boldsymbol{W}_{ij}^{(T_a,T_b)}$ is likely to have large values if and only if $v_i^{(T_a)}$ and $v_j^{(T_b)}$ belong to the same cluster, i.e., $v_i^{(T_a)}, v_j^{(T_b)} \in C_k$ for a certain $1 \le k \le K$.

2. Ranking: We want to find an approach to measure the relative importance, in terms of *popularity* (number of occurrence), of each vertex in each cluster.

One intuitive solution to solve this problem is to assign a clustering vector and a within-cluster ranking vector to each vertex, as described in RankClus [12] and NetClus [13]. In *GPNRankClus*, we adopt a different approach that assigns each vertex $v_n^{(T_m)}$ one positive K-element *ranking score* vector, denoted by $\boldsymbol{r}_n^{(T_m)} = \{r_{nk}^{(T_m)}\}_{k=1}^K$, that contains both clustering and ranking information, such that

$$(3.1)\quad v_n^{(T_m)} \in C_k \;\Leftrightarrow\; k = \mathbf{argmax}_l(r_{nl}^{(T_m)})$$

$$(3.2)\quad \mathbf{rank}_k(v_i^{(T_m)}) < \mathbf{rank}_k(v_j^{(T_m)}) \;\Leftrightarrow\; r_{ik}^{(T_m)} > r_{jk}^{(T_m)}$$

where $\mathbf{rank}_k(v_n^{(T_m)})$ is a positive integer that defines the rank of the vertex $v_n^{(T_m)}$ for type T_m in the k-th cluster, such that $\mathbf{rank}_k(v_i^{(T_m)}) < \mathbf{rank}_k(v_j^{(T_m)})$ if and only if $v_i^{(T_m)}$ is more important than $v_j^{(T_m)}$ in the k-th cluster. Note that since it is usually not meaningful to compare the importance for vertices from different types, the ranks are defined within each type for each cluster independently. Different from RankClus and NetClus, we let the ranking score $r_{nk}^{(T_m)}$ be unbounded positive real numbers rather than probabilities that sum up to 1, because we notice that one object can have high ranking scores in multiple clusters.

Due to this requirement, we model the ranking score for each object in each cluster as a gamma distribution, which defines the probability over positive real numbers:

$$(3.3)\qquad r_{nk}^{(T_m)} \sim Gamma(\alpha_r, \beta_r).$$

where $\theta_r = (\alpha_r, \beta_r)$ are parameters that determine the shape of a Gamma distribution, and the ratio $\frac{\alpha_r}{\beta_r}$ defines its expected value.

Next, we model the probability of observed links in the network according to these ranking score vectors. As we allow multiple edges between two vertices, we treat the connections between vertices as repeated events, and the number of connections $\boldsymbol{W}_{ij}^{(T_a,T_b)}$ can be naturally modeled as a Poisson distribution:

$$(3.4)\qquad \mathbf{W}_{ij}^{(T_a,T_b)} \sim Pois(\lambda_{(T_a,T_b)}(\mathbf{r}_i^{(T_a)} \cdot \mathbf{r}_j^{(T_a)}))$$

where $\lambda_{(T_a,T_b)}$ is a positive parameter that represents the strength for a specified edge type (T_a, T_b) and $\mathbf{r}_i^{(T_a)} \cdot \mathbf{r}_j^{(T_b)}$ represents the dot product of the two ranking score vectors. We can see that the parameter of the Possion distribution is determined by two parts: (1) $\lambda_{(T_a,T_b)}$, the intensity of edges from type (T_a, T_b), as some relation type might tend to generate more links between vertices than others; and (2) $\mathbf{r}_i^{(T_a)} \cdot \mathbf{r}_j^{(T_b)}$, the dot product of the ranking score vectors of the two vertices. In order to have a large dot product

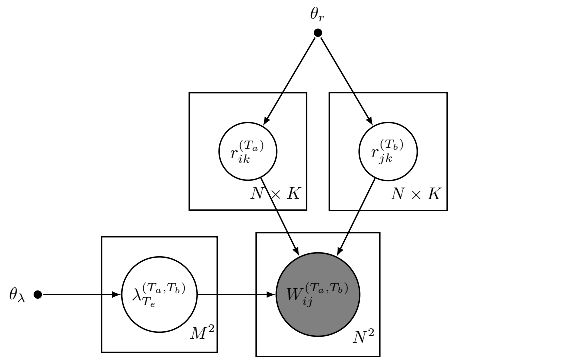


Figure 2: Directed graphical model for the proposed method. The arrows in the graph represent the dependency between random variables.

value, vertices $v_i^{(T_a)}$ and $v_j^{(T_b)}$ need to both have significant influence in at least one of the cluster C_k, i.e., $r_{ik}^{(T_a)} \times r_{jk}^{(T_b)}$ is large. This satisfies the clustering structure assumption we made. A more detailed interpretation of the dot product is introduced in Section 4.

Note that, in an undirected network, relationships (T_a, T_b) are symmetric. Therefore, we let $\lambda_{(T_b,T_a)} = \lambda_{(T_a,T_b)}$. If there are no edges for type (T_a, T_b), we let $\lambda_{(T_a,T_b)} = 0$. We assume each non-zero $\lambda_{(T_a,T_b)}$ also follows a gamma distribution, parameterized by $\theta_\lambda = (\alpha_\lambda, \beta_\lambda)$:

$$(3.5)\qquad \lambda_{(T_a,T_b)} \sim Gamma(\alpha_\lambda, \beta_\lambda)$$

Equations (3.3), (3.4) and (3.5) completely define our generative heterogeneous network, which is summarized in a directed graphical model in Figure 2 and described as follows:

1. For each type $T_m = T_1, T_2, \ldots, T_M$,
   For each vertex $n = 1, \ldots N_{T_m}$,
      For each cluster $k = 1, \ldots, K$,
         Draw $r_{nk}^{(T_m)} \sim Gamma(\alpha_r, \beta_r)$
2. For each non-zero edge type (T_a, T_b), $1 \le a \le b \le M$,
   Draw $\lambda_{(T_a,T_b)} \sim Gamma(\alpha_\lambda, \beta_\lambda)$
   Assign $\lambda_{(T_b,T_a)} = \lambda_{(T_a,T_b)}$
3. For each pair of different vertices $(v_i^{(T_a)}, v_j^{(T_b)})$, s.t. $v_i^{(T_a)} \neq v_j^{(T_b)}$
   Draw $\mathbf{W}_{ij}^{(T_a,T_b)} \sim Pois(\lambda_{(T_a,T_b)}(\mathbf{r}_i^{(T_a)} \cdot \mathbf{r}_j^{(T_b)}))$
   Assign $\mathbf{W}_{ji}^{(T_b,T_a)} = \mathbf{W}_{ij}^{(T_a,T_b)}$

Note that since we are considering an undirected network, $\lambda(T_a, T_b)$ and $\mathbf{W}_{ij}^{(T_a,T_b)}$ are symmetric.

4 Interpretation of the Model

In this section, we explain why and how our model can achieve clustering and ranking within a heterogeneous network from a geometric perspective. We start by examining our model in the ideal case where clusters are well-separated.

4.1 The Ideal Case In Equation (3.4), we assume that $\mathbf{W}_{ij}^{(T_a,T_b)}$ follows a Poisson Distribution parameterized by the intensity parameter $\lambda_{(T_a,T_b)}(\mathbf{r}_i^{(T_a)} \cdot \mathbf{r}_j^{(T_b)})$. The dot product can be expressed as

$$\mathbf{r}_i^{(T_a)} \cdot \mathbf{r}_j^{(T_b)} = \cos\theta \times ||\mathbf{r}_i^{(T_a)}|| \times ||\mathbf{r}_j^{(T_b)}|| \tag{4.6}$$

where θ denotes the angle between the vector $\mathbf{r}_i^{(T_a)}$ and $\mathbf{r}_j^{(T_b)}$, and $||\cdot||$ represents the Euclidean norm of a vector. In order to have a high occurrence of edges between vertices $v_i^{(T_a)}$ and $v_j^{(T_b)}$, our model imposes that the two vertices need to be similar such that (1) they are in the same cluster (i.e., $\cos(\theta)$ is large), and (2) the popularity of both $v_i^{(T_a)}$ and $v_j^{(T_b)}$ need to be high (i.e., both $||\mathbf{r}_i^{(T_a)}||$ and $||\mathbf{r}_j^{(T_b)}||$ are large). Therefore, having the intensity parameter to be proportional to the dot product, the impact of clustering and ranking to link generation are both considered.

Now assume that the network can be divided into two well-separated clusters C_1 and C_2, meaning that the vertices within clusters C_1 and C_2 are well connected and that there are almost no edges between clusters C_1 and C_2. If vertices $v_i^{(T_a)}$ and $v_j^{(T_a)}$ are from the same cluster (e.g., $v_i^{(T_a)}, v_j^{(T_b)} \in C_1$), we expect $\cos\theta$ to be large, i.e., $\mathbf{r}_i$ and $\mathbf{r}_j$ are almost parallel with each other (e.g., vectors $(1,0)$ and $(2,0)$ in $\mathbb{R}^2$). If $v_i^{(T_a)}$ and $v_j^{(T_b)}$ are from different clusters (e.g., $v_i^{(T_a)} \in C_1, v_j^{(T_b)} \in C_2$), we expect $\cos\theta$ to be very small, i.e., the latent vectors $\mathbf{r}_i^{(T_a)}$ and $\mathbf{r}_j^{(T_b)}$ are almost perpendicular with each other (e.g., $(1,0)$ and $(0,1)$ in $\mathbb{R}^2$). Because of the property described above, we can represent all the clusters by a set of perpendicular unit vectors $\{\mathbf{c}_k\}_{k=1}^K$, each of which can be considered as the center of all the $\mathbf{r}_n^{(T_m)}$ vectors in that cluster. Since we assume all the elements of $\mathbf{r}_n^{(T_m)}$ are always positive, the only set of K unit vectors in the space $\mathbb{R}_0^{+K}$ that are perpendicular to each other are the set that are located at the K axes, which is illustrated in $\mathbb{R}^2$ in Figure 3. Therefore, if we choose the dimensionality of the vectors $\mathbf{r}_n^{(T_m)}$ to be equal to the number of clusters K, the vectors $\{\mathbf{c}_k\}_{k=1}^K$ will be located at the axes. In this case, the cluster that vertex $v_n^{(T_m)}$ belongs to is given by

$$v_n^{(T_m)} \in C_k,\ where\ k = \mathbf{argmax}_l(\mathbf{r}_{nl}^{(T_m)}). \tag{4.7}$$

Because the number of links that connect to a given vertex $v_n^{(T_m)}$ depends on the value of $||\mathbf{r}_n^{(T_m)}||$, we can determine the relative popularity of $v_i^{(T_m)}$ in each cluster by observing the projection of the vector $\mathbf{r}_n^{(T_m)}$ to the corresponding axis. Therefore, the rank for vertex $v_n^{(T_m)}$ within vertices of type T_m in cluster k is given by

$$\mathbf{rank}_k(v_n^{(T_m)}) = \mathbf{argsort}_i(\mathbf{r}_{ik}^{(T_m)}). \tag{4.8}$$

4.2 The Non-Ideal Case In a non-ideal case, the vectors in the same cluster are not parallel to each other, and the

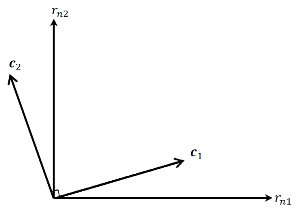


Figure 3: Plot of ranking score vectors in $\mathbb{R}^2$ space. Assuming that $\mathbf{c_1}$ and $\mathbf{c_2}$ are orthogonal, if $\mathbf{c_1}$ is not aligned with one of the axes, then either $\mathbf{c}_{21} < 0$ or $\mathbf{c}_{22} < 0$, which contradicts with the assumption $\mathbf{c_2} \in \mathbb{R}_0^{+2}$. Therefore, both $\mathbf{c_1}$ and $\mathbf{c_2}$ have to align with the axes.

