Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

基于知识图谱的推荐算法研究

图注意力网络

文本安全
硕士研究生 潘珲

2023年05月21日

内容提要

- 背景简介
  - 推荐算法
- 基本概念
  - GAT
- 算法原理
  - KGAT
  - metaKG
- 总结
- 前沿发展
- 参考文献

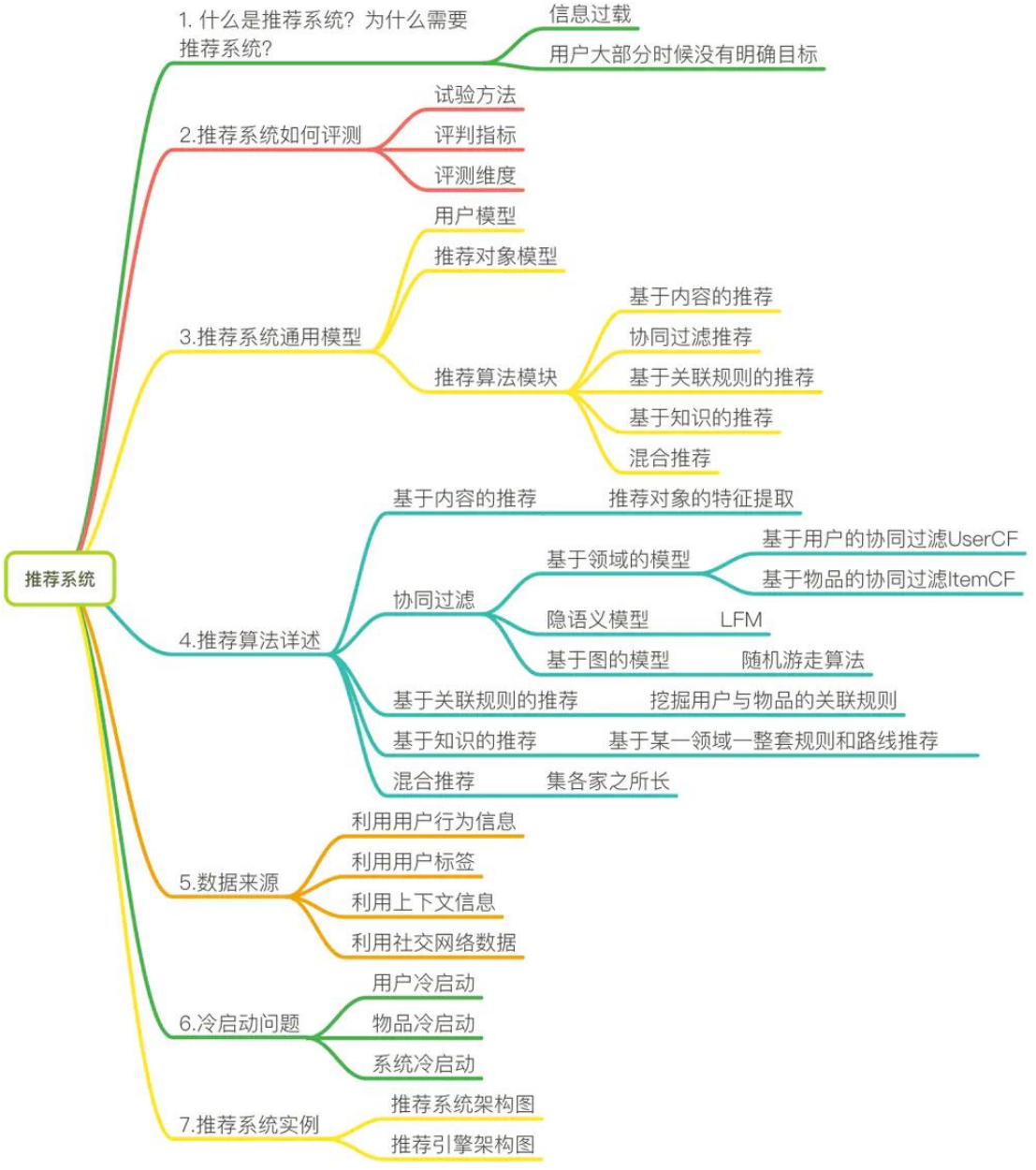
预期收获

- 预期收获
  - 了解推荐算法的基本概念
  - 熟悉推荐算法的应用场景
  - 理解图推荐算法的原理
  - 了解图推荐算法的未来发展

背景简介

- 推荐系统
  - 推荐系统通过发掘用户的行为，找到用户的个性化需求，从而将物品准确推荐给需要它的用户，帮助用户找到他们感兴趣但很难发现的物品

- 作用
  - **解决互联网时代下的信息超载问题**
  - 提供用户个性化服务，建立用户粘性



背景简介

- 通用模型
  - 用户建模模块
    - 通过用户行为，建立用户模型
  - 推荐对象建模模块
    - 通过物品的信息，建立推荐对象模型
  - 推荐算法模块
    - 通过用户兴趣匹配物品的特征信息，再经过推荐算法计算筛选，找到用户可能感兴趣的推荐对象，然后推荐给用户

