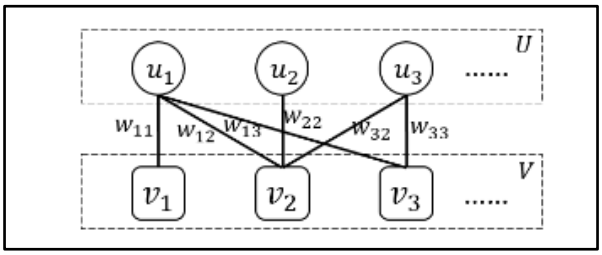
《BiNE: Bipartite Network Embedding》

《二部网络嵌入》

摘要：

1. **本文的背景：**在图神经网络早期研究中，大部分有效的方法（如 DeepWalk、LINE 等）都集中在同质网络的问题处理上，而对于像二部图这种特殊的异质网络，直接将这些方法应用在其上取得的效果并不理想。因此，本文的目的是找出一种框架将图嵌入学习更好地应用在二部图（bipartite network）上。
2. **本文的贡献：**提出了一种新的图神经网络框架——BiNE （Bipartite Network Embedding）来学习二部网络的节点嵌入表示。
3. **主要创新点：**设计了一种有偏的随机游走（random walk）来生成节点序列；并提出了一种同时考虑节点显式关系（explicit relations）和隐式关系（implicit relations）的联合优化框架来更好地学习节点的嵌入表示。
4. **实验结果：**在链接预测（link prediction）和推荐（recommendation）的应用实验中，BiNE 相比当时的 baseline 模型取得了更好的效果，这说明通过 BiNE 训练得到的节点嵌入表示可有效用于各种实际的应用任务。此外，其他一系列的定量和定性的分析也都证明了 BiNE 的有效性。

二部图的定义：



一个二部网络的结构如上图所示。对于一个二部图$G = (U, V, E)$，其中，U, V分别表示两种类型的节点的集合，而$E \subseteq U \times V$代表二部图内连接两类节点的边的集合。并且，每一条边都有一个非负的权重w_{ij}，用于表示两节点u_i与v_j关系的强弱。

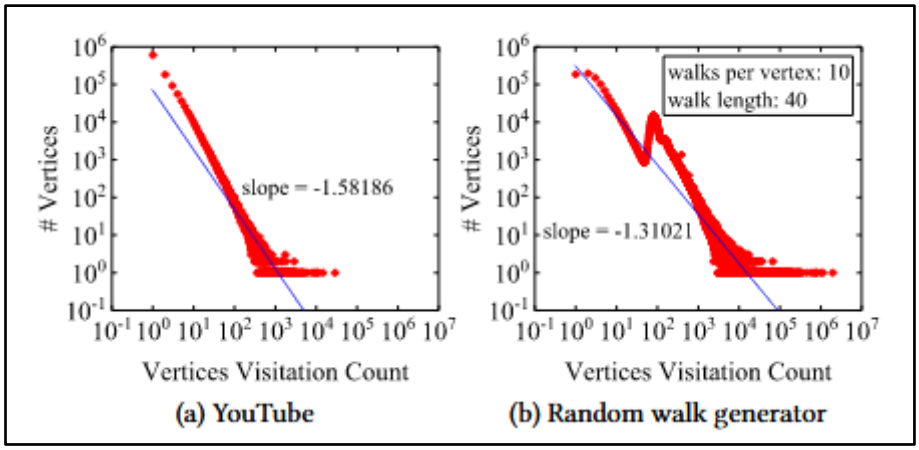
而二部图嵌入表示学习的目标是，给定一个二部图G及其边的权重$W \in \mathbb{R}^{|U|\times|V|}$，将网络中所有的节点（$U \cup V$）映射到一个低维的向量空间$\mathbb{R}^d$中，其中$d$表示每个节点嵌入向量的维度。

传统图嵌入方法的缺陷：（以 DeepWalk 为例）

二部图（bipartite graph）是一种现实中常见的数据结构，它被广泛应用于各种场景，如推荐系统、搜索引擎、问答系统等。而常见的图嵌入方法，如 DeepWalk，通常分为两步：1）通过随机游走得到顶点序列；2）使用词嵌入方法进行计算得到嵌入表示。

