HID: Hierarchical Multiscale Representation Learning for Information Diffusion

摘要

多尺度建模在各种机器学习任务中取得了巨大成功。然而， 信息传播这一突出任务并没有得到适当的探索， 该任务旨在了解信息如何在在线社交网络中沿着用户传播。对于特定用户来说， 是否以及何时采用从另一个用户传播的信息会受到复杂交互的影响， 因此建模非常困难。当前最先进的技术调用具有用户矢量表示的深度神经模型。在本文中通过集成用户表示学习和多尺度建模， 提出了一个层次信息扩散（HID）框架。所提出的框架可以分层在利用用户表示的所有信息扩散技术之上， 以提高原始技术的预测能力和学习效率。在三个真实世界数据集上进行的大量实验展示了本文方法的优越性。

关键词： 信息级联；机器学习；社交网络；

1 引言

本文的工作受到用户在信息传播过程中角色重要性的推动。为了捕捉用户的扩散行为，以前的工作要么设计复杂的自定义模型， 要么依赖于外部知识， 如友谊网络和用户档案， 由于隐私政策， 这些知识往往不太有组织、匿名或无法访问。因此， 我们试图利用用户扩散行为本身的痕迹（即扩散路径），从中提取用户行为的多个方面。来自现实世界的数据可以自然地在多个尺度上进行编码， 这可以作为深度学习中特征表示的丰富可靠的资源。迫切需要深层模型来利用这些隐含的多重尺度。然而， 对信息扩散中多尺度建模的探索在很大程度上被忽视了。在此背景下， 本文探索了用户表示学习框架中的多尺度建模。挑战主要有三个方面：（1）如何发现和归纳多个量表， 同时在量表隐含的情况下保持原始数据中的个体行为模式；（2）如何以高效有效的方式在多个尺度之间传递知识； 以及（3）如何设计适用于大多数最先进的扩散模型的可推广策略。为了解决上述问题， 本文提出了层次信息扩散（HID）框架。该方法有几个吸引人的特性， 最值得注意的是： HID 有助于学习用户表示， 在扩散路径上有多个规模的摘要。这是通过对用户进行分组， 然后在每个规模上粗化扩散路径来实现的。提出了一个 k 阶扩散邻近矩阵来支持这一点。更少的用户和粗化规模的活动也带来了效率。受人工神经网络研究的启发， 缩小规模实现了跨规模的用户知识交流揭示了神经元对模型性能的重要性与训练后其权重分布与初始化不同的程度相关将放大和缩小相结合产生了一个分层学习框架，该框架有可能将非活跃用户与其他用户分组，因此与不进行多重放大的基于采样的优化相比，非活跃用户可以获得更多的优化周期。本文提出用于在线网络中信息传播的 HID，通过大量实验， 在 3 个真实数据集上展示了多尺度学习的优势。该方法产生了更高的预测能力和学习效率，并实现了高达 14.76

2 相关工作

具有用户表示的信息扩散： [[4]，[3]] 利用表示学习来预测看不见的扩散。 [[2]] 通过一种 新颖的路径采样程序来学习用户表示， 以收集每个用户的上下文。 [[7]] 学习网络规则化的基 于角色的用户表示。 [[6]] 学习用户嵌入， 同时利用社区结构规范信息传播过程。 [[1]] 采用了 注意力机制和具有用户嵌入层的卷积网络。 [[7]] 采用具有用户嵌入层的递归网络， 通过结构 注意力来探索扩散路径和图。多尺度表示学习: 用于信息扩散的多尺度表示学习尚未被探索。 [[1]] 联合处理微观和宏观预测， [[5]] 采用层次关注机制。这两种型号都可以从 HID 中受益。 此外， [[6]] 捕捉了分层社区级的信息扩散， 但作为一个概率模型。相比之下， 图的多尺度表 示学习最近受到了极大的关注。

3 本文方法

3.1 问题分析

在线社交网络（OSN）可以被描述为允许其用户共享信息的平台[Guille等人，2013]。形式上，OSN所拥有的用户集表示为V。语料库D是一组扩散路径，通常从OSN中提取，并由其|V|用户在信息共享过程中生成。我们的预测任务是信息传播。

3.1.1 问题定义

扩散路径：共享某个信息单元的非重复用户序列，表示有序的用户路径，捕获OSN的用户采用该信息的顺序。

用户表示：我们将用户表示定义为映射Φ：v∈V→R_d，它将每个用户映射到欧几里得空间（d《|V|）中的d维向量。在实践中，Φ是根据训练语料库估计的|V|×d矩阵。

信息扩散：给定|V|个用户和一个语料库D_{test}，该任务旨在与V中的用户一起完成语料库D_{test}中的扩散路径。D_{test}中每个扩散路径的第一个用户（即源用户）是已知的。

3.1.2 问题辨析

本文希望学习Φ，一种潜在的用户表示，以更好地预测信息传播。目前估计Φ的方法有两个主要缺点：（1）没有考虑多尺度特性，（2）由于初始配置麻烦，它们的随机优化很容易陷入局部极小值。针对这些不足，我们引入了分层用户表示学习来建模OSN中的信息扩散：给定粗尺度的数量s，将语料库D_{train}简化为一系列连续的粗语料库，D_1，D_2，D_s，具有相应的用户集V_1、V_2、V_s（|V|>|V_1|>・・・>|V_s|）。学习相应的用户表示族Φ_1，$\Phi_2 \ldots \Phi_s$然后获得最细粒度的用户表示Φ来预测D_{test}中的扩散路径。