vectors in different clusters will no longer be perpendicular. Thus, the vectors represented by cluster $\{\mathbf{c}_k\}_{k=1}^K$ are not perpendicular with each other and can locate arbitrarily in the space. We solve this problem by introducing "seeds", i.e., for each cluster $1 \le k \le K$, we seed (select) L representative objects of one or more types, which are indexed by $\{S_{kl}\}_{l=1}^L$ (we only used 1 seed for each cluster in our experiments). We incorporate this *seed* information through their respective prior distributions. Rather than assign the default prior for $\mathbf{r}$ as described in Equation (3.3), we assign a special prior distribution for these "seeds" such that

$$\begin{aligned} &\mathbf{r}_{S_{kl},k} \sim Gamma(\alpha_{S0}, \beta_{S0}) \\ &\mathbf{r}_{S_{kl},t} \sim Gamma(\alpha_{S1}, \beta_{S1}) \quad for\ all\ t \neq k \end{aligned} \tag{4.9}$$

where α_{S0}, β_{S0}, α_{S1} and β_{S1} are pre-defined hyper-parameters that satisfies $\frac{\alpha_{S0}}{\beta_{S0}} > \frac{\alpha_r}{\beta_r} > \frac{\alpha_{S1}}{\beta_{S1}}$. Note that, the expected value of a gamma distribution $Gamma(\alpha, \beta)$ is given by $\frac{\alpha}{\beta}$. The above constraint is equivalent to force $\mathbb{E}[\mathbf{r}_{S_{kl},k}] > \mathbb{E}[\mathbf{r}_{NS,j}] > \mathbb{E}[\mathbf{r}_{S_{kl},t}]$, where we use $\mathbf{r}_{NS,j}$ to represent the ranking score for arbitrary data points that are not seeds. By assigning such special priors for the "seeds," $\mathbf{r}_{S_{kl}}$ will tend to have a large value at the direction of the k-th axis, and to have a small value at all the orthogonal directions. Thus, we can force the seeds for the k-th cluster close to the k-th axis. Since the $\mathbf{r}_n^{(T_m)}$ vector for each vertex in a cluster tends to be parallel to each other, we now are able to force $\mathbf{r}_n^{(T_m)}$ vectors for objects in the k-th cluster close to the k-th axis.

In practice we choose $\alpha_{S0} = \alpha_{S1} = \alpha_r = \beta_r = 1$, $\beta_{S0} = 0.01$ and $\beta_{S0} = 100$. We choose α's to be small because this is a prior distribution, and we do not want to be too confident about the choice of the expected values before observing data. We make $\mathbb{E}[\mathbf{r}_{S_{kl},k}] = \frac{\alpha_{S0}}{\beta_{S0}} = 100$ and $\mathbb{E}[\mathbf{r}_{S_{kl},t}] = \frac{\alpha_{S0}}{\beta_{S0}} = 0.01$ such that the seeds are more likely to be located at the axes. Note that the posterior distribution tends to differ significantly from these values especially when we choose small α's.

After introducing seeds, the object rank in each cluster can be obtained from Equation (4.8). However, now it is tricky to get the clustering results, as the connectivity in each cluster might differ, i.e., some clusters might be more well-connected than others. If we simply follow Equation (4.7), we will tend to misclassify objects in the less-connected cluster (tends to have a lower ranking score) into the well-connected cluster (tends to have a higher ranking score). Therefore, we want to normalize the clustering results with respect to the degrees in each of the clusters. As a result, rather than directly utilize the vector $\mathbf{r}_n^{(T_m)}$, we introduce a heuristic by normalizing the vector as follows

$$v_n^{(T_m)} \in C_k, \; where \; k = \mathbf{argmax}_l \left(\frac{\mathbf{r}_{nl}^{(T_m)}}{\tilde{\mathbf{r}}_{\cdot l}^{(T_m)}}\right) \tag{4.10}$$

where $\tilde{\mathbf{r}}_{\cdot l}^{(T_m)}$ denotes the median value for all the l-th element of vectors $\mathbf{r}_n^{(T_m)}$, $1 \le n \le N_{T_m}$. Note that for vertices of different types, the normalization constant in the denominator in Equation (4.10) are different. We choose to normalize the vectors with respect to the median values rather than mean or the maximum, because the values in the vector differ significantly on magnitude, ranging from 10^2 to 10^{-2}, and median is less sensitive to the skewness of the data.

5 Variational Inference

In this section, we introduce how the parameters in the proposed *GPNRankClus* model can be learned via a variational inference algorithm. The joint probability of our model as shown in Figure 2 is given by

$$\begin{aligned} & p(\boldsymbol{\lambda}, \boldsymbol{R}, \boldsymbol{W} | \theta_R, \theta_\lambda) \\ & = \prod_{a,b,i,j}^{v_i^{(T_a)} \neq v_j^{(T_b)}} p(\boldsymbol{W}_{ij}^{(T_a,T_b)} | \lambda_{(T_a,T_b)}, \boldsymbol{r}_i^{(T_a)}, \boldsymbol{r}_j^{(T_a)}) \\ & \prod_{m=1}^{M} \prod_{n=1}^{N_{T_m}} \prod_{k=1}^{K} p(\boldsymbol{r}_{nk}^{(T_m)} | \theta_R) \prod_{a=1}^{M} \prod_{b=1}^{a} p(\lambda_{(T_a,T_b)} | \theta_\lambda) \end{aligned} \tag{5.11}$$

We now learn the latent variables $\mathbf{R}$ and parameters λ using variational inference [15].