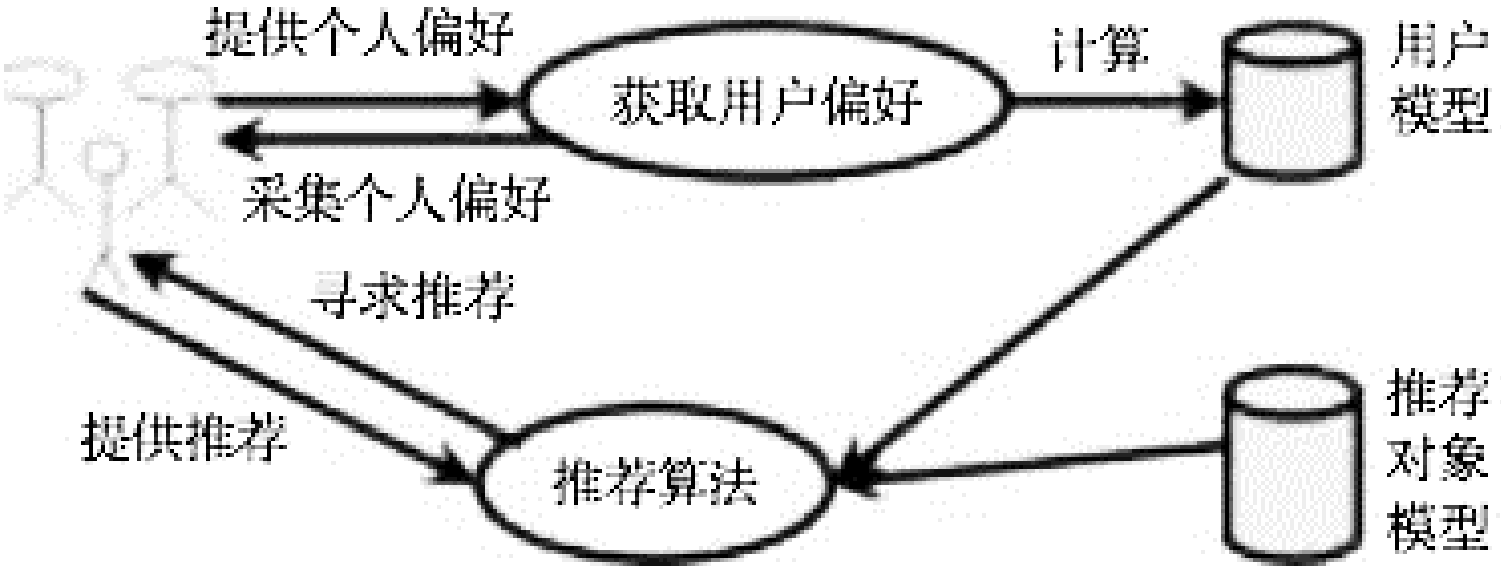


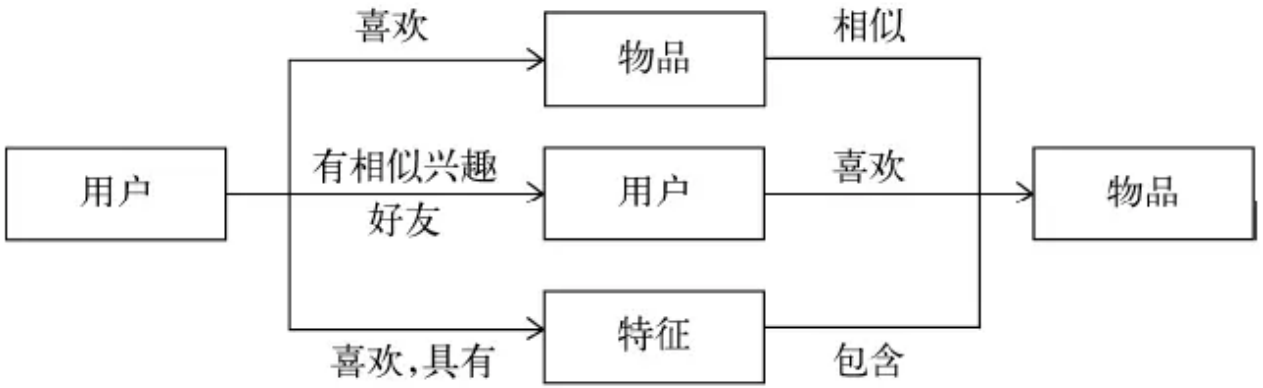
图1　推荐系统通用模型

背景简介 用户建模

- 用户建模
  - 获取用户信息就是解决模型输入数据的问题，输入数据主要有以下几种
    - 用户属性
      - 人口统计学信息（性别、年龄、国籍等）
    - 用户手动输入的信息
      - 包括用户在搜索引擎中输入的关键词，用户反馈的信息，对推荐对象的喜好程度等
    - 用户的浏览行为和浏览内容
      - 包括浏览次数、频率、停留时间等，浏览页面时的操作(收藏、保存、复制等)等
      - 服务器端保存的日志也能较好地记录用户的浏览行为和内容
  - 用户模型的建模方法主要有遗传算法、基于机器学习的方法，例如 TF-IDF、自动聚类、贝叶斯分类器、决策树归纳和神经网络方法等

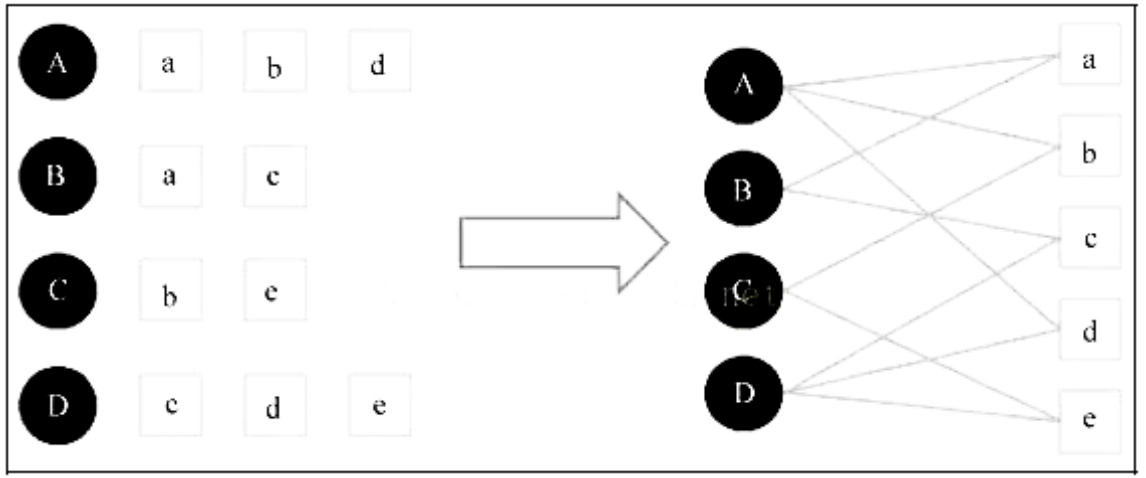
背景简介 推荐算法

- 基于内容的推荐
  - 基于用户感兴趣的物品A，找到和A内容信息相近的物品B
  - 利用用户和物品本身的内容特征，如用户的地理位置、性别、年龄，电影物品的导演、演员、发布时间等
- 协同过滤推荐
  - 基于用户行为数据设计的推荐算法，称为协同过滤算法
  - 此方法主要根据用户对物品的历史行为，寻找用户或物品的近邻集合，以此计算用户对物品的偏好



背景简介 推荐算法

- 基于图的模型
  - 将用户行为数据表示为一系列的二元组
  - 每一个二元组(u,i)代表用户u对物品 i 产生过交互行为，这样便可以将这个数据集表示为一个二分图
  - 给用户u推荐物品，可以转化为计算用户顶点u和与所有物品顶点之间的相关性，然后取与用户没有直接边相连的物品，按照相关性的高低生成推荐列表