3.2 HID 算法

3.2.1 HID 算法概述

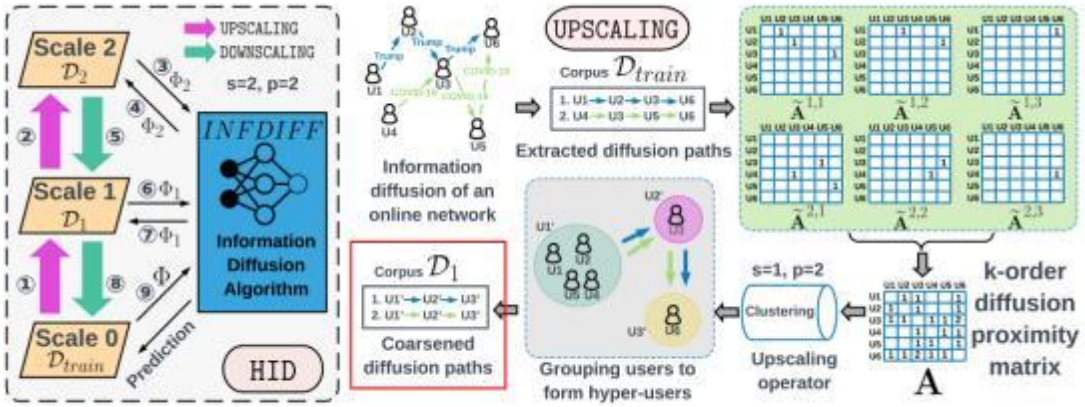


Figure 1: Methodology overview. *Left*: Framework overview of HID with a toy example. Numbers next to the arrow indicate the order of the algorithm workflow. *Right*: The UPSCALING procedure for generating multiple scales of information diffusion data corpus.

图 1. 方法概述。左图： HID 的框架概述和一个玩具示例。箭头旁边的数字表示算法工作流的 顺序。右图：生成多尺度信息扩散数据语料库的 UPSCALING 过程

图 2 算法 1 概述了整个 HID 方法。该方法可以应用于任何利用用户表示的信息传播技 术。

Algorithm 1 $\texttt{HID}(s, p, d, \mathcal{D}_{\text{train}}, \mathcal{F})$

Input:
the number of coarse scales s
the coarsening rate between adjacent two scales p
the user embedding dimensionality d
an arbitrary information diffusion algorithm that leverages user representations $\mathcal{F}$
a corpus $\mathcal{D}_{\text{train}}$ with user set V
Output: latent representation of users $\Phi : v \in \mathrm{V} \mapsto \mathbb{R}^d$

$\mathcal{D}_0 \leftarrow \mathcal{D}_{\text{train}}$
$\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_s \leftarrow \texttt{UPSCALING}(s, p, \mathcal{D}_0)$
Initialize Φ_s
$\Phi_s \leftarrow \mathit{INFDIFF}(\Phi_s, \mathcal{D}_s, d, \mathcal{F})$
for $i = s - 1$ to 0 **do**
    $\Phi_i \leftarrow \texttt{DOWNSCALING}(\Phi_{i+1}, \mathcal{D}_i, \mathcal{D}_{i+1})$
    $\Phi_i \leftarrow \mathit{INFDIFF}(\Phi_i, \mathcal{D}_i, d, \mathcal{F})$
end for
$\Phi \leftarrow \Phi_0$
return Φ

图 2. HID 算法伪码

算法 1 将可以从 HID 中受益的信息扩散算法表示为 INFDIFF。INFDIFF 过程将学习用 户表示并预测信息扩散。

粗尺度的数量 s 表示得到的粗化语料库的数量， 相邻两个尺度之间的粗化率用户计算尺 度 i 的用户数量除以 p。

UPSCALING：该方法对应图 1 中 UPSCALING 过程， UPSCALING 过程实现了对用户 的连续抽象。抽象是这样做的， 即较粗的规模将有较少的用户， 但扩散路径的关键特征大多 被保留了下来。具体地，扩散路径的变化被最小化，并且用户排序被保留。

DOWSCALING ：DOWNSCALING 过程将尺度 i+1 的较粗用户表示映射到尺度 i 的随 后的精细训练的用户表示， 该尺度 i 用作学习更细粒度用户表示的初始化点。具体而言， 对 于尺度 i 处的每个用户， DOWNSCALING 在尺度 i+1 处找到用户对应的超级用户， 然后将 学习到的超级用户的表示（由 INFDIFF 返回） 分配为该用户的初始表示。向下缩放跨尺度传 递用户表示的知识， 从而为学习提供良好的初始化， 有效地避免了在非凸优化中更容易出现 局部极小值的麻烦配置。

3.2.2 k 阶扩散邻近矩阵 A

流行的图嵌入方法是隐式分解包含A^1、A^2、…A^k项的矩阵。其中k是随机游动上的窗口大小，条目A_{ij}^k是长度为k的节点i和j之间的路径数。我们为每个扩散路径定义矩阵$\tilde{A}m,k$，是一个|V|×|V|矩阵，表示步长为k的扩散路径m的邻接矩阵。条目$\tilde{A}_{ij}m,k$是具有步骤k的用户i和用户j的有序对出现在扩散路径m中的次数，因此只能是1或0。

在本文中，多尺度属性具有以下功能:(1)捕获两个不同用户之间的定向、本地和远程信息采用接近性;(2)考虑不同过渡顺序和扩散路径下信息采用模式的不同连接。为了捕捉语料库D_{train}的多尺度特性，定义了以下k阶扩散邻近矩阵:

$$\mathbf{A}=\sum_{m=1}^{|\mathcal{D}_{\text{train}}|}\sum_{k=1}^{l}\tilde{\mathbf{A}}^{m,k}+\left(\sum_{m=1}^{|\mathcal{D}_{\text{train}}|}\sum_{k=1}^{l}\tilde{\mathbf{A}}^{m,k}\right)^{\mathrm{T}},\quad(1)$$

图 3

其中$|D_{train}|$是D_{train}中的扩散路径的总数，l是D_{train}中的最大可能步长（即，最长扩散路径的长度减去1）。计算A的步骤称为获得近似。A说明了用户沿着扩散路径的双向共现模式，并且可以被分解或转换，其中子组件具有真正的实际解释，例如：

$$\mathbf{A_n}=\sum_{m=1}^{|\mathcal{D}_{\text{train}}|}\sum_{k=1}^{\tau}\tilde{\mathbf{A}}^{m,k},\quad(2)$$