5.1 The Variational Distribution It is computationally intractable to evaluate the marginal likelihood, $\log p(\mathbf{W}|\theta_R, \theta_\lambda)$ directly. We use variational methods to approximate the marginal likelihood by maximizing a lower bound, $\mathcal{L}(q)$, on the true log marginal likelihood [16], which can be obtained by using Jensen's inequality as follows:

(5.12)

$$\begin{aligned} & \log p(\mathbf{W}|\theta_r, \theta_\lambda) \ge \mathcal{L}(q) = \int q(\boldsymbol{\lambda}, \boldsymbol{R}) \log \frac{p(\boldsymbol{\lambda}, \boldsymbol{R}, \boldsymbol{W})}{q(\boldsymbol{\lambda}, \boldsymbol{R})} d\boldsymbol{\lambda} d\boldsymbol{R} \\ & = \sum_{a,b,i,j}^{v_i^{(T_a)} \neq v_j^{(T_b)}} \mathbb{E}_{\boldsymbol{R},\boldsymbol{\lambda}}[\log p(\boldsymbol{W}_{ij}^{(T_a,T_b)} | \lambda_{(T_a,T_b)} (\boldsymbol{r}_i^{(T_a)} \cdot \boldsymbol{r}_j^{(T_b)}))] \\ & + \sum_{m=1}^{M} \sum_{n=1}^{N_{T_m}} \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{R}}[\log p(r_{nk}^{(T_m)} | \theta_r)] \\ & + \sum_{a=1}^{M} \sum_{b=1}^{a} \mathbb{E}_{\boldsymbol{\lambda}}[\log p(\lambda_{(T_a,T_b)} | \theta_\lambda)] + \mathcal{H}(q(\boldsymbol{\lambda}, \boldsymbol{R})) \end{aligned}$$

where $q(\boldsymbol{\lambda}, \boldsymbol{R})$ is an approximate variational distribution of the posterior probability $p(\boldsymbol{\lambda}, \boldsymbol{R}|\mathbf{W}, \theta_r, \theta_\lambda)$ and $\mathcal{H}(q(\boldsymbol{\lambda}, \boldsymbol{R}))$ denotes the entropy of distribution $q(\boldsymbol{\lambda}, \boldsymbol{R})$. We apply mean-field approximation and assume that the variational distribution can be factorized into disjoint groups, such that

(5.13)

$$q(\boldsymbol{\lambda}, \boldsymbol{R}) = \left(\prod_{a=1}^{M} \prod_{b=1}^{a} q(\lambda_{(T_a,T_b)}) \right) \left(\prod_{m=1}^{M} \prod_{n=1}^{N_{T_m}} \prod_{k=1}^{K} q(r_{nk}^{(T_m)}) \right)$$

5.2 Auxiliary Variables With this approximation, all the expected values in Equation (5.12) are straightforward to compute, except for $\mathbb{E}_{\boldsymbol{R},\boldsymbol{\lambda}}[\log p(\boldsymbol{W}_{ij}^{(T_a,T_b)} | \lambda_{(T_a,T_b)} (\boldsymbol{r}_i^{(T_a)} \cdot \boldsymbol{r}_j^{(T_b)}))]$, which is given by

(5.14)

$$\begin{aligned} & \mathbb{E}_{\boldsymbol{R},\boldsymbol{\lambda}}[\log p(\boldsymbol{W}_{ij}^{(T_a,T_b)} | \lambda_{(T_a,T_b)} (\boldsymbol{r}_i^{(T_a)} \cdot \boldsymbol{r}_j^{(T_b)}))] \\ & = \boldsymbol{W}_{ij}^{(T_a,T_b)} \mathbb{E}_{\boldsymbol{\lambda}}[\log \lambda_{(T_a,T_b)}] + \boldsymbol{W}_{ij}^{(T_a,T_b)} \mathbb{E}_{\boldsymbol{R}}[\log \sum_{k=1}^{K} r_{ik}^{(T_a)} r_{jk}^{(T_b)}] \\ & - \mathbb{E}_{\boldsymbol{\lambda}}[\lambda_{(T_a,T_b)}] \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{R}}[r_{ik}^{(T_a)}] \mathbb{E}_{\boldsymbol{R}}[r_{jk}^{(T_b)}] + const \end{aligned}$$

where $const$ represents constant terms that do not contain the variables we are interested in. It is difficult to directly estimate $\mathbb{E}_{\boldsymbol{R}}[\log \sum_{k=1}^{K} r_{ik}^{(T_a)} r_{jk}^{(T_b)}]$. Therefore, we introduce auxiliary variables $y_{ij}^{(T_a,T_b)}$ that follow multivariate distributions denoted by $q_{y_{ij}^{(T_a,T_b)}}(y)$ and apply Jensen's inequality again, such that

(5.15)

$$\begin{aligned} & \mathbb{E}_{\boldsymbol{R}}[\log \sum_{k=1}^{K} r_{ik}^{(T_a)} r_{jk}^{(T_b)}] \\ & = \mathbb{E}_{\boldsymbol{R}}[\log \left\{ \sum_{k=1}^{K} q_{y_{ij}^{(T_a,T_b)}}(k) \frac{r_{ik} r_{jk}}{q_{y_{ij}^{(T_a,T_b)}}(k)} \right\}] \\ & \ge \mathbb{E}_{\boldsymbol{R}}[\mathbb{E}_{y_{ij}^{(T_a,T_b)}}[\log(r_{iy}^{(T_a)} r_{jy}^{(T_b)})]] + \mathcal{H}(q_{y_{ij}^{(T_a,T_b)}}) \\ & = \mathbb{E}_{\boldsymbol{R}, y_{ij}^{(T_a,T_b)}}[\log r_{iy}^{(T_a)}] + \mathbb{E}_{\boldsymbol{R}, y_{ij}^{(T_a,T_b)}}[\log r_{jy}^{(T_b)}] + \mathcal{H}(q_{y_{ij}^{(T_a,T_b)}}) \end{aligned}$$

where $\mathcal{H}(q_{y_{ij}^{(T_a,T_b)}})$ denotes the entropy of the distribution $q_{y_{ij}^{(T_a,T_b)}}(y)$.