背景简介 基于图的推荐

- 度量图中两个顶点之间相关性
  - 两个顶点之间的路径数
  - 两个顶点之间的路径长度
  - 两个顶点之间的路径经过的顶点特征

- 相关性高的特征
  - 两个顶点之间有很多路径相连
  - 链接两个顶点之间的路径长度都比较短
  - 链接两个顶点之间的路径不会经过出度比较大的顶点

- 这其实是一个排名问题，通过衡量相关性，给出推荐列表

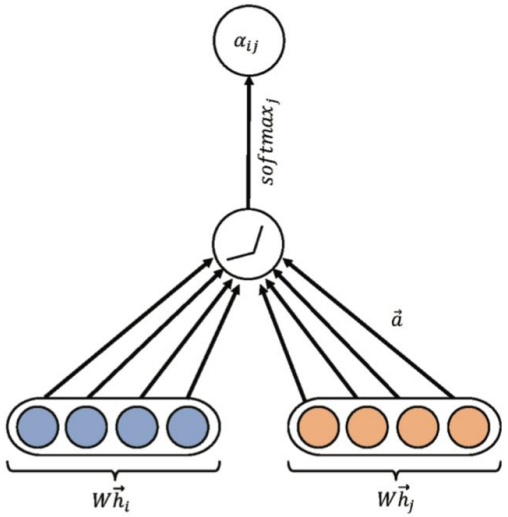
基本概念

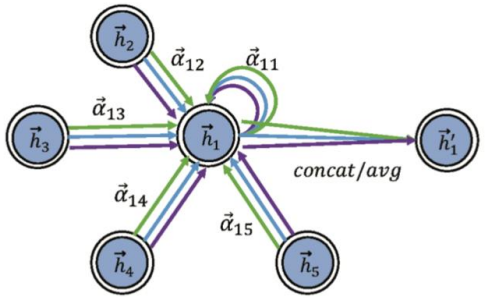
基本概念

- 在 GCN 里，一次图卷积操作包含对邻节点特征的标准化求和

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \frac{1}{c_{ij}} W^{(l)} h_j^{(l)}\right),$$

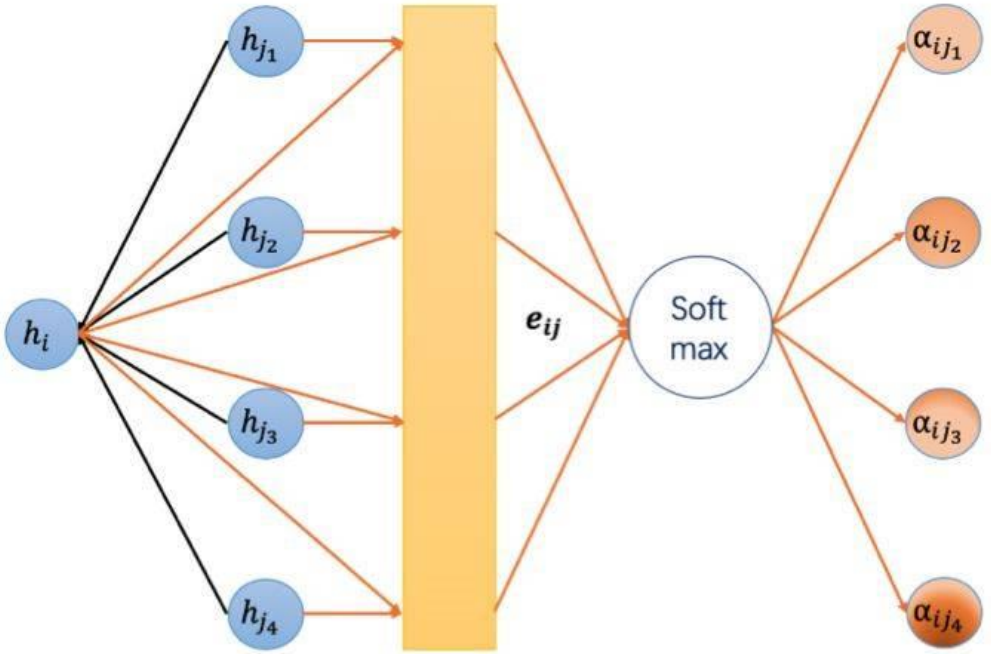
  - c_{ij}是一个基于图结构的标准化常数
  - 这使得在一张图上学习到的 GCN 模型比较难直接应用到另一张图上
- GAT 和 GCN 的核心区别在于如何收集并累和距离为 1 的邻居节点的特征表示
- 图注意力网络—**GAT**
  - 用注意力机制对邻近节点特征加权求和
  - 邻近节点特征的权重完全取决于节点特征





基本概念

- 图注意力模型 GAT
  - 用注意力机制替代了图卷积中固定的标准化操作
  - 右图和公式定义了如何对第 l 层节点特征做更新得到第 l+1 层节点特征
    - 对l层节点嵌入做线性变换
    - 计算成对节点间的原始注意力分数
    - 对于一个节点所有边得到的原始注意力分数应用一个softmax得到注意力权重
    - 节点特征更新，对所有邻居节点的特征做基于注意力的加权求和



$$z_i^{(l)} = W^{(l)} h_i^{(l)}, \tag{1}$$

$$e_{ij}^{(l)} = \mathrm{LeakyReLU}(\vec{a}^{(l)^T}(z_i^{(l)} \| z_j^{(l)})), \tag{2}$$

$$\alpha_{ij}^{(l)} = \frac{\exp(e_{ij}^{(l)})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik}^{(l)})}, \tag{3}$$

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} z_j^{(l)} \right), \tag{4}$$

算法原理

TIPO-KGAT

T	目标	在推荐任务中引入知识图谱提供推荐
I	输入	用户物品交互、物品属性
P	处理	1.提取用户物品交互和物品属性 2.构建知识图谱 3.使用TransR进行嵌入 4.使用注意力模块传播信息 5.计算用户偏好
O	输出	用户偏好推荐