图 4

其中τ是预定义的邻域阈值（τ<1），因此A_n根据用户的相似信息采用时间来考虑相邻模式。进一步的：

$$\mathbf{A_o} = \sum_{m=1}^{|\mathcal{D}_{\text{train}}|} \left[\mathbf{i}_{e_m} \cdot \sum_{k=1}^{l} \widetilde{\mathbf{A}}_{e_m}^{m,k} \right], \tag{3}$$

图 5

其中e_m是扩散路径m的Φ中的源用户索引，i_{e_m}是|V|维向量，它用作指示函数（在第e_m个条目中有1，在其他地方有0）。以这种方式，A_o考虑用户是否会传播源自另一用户的信息的模式。

3.2.3 UPSACLING 算法过程

图 6 展示了 UPSCALING 算法的伪代码， 描述了对用户的连续抽象的过程， 不断对用户 规模和传播路径进行粗粒化的一个过程。

Algorithm 2 UPSCALING$(s, p, \mathcal{D}_{\text{train}})$

Input:
the number of coarse scales s
the coarsening rate between adjacent two scales p
a corpus $\mathcal{D}_{\text{train}}$ with user set V
Output: a series of successively coarser corpora, $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_s$

$\mathcal{D}_0 \leftarrow \mathcal{D}_{\text{train}}$
for $i = 0$ to $s - 1$ **do**
  $\mathbf{A}_i \leftarrow ObtainProximity(\mathcal{D}_i)$
  $\mathrm{V}_{i+1} \leftarrow UpscalingOperator(\mathbf{A}_i, \mathrm{V}_i, p)$
  $\mathcal{D}_{i+1} \leftarrow RewriteCorpus(\mathcal{D}_i, \mathrm{V}_i, \mathrm{V}_{i+1})$
end for
return $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_s$

图 6. UPSCALING 算法伪码

其中 ObtainProximity: 对当前阶段获取的语料库计算所得到的 K 阶扩散矩阵 A 的一个 过程。UpscalingOperator: 通过集群将用户分组以形成超级用户。UPSCALING 以自下而上的 方式在所有规模上运行， 在每个规模上， 具有相似采用模式的用户将形成超级用户。基于尺 度 i 上的 A，我们应用 U pscalingOperator 来形成尺度 i+1 上的超用户。可能的升级算子包 括层次聚集聚类（HAC）、谱聚类、 K-means 等。尺度

i+1 的用户数量由 p 定义， 即相邻两 个尺度之间的粗化率， 并计算为尺度 i 的用户数量除以 p。RewriteCorpus: 然后， 重写语料库 过程更新语料库， 并通过用尺度 i+1 的相应超用户 ID 替换尺度 i 的用户 ID 来获得， 同时确 保所得到的扩散路径仍然有效。由于我们希望确保扩散路径的变化最小化， 并且用户排序是 保留的， 我们只考虑每个超级用户的第一次出现。此外， 如果粗化扩散路径上的用户数量小 于 3，我们将丢弃该扩散路径，因为许多扩散模型需要在扩散路径上至少有 3 个用户。

4 实验结果

4.1 实验设置

4.1.1 实验对比基线

本文将 HID 应用于当前涉及用户表示学习的基于深度学习的信息扩散模型。对于每种情 况， 比较了应用 HID 和不应用 HID 时的准确性和效率变化。此外， HARP 和 Walklets 是多 尺度图表示技术，用于比较多尺度表示学习方面。用与作为基线对比的其他算法：

CDK：诱导用户嵌入， 使得较早被污染的用户比较晚被污染或未被污染的使用者更接近 源用户。

CSDK：生成用户嵌入， 使得首先被污染的用户在应用信息嵌入作为偏移后更接近源用 户，而不是后来被污染或根本没有被污染的。

Forest：是一个基于强化学习（RL）的深度扩散模型。 RL 将扩散大小信息合并到递归神 经网络模型中。

HARP：是一种通过在嵌入图之前压缩输入图来学习图的节点嵌入的方法。HARP 是一 种元策略，已被证明可以改进最先进的嵌入图算法。

Walklets：是一种通过对短随机游动进行二次采样（即“跳过”步骤） 来学习图中节点的 多尺度表示的方法。

4.1.2 实验数据集

Dataset	Memetracker	Twitter	Digg
Number of Users	994	1,222	500
Number of Links	32,652	166,889	486,354
Number of Diffusion Paths	4,319	9,761	3,553
Avg. Diffusion Path Length	8.56	18.10	137.89

图 7. 数据集示例

4.1.3 实验评价指标

本文对于每个数据集，扩散路径集随机分为两部分：80%用于训练和验证（D_{train}），其余用于测试（D_{test}）。我们通过广泛采用的指标平均精度$(MAP)^2$评估性能，计算如下：

$$\mathrm{MAP} = \frac{1}{|\mathcal{D}_{\mathrm{test}}|} \sum_{t \in \mathcal{D}_{\mathrm{test}}} \frac{\sum_{k=1}^{|t|} P@k \times \mathrm{isInfected}(k)}{|t|}, \quad (4)$$

图 8

其中t是D_{test}中的扩散路径，P@k是秩k处的精度，即前k个用户中受感染用户的百分比。当第k个用户真正参与该扩散路径时，isInfected（k）为1，否则为0。