然而，将这些传统的方法应用在二部图上（一种特殊的异构图）它们的效果会是次优的，主要原因有以下两点：

1. **未考虑节点的类型信息**，虽然边只存在于不同类型的节点之间，但是同类型的节点之间也可能存在着某种重要的隐含关系。
2. **通过随机游走生成的序列并不能很好的保持二部图的特征**。因为二部图的分布通常遵循幂律分布，但是像 DeepWalk 等的随机游走的设计存在不足，难以反映二部图的这种分布情况。具体情况如下图所示：



可见，由于 DeepWalk 随机游走生成器的设计不当，生成的语料库（序列）没有显示出二部图所需的幂律分布。

而本文提出的 BiNE 有效解决了传统图嵌入方法上述的两点不足：

1. 对于网络中的隐式关系和显示关系，BiNE 分别设计了对应的目标函数，使得被忽略的隐式关系也能被显式的建模出来，并且实验证明这样做是有助于提高模型的表现成绩的。
2. 虽然 BiNE 同样采用了随机游走的策略，但与 DeepWalk 不同，它是一种有偏的随机游走策略。一是根据节点的重要性，来决定有多少序列以该节点为起点。二是生成的序列长度并不是完全相同的，而是设置了一个概率来控制游走的停止。

显式关系建模：

与$LINE$中的一阶相似性建模类似，$BiNE$通过考虑两个连接顶点之间的局部相似性来建模显式关系。所以，顶点u_i与v_j之间的联合概率被定义为：

$$P(i,j)=\frac{w_{ij}}{\sum_{e_{ij}\in E} w_{ij}}$$

而度量两个节点的嵌入表示在映射空间内的相近性则参考了$word2vec$取向量内积的思想，定义如下：

$$\hat{P}(i,j)=\frac{1}{1+\exp\left(-\vec{u}_i^{\,T}\vec{v}_j\right)}$$

其中，$\vec{u}_i \in \mathbb{R}^d$，$\vec{v}_j \in \mathbb{R}^d$分别代表节点$u_i$与$v_j$的嵌入向量。

最后，我们再通过KL散度来衡量上述两个分布的差异，以此定义显式关系的目标优化函数O_1，即最小化下式：

$$minimize\ \ O_1 = KL(P||\hat{P}) = \sum_{e_{ij}\in E} P(i,j)\log\left(\frac{P(i,j)}{\hat{P}(i,j)}\right)$$

$$\propto -\sum_{e_{ij}\in E} w_{ij}\log\hat{P}(i,j)$$

直观理解：对于两个相连紧密的节点，如果学习到的两节点的表示在低维的向量空间中也是彼此靠近的，则保持了所谓的局部相似性。

隐式关系建模：

隐式关系定义：

对于二部图，两个相同类型的节点，尽管它们之间不存在直接相连的边，但如果存在一条从u_i到u_j的路径，我们就认为这两个节点之间存在着某种隐式关系。**也就是说，本文所指的节点之间的隐式关系，是相对于两个同类型的节点来说的。**

构建节点序列的语料库：

为此，为了学习到二部图同类型节点上的隐式关系，我们需要把一个二部图网络拆分为两个同质网络，而定义两相同类型的节点的隐式关系（二阶相似度）的公式如下：

$$w_{ij}^U=\sum_{k\epsilon V} w_{ik}w_{jk};\quad w_{ij}^V=\sum_{k\epsilon U} w_{ki}w_{kj}$$

于是，我们就可以得到两个同质网络的权重矩阵W^U和W^V：

$$W^U = WW^T,\ \ W^V = W^TW$$

其中，$W^U \in \mathbb{R}^{|U|\times|U|}$，$W^V \in \mathbb{R}^{|V|\times|V|}$，即它们的维度大小分别为$|U| \times |U|$和$|V| \times |V|$。而$W \in \mathbb{R}^{|U|\times|V|}$代表原二部图的权重矩阵。

