LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation

摘要

图卷积网络（GCN）已成为协同过滤的最新技术。 然而，其推荐效果的原因尚不清楚。现有的使 GCN 适应推荐的工作缺乏对 GCN 的彻底的消融分析，GCN 最初是为图分类任务设计的，并配备了许多神经网络操作。 然而，我们凭经验发现 GCN 中最常见的两种设计——特征变换和非线性激活——对协同过滤的性能贡献不大。 更糟糕的是，包含它们会增加训练难度并降低推荐性能。在这项工作中，我们的目标是简化 GCN 的设计，使其更加简洁且适合推荐。 我们提出了一个名为 LightGCN 的新模型，仅包含 GCN 中最重要的组件——邻域聚合——用于协同过滤。 具体来说，LightGCN 通过在用户-物品交互图上线性传播用户和物品嵌入来学习用户和物品嵌入，并使用所有层学习到的嵌入的加权和作为最终嵌入。 这种简单、线性和简洁的模型更容易实现和训练，表现出显着的改进（平均相对改进约 16.0%） ——在完全相同的实验环境下。 从分析和实证的角度对简单 LightGCN 的合理性进行了进一步的分析。 我们的实现可在 TensorFlow和 PyTorch 中使用。

关键词：协同过滤，推荐，嵌入传播，图神经网络

1 引言

本篇论文是关于推荐系统中的图神经网络以及LightGCN模型的研究。推荐系统是指利用用户的历史行为和偏好，为其推荐可能感兴趣的物品或服务的技术。图神经网络是一种用于处理图数据的深度学习模型，能够有效地捕捉节点之间的结构和关联信息。LightGCN模型是针对推荐系统中的图数据设计的一种简化的图卷积网络模型，旨在提高推荐的准确度并且简化模型的复杂性。研究人员注意到Graph Convolution Network(GCN)已成为协同过滤领域的最新技术，但是对于它在推荐系统中的有效性，还存在着不太清晰的理解。因此，研究人员认为有必要对GCN在推荐系统中的应用进行全面的剖析。此外，现有工作在将GCN应用于推荐任务时，缺乏对GCN的深入剖析，GCN最初是图分类任务设计的，并且有许多神经网络操作。因此，选题依据主要包括对GCN在推荐系统中的应用进行深入理解，以及对GCN的详尽分析，尤其是GCN在推荐系统中的有效性和操作机理。通过研究图神经网络在推荐系统中的应用，有助于推动推荐系统的技术发展，提高推荐系统的准确性和效率。推荐系统在处理用户偏好和信息的过程中面临着诸多挑战，如稀疏性，冷启动问题等。通过研究图神经网络在推荐系统中的应用，可能为解决这些问题提供新的思路和办法。还可以拓展图神经网络的应用领域，通过在推荐系统中探索图神经网络的应用，可以为图神经网络的应用领域提供新的方向，促进图神经网络在其他领域的应用和发展。

2 相关工作

2.1 协同过滤

协同过滤是一种常用于推荐系统中的技术，它基于用户对物品的历史行为信息（比如评分、喜好等），通过分析用户之间的相似度或物品之间的相似度来进行推荐。这种算法主要基于两类协同信息：用户之间的协同关系（User-User Collaborative Filtering）和物品之间的协同关系（Item-Item Collaborative Filtering）。这些信息可以通过不同的方式进行利用，一种常见的方式是基于矩阵分解的技术，比如矩阵分解模型（Matrix Factorization）等。

近年来，随着深度学习和图神经网络的发展，还出现了基于图神经网络的协同过滤算法，如图卷积网络（Graph Convolutional Network, GCN）等。这些算法能够更好地处理推荐系统中的图数据，从而提升推荐的准确性和效率。

总的来说，协同过滤算法通过分析用户行为数据或物品属性，利用用户之间的相似性或物品之间的相关性，来实现个性化的推荐。

2.2 Graph Convolution Network

Graph Convolution Network (GCN)是一种用于处理图数据的深度学习模型。它的基本思想是通过在图上执行图卷积操作来学习节点的表示。在这个过程中，GCN迭代地聚合邻居节点的特征，用于更新目标节点的表示。GCN进行卷积操作主要有两种方法：一种是基于谱分解， 即谱分解图卷积。 另一种是基于节点空间变换，即空间图卷积。