P	问题	基于路径的方法工作量大
C	条件	基于知识图谱
D	难点	如何捕获高阶相关性
L	水平	KDD2019

算法原理 KGAT

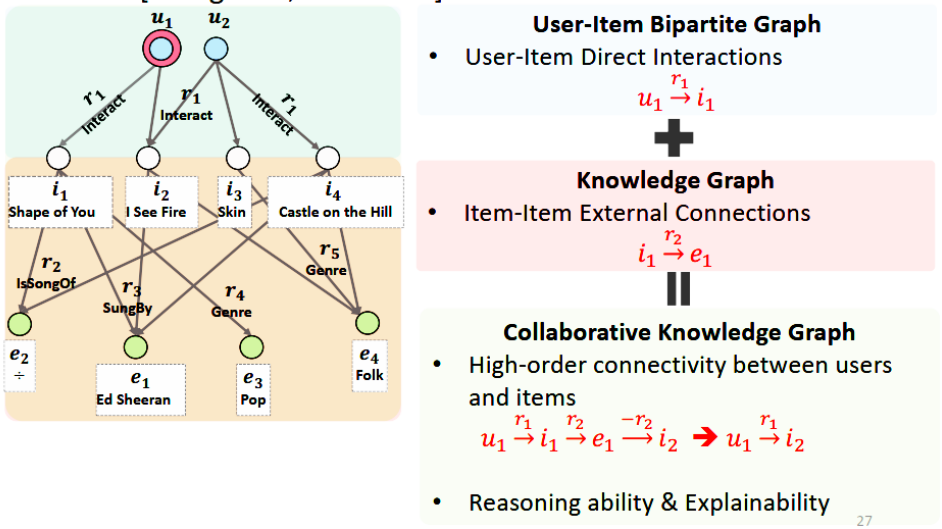
- **KGAT**
  - 通过邻居节点来更新当前节点的表示，并且该算法能够在线性的时间复杂度上进行递归来捕获高阶连接性
  - 利用注意力机制来学习传播过程中每个邻居的权重，使得这种权重可以揭示不同高阶连通的重要程度

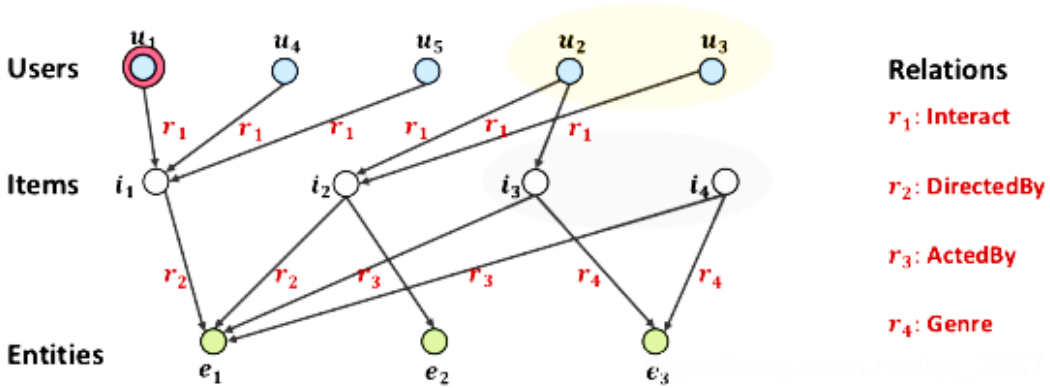
- 优点：
  - 避免了Path-based提取路径工作量大的问题
  - 避免了高阶相关性无法捕获的问题：显式地将高阶关系纳入预测模型，对所有相关参数进行定制以优化推荐目标

算法原理

- 协作知识图（CKG）
  - 知识图谱（物品间的连接）和用户-物品二分图（主要是用户和物品的交互）的统一整体流程
- 高阶连通性
  - 高阶关系：通过一个或多个属性来连接两个item（用户或物品）
  - 利用高阶关系对于执行高质量推荐非常重要
  - 将节点之间的 L阶连接 定义为一个多跳关系路径：$e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} \ldots \xrightarrow{r_L} e_L$

KGAT from [Wang et al, KDD'2019]





算法原理 KGAT

- 研究方法：
  - KGAT模型如下图所示，主要包含三个部分：
  - Embedding Layer：通过保留CKG的结构将每个节点参数化为一个向量
  - Attentive Embedding Propagation Layer：递归地传播节点邻居的Embedding信息以更新其表示，并利用知识感知的注意力机制在传播过程中学习每个邻居的权值
  - Prediction Layer：集成来自所有传播层的用户和物品的表示，并输出相应的预测评分

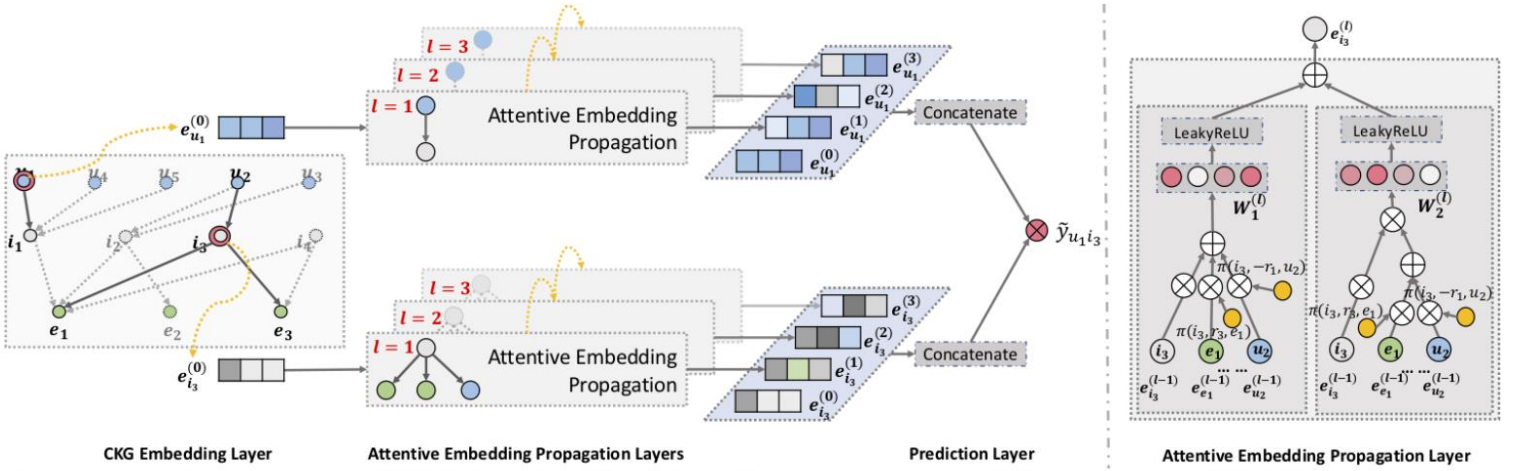


Figure 2: Illustration of the proposed KGAT model. The left subfigure shows model framework of KGAT, and the right subfigure presents the attentive embedding propagation layer of KGAT.