By plugging Equation (5.15) into (5.14) and then into (5.12), we get a looser lower bound $\mathcal{L}'(q)$. We approximately maximize the marginal likelihood by maximizing this lower bound $\mathcal{L}'(q)$.

We first take the derivative of $\mathcal{L}'(q)$ with respect to $q_{y_{ij}^{(T_a,T_b)}}(y)$ to find the optimal $q^*_{y_{ij}^{(T_a,T_b)}}(y)$ that maximizes the lower bound, which is given by

$$q^*_{y_{ij}^{(T_a,T_b)}}(y) \propto \exp\{\mathbb{E}_{\boldsymbol{R}}[\log r_{iy}^{(T_a)}] + \mathbb{E}_{\boldsymbol{R}}[\log r_{jy}^{(T_b)}]\} \tag{5.16}$$

The normalization factor can be calculated by ensuring

$\sum_{y=1}^{K} q^*_{y_{ij}^{(T_a,T_b)}}(y) = 1$.

5.3 Optimal Variational Distributions With the factorized mean-field approximation above, we can obtain the optimal distributions for q that maximize the lower bound, by applying the following Equation [15]:

$$\log q^*_{\mathbf{Z}_j}(\mathbf{Z}) = \mathbb{E}_{\mathbf{Z}_{-j}}[\log p(\mathbf{X}, \mathbf{Z})] + const, \tag{5.17}$$

where $\mathbf{Z}$ represents the latent variables, $\mathbf{X}$ represents the observations, $p(\mathbf{X}, \mathbf{Z})$ represents the joint distribution , $q^*_{\mathbf{Z}_j}(\mathbf{Z})$ represents the optimal distribution for the latent variable $\mathbf{Z}_j$ that maximizes the lower bound, and $\mathbb{E}_{\mathbf{Z}_{-j}}$ represents that the expected value is taken with respect to all other latent variables except for $\mathbf{Z}_j$. In our model, $\mathbf{X} = \mathbf{W}$ and $\mathbf{Z} = \{\mathbf{R}, \boldsymbol{\lambda}\}$.

By applying Equation (5.17), we observe that the optimal distribution for $\lambda_{(T_a,T_b)}$ is a gamma distribution such that $q^*_{\boldsymbol{\lambda}}(\lambda_{(T_a,T_b)}) = Gamma(\alpha_{(T_a,T_b)}, \beta_{(T_a,T_b)})$, where

$$\begin{aligned} \alpha_{(T_a,T_b)} &= \alpha_{\lambda} + \sum_{i,j}^{v_i^{(T_a)} \neq v_j^{(T_b)}} \boldsymbol{W}_{ij}^{(T_a,T_b)} \\ \beta_{(T_a,T_b)} &= \beta_{\lambda} + \sum_{i,j}^{v_i^{(T_a)} \neq v_j^{(T_b)}} \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{R}}[r_{ik}^{(T_a)}]\mathbb{E}_{\boldsymbol{R}}[r_{jk}^{(T_b)}] \end{aligned} \tag{5.18}$$

The optimal distribution for $r_{nk}^{(T_m)}$ is a gamma distribution such that $q^*_{\boldsymbol{R}}(r_{nk}^{(T_m)}) = Gamma(\alpha_{nk}^{(T_m)}, \beta_{nk}^{(T_m)})$, where

$$\begin{aligned} \alpha_{nk}^{(T_m)} &= \alpha_r + \sum_{a=1}^{M} \sum_{i=1}^{N_{T_a}} \boldsymbol{W}_{ni}^{(T_m,T_a)} q_{y_{ni}^{(T_m,T_a)}}(k) \\ \beta_{nk}^{(T_m)} &= \beta_r + \sum_{a,i}^{v_n^{(T_m)} \neq v_i^{(T_a)}} \mathbb{E}_{\boldsymbol{\lambda}}[\lambda_{(T_m,T_a)}]\mathbb{E}_{\boldsymbol{R_{-n}}}[r_{ik}^{(T_a)}] \end{aligned} \tag{5.19}$$

The expected values are given as:

$$\begin{aligned} \mathbb{E}_{\boldsymbol{R}}[r_{nk}^{(T_m)}] &= \frac{\alpha_{nk}^{(T_m)}}{\beta_{nk}^{(T_m)}} \\ \mathbb{E}_{\boldsymbol{R}}[\log r_{nk}^{(T_m)}] &= \Psi(\alpha_{nk}^{(T_m)}) - \log \beta_{nk^{(T_m)}} \\ \mathbb{E}_{\boldsymbol{\lambda}}[\lambda_{(T_a,T_b)}] &= \frac{\alpha_{(T_a,T_b)}}{\beta_{(T_a,T_b)}} \\ \mathbb{E}_{\boldsymbol{\lambda}}[\log \lambda_{(T_a,T_b)}] &= \Psi(\alpha_{(T_a,T_b)}) - \log \beta_{(T_a,T_b)} \end{aligned} \tag{5.20}$$

where $\Psi(\cdot)$ represents the digamma function, i.e., the first derivative of the logarithm of a gamma function. We plug in these expected values into the update equations (5.16), (5.18) and (5.19), when updating the variational distributions. Because each time when we apply each of these update equations, the lower bound $\mathcal{L}'(q)$ increases, the algorithm is guaranteed to converge.

We repeat the update for each of the variational distribution until the algorithm converges. In practice, we stop the algorithm when the increase of lower bound $\mathcal{L}'(q)$ during an update is less than 10^{-5} of its original value. Algorithm 1 provides a pseudo-code and summary of our approach.