接着，我们就需要在这两个新的同质图上进行随机游走，以获取对应节点类型的序列来学习相关的隐式关系。但是，需要注意的是，由于$DeepWalk$的随机游走策略并不是最优的，因此本文重新设计了一种偏置的和自适应的随机游走发生器（biased and self-adaptive random walk generator）。具体方式如下：

1. 对于每一个节点，它的中心性（centrality）越强，则从它开始的随机游走序列越多，即随机游走的迭代次数越多。
2. 定义一个概率值p，使得随机游走在每一步都有可能立即停止，从而得到一系列长度不同的随机节点序列。

相关算法伪代码如下：

Algorithm 1: WalkGenerator(W, R, $maxT$, $minT$, p)

Input : weight matrix of the bipartite network **W**, vertex set R (can be U or V), maximal walks per vertex $maxT$, minimal walks per vertex $minT$, walk stopping probability p

Output: a set of vertex sequences D^R

```
Calculate vertices' centrality: H = CentralityMeasure(W);
Calculate W^R w.r.t. Equation (4);
foreach vertex v_i ∈ R do
    l = max(H(v_i) × maxT, minT);
    for i = 0 to l do
        D_{v_i} = BiasedRamdomWalk(W^R, v_i, p);
        Add D_{v_i} into D^R;
return D^R;
```

建模隐式关系：

在前文生成的两个语料库上分别使用$Skip-Gram$模型来学习节点的表示。其目标函数如下：

$$maximize\ O_2 = \prod_{u_i \in S \wedge S \in D^U} \prod_{u_c \in C_S(u_i)} P(u_c|u_i)$$

其中，S为语料库中的节点序列，$C_S(u_i)$为在该序列中节点u_i的上下文节点的

集合，$P(u_c|u_i)=\frac{\exp(\vec{u}_i^{\,T}\vec{\theta}_c)}{\sum_{k=1}^{|U|}\exp(\vec{u}_i^{\,T}\vec{\theta}_k)}$。可以看到，在计算$P(u_c|u_i)$时，公式采用了$softmax$，会非常花时间。所以，我们需要通过使用负采样的方法来进一步优化目标函数，从而得到最终的优化公式：

$$maximize\ O_2=\prod_{u_i\in S\wedge S\in D^U}\prod_{u_c\in C_S(u_i)}P(u_c,N_S^{ns}(u_i)|u_i)$$

其中，$N_S^{ns}(u_i)$代表节点u_i的负采样节点集合。

$$P(u_c,N_S^{ns}(u_i)|u_i)=\prod_{z\in\{u_c\}\cup N_S^{ns}(u_i)}P(z|u_i)$$

而$P(z|u_i)$为：

$$P(z|u_i)=\begin{cases}\sigma(\vec{u}_i^{\,T}\vec{\theta}_z), & if\ z\ is\ acontext\ of\ u_i\\ 1-(\vec{u}_i^{\,T}\vec{\theta}_z), & z\in N_S^{ns}(u_i)\end{cases}$$

其中，σ代表激活函数，$\vec{u}_i$代表节点u_i的嵌入向量，$\vec{\theta}_z$代表节点z的上下文（二阶）嵌入向量。

同理，节点类型为V的目标函数也是如此：

$$maximize\ O_3=\prod_{v_j\in S\wedge S\in D^V}\prod_{v_c\in C_S(v_j)}P(v_c,N_S^{ns}(v_j)|v_j)$$

其中，$N_S^{ns}(v_j)$代表节点v_j的负采样节点集合。

$$P(v_c,N_S^{ns}(v_j)|v_j)=\prod_{z\in\{v_c\}\cup N_S^{ns}(v_j)}P(z|v_j)$$

而$P(z|v_j)$为：

$$P(z|v_j)=\begin{cases}\sigma(\vec{v}_j^{\,T}\vec{\theta}_z), & if\ z\ is\ acontext\ of\ v_j\\ 1-(\vec{v}_j^{\,T}\vec{\theta}_z), & z\in N_S^{ns}(v_j)\end{cases}$$