算法原理 KGAT

- Embedding Layer
  - 知识图谱的嵌入，是参数化实体和关系作为向量表示的一种有效方法，同时能够保留图的结构信息
  - 采用了TransR方法，对于三元组(h , r , t) 的似然得分

$$g(h,r,t) = ||W_r e_h + e_r - W_r e_t||_2^2$$

  - 通过有效三元组和无效三元组进行区分的思想，训练成对损失函数

$$\mathcal{L}_{\mathrm{KG}} = \sum_{(\mathrm{h},\mathrm{r},\mathrm{t},\mathrm{t}')\in\mathcal{T}} -ln\,\sigma\big(g(h,r,t') - g(h,r,t)\big)$$

- 该层相当于知识表示的规则化，提升了模型的表示能力

算法原理 KGAT

- Attentive Embedding Propagation Layers
  - 信息传播
    - 对于实体h，通过$N_{\mathrm{h}}=\{(h,r,t)\mid(h,r,t)\in G\}$表示三元组集合，称为ego-network。这种线性组合的思想刻画了实体的一阶连接结构

$$e_{N_h}=\sum_{(h,r,t)\in N_h}\pi(h,r,t)e_t$$

    - 其中$\pi(h,r,t)$控制在关系(h,r,t)中实体间传播的衰减系数，其作用是：显示出通过关系r有多少信息能够从t传播到h
  - 知识感知注意力
    - 通过注意力机制对$\pi(h,r,t)$公式化，在关系上距离更近的实体间会传递更多的信息

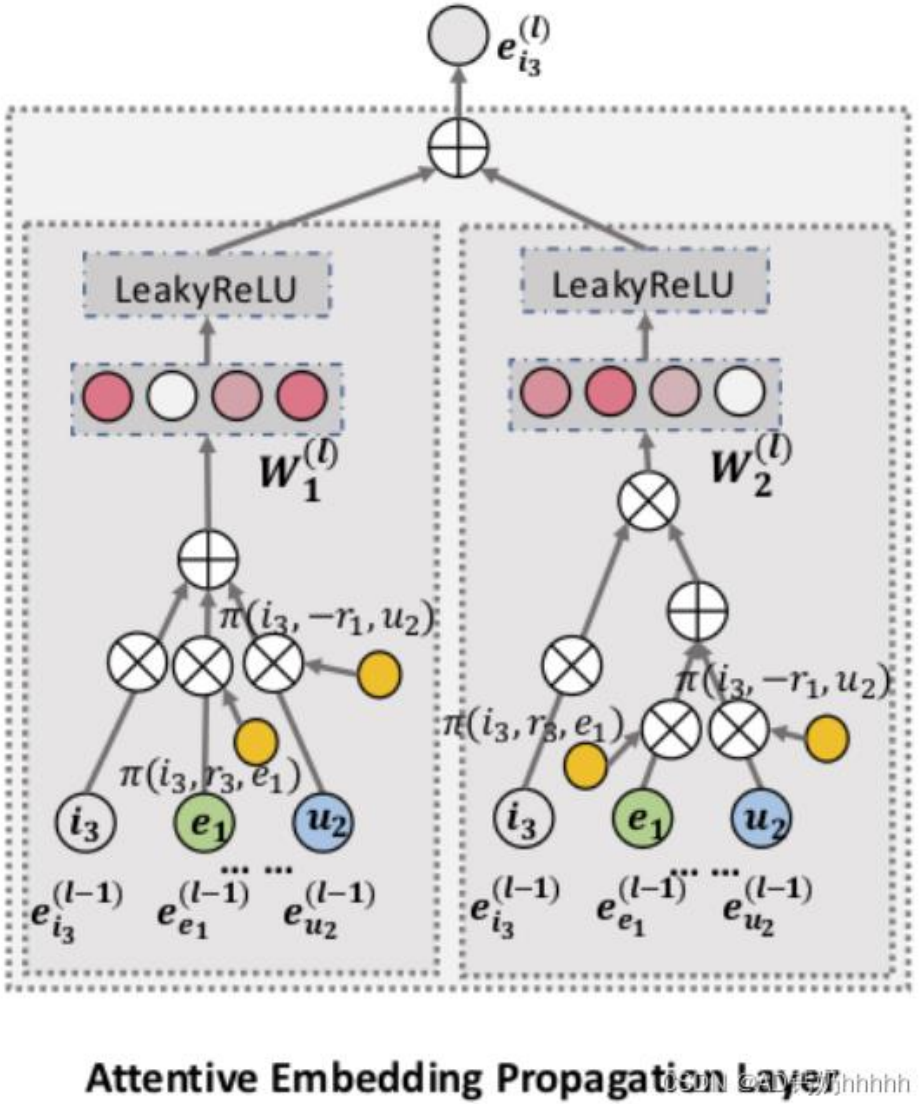
$$\pi(h,r,t)=(\mathrm{W_r e_t})^{\top}\tanh(\mathrm{W_r e}_h+\mathrm{e_r})$$

$$\pi(h,r,t)=\mathrm{softmax}\big(\pi(h,r,t)\big)$$

算法原理 KGAT

- Attentive Embedding Propagation Layers
  - 信息聚合
    - 该模块的作用：将前两层的结果（实体表示e_h和 ego-networke_{N_h} ）进行集成，作为实体h新的表示形式
    - 选择双线性交互聚合器

$$f_{\text{Bi-Interaction}} = \text{LeakyReLU}\left(W_1\left(e_h^l + \overrightarrow{e_h}^l\right)\right) + \text{LeakyReLU}\left(W_1\left(e_h^l \odot \overrightarrow{e_h}^l\right)\right)$$



Attentive Embedding Propagation Layer

算法原理 KGAT

- Attentive Embedding Propagation Layers
  - 多层传播
    - 通过多层传播可以捕获更多信息（多跳邻居所传播的）递归地将一个实体表示为

$$e_h^l = f\left(e_h^{(l-1)}, e_{N_h}^{(l-1)}\right)$$

- 模型预测
  - 预测评分：

$$\hat{y}(u,i) = e_u^{*T} e_i^*$$

  - 其中，用户和物品表示是通过layer-aggregation机制，将不同层的输出所强调的连接信息进行集成：

$$e_u^* = e_u^0 \| \cdots \| e_u^L, e_i^* = e_i^0 \| \cdots \| e_i^L$$

算法原理 KGAT

- 优化
  - 通过BPR损失函数对推荐模型进行优化
  - 优化的意图是：观察到的交互表示出用户的偏好，那么相应就应该有更高的预测评分

$$\mathcal{L}_{CF} = \sum_{(u,i,j)\in\mathcal{O}} -\ln \sigma\left(e_u^{\star\top} e_i^{\star} - e_u^{\star\top} e_j^{\star}\right)$$

  - 将该损失项与前面结合，定义最终的目标优化函数

$$\mathcal{L}_{\mathrm{KGAT}} = \mathcal{L}_{KG} + \mathcal{L}_{CF} + \lambda||\Theta||_2^2$$

实验结果 KGAT

- 整体实验
  - 评价指标: **recall ndcg**
  - 整体指标提升至少5%
- 参数实验：
  - 传播层数效果
    - 基本层数越多越好
  - 聚类器实验
    - 论文选择了三个聚类器，最终选择了效果最好的Bi-Interaction聚类器

Table 2: Overall Performance Comparison.