Algorithm 1 Variational Inference for GPNRankClus

```
 1: repeat
 2:     for each $(a, b, i, j)$ s.t. $\mathbf{W}_{ij}^{(T_a,T_b)} > 0$ do
 3:         Update $q^*_{y_{ij}}(y_{ij})$ according to Equation (5.16)
 4:         Normalize $q^*_{y_{ij}}(y_{ij})$
 5:     end for
 6:     for $m \leftarrow 1$ to $M$ do
 7:         for $r \leftarrow 1$ to $N_{T_m}$ do
 8:             for $k \leftarrow 1$ to $K$ do
 9:                 Update $q^*_{\boldsymbol{R}}(r_{nk}^{(T_m)})$ according to Equation (5.19)
10:             end for
11:         end for
12:     end for
13:     for each (a,b) s.t. $1 \leq a \leq b \leq M$ do
14:         Update $q^*_{\boldsymbol{\lambda}}(\lambda_{(T_a,T_b)})$ according to Equation (5.18)
15:     end for
16: until Convergence
```

6 Experiments

In this section, we investigate the effectiveness of our algorithm using both synthetic and real datasets.

6.1 Synthetic data We generate synthetic data based on the generative model mentioned in Section 3. As a sanity check, we first examine whether or not our algorithm can recover the clusters if the network was generated from the same modeling assumption as our approach. Our synthetic data consists of four different types of objects, represented by T_1, T_2, T_3 and T_4 respectively, clustered into two clusters, denoted by C_1 and C_2. We generate $\{\{\mathbf{r}_n^{T_m}\}_{n=1}^{N_{(T_m)}}\}_{m=1}^{M}$ as follows:

$$\begin{aligned} &\left.\begin{aligned} r_{n1}^{(T_m)} &\sim Gamma(1, 0.1) \\ r_{n2}^{(T_m)} &\sim Gamma(1, 1) \end{aligned}\right\} \text{if } v_n \in C_1, \\ &\left.\begin{aligned} r_{n1}^{(T_m)} &\sim Gamma(1, 1) \\ r_{n2}^{(T_m)} &\sim Gamma(1, 0.1) \end{aligned}\right\} \text{if } v_n \in C_2, \end{aligned} \tag{6.21}$$

In the equations above, $r_{n1}^{(T_m)}$ is more likely to be larger than $r_{n2}^{(T_m)}$, if $v_n^{(T_m)} \in C_1$, and vice versa. We generate $\{\lambda_{(T_a,T_b)}\}_{1 \leq a \leq b \leq 4}$ using

$$\lambda_{(T_a,T_b)} \sim Gamma(0.1, 0.02) \tag{6.22}$$

such that the intensity for each edge type varies significantly. To make the network more realistic, we add noise in generating the edges, i.e., instead of using Equation (3.4), we generate $\mathbf{W}_{ij}^{(T_a,T_b)}$ using

$$\mathbf{W}_{ij}^{(T_a,T_b)} \sim Pois(\lambda_{(T_a,T_b)}(\mathbf{r}_i^{(T_a)} \cdot \mathbf{r}_j^{(T_b)}) + \epsilon) \tag{6.23}$$

where ϵ follows a zero-mean normal distribution with variance σ^2. We let $\mathbf{W}_{ij}^{(T_a,T_b)} = 0$, if $\lambda_{(T_a,T_b)}(\mathbf{r}_i^{(T_a)} \cdot \mathbf{r}_j^{(T_b)}) + \epsilon \leq 0$

We generated 50 objects of each type in each of the clusters resulting in a data size of 400. We test our model on this data. Because GPNRankClus is based on a similar generative process, it would be unfair to compare the experimental results with other methods. Therefore, in this section we concentrate on testing the GPNRankClus at different noise level. We set the hyper-parameters of our model as $\alpha_r = \beta_r = \alpha_\lambda = \beta_\lambda = 1$. We do not use any "seeds" in this experiment. We vary the noise level σ^2, and report clustering accuracy as we vary the noise level in Figure 4(a).

As expected, we observe from Figure 4(a) that when we increase the noise level from $\sigma^2 = 0.1$ to $\sigma^2 = 10$, the clustering accuracy decreases from 93% to 75%. To better understand how noise affects the ranking score and therefore influence the clustering results, we plot the expected value of ranking score $\{\mathbb{E}(\mathbf{r}_n^{(T_1)})\}_{n=1}^{N_{T_1}}$ at different noise level in Figure 4. We observe from the figure that, when the noise level is very low, i.e. $\sigma^2 = 0.1$ illustrated in Figure 4(b), the data points that belong to each cluster are located close to the axes, which is consistent to the analysis in Section 4. In this case, the two clusters can be clearly separated by the decision boundary defined in Equation (4.10). When we increase the noise to the intermediate level, i.e. $\sigma^2 = 2$, illustrated in Figure 4(c), the data points deviate from the axes, but our approach is still able to cluster most of the points correctly. However, when we increase the noise level to $\sigma^2 = 10$, illustrated in Figure 4(d), the data points will not be located close to the axes, and we can only separate a subset of the data points into the correct clusters.

6.2 Experiments on Real Data We now test the performance of our model on two real heterogeneous network data sets: DBLP and YELP data. In these experiments, we set the parameters for GPNRankClus as follows: We let $\alpha_r = \beta_r = \alpha_\lambda = \beta_\lambda = 1$. These parameter indicate we apply non-informative priors for our model. The parameters for the seeds are set as follows: $\alpha_{S0} = \alpha_{S1} = 1$, $\beta_{S0} = 0.01$, $\beta_{S1} = 100$. Note that these parameters will affect the prior distribution for only the seed vertices, and experiments show that it does not significantly influence the experimental results as long as these values are set in a reasonable range.

We compare GPNRankClus with state-of-the-art algorithms in terms of clustering ability. In addition to NetClus [13], which is introduced in Section 2, we also compare GPNRankClus with the following two existing classification methods for heterogeneous networks:

- GNetMine [17], a transductive classification method in heterogeneous networks; and
- RankClass [18], a ranking-based classification method in heterogeneous networks.

We are not able to run the multi-type spectral clustering [3] because of the size of the dataset. For NetClus, we set the parameters $\lambda_P = 0.9$ and $\lambda_S = 0.5$. For both GNetMine and RankClass, we follow the parameters used in [17] and [18] and set parameters $\alpha_i = 0.1$ and $\lambda_{ij} = 0.2$.