联合优化：

根据显式关系和隐式关系建模得到的目标函数，我们可以得到最终的目标函数公式为：

$$maximize\ L=\alpha O_2+\beta O_3-\gamma O_1$$

使用 Stochastic Gradient Ascent algorithm (SGA)来优化这一函数。但是，由于优化这个函数的不同部分需要使用不同的训练实例，因此我们必须分步进行优化操作。

首先，是显式关系的优化，即L中的O_1部分，更新节点表示的公式如下所示：

$$\vec{u}_i = \vec{u}_i + \lambda\{\gamma w_{ij}[1 - \sigma(\vec{u}_i{}^T \vec{v}_j)] \cdot \vec{v}_j\}$$

$$\vec{v}_j = \vec{v}_j + \lambda\{\gamma w_{ij}[1 - \sigma(\vec{u}_i{}^T \vec{v}_j)] \cdot \vec{u}_i\}$$

接着，是隐式关系的优化，公式如下：

$$\vec{u}_i = \vec{u}_i + \lambda\{\prod_{z \in \{u_c\} \cup N_S^{ns}(u_i)} \alpha\left[I(z, u_i) - \sigma\left(\vec{u}_i{}^T \vec{\theta}_z\right)\right] \cdot \vec{\theta}_z\}$$

$$\vec{v}_j = \vec{v}_j + \lambda\{\prod_{z \in \{v_c\} \cup N_S^{ns}(v_j)} \beta\left[I\left(z, v_j\right) - \sigma\left(\vec{v}_j{}^T \vec{\theta}_z\right)\right] \cdot \vec{\theta}_z\}$$

其中，$I(z, u_i)$用于判断节点z是否为节点u_i的上下文节点，同理$I\left(z, v_j\right)$。最后，上下文向量的更新公式如下：

$$\vec{\theta}_z = \vec{\theta}_z + \lambda\{\alpha\left[I(z, u_i) - \sigma\left(\vec{u}_i{}^T \vec{\theta}_z\right)\right] \cdot \vec{u}_i\}$$

$$\vec{\theta}_z = \vec{\theta}_z + \lambda\{\beta\left[I\left(z, v_j\right) - \sigma\left(\vec{v}_j{}^T \vec{\theta}_z\right)\right] \cdot \vec{v}_j\}$$

BiNE 模型的算法伪代码如下：

Algorithm 2: Training algorithm of BiNE

Input : bipartite network $G = (U, V, E)$, weight matrix of the bipartite network $\mathbf{W}$, window size ws, number of negative samples ns, embedding size d, maximal walks per vertex $maxT$, minimal walks per vertex $minT$, walk stopping probability p

Output: vertex embeding matrices $\mathbf{U}$ and $\mathbf{V}$

Initialize embedding vectors $\vec{\mathbf{u}}_i$ and $\vec{\mathbf{v}}_j$;
Initialize context vectors $\vec{\boldsymbol{\theta}}_i$ and $\vec{\boldsymbol{\theta}}_j$;
D^U = WalkGenerator($\mathbf{W}, U, maxT, minT, p$);
D^V = WalkGenerator($\mathbf{W}, V, maxT, minT, p$);
foreach *edge* $(u_i, v_j) \in E$ **do**
    Update $\vec{\mathbf{u}}_i$ and $\vec{\mathbf{v}}_j$ using Equations (10) and (11);
    foreach (u_i, u_c) *in the sequence* $S \in D^U$ **do**
        Negative sampling to generate $N_S^{ns}(u_i)$;
        Update $\vec{\mathbf{u}}_i$ using Equation (12);
        Update $\vec{\boldsymbol{\theta}}_z$ using Equation (14) where $z \in \{u_c\} \cup N_S^{ns}(u_i)$;
    foreach (v_j, v_c) *in the sequence* $S \in D^V$ **do**
        Negative sampling to generate $N_S^{ns}(v_j)$;
        Update $\vec{\mathbf{v}}_j$ using Equation (13);
        Update $\vec{\boldsymbol{\theta}}_z$ using Equation (15) where $z \in \{v_c\} \cup N_S^{ns}(v_j)$;
return Vertex embeding matrices $\mathbf{U}$ and $\mathbf{V}$