4.2 实验结果

Algo	Memetracker		Twitter		Digg	
	MAP	Gain	MAP	Gain	MAP	Gain
CDK	*0.1852*	N/A	*0.5763*	N/A	*0.1397*	N/A
HID_H	0.1965*	**6.1**	0.5971*	**3.61**	0.1603*	**14.75**
HID_K	0.1899*	**2.54**	0.5885*	**2.12**	0.1455*	**4.15**
HID_S	0.1855	**0.16**	0.5773	**0.17**	0.1451*	**3.87**
CSDK	*0.2074*	N/A	*0.5712*	N/A	*0.1492*	N/A
HID_H	0.2164*	**4.34**	0.5844*	**2.31**	0.1541*	**3.28**
HID_K	0.2098*	**1.16**	0.5804*	**1.61**	0.1516*	**1.61**
HID_S	0.2082	**0.39**	0.5853*	**2.47**	0.1501	**0.6**
Forest	*0.3244*	N/A	*0.5915*	N/A	*0.1500*	N/A
HID_H	0.3397*	**4.72**	0.6019*	**1.76**	0.1689*	**12.6**
HID_K	0.3376*	**4.07**	0.6043*	**2.16**	0.1583*	**5.53**
HID_S	0.3296*	**1.6**	0.6029*	**1.93**	0.1564*	**4.27**
HARP	*0.1704*	**99.35***	*0.5745*	**5.19***	*0.1376*	**22.75***
Walklets	*0.1581*	**114.86***	*0.5744*	**5.21***	*0.1208*	**39.82***

图 9. HID 与其他对比基线的 MAP 精度对比（HID 的下标显示放大运算符， 其中“H”、“K” 和“S”代表“HAC”、“K-means”和“Spectral”聚类。）

图 9 显示了HID对比基线方法的MAP精度提升比例。HID始终比所有比较方法产生更好的结果。在Memetracker上，HID相对于CDK、CSDK和Forest的相对收益分别为6.1%、4.34%和4.72%。在Twitter上，基线往往表现得很好，MAP得分很高（例如，CDK的MAP得分为0.5763）。尽管如此，HID仍能带来性能提升。在Twitter上，HID（CDK）、HID（CSDK）和HID（Forest）引入的改进分别为3.61%、2.47%和2.16%对于Digg，Digg的收益尤其显著：HID（CDK）、HID（CSDK）和HID（Forest）分别比基线高14.75%、

3.28%和12.6%。HID（CSDK）在Digg上的适度增益可能源于对信息表示的额外学习，这在CSDK的扩散建模过程中起着主导作用。在图9中，还展示了不同放大算子的结果。在大多数情况下，HAC产生的结果最好，偶尔也明显优于K-means和Spectral Clustering（例如，在Digg上）。这可能是因为HID以符合HAC的分层方式运行。频谱聚类往往给出最低的性能增益。光谱聚类将数据点视为图的顶点，并连接足够近的顶点。没有良好分离的连接组件的嘈杂的在线数据可能是频谱聚类不太合适的原因。在图9中，还将HID与HARP和Walklets进行了比较。HARP和Walklets是多尺度图表示学习技术。在两者之间，HARP给出了更好的结果。然而，由于受到封闭世界假设的影响，他们都无法有效地解决信息传播的预测任务。在这3个数据集上，HARP和Walklets给出的结果比HID更差。

4.3 参数敏感度

HID 包括两个参数： 粗尺度的数量 s 和粗化率 p。使用 CDK 作为 HAC 的基线，我们研究了 s 和 p 的不同选择如何影响原始语料库的粗化程度和 HID 性能。

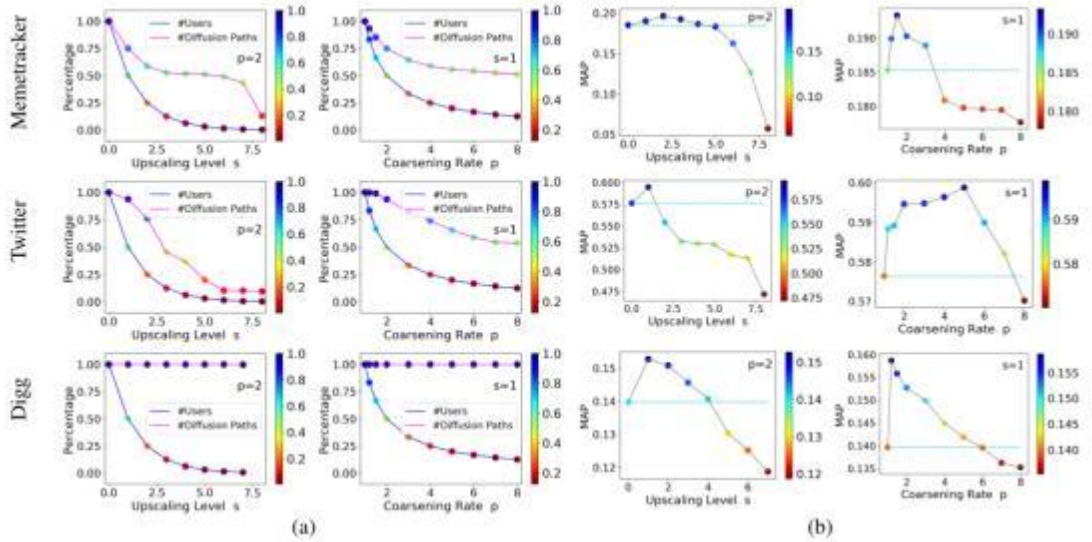


图 10. 参数 s 和 P 对 HID 算法的影响图

图 10 参数敏感性分析显示了对原始语料库（a）和HID性能（b）的放大效应。青色的水平虚线显示了通过基线CDK（即，在应用HID之前）实现的MAP得分。

第一列报告了对用户和扩散路径的相对总数的影响，即粗化尺度的用户/扩散路径的数量与原始语料库的用户/传播路径的数量之比，当p＝2时，s的值不同；并且第二列示出了当s＝1时相同但具有变化的p值。前三个量表对语料库的影响最大。当p=2时，只有不到10%的用户在4次缩放后仍保留，当p大于5且s=1时，只有大约20%的用户保留，这两种情况都表明了可能遭受性能损失的临界点。粗化程度因数据集而异，这意味着HID带来改进的难度。在这3个数据集中，s和p往往对Twitter的影响最大.