	Amazon-Book		Last-FM		Yelp2018	
	recall	ndcg	recall	ndcg	recall	ndcg
FM	0.1345	0.0886	0.0778	0.1181	0.0627	0.0768
NFM	**0.1366**	0.0913	**0.0829**	0.1214	0.0660	0.0810
CKE	0.1343	0.0885	0.0736	0.1184	0.0657	0.0805
CFKG	0.1142	0.0770	0.0723	0.1143	0.0522	0.0644
MCRec	0.1113	0.0783	-	-	-	-
RippleNet	0.1336	0.0910	0.0791	0.1238	**0.0664**	**0.0822**
GC-MC	0.1316	0.0874	0.0818	**0.1253**	0.0659	0.0790
KGAT	**0.1489***	**0.1006***	**0.0870***	**0.1325***	**0.0712***	**0.0867***
%Improv.	8.95%	10.05%	4.93%	5.77%	7.18%	5.54%

Table 3: Effect of embedding propagation layer numbers (L).

	Amazon-Book		Last-FM		Yelp2018	
	recall	ndcg	recall	ndcg	recall	ndcg
KGAT-1	0.1393	0.0948	0.0834	0.1286	0.0693	0.0848
KGAT-2	0.1464	0.1002	0.0863	0.1318	0.0714	**0.0872**
KGAT-3	0.1489	0.1006	0.0870	0.1325	0.0712	0.0867
KGAT-4	**0.1503**	**0.1015**	**0.0871**	**0.1329**	**0.0722**	0.0871

Table 4: Effect of aggregators.

	Amazon-Book		Last-FM		Yelp2018	
Aggregator	recall	ndcg	recall	ndcg	recall	ndcg
GCN	0.1381	0.0931	0.0824	0.1278	0.0688	0.0847
GraphSage	0.1372	0.0929	0.0822	0.1268	0.0666	0.0831
Bi-Interaction	**0.1393**	**0.0948**	**0.0834**	**0.1286**	**0.0693**	**0.0848**

贡献 KGAT

- 主要贡献
  - 强调了在协同知识图谱(CKG)中显式建模高阶关系(high-order relation)的重要性，以通过项目的边信息来提供更好的推荐
  - 提出KGAT方法，在图神经网络框架下，以显式地、端到端的方式实现了high-order relation的建模
  - 在三个公开的benchmark上对该方法进行了大量实验，证明了KGAT的有效性及其在理解高阶关系的重要性方面的可解释性

算法原理

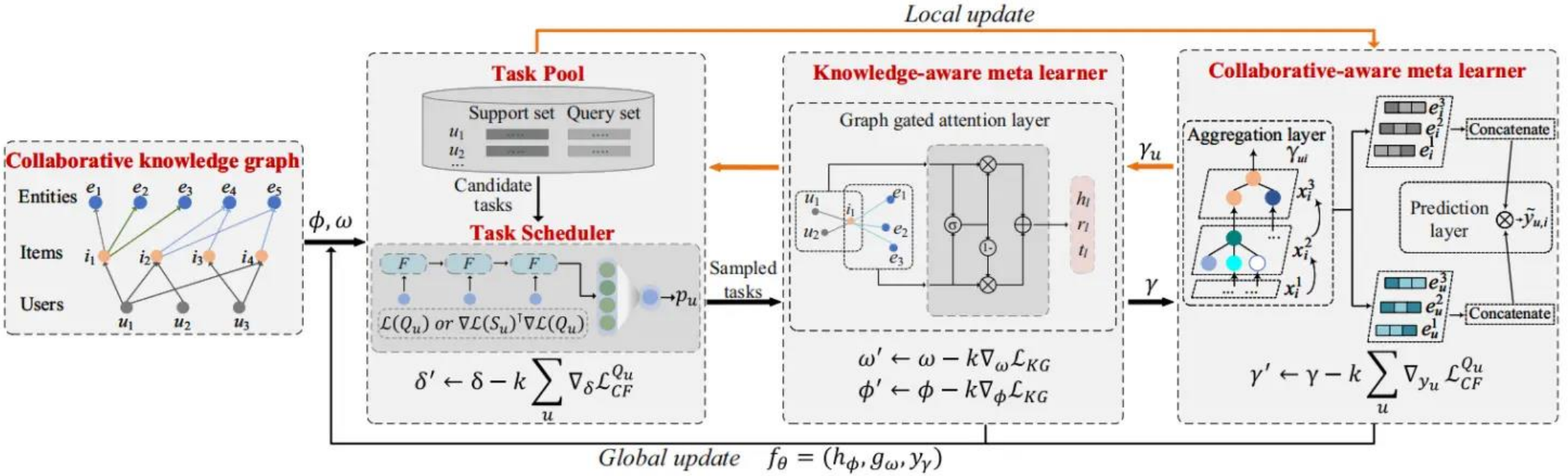
MetaKG

TIPO-MetaKG

T	目标	在冷启动情况下推荐
I	输入	用户物品交互、物品属性
P	处理	1.提取用户物品交互和物品属性 2.构建知识图谱 3.使用TransR进行嵌入 4.使用注意力模块传播信息 5.使用元学习更新嵌入参数 5.计算用户偏好
O	输出	用户偏好推荐
P	问题	当用户-项目稀缺，KG的性能就下降
C	条件	知识图谱元学习
D	难点	在冷启动如何通过KG捕获更准确的先验知识
L	水平	TKDE2022

算法原理 MetaKG

- 核心思想-元学习
  - 设计两个元学习器（collaborative-aware meta learner and knowledge-aware meta learner）可以有效捕获冷启动中有限用户-项目交互的高阶协作关系和语义表示
  - 自适应任务调度程序：可以通过任务采样的概率来决定调度任务的先后，以及两个learner的优化



算法原理 MetaKG

- 协作感知元学习器-从CKG中学习多方位的偏好
  - 与KGAT一样使用BPR损失来学习每个用户的偏好

$$\mathcal{L}_{CF}^{S_u} = \sum_{(u,i,j)\in\mathcal{O}} -\ln \sigma\left(e_u^{\star\top} e_i^{\star} - e_u^{\star\top} e_j^{\star}\right)$$