实验及结果：

本文进行了两类应用实验：链接预测和推荐，并尝试探究以下四个问题：

1. 和$baseline$方法与其他新型$embedding$方法相比，$BiNE$的实际表现到底如何？
2. 明确隐式关系建模是否真的有助于学习节点的表示？
3. 本文设计的有偏随机游走策略是否真的有效？
4. 超参数的设置如何影响$BiNE$模型的性能？（参数敏感度分析）

RQ1：

对于链接预测任务，本文使用了来自 Wikepedia 和 Tencent 的二部图数据集，分别表示了网页—作者和 QQlive 上用户—电影的关系信息。数据集情况如下：

Table 1: Statistics of bipartite networks and metrics adopted in experimets for different tasks.

Task	Link Prediction		Recommendation		
Type	undirected, unweighted		undirected, weighted		
Metric	AUC-ROC,AUC-PR		F1, NDCG, MAP, MRR		
Name	Tencent	Wikipedia	VisualizeUs	DBLP	MovieLens
$\|U\|$	14,259	15,000	6,000	6,001	69,878
$\|V\|$	1,149	3,214	3,355	1,308	10,677
$\|E\|$	196,290	172,426	35,639	29,256	10,000,054
Density	1.2%	0.4%	0.2%	0.4%	1.3%

而与$baseline$方法的对比结果如下：

Table 3: Link prediction performance on Tencent and Wikipedia.

Algorithm	Tencent		Wikipedia	
	AUC-ROC	AUC-PR	AUC-ROC	AUC-PR
CN	50.63%	65.66%	86.85%	90.68%
JC	51.49%	66.18%	63.90%	73.04%
AA	50.63%	65.66%	87.37%	91.12%
AL	50.44%	65.70%	90.28%	91.81%
Katz	50.90%	65.06%	90.84%	92.42%
PA	55.60%	68.99%	90.71%	93.37%
DeepWalk	57.62%	71.32%	89.71%	91.20%
LINE	59.68%	73.48%	91.62%	93.28%
Node2vec	59.28%	72.62%	89.93%	91.23%
Metapath2vec++	60.70%	73.69%	89.56%	91.72%
BiNE	60.98%**	73.77%**	92.91%**	94.45%**

** indicates that the improvements are statistically significant for $p < 0.01$ judged by paired t-test.

可见在两类数据集的链接预测任务上，BiNE 模型都取得了最佳地效果。

而对于推荐任务，实验中使用了 DBLP，Movielens 和 VisualizeUs 这三个常见的数据集，实验结果如下：

Table 4: Performance comparison of Top-10 Recommendation on VisualizeUs, DBLP, and MovieLens.

Algorithm	VisualizeUs				DBLP				Movielens			
	F1@10	NDCG@10	MAP@10	MRR@10	F1@10	NDCG@10	MAP@10	MRR@10	F1@10	NDCG@10	MAP@10	MRR@10
BPR	6.22%	9.52%	5.51%	13.71%	8.95%	18.38%	13.55%	22.25%	8.03%	7.58%	2.23%	40.81%
RankALS	2.72%	3.29%	1.50%	3.81%	7.62%	11.50%	7.52%	14.87%	8.48%	7.95%	2.66%	38.93%
FISMauc	10.25%	15.46%	8.86%	16.67%	9.81%	13.77%	7.38%	14.51%	6.77%	6.13%	1.63%	34.04%
DeepWalk	5.82%	8.83%	4.28%	12.12%	8.50%	24.14%	19.71%	31.53%	3.73%	3.21%	0.90%	15.40%
LINE	9.62%	13.76%	7.81%	14.99%	8.99%	14.41%	9.62%	17.13%	6.91%	6.50%	1.74%	38.12%
Node2vec	6.73%	9.71%	6.25%	13.95%	8.54%	23.89%	19.44%	31.11%	4.16%	3.68%	1.05%	18.33%
Metapath2vec++	5.92%	8.96%	5.35%	13.54%	8.65%	25.14%	19.06%	31.97%	4.65%	4.39%	1.91%	16.60%
BiNE	13.63%**	24.50%**	16.46%**	34.23%**	11.37%**	26.19%**	20.47%**	33.36%**	9.14%**	9.02%**	3.01%**	45.95%**

** indicates that the improvements are statistically significant for $p < 0.01$ judged by paired t-test.

可见在三类数据集的推荐任务上，BiNE 模型都取得了最佳地效果。

RQ2：

Table 5: BiNE with and without implicit relations.