GCN基于图信号的谱分解的卷积操作。在谱图理论中，一个图的拉普拉斯矩阵（Laplacian Matrix）具有特征值分解（Eigen Decomposition）的性质，其特征向量和特征值可以被用来表示图结构的特征和性质。具体来说，在GCN中，图卷积操作的实现借助了图的拉普拉斯矩阵。对于一个给定的图G，它的拉普拉斯矩阵通常定义为L = D - A，其中D是度矩阵（Degree Matrix），A是邻接矩阵（Adjacency Matrix）。根据谱图理论，拉普拉斯矩阵的特征向量和特征值可以提供关于图结构的有用信息。在GCN中，拉普拉斯矩阵的谱分解可以被用来定义图卷积操作。通过使用拉普拉斯矩阵的特征向量，可以将图信号进行傅里叶变换，从而实现图卷积。这样的操作使得GCN能够更好地利用图的结构信息，从而有效地学习节点的表示。

GCN基于节点空间变换的卷积操作。该模型的基本思想是通过在图上执行图卷积操作来对节点进行表示学习。通过利用图信号的傅里叶变换，GCN的卷积操作可以被理解为对节点的空间变换，从而实现对节点表示的更新和学习。

具体来说，GCN的基于节点的空间变换的卷积操作可以通过以下公式进行描述：

$$e_u^{(k+1)} = AGG(e_u^{(k)}, \{e_i^{(k)} : i \in N_u\}) \quad (1)$$

其中，$e_u^{(k+1)}$ 表示第k层节点u的表示，AGG为邻域聚合的函数，$\{e_i^{(k)} : i \in N_u\}$是节点u的邻居节点表示集合，这个过程实际上可以被理解为对节点特征的变换和聚合，从而实现节点的表示的更新和学习。

利用基于节点空间变换的卷积操作，GCN能够有效地捕获图结构中节点之间的关联信息，从而实现更加精准的节点表示学习。这种基于节点空间变换的卷积操作为推荐系统提供了一种新的图谱分解视角和手段，有助于提升推荐的准确性和个性化程度。

2.3 Neural Graph Collaborative Filtering (NGCF)

Neural Graph Collaborative Filtering (NGCF) 是一种图神经网络模型。NGCF 结合了协同过滤和图神经网络的思想，以便更好地利用用户和物品之间的交互信息。NGCF的核心思想是使用图神经网络来学习用户和物品的嵌入表达，以捕获它们之间的复杂交互关系。具体来说，NGCF通过以下方式实现：图的建模：将用户和物品之间的交互关系建模为一个二分图，图中包含用户节点和物品节点。邻域聚合：采用图卷积网络的邻域聚合思想，将用户节点和物品节点的邻居节点特征进行聚合，以更新目标节点的表示。特征转换：通过可训练的权重矩阵来进行特征转换，以更好地表达节点的特征。非线性激活：采用非线性激活函数来增强模型的表达能力。

在初始步骤中，每个用户和物品都与一个ID嵌入相关联。设$e_u^{(0)}$表示用户u的ID嵌入，$e_i^{(0)}$表示物品i的物品嵌入。则NGCF利用用户项交互图将嵌入传播为：

$$e_u^{(k+1)} = \sigma\left(W_1 e_u^{(k)} + \sum_{i \in N_u} \frac{1}{\sqrt{|N_u||N_i|}}\left(W_1 e_i^{(k)} + W_2\left(e_i^{(k)} \bigodot e_u^{(k)}\right)\right)\right)$$
$$e_i^{(k+1)} = \sigma\left(W_1 e_i^{(k)} + \sum_{u \in N_i} \frac{1}{\sqrt{|N_u||N_i|}}\left(W_1 e_u^{(k)} + W_2\left(e_u^{(k)} \bigodot e_i^{(k)}\right)\right)\right) \quad (2)$$

其中，$e_u^{(k)}$和$e_i^{(k)}$分别表示用户u和物品i经过k层传播后的精细化嵌入，σ为非线性激活函数，N_u表示用户u交互的物品集合，N_i表示与物品i交互的用户的集合，W_1与W_2为可训练权矩阵，在每一层进行特征变换。通过传播L层，NGCF得到L+1个嵌入来描述用户$(e_u^{(0)}, e_u^{(1)} \ldots\ldots e_u^{(L)})$和物品$(e_i^{(0)}, e_i^{(1)} \ldots\ldots e_i^{(k)})$。然后将这些L+1嵌入连接起来，得到最终的用户嵌入和物品嵌入，使用内积生成预测分数。

NGCF模型通过多层的邻域聚合和特征转换，学习到了用户和物品的多层嵌入表示，从而能够更好地捕捉它们之间的复杂关系，并用于个性化推荐。为图神经网络在推荐任务中的应用提供了重要的范例。