本文将测试集上的HID性能测量为图中s和p的函数。第2（b）段。关于s，HID的性能首先随着s的增加而提高，然后随着s的不断增加而降低。当粗化适当时，性能有望提高。在HID中，粗化对用户进行操作。粗尺度学习精细尺度的初始用户表示，因此适当程度的s采用适当的粗化策略可以确保用户行为的特征在粗化的尺度上得到保留。当s变得太大时，保留在最粗略尺度中的用户数量和扩散路径将太小，无法作为原始语料库的摘要。这就是为什么之后性能会下降的原因，这并不奇怪。类似地，可以观察到p有一个可接受的值范

围，在此之后p变得太大并破坏了对信息扩散预测至关重要的原始模式。这两个参数对性能的影响都相当大。最佳s（或p）的范围可以由图10（a）所示。

4.4 可扩展性

在图 11 中， 比较了 HID 和其他技术的运行时间。HID 与基线相比， HID 总是更高效， 并 且随着 s 的增加而变得更高效。

这是因为在粗化的尺度上参与的用户和活动较少。运行时间相对于剩余的语料库大小几 乎线性地减小。当 s>1 时， 有时会观察到效率和性能之间的权衡（例如， Twitter）。尽管如 此， HID 由于其巨大的可扩展性，对于大规模在线网络来说可能是一个有价值的工具。

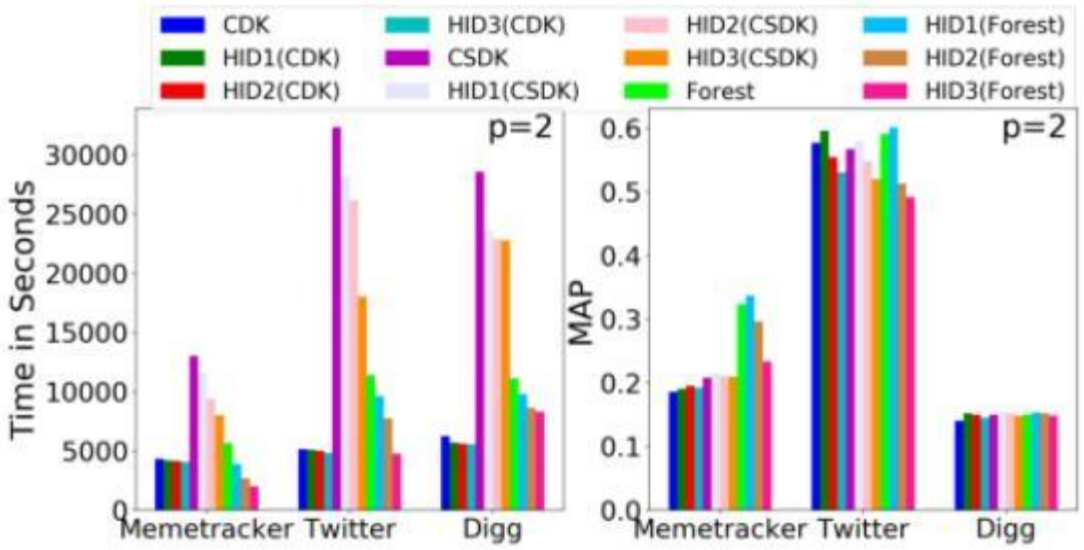


图 11. 方法之间的效率比较及其 MAP 评分

5 复现细节

5.1 与已有开源代码对比

本篇论文具有开源代码，图 12、13 为关键公式 k 阶扩散矩阵计算的代码呈现：

```
"""
A_pk: adjacency matrix of diffusion path p with step k
"""
def adjacency_matrix(diffpath, num_user, step, uname_uid):
    adjmtx = lil_matrix((num_user, num_user), dtype=np.int8)
    for i in range(len(diffpath) - step):
        srt_node = uname_uid[diffpath[i]]
        dst_node = uname_uid[diffpath[i + step]]
        adjmtx[srt_node, dst_node] += 1
    return adjmtx


"""
A_p: adjacency matrix of diffusion path p with all accumulated steps
"""
def diffpath_adjacency_matrix_cooccur(diffpath, num_user, uname_uid):
    diffpath_adjmtx = lil_matrix((num_user, num_user), dtype=np.int8)
    max_step = len(diffpath) - 1
    for step in range(1, max_step + 1):
        adjmtx = adjacency_matrix(diffpath, num_user, step, uname_uid)
        diffpath_adjmtx += adjmtx
    return diffpath_adjmtx
```

图 12

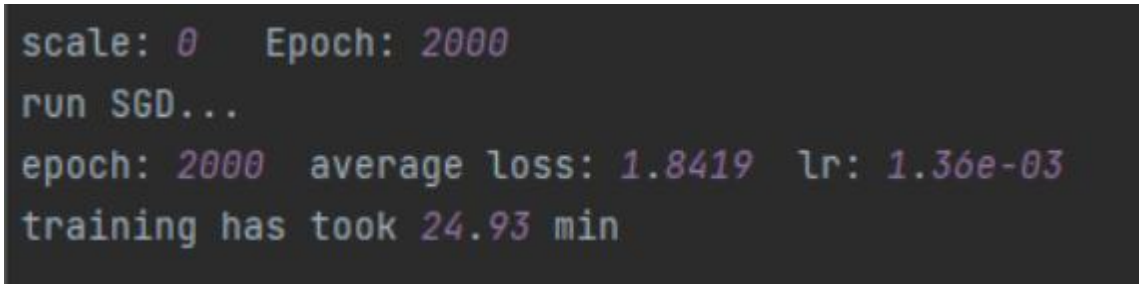
```
A_p_neighbor: adjacency matrix of diffusion path p with accumulated steps
              up to the neighboring threshold
"""
def diffpath_adjacency_matrix_thresh(diffpath, num_user, uname_uid, threshold):
    diffpath_adjmtx = lil_matrix((num_user, num_user), dtype=np.int8)
    max_step = min(len(diffpath) - 1, threshold)
    for step in range(1, max_step + 1):
        adjmtx = adjacency_matrix(diffpath, num_user, step, uname_uid)
        diffpath_adjmtx += adjmtx
    return diffpath_adjmtx


"""
A_p_origin: adjacency matrix of diffusion path p regarding origin
"""
def diffpath_adjacency_matrix_origin(diffpath, num_user, uname_uid):
    # diffpath_adjmtx = lil_matrix((num_user, num_user), dtype=np.int8)
    max_step = len(diffpath) - 1
    EA = uname_uid[diffpath[0]]
    iEA = lil_matrix((num_user, 1), dtype=np.int8)
    iEA[EA] = 1

    tmp = lil_matrix((num_user, num_user), dtype=np.int8)
    for step in range(1, max_step + 1):
        adjmtx = adjacency_matrix(diffpath, num_user, step, uname_uid)
        tmp += adjmtx
    diffpath_adjmtx = iEA.dot((tmp[EA]).reshape(1, num_user))
    return diffpath_adjmtx
```