	Without Implicit Relations		With Implicit Relations	
Link Prediction				
Dataset	**AUC-ROC**	**AUC-PR**	**AUC-ROC**	**AUC-PR**
Tencent	59.78%	73.05%	60.98%**	73.77%**
WikiPedia	91.47%	93.73%	92.91%**	94.45%**
Recommendation				
Dataset	**MAP@10**	**MRR@10**	**MAP@10**	**MRR@10**
VisualizeUS	7.91%	15.65%	16.46%**	34.23%**
DBLP	20.20%	32.95%	20.47%**	33.36%**
MovieLens	2.86%	43.98%	3.01%**	45.95%**

** indicates that the improvements are statistically significant for $p < 0.01$ judged by paired t-test.

可以看到，隐式关系建模确实能帮助模型提高节点嵌入表示的性能。

RQ3：

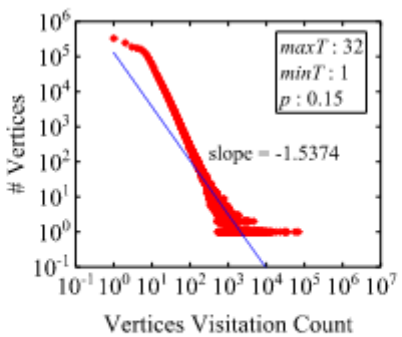


Figure 3: Distribution of vertices in the biased and self-adpative random walk generator.

Table 6: BiNE with different random walk generators.

	Uniform Random Walk Generator		Biased and Self-adaptive Random Walk Generator	
Link Prediction				
Dataset	**AUC-ROC**	**AUC-PR**	**AUC-ROC**	**AUC-PR**
Tencent	59.75%	73.06%	**60.98%****	**73.77%****
WikiPedia	88.77%	91.91%	**92.91%****	**94.45%****
Recommendation				
Dataset	**MAP@10**	**MRR@10**	**MAP@10**	**MRR@10**
VisualizeUS	15.93%	33.66%	**16.46%****	**34.23%****
DBLP	11.79%	23.41%	**20.47%****	**33.66%****
MovieLens	2.91%	46.12%	**3.04%****	**46.20%****

** indicates that the improvements are statistically significant for $p < 0.01$ judged by paired t-test.

可以看到，本文设计的有偏随机游走策略的确能够提高模型的嵌入学习能力。

RQ4：

Table 2: The search range and optimal setting (highlighted in red) of hyper-parameters for our BiNE method.

Parameter	Meaning	Test values
ns	number of negative samples	[1, 2, **4**, 6, 8, 10]
ws	size of window	[1, 3, **5**, 7, 9]
p	walk stopping probability	[0.05, 0.1, **0.15**, 0.2, 0.3, 0.4, 0.5]
β	trade-off parameter	[0.0001, 0.001, **0.01**, 0.1, 1]
γ	trade-off parameter	[0.01, 0.05, **0.1**, 0.5, **1**, 5]