2.4 SGCN

针对GCN在节点分类方面存在的不必要的复杂性，提出了SGCN算法，该算法通过去除非线性并将权矩阵压缩为一个权矩阵来简化GCN算法。SGCN中的图卷积定义为:

$$E^{(K+1)} = (D + I)^{-\frac{1}{2}}(A + I)(D + I)^{-\frac{1}{2}} \quad (3)$$

其中$I \in R^{(m+n)*(m+n)}$是一个单位矩阵，加到A上包含自连接。在SGCN中，最后一层得到的嵌入用于下游预测任务，可以表示为:

$$E^{(K)} = (A + I)E^{(K-1)} = (A + I)^K E^{(0)}$$
$$= \binom{k}{0} E^{(0)} + \binom{k}{1} A E^{(0)} + \binom{k}{2} A^2 E^{(0)} + \ldots + \binom{k}{K} A^k E^{(0)} \quad (4)$$

由上述推导可知，在A中插入自连接并在其上传播嵌入，本质上相当于在每个LGC层上传播的嵌入的加权和。

2.5 APPNP

在最近的一项工作中，作者将GCN与个性化PageRank联系起来，并由此提出了一种名为APPNP的GCN变体，该变体可以长距离传播而不会有过平滑的风险。受Personalized PageRank中的传送设计的启发，APPNP用起始特征(即第0层嵌入)补充每个传播层，这可以平衡保持局域性(即，保持靠近根节点以减轻过度平滑)的需求和利用来自大型邻域的信息。APPNP中的传播层定义为:

$$E^{(k+1)} = \beta E^{(0)} + (1 - \beta) A E^{(k)} \quad (5)$$

β为控制传播过程中起始特征保留的传送概率，A为归一化邻接矩阵。在APPNP中，最后一层用于最终预测，即:

$$E^{(k)} = \beta E^{(0)} + (1 - \beta) A E^{(k-1)}$$
$$= \beta E^{(0)} + \beta(1 - \beta) A E^{(0)} + (1 - \beta)^2 A^{-2} E^{(k-2)}$$
$$= \beta E^{(0)} + \beta(1 - \beta) A E^{(0)} + \beta(1 - \beta)^2 A^{-2} E^{(0)} + \ldots + (1 - \beta)^K A^{-k} E^{(0)} \quad (6)$$

通过相应设置α_k, LightGCN 可以完全恢复 APPNP 使用的预测嵌入。因此，LightGCN 在对抗过平滑方面与 APPNP 具有相同的优势——通过适当设置α，我们允许使用大 K 进行远程建模，并具有可控的过平滑。

另一个小区别是 APPNP 将自连接添加到邻接矩阵中。然而，由于不同层的加权和，这是冗余的。

3 本文方法

3.1 本文方法概述

这篇论文主要介绍了一种名为LightGCN的推荐系统模型，它是对图卷积网络（GCN）进行了简化和优化的模型。GCN已经成为协同过滤任务中的一种最先进的技术，但对于它在推荐系统中的有效性的原因尚不十分清楚。而现有工作中对GCN在推荐系统中的应用缺乏深入的分析，LightGCN模型旨在简化图卷积网络，提高推荐系统的性能。

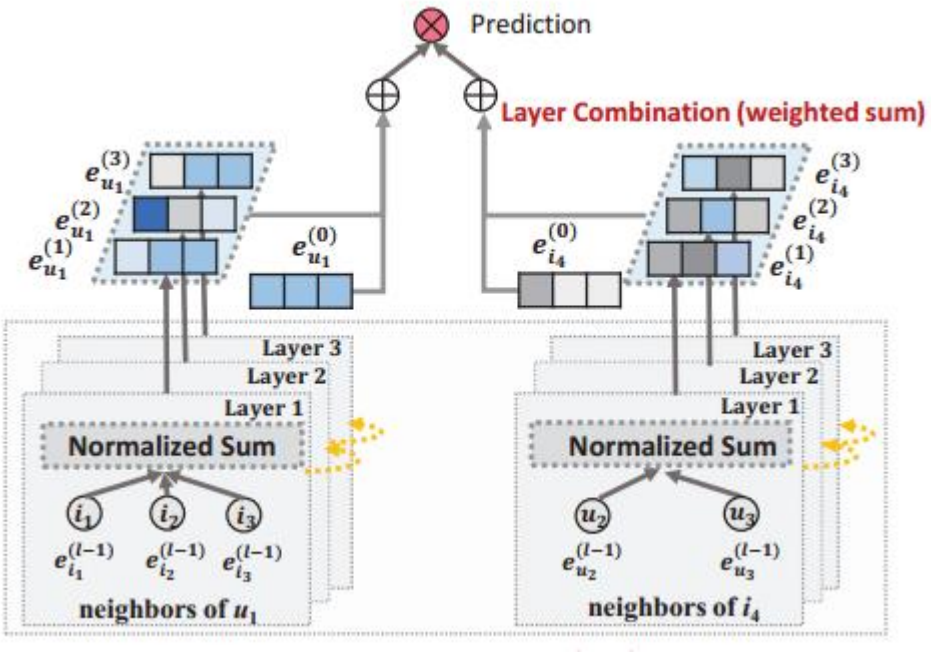

图 1. Light Graph Convolution(LGC)