  - 对于嵌入来说每一层的特征交互可以进行编码，并串联成一个最终表示：

$$e_u^* = ||_{l\in L} y_\gamma^l(e_u^l, \overrightarrow{e_u{}^l})$$

  - 那么协作信号就可以在元学习器中通过任务的可学习聚合器自动捕捉

$$\gamma_u = \gamma - v\frac{\partial \mathcal{L}_{CF}^{S_u}(e_u^*)}{\partial \gamma}$$

算法原理 MetaKG

- 知识感知元学习器-学习知识关联和任务的语义关联
  - 在三元组粒度上对实体进行建模，与KGAT一样对于给定三元组的评分：

$$g(h,r,t) = \left|\left|W_r e_h + e_r - W_r e_t\right|\right|_2^2$$

  - 其损失函数为

$$\mathcal{L}_{KG} = \sum_{(h,r,t,t')\in\mathcal{T}} -ln\,\sigma\big(g(h,r,t') - g(h,r,t)\big)$$

  - 实体表征是由φ和ω参数化的，实体的语义信息不应该与特定的任务相适应，而是应该在整个知识图谱中共享

$$\phi' = \phi - k\frac{\partial \mathcal{L}_{KG}(\gamma, c)}{\partial \phi},\quad \omega' = \omega - k\frac{\partial \mathcal{L}_{KG}(\gamma, c)}{\partial \omega}$$

算法原理 MetaKG

- 自适应调度函数
  - 有些任务是有噪声的
  - 任务对用户偏好学习有不同的贡献
- 构建自适应的任务调度器，以选择信息量最大的任务
  - 把调度器定义为参数δ的z，并利用两个代表性的因素来量化T_u中涵盖的信息：

$$p_u = Z_\delta\left(L(Q_u), \nabla L(S_u)^T \nabla L(Q_u)\right)$$

  - p_u是候选任务T_u的采样概率
  - $L(Q_u)$表示查询集的损失
  - 第二部分是任务T_u的支持集和查询集的梯度相似度
    - 如果梯度相似度大，大的查询集损失可能代表真正的困难任务
    - 一个查询集有噪声的任务可能导致大的查询损失，但梯度相似度小
- 选用LSTM网络作为调度器，编码每个任务的历史信息

算法原理 MetaKG

- 门控注意力模块
  - 先前的方法通常汇总所有邻居的三元组用来生成上下文表示，没有区分邻居实体所携带的语义信息

$$\overrightarrow{e_i^c} = g_\omega(h, r, u) = \sum_{(i,r,u)\in\mathcal{N}_h^c} \alpha(i, r, u)e_u$$

  - 对于项目h,邻居实体t和邻居用户u分别代表他的知识相关性和协作相关性
  - 设计一个门控注意力模块，明确的聚合两种信息

$$g_i = \sigma\left(W_c^T \overrightarrow{e_i^c} + W_k \overrightarrow{e_i^k}\right)$$

$$\overrightarrow{e_i} = g_i \cdot \overrightarrow{e_i^c} + (1 - g_i)\overrightarrow{e_i^k}$$