图 13

图 14、15 为训练 2000 代的损失和时间开销：



```
scale: 0   Epoch: 2000
run SGD...
epoch: 2000  average loss: 1.8419  lr: 1.36e-03
training has took 24.93 min
```

图 14

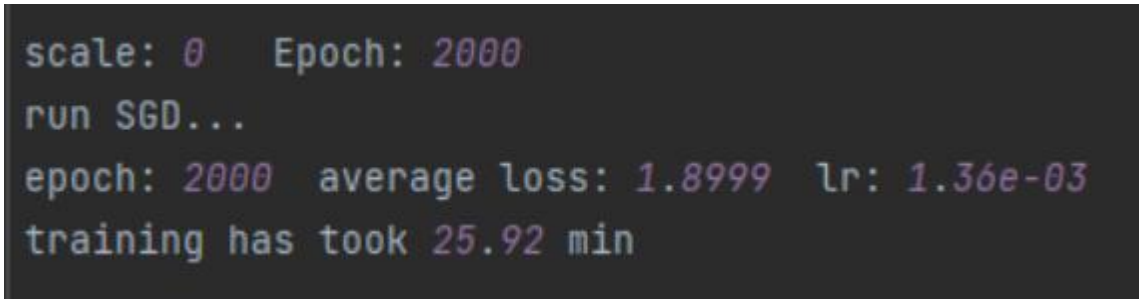


图 15

5.2 创新点

原文中使用了 HAC”、“K-means”和“Spectral”三种聚类方法， 在这里通过改变 upsaling operator 中的聚类算法于层次密度聚类 HDBSCAN。这是一个对 DBSCAN 的改进算法， 结合了密度聚类和层次聚类。它的优化点主要如下：