图1为LightGCN模型体系结构图，在LightGCN中。采用简单的加权和聚合，放弃了特征变换和非线性激活的使用。LightGCN中的图卷积操作定义为：

$$e_u^{(k+1)} = \left(\sum_{i \in N_u} \frac{1}{\sqrt{|N_u||N_i|}} e_i^{(k)} \right)$$
$$e_i^{(k+1)} = \left(\sum_{u \in N_i} \frac{1}{\sqrt{|N_u||N_i|}} e_u^{(k)} \right) \quad (7)$$

对称归一化项$\frac{1}{\sqrt{|N_u||N_i|}}$遵循标准GCN的设计，可以避免嵌入的规模随着图卷积操作的增加而增加。值得注意的是，在LGC中，我们只聚合已连接的邻居，而不集成目标节点本身（即自连接）。因为层组合操作本质上具有与自连接相同的效果，所以LGC中不需要包含自连接操作。

在LightGCN中，唯一可训练的模型参数是第0层的嵌入，即所有用户的$e_u^{(0)}$和所有物品的e_i^0，当它们给定时，可以通过式(7)定义的LGC计算更高层的嵌入，在K层LGC之后，我们进一步将每层得到的嵌入组合起来，形成用户(一个物品)的最终表示：

$$e_u = \sum_{k=0}^{K} \alpha_k e_u^{(k)} \; ; \; e_i = \sum_{k=0}^{K} \alpha_k e_i^{(k)} \quad (8)$$

其中$\alpha_k \geq 0$表示第k层嵌入对构成最终嵌入的重要性。它可以被视为手动调整的超参数，也可

以被视为自动优化的模型参数。在实验中将α_k统一设置为$\frac{1}{(K+1)}$,通常可以获得良好的性能，同时避免不必要地使LightGCN复杂化，保持其简单性。执行层组合以获得最终表示的原因有三个：(1)随着层数的增加，嵌入会出现过平滑现象。因此，仅仅使用最后一层是有问题的。(2)不同嵌入层的嵌入捕获不同的语义。例如，第一层对有交互的用户和物品进行平滑处理，第二层对交互物品(用户)上有重叠的用户(物品)进行平滑处理，更高层捕获高阶接近度。因此，把它们结合起来会使表现更加全面。(3)将不同层的嵌入与加权和结合起来，捕获了具有自连接图卷积的效果。

模型预测定义为用户与物品最终表示的内积：

$$\hat{y}_{ui} = e_u^T e_i \quad (9)$$

用作推荐生成的排名分数。

3.2 特征提取模块

LightGCN模型在特征提取方面非常简化，因为在协同过滤任务中，每个节点（用户或物品）仅具有一个ID特征，而没有其他显式的特征。因此，LightGCN的特征提取模块并不涉及传统意义上的特征工程或特征提取过程。相比之下，LightGCN主要关注于通过图卷积操作来学习节点的表示，而不是提取显式的特征。

在LightGCN中，特征的提取主要依赖于图卷积操作，通过在用户-物品交互图上执行图卷积操作来学习节点的隐含表示。该模型通过将邻居节点的嵌入进行加权和归一化求和来更新目标节点的表示，从而实现特征的隐含学习和表示。这种轻量化的特征提取模块是LightGCN模型简化的重要组成部分，也是其高效性和灵活性的体现。

3.3 损失函数定义

论文中提到采用了贝叶斯个性化排名（Bayesian Personalized Ranking, BPR）损失函数来训练LightGCN模型。BPR是一种成对损失函数，它鼓励模型将已观察到的用户-物品对的预测评分设为高于未观察到的用户-物品对的预测评分。其数学表达如下：

$$\boldsymbol{L_{BPR}} = -\sum_{u=1}^{M}\sum_{i\in N_u}\sum_{j\notin N_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda\left\|E^{(0)}\right\|^2 \quad (10)$$

其中，$\boldsymbol{L_{BPR}}$代表BPR损失函数，M表示用户的数量，$\boldsymbol{N_u}$表示用户u的已观察物品集合，$\hat{y}_{ui}$表示用户u对物品i的观测评分，而$\hat{y}_{uj}$表示用户u未观察到的物品j的评分。σ代表sigmoid函数，λ表示L2正则化的强度。