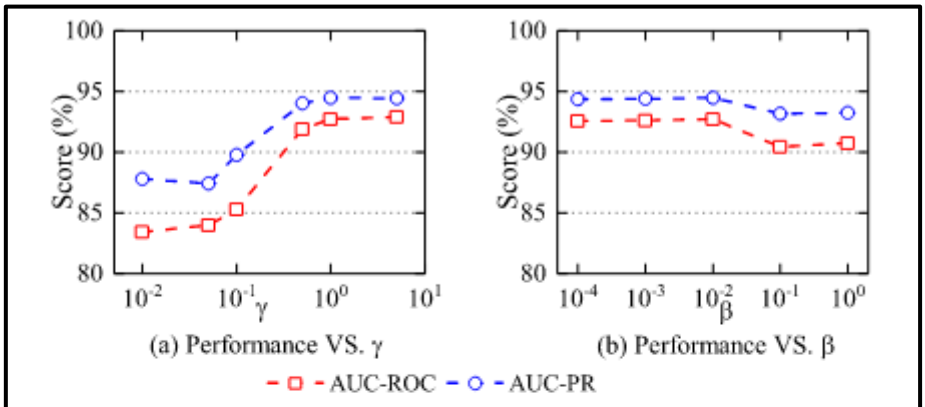


Figure 4: Impact of hyper-parameters on link prediction.

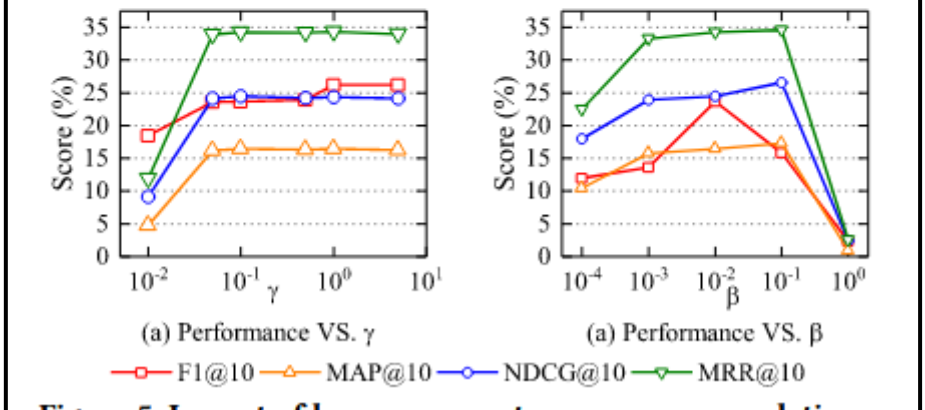
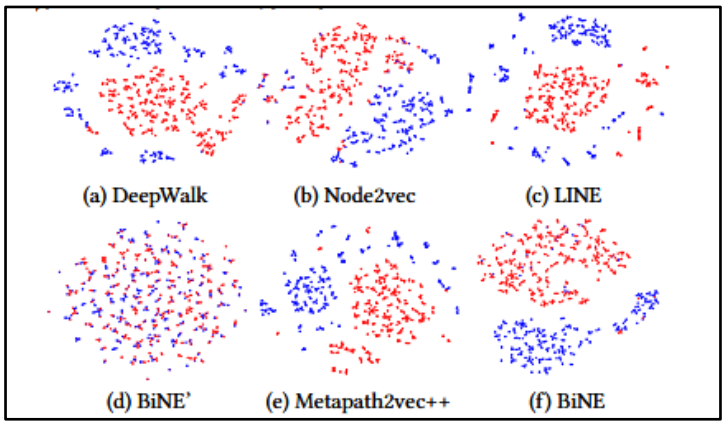


Figure 5: Impact of hyper-parameters on recommendation.

不同超参数的最佳取值以及超参数γ和β取值对模型性能的影响。

不同方法得到的嵌入向量可视化结果：



总结：

1. 相比于传统的同质图方法，$BiNE$能够有效地利用二部图（异质图）的特点，生成不同类型节点的 embedding，且效果更好；
2. 异质图的 SOTA 方法$Metapath++$认为显式关系和隐式关系的作用相等，并且忽略了图上节点之间的权重信息；而$BiNE$综合考虑了显式关系和隐式关系的作用，并且很好地利用了权重信息。
3. 本文提出的有偏随机游走策略，每个节点采样的游走序列数与节点的重要性相关，并且序列长度不固定，更符合图的原始分布。

对本文的感悟：

在我看来，本文最精妙地创新点在于将二部图转化为两个同构图的大胆创新，使得在异构图上的随机游走能够间接等价于在同构图上的随机游走。而这种对于隐式关系的定义无疑是创新且大胆的，这也是我们在研究中值得学习和借鉴的地方。