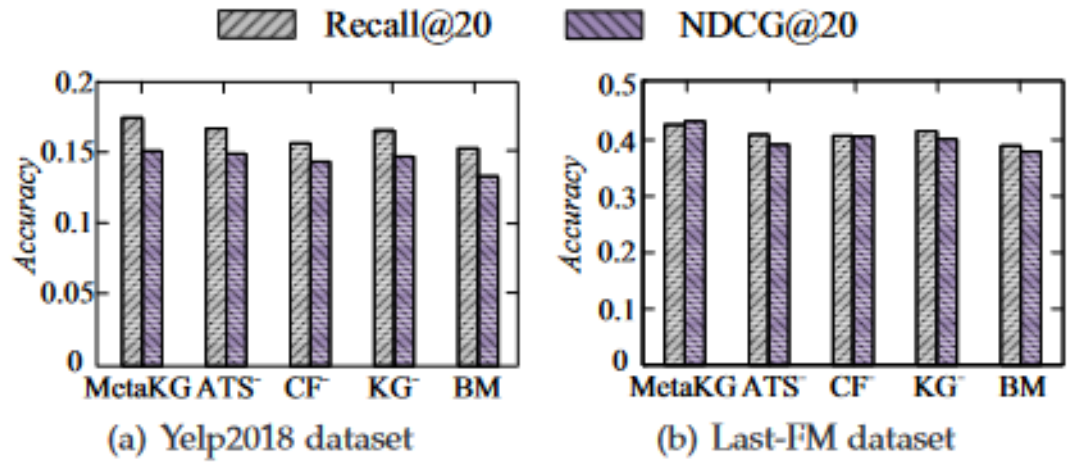
实验效果

- 效果实验
  - **MetaKG 模型在四种情况下均取得最佳性能**
  - **MetaKG模型比其他基线模型至少提高了 5%**

Scenario	Methods	Yelp2018			Last-FM			Amazon-book		
		Recall@20	NDCG@20	RI	Recall@20	NDCG@20	RI	Recall@20	NDCG@20	RI
User Cold-start (UC) Old items for new users	FM	0.0978	0.0812	45.74%	0.1917	0.1659	110.86%	0.1597	0.1420	116.17%
	NFM	0.1174	0.0998	19.93%	0.2143	0.1905	86.06%	0.1791	0.1573	94.00%
	CKE	0.1188	0.1050	16.18%	0.2632	0.2521	45.96%	0.2401	0.2391	35.72%
	CFKG	0.0936	0.0811	48.95%	0.2525	0.2362	53.92%	0.2180	0.2050	53.84%
	UGRec	0.0711	0.0624	94.81%	0.1912	0.1956	94.66%	0.0778	0.0717	335.56%
	MCRec	0.0920	0.0793	51.95%	0.1580	0.1532	141.70%	0.1529	0.1641	105.51%
	RippleNet	0.1001	0.0833	42.22%	0.1760	0.1607	123.51%	0.1815	0.1741	82.93%
	KGCN	0.1123	0.0901	29.23%	0.2206	0.2067	76.04%	0.2445	0.2279	37.79%
	KGAT	0.1240	0.1110	10.59%	0.2821	0.2744	35.16%	0.2804	0.2712	17.92%
	CKAN	0.0844	0.0802	57.74%	0.1634	0.1649	129.26%	0.2650	0.2567	24.67%
	MeLU	0.1102	0.1051	20.59%	0.2439	0.2344	57.25%	0.2265	0.2207	45.44%
	MetaHIN	0.1260	0.1204	5.38%	0.2610	0.2628	43.68%	0.2569	0.2504	28.21%
	MetaKG	**0.1362**	**0.1236**	–	**0.3927**	**0.3598**	–	**0.3291**	**0.3213**	–
Item Cold-start (IC) New items for old users	FM	0.0888	0.0755	78.25%	0.3574	0.3234	38.23%	0.2055	0.1792	16.57%
	NFM	0.0844	0.0740	84.69%	0.3582	0.3192	39.02%	0.2084	0.1829	14.57%
	CKE	0.0888	0.0809	72.26%	0.3476	0.3112	42.92%	0.2167	0.1931	9.33%
	CFKG	0.0917	0.0775	73.14%	0.3523	0.3259	38.65%	0.2073	0.1809	15.51%
	UGRec	0.1349	0.1188	15.29%	0.2623	0.2334	90.00%	0.1716	0.1519	38.53%
	MCRec	0.0750	0.0691	102.85%	0.3605	0.3312	35.98%	0.2070	0.1573	24.47%
	RippleNet	0.0920	0.0737	77.37%	0.3090	0.2777	60.45%	0.1710	0.1492	40.05%
	KGCN	0.0904	0.0718	81.32%	0.3270	0.3111	47.27%	0.1748	0.1529	36.83%
	KGAT	0.0905	0.0788	72.83%	0.3445	0.3272	39.91%	0.2143	0.1902	10.78%
	CKAN	0.1360	0.1204	14.06%	0.2758	0.2946	65.00%	0.1762	0.1693	29.51%
	MeLU	0.0712	0.0620	119.68%	0.2217	0.2119	116.72%	0.1920	0.1869	18.11%
	MetaHIN	0.0670	0.0528	145.64%	0.2399	0.2266	101.47%	0.1990	0.1705	21.47%
	MetaKG	**0.1571**	**0.1356**	–	**0.4780**	**0.4616**	–	**0.2336**	**0.2141**	–
User-Item Cold-start (UIC) New items for new users	FM	0.1181	0.0949	53.67%	0.3367	0.3043	33.21%	0.2412	0.1825	30.10%
	NFM	0.1121	0.0982	54.95%	0.3390	0.3020	33.32%	0.2721	0.2265	9.60%
	CKE	0.1293	0.1156	32.99%	0.3118	0.2976	39.81%	0.2691	0.2206	11.71%
	CFKG	0.0992	0.0819	80.42%	0.3208	0.3013	37.02%	0.2444	0.2102	19.98%
	UGRec	0.1173	0.1023	48.41%	0.2571	0.2369	72.68%	0.2535	0.2216	14.72%
	MCRec	0.0910	0.0859	84.05%	0.1990	0.1857	121.63%	0.1800	0.1634	58.52%
	RippleNet	0.1150	0.1025	49.76%	0.2150	0.2110	99.89%	0.1969	0.1435	62.68%
	KGCN	0.1189	0.1004	48.81%	0.3473	0.3060	30.90%	0.2653	0.2298	10.13%
	KGAT	0.1498	0.1360	13.93%	0.3352	0.3182	30.41%	0.2670	0.2305	9.62%
	CKAN	0.1462	0.1166	24.62%	0.2854	0.2943	46.88%	0.2750	0.2273	8.85%
	MeLU	0.1308	0.1237	27.94%	0.2071	0.1875	116.37%	0.2498	0.2014	21.41%
	MetaHIN	0.1430	0.1382	15.82%	0.2030	0.1871	118.67%	0.2656	0.2285	10.39%
	MetaKG	**0.1747**	**0.1513**	–	**0.4227**	**0.4287**	–	**0.2857**	**0.2587**	–
Non-cold-start Old items for old users	FM	0.0916	0.0779	77.43%	0.2193	0.1948	103.42%	0.1168	0.1073	111.06%
	NFM	0.1002	0.0870	60.47%	0.2677	0.2344	67.91%	0.1368	0.1256	80.25%
	CKE	0.1353	0.1214	16.86%	0.2952	0.2804	45.95%	0.2233	0.2181	6.95%
	CFKG	0.0802	0.0693	100.98%	0.2850	0.2625	53.59%	0.1239	0.1162	96.82%
	UGRec	0.0592	0.0499	175.81%	0.1898	0.1927	119.55%	0.0398	0.0351	532.93%
	MCRec	0.0670	0.0529	152.29%	0.1220	0.1078	266.68%	0.0670	0.0619	266.84%
	RippleNet	0.0690	0.0631	126.97%	0.1910	0.1721	131.82%	0.1061	0.0864	147.96%
	KGCN	0.0645	0.0579	145.09%	0.2006	0.1882	116.15%	0.1117	0.0910	135.47%
	KGAT	0.1270	0.1166	23.07%	0.3004	0.2892	42.45%	0.2221	0.2188	7.06%
	CKAN	0.0868	0.0692	93.73%	0.2071	0.2042	104.16%	0.1040	0.0960	136.44%
	MeLU	0.1266	0.1183	22.38%	0.2828	0.2755	50.42%	0.1709	0.1630	41.46%
	MetaHIN	0.1420	0.1226	13.56%	0.3029	0.2886	42.02%	0.2160	0.2181	8.72%
	MetaKG	**0.1562**	**0.1436**	–	**0.4221**	**0.4176**	–	**0.2344**	**0.2376**	–

实验效果

- 消融实验
  - 在Yelp2018和Last-Fm上进行
  - 可以看出每一个模块都是有必要的
  - ATS：自适应任务调度
  - CF：协同感知元学习器
  - KG：任务感知元学习器
  - BM：同时没有CF和KG



(a) Yelp2018 dataset (b) Last-FM dataset

Fig. 4. Ablation Study of MetaKG in UIC Scenario

总结

优劣分析

- **KGAT**
  - 优势
    - 避免了Path-based中提取路径工作量大的问题
    - 避免了高阶相关性无法捕获的问题
  - 劣势：未处理冷启动问题
- **MetaKG**
  - 优势
    - 利用门控注意力捕获交互种类信息
    - 利用元学习捕获新节点信息
  - 劣势：图模型较大

应用总结

- 应用领域
  - 冷启动推荐
  - 点击率预测
  - 线上推荐系统

- 发展方向
  - 深入挖掘不同边的注意力更新方法
  - 探讨图神经网络在推荐中领域的潜力
  - 其他的结构化知识可以考虑：社交网络、物品上下文等
  - 信息传播机制和决策过程的集成，有助于推进可解释推荐研究的进展
  - 推荐隐私与安全

参考文献

[1] Finn C, Abbeel P, Levine S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks[J].

[2] Wang X, He X, Cao Y, et al. Kgat: Knowledge graph attention network for recommendation[C]//Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019: 950-958.

[3] Du Y, Zhu X, Chen L, et al. Metakg: Meta-learning on knowledge graph for cold-start recommendation[J]. IEEE Transactions on Knowledge and Data Engineering, 2022.

道德经

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！