使用相互可达距离替换欧氏距离， 该距离可以使得密度低的点离密度高的区域更远， 减少 dbscan 对 Eps 阈值的依赖性

使用最小生成树构建层次聚类模型，引入层次聚类思想

对最小生成树的最小子树做了限制，减少计算量，同时保证生成的类簇不要过小

使用“簇稳定性”的度量方式自动划分类簇，不需要自行设定阈值

5.3 复现结果

图 16、17 为应用 K-means 聚类算法下采用 HID 和不采用 HID 的复现结果

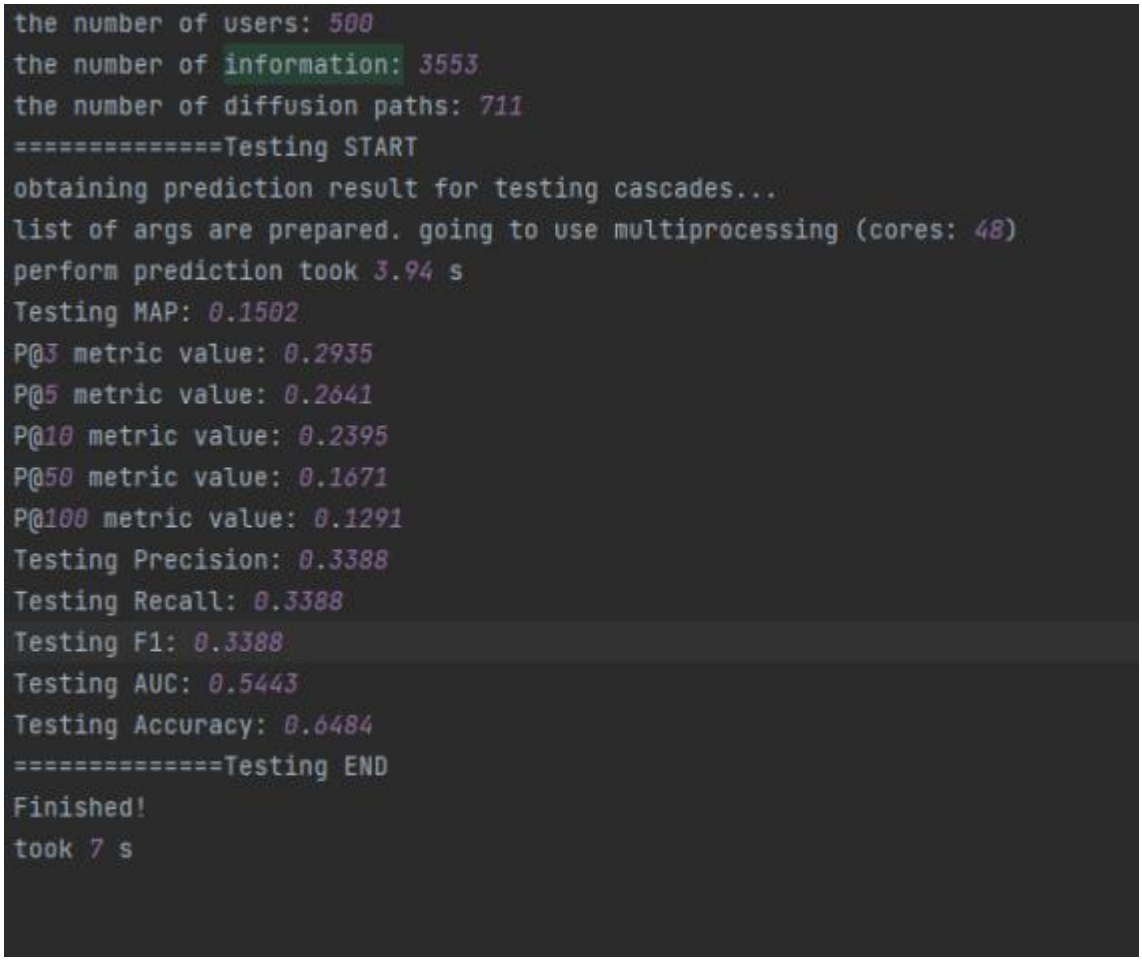

the number of users: 500
the number of information: 3553
the number of diffusion paths: 711
==========Testing START
obtaining prediction result for testing cascades...
list of args are prepared. going to use multiprocessing (cores: 48)
perform prediction took 3.94 s
Testing MAP: 0.1502
P@3 metric value: 0.2935
P@5 metric value: 0.2641
P@10 metric value: 0.2395
P@50 metric value: 0.1671
P@100 metric value: 0.1291
Testing Precision: 0.3388
Testing Recall: 0.3388
Testing F1: 0.3388
Testing AUC: 0.5443
Testing Accuracy: 0.6484
==========Testing END
Finished!
took 7 s