Table 1: Classification Results on DBLP Data

Classification accuracy on authors

	GPNRankClus	NetClus	GNetMine	RankClass
Accuracy	**92.28%**	76.11%‡	80.67%	91.12%

Classification Accuracy on Conferences

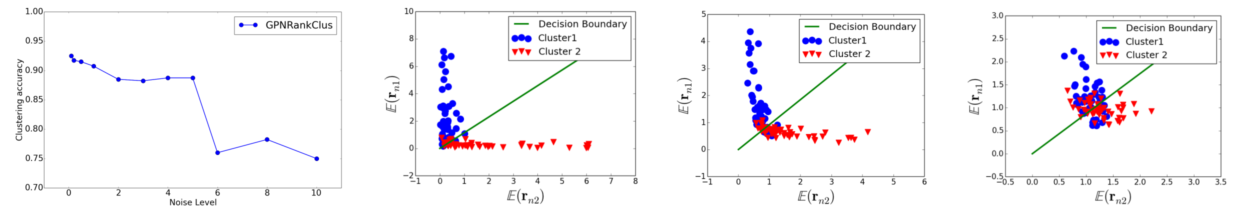
	GPNRankClus	NetClus	GNetMine	RankClass
Accuracy	**100%**	85%‡	**100%**	**100%**

‡We test NetClus on the star-schema version of the DBLP dataset.

6.2.1 DBLP Dataset The DBLP network data includes 20 conferences from four related areas (*database (DB), data mining (DM), machine learning (ML) and information retrieval (IR)*). It includes 28,702 authors and their publications in these conferences. As described in Figure 1, this dataset has three types of vertices (*conferences, author, words*). We consider four types of edges (*conference-author, conference-word, author-word, author-author*) in this network. In this experiment, we used SIGMOD, KDD, ICML and SIGIR as seeds in each of the research area respectively.

The classification accuracy on authors and conferences are summarized in Table 1. In this table, we observe that GPNRankClus outperforms all other methods in terms of author classification accuracy, achieving 92.28% accuracy. The performance of RankClass perform slightly worse. GNetMine is less effective in this task, achieving 80.67% accuracy. GPNRankClus, GNetMine and RankClass achieve perfect accuracy in classifying the conferences. Because the DBLP data is not in star-schema, we can not directly apply NetClus on this dataset. Therefore, we test NetClus using the star-schema version of the DBLP dataset, in which paper is a center type and author, venue and words are linked via papers. The performance of NetClus is worse, achieving 76.11% accuracy on authors and 85% accuracy on conferences. Note that the clustering performance of NetClus is better in [13], but more seeds are used in their experiment.

Table 2 provides a summary of the ranking results obtained by GPNRankClus. Note that in this dataset, we do not have data that reflects the influence or reputation of the authors and conferences, such as citations; therefore, the ranking results is based on popularity. We observe that the ranking results make sense. The highly ranked terms coincide the representative keywords in each of the fields. The top ranked conferences also agree with the top conferences in each research field. Because we do not have the ground truth, we are not able to measure the ranking results quantitatively.



(a) Clustering Accuracy vs. Noise Level (b) $\mathbb{E}(\mathbf{r}_n^{(T_1)})$ under noise level $\sigma^2 = 0.1$ (c) $\mathbb{E}(\mathbf{r}_n^{(T_1)})$ under noise level $\sigma^2 = 2$ (d) $\mathbb{E}(\mathbf{r}_n^{(T_1)})$ under noise level $\sigma^2 = 10$

Figure 4: Plots for synthetic data

Table 2: Ranking Results on DBLP Data

Top-5 Terms in Each Cluster

	DB	**DM**	**ML**	**IR**
1	data	data	learning	web
2	database	mining	knowledge	retrieval
3	databases	learning	system	information
4	query	clustering	reasoning	search
5	system	classification	model	text

Top-5 Conferences in Each Cluster

	DB	**DM**	**ML**	**IR**
1	VLDB	KDD	IJCAI	SIGIR
2	ICDE	PAKDD	AAAI	WWW
3	SIGMOD	ICDM	ICML	CIKM
4	PODS	PKDD	CVPR	ECIR
5	EDBT	SDM	ECML	AAAI

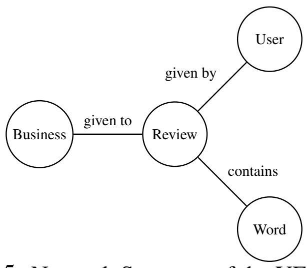


Figure 5: Network Structure of the YELP Data.

6.2.2 YELP Dataset The YELP network data contains four types of vertices: *businesses (B), reviews (R), users (U), and words(W)*. There are three types of edges between various types of objects, including edges between businesses and reviews, representing the review given to the business; edges between reviews and users, representing the review given by the user; edges between reviews and words, representing the review containing the corresponding word. The relationship network of the YELP data is shown in Figure 5.

In this dataset, each business is associated with one or more hierarchical categories. This data can also be interpreted with different clustering tasks. We focus our analysis by examining a subset of the YELP dataset for the following three different clustering tasks:

- Level-1 categories: We deal with the businesses in the following categories: health & medical; food; shopping; beauty & spas. It involves $2,747$ businesses, $49,429$ reviews and $29,612$ users.
- Restaurant categories: We deal with the restaurant businesses in the following categories: Sandwiches; Thai; American (New); Mexican; Italian; Chinese. It involves 966 businesses, $37,066$ reviews and $23,946$ users.
- Shopping categories: We deal with the shopping businesses in the following categories: Eyewear & Opticians; Books, Mags, Music & Video; Sporting Goods; Fashion; Drugstores; Home & Garden. It involves 644 businesses, $7,996$ reviews and $5,717$ users.

In each task, we remove the businesses that do not fall in the categories we are interested in. We also remove the businesses with one or no review, because we do not have enough information to cluster these businesses. In each task, we select one business with the most reviews in each of the categories as *seeds*.

In addition to classification accuracy, we also report the Normalized Mutual Information (NMI) between the predicted labels and the actual labels. The NMI between two random variables X and Y is defined as [19]

$$\frac{\sum_{x\in X}\sum_{y\in Y} p(x,y)[\log p(x,y) - \log p(x)p(y)]}{\sqrt{\mathcal{H}(X)\mathcal{H}(Y)}} \quad (6.24)$$

where $\mathcal{H}(X)$ and $\mathcal{H}(Y)$ are the entropy for random variables X and Y respectively. NMI is a measure of the variables mutual dependence. The NMI ranges from 0 to 1, where a higher value indicates X and Y agree with each other. We measure NMI here but not in DBLP, because the classification accuracy in DBLP is very high. A high accuracy already implies a high NMI.

The accuracy and NMI are summarised in Table 3. We observe from the table that the classification accuracy is lower in general compared to the DBLP data. This is due to that the YELP data is much noisier. The businesses are classified merely according to online reviews; however, there are many reviews conveying no categorical information, containing only words such as "place", "good" and "like." In Table 3, we observe that GPNRankClus achieves higher classification accuracy than other methods in all of the tasks. The classification accuracy of GPNRankClus is higher than GNetMine by 9.09% and 17.45% in the Level 1 and Restaurant tasks, respectively. In the Shopping task, GPNRankClus achieves almost identical accuracy as GNetMine. The accuracy difference between GPNRankClus and RankClass is even higher

Table 3: Classification Results on the YELP Data

Classification accuracy on businesses

	GPNRankClus	NetClus	GNetMine	RankClass
Level 1	**56.25%**	17.78%	47.16%	37.19%
Restaurant	**66.81%**	15.31%	49.36%	57.11%
Shopping	**64.62%**	13.28%	64.45%	32.58%

NMI on businesses

	GPNRankClus	NetClus	GNetMine	RankClass
Level 1	**0.5590**	0.0168	0.1387	0.1579
Restaurant	**0.6606**	0.0187	0.2346	0.3044
Shopping	**0.4721**	0.0313	0.3617	0.2335

in these three tasks, ranging from 9.7% to 32.04%. The clustering results given by NetClus are not significantly different from random guess. To get a more reasonable results, NetClus usually require more and stronger seeds.

In terms of NMI, GPNRankClus outperforms other methods more significantly. By observing the confusion matrix, we found that although the predicted labels given by GPNRankClus do not perfectly agree with the actual labels, GPNRankClus found other meaningful clustering structures. However, the misclassification results for GNetMine and RankClass are more random, resulting in significantly lower NMI values. Due to the limited space, we introduce the detailed confusion matrix in the supplemental materials. The supplemental materials and the code are available at `http://www1.ece.neu.edu/~jchen`.

7 Conclusion

In this paper, we propose to simultaneously cluster and rank items in the heterogeneous information network with arbitrary schema. In order to solve this problem, we introduce a new concept *ranking score* that conveys both ranking and clustering results. Based on this concept, we propose a novel generative model, called *GPNRankClus*, to model the likelihood of the observed heterogeneous information network. More specifically, Gamma-Poisson model is used, where we model the ranking score of each vertex in each cluster as a gamma distribution, and the number of edges between two vertices as a Poisson distribution that takes the ranking scores of the two vertices and the strength of link type as parameters. Experiments on DBLP and YELP data show that our method outperforms the state-of-the-art methods.

Acknowledgement

This work was partially supported by NIH/NHLBI grants R01HL089856 & R01HL089857.

References

[1] X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger, "Scan: a structural clustering algorithm for networks," in *KDD'2007, San Jose, CA*, 2007, pp. 824–833.

[2] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[3] B. Long, Z. M. Zhang, X. Wu, and P. S. Yu, "Spectral clustering for multi-type relational data," in *ICML'06, Pittsburgh, Pennsylvania, USA*, 2006, pp. 585–592.

[4] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[5] T. A. Snijders and K. Nowicki, "Estimation and prediction for stochastic blockmodels for graphs with latent block structure," *Journal of Classification*, vol. 14, no. 1, pp. 75–100, 1997.

[6] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels." *Journal of Machine Learning Research*, vol. 9, no. 1981-2014, p. 3, 2008.

[7] P. Latouche, E. Birmelé, C. Ambroise *et al.*, "Overlapping stochastic block models with application to the french political blogosphere," *The Annals of Applied Statistics*, vol. 5, no. 1, pp. 309–336, 2011.

[8] B. Karrer and M. E. Newman, "Stochastic blockmodels and community structure in networks," *Physical Review E*, vol. 83, no. 1, 2011.

[9] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 604–632, 1999.

[10] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." *Computer Networks*, vol. 30, 1998.

[11] Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma, "Object-level ranking: bringing order to web objects," in *WWW'05, Chiba, Japan*, 2005, pp. 567–574.

[12] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu, "Rankclus: integrating clustering with ranking for heterogeneous information network analysis," in *EDBT'09, Saint-Petersburg, Russia*, 2009, pp. 565–576.

[13] Y. Sun, Y. Yu, and J. Han, "Ranking-based clustering of heterogeneous information networks with star network schema," in *KDD'09, Paris, France*, 2009, pp. 797–806.

[14] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 888–905, 2000.

[15] C. M. Bishop *et al.*, *Pattern recognition and machine learning*. springer New York, 2006, vol. 1, ch. 10 Approximate Inference, pp. 461 – 474.

[16] C. M. Bishop, "Variational principal components," in *ICANN99, Edinburgh, UK*, 1999.

[17] M. Ji, Y. Sun, M. Danilevsky, J. Han, and J. Gao, "Graph regularized transductive classification on heterogeneous information networks," in *ECML PKDD'2010, Casa Convalescncia, Barcelona, Catalonia, Spain*, 2010, pp. 570–586.

[18] M. Ji, J. Han, and M. Danilevsky, "Ranking-based classification of heterogeneous information networks," in *KDD'11, San Diego, CA*, 2011, pp. 1298–1306.

[19] A. Strehl and J. Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *The Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2003.