图 16. 应用 HID

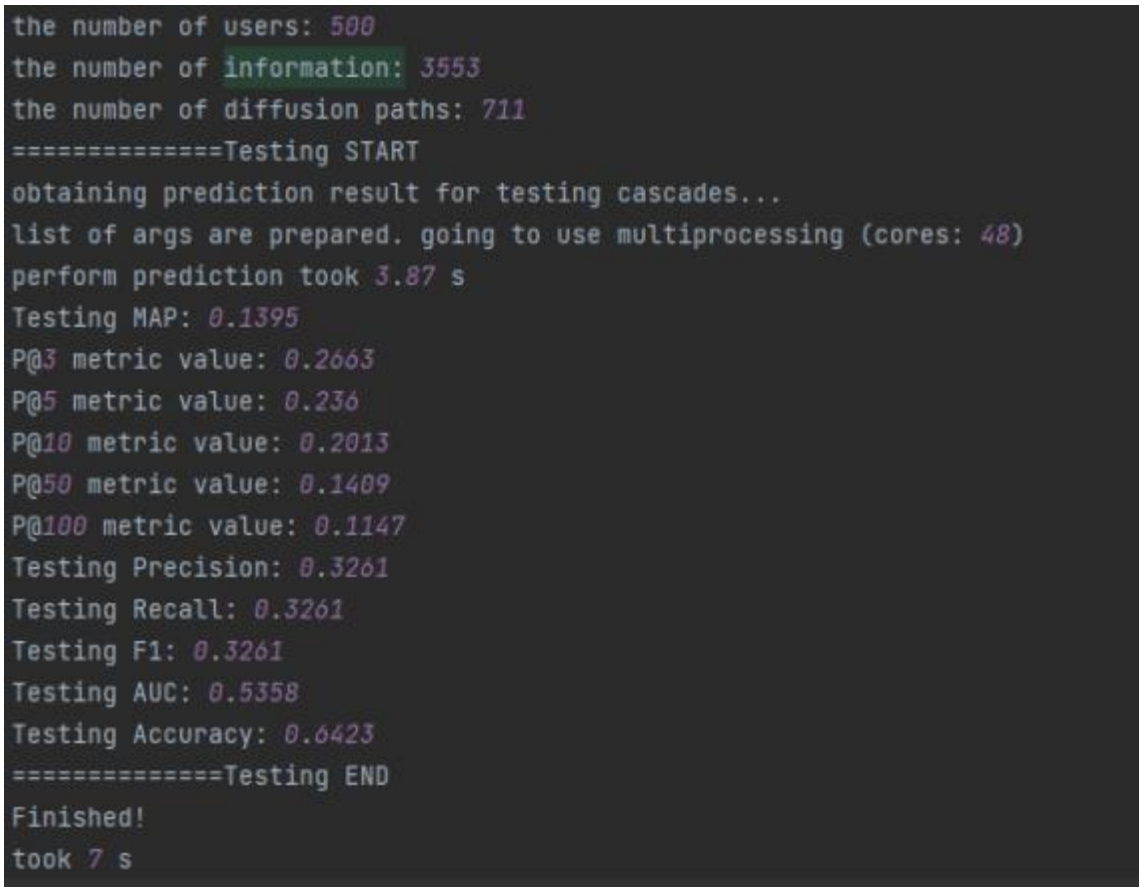


图 17. 不应用 HID

图 18、19 为应用 HDBSCAN 下采用 HID 和不采用 HID 的复现结果

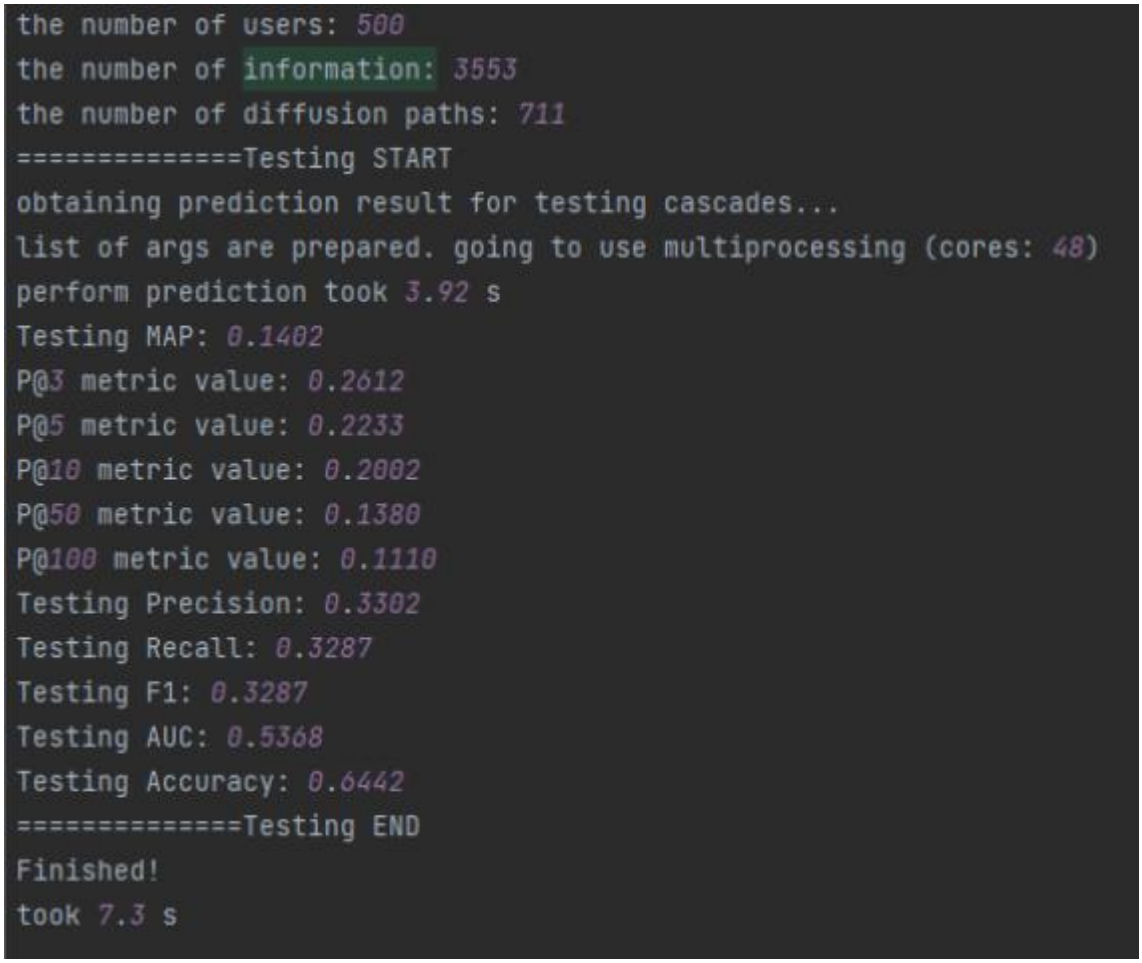

the number of users: 500
the number of information: 3553
the number of diffusion paths: 711
==========Testing START
obtaining prediction result for testing cascades...
list of args are prepared. going to use multiprocessing (cores: 48)
perform prediction took 3.92 s
Testing MAP: 0.1402
P@3 metric value: 0.2612
P@5 metric value: 0.2233
P@10 metric value: 0.2002
P@50 metric value: 0.1380
P@100 metric value: 0.1110
Testing Precision: 0.3302
Testing Recall: 0.3287
Testing F1: 0.3287
Testing AUC: 0.5368
Testing Accuracy: 0.6442
==========Testing END
Finished!
took 7.3 s

图 18. 应用 HID

```
the number of users: 500
the number of information: 3553
the number of diffusion paths: 711
==============Testing START
obtaining prediction result for testing cascades...
list of args are prepared. going to use multiprocessing (cores: 48)
perform prediction took 4.03 s
Testing MAP: 0.1522
P@3 metric value: 0.2945
P@5 metric value: 0.2653
P@10 metric value: 0.2399
P@50 metric value: 0.1685
P@100 metric value: 0.1298
Testing Precision: 0.3395
Testing Recall: 0.3392
Testing F1: 0.3391
Testing AUC: 0.5456
Testing Accuracy: 0.6492
==============Testing END
Finished!
took 7.3 s
```

图 19. 不应用 HID

可见应用 HID 比不应用 HID 的效果在 MAP 评价上提高了一定的性能，采用 HDBSCAN 后有略微提升， 但是可以忽略不记， 因为本篇论文的核心在于 HID 的引入对性能的改进， 聚类算法对本应用产生的影响不大。

6 总结与展望

本文提出了一种新的信息扩散任务的层次框架。所提出的框架 HID 可以被分层在利用用户表示的所有信息扩散技术之上。 HID 有助于更高效的学习和准确的预测。大量实验证明了其优越的性能。在复现期间， 需要用到源代码， 还无法自己从零复现一篇论文的能力， 希望后续能够提升自己的代码能力和对整篇论文的框架掌握能力。在未来， 希望将 HID 扩展到对其他方面进行建模（例如， 信息表示、不同放大算子的组合、通过缩小来混合知识）， 以便获得进一步的改进。

参考文献

[1] Maosong Sun Ganqu Cui et al Cheng Y ang, Jian Tang. Multi-scale information diffusion prediction with reinforced recurrent networks. *Int. J. In AAAI*, 2019.

[2] andParag Singla Harvineet Singh, Amitabha Bagchi. Learning user representations in online social networks using temporal dynamics of information diffusion. *arXiv preprint arXiv*, page 1710.07622, 2017.

[3] Patrick Galli-nari et al Sheng Gao, Huacan Pang. A novel embedding method for information diffusion prediction in social network big data. *Int. J.IEEE Transactions on Industrial Informatics*, 13(4):2097–2105, 2017.

[4] Sylvain Lam-prier Simon Bourigault and Patrick Gallinari. Representation learning for information diffusion through social networks: an embedded cascade model. *In WSDM,ACM*, pages 573–582, 2016.

[5] Zhitao Wang and Wenjie Li. Hierarchical diffusion attention network. *Int. J.In IJCAI AAAI Press*, pages 3828–3834, 2019.

[6] et al Y uan Zhang, Tianshu Lyu. Cosine: Community-preserving social network embedding from information diffusion cascades. *Int. J. In AAAI*, 2018.

[7] et al Zhitao Wang, Chengyao Chen. Information diffusion prediction with network regularized role-based user representation learning. *Int. J. TKDD*, 2019.