通过使用BPR损失函数，模型被鼓励更好地对用户-物品对进行排序，从而提高推荐的准确性和个性化程度。这种损失函数的使用使得训练出的LightGCN模型能够更好地契合推荐系统的实际需求。

4 复现细节

4.1 与已有开源代码对比

以gowalla数据集为例：

首先进行数据处理部分

```
dataset = dataloader1.Loader(path="../data/"+'gowalla')
```

```
810128 interactions for training
217242 interactions for testing
```

其中引用了dataloader1.py中的Loader类对gowalla数据集进行预处理，经过处理后810128个用来训练，217242个用来测试。

然后编写LightGCN类

```
class LightGCN(torch.nn.Module):
    def __init__(self, dataset,dim=1) -> None:
        super().__init__()
        self.dim=dim
        self.dataset=dataset
        self.init_weight()
    def init_weight(self):
        self.user_num=self.dataset.n_users
        self.item_num=self.dataset.m_items
        self.user_embed=torch.nn.Embedding(num_embeddings=self.user_num,embedding_dim=self.dim)
        self.item_embed=torch.nn.Embedding(num_embeddings=self.item_num,embedding_dim=self.dim)
        torch.nn.init.normal_(self.user_embed.weight,std=0.1)
        torch.nn.init.normal_(self.item_embed.weight,std=0.1)
        self.Graph = self.dataset.getSparseGraph()
    def propagate(self,layers):
        user_w=self.user_embed.weight
        item_w=self.item_embed.weight
        all_w=torch.concat([user_w,item_w],dim=0)
        all_w=all_w.to(device)
        w_layers=[all_w]
        for i in range(layers):
            all_w=torch.sparse.mm(self.Graph,all_w)
            w_layers.append(all_w)
        w_layers=torch.stack(w_layers,dim=1)
        output=w_layers.mean(dim=1)
        return output
    def train_step(self,u,i,j,decay,opt):
        output=self.propagate(3)
        user,item=(output[:self.user_num],output[self.user_num:])
        um=user[u]
        im=item[i]
        jm=item[j]
        score_ui=torch.mul(um,im)
        score_ui=torch.sum(score_ui,dim=1)
        score_uj=torch.mul(um,jm)
        score_uj=torch.sum(score_uj,dim=1)
        loss=torch.mean(torch.nn.functional.softplus(score_uj-score_ui))
        #L2
        u0=self.user_embed(u)
        i0=self.item_embed(i)
        j0=self.item_embed(j)
        L2=(u0.norm(2).pow(2)+i0.norm(2).pow(2)+j0.norm(2).pow(2))/(2*float(len(u)))
        loss=loss+decay*L2
        opt.zero_grad()
        loss.backward()
        opt.step()
        return loss.cpu().item()
    @torch.no_grad
    def forward(self,u):
        output=self.propagate(3)
        user,item=(output[:self.user_num],output[self.user_num:])
        um=user[u]
        score=torch.matmul(um,item.t())
        return score
```

对处理后的gowalla数据集使用LightGCN

```
lgcn=LightGCN(dataset,1024)
lgcn=lgcn.to(device)
```

```
loading adjacency matrix
```

经过10个epoch后，可以看到loss呈现降低趋势，precision为0.03696832，recall为0.11968472，ndcg为0.09953653。

```
loss=torch.zeros(1)
for epoch in range(1,500):
    loss=0.
    for batch,(u,i,j) in enumerate(minibatch(4096,U,I,J)):
        loss+=lgcn.train_step(u,i,j,0.00001,opt)
    if epoch%10==0:
        Test(dataset, lgcn, 1,  multicore=0)
    res.append(loss/batch)
    print(epoch,':',batch,' ',loss/batch)
```

```
1 : 197   0.003436712661597329
2 : 197   0.00318760794072133
3 : 197   0.002982056355765704
4 : 197   0.00278938845920419
5 : 197   0.0026084282388201523
6 : 197   0.002439148724079132
7 : 197   0.0022814632192499898
8 : 197   0.002135195284798212
9 : 197   0.0020000327497881453
{'precision': array([0.03696832]), 'recall': array([0.11968472]), 'ndcg': array([0.09953653])}
```

4.2 实验环境搭建

编译器：Pycharm , Jupyter Notebook
环境：torch==1.4.0, pandas==0.24.2, scipy==1.3.0, numpy==1.22.0,
tensorboardX==1.8,scikit-learn==0.23.2 , tqdm==4.48.2

4.3 创新点

先进的负采样策略可能会改善LightGCN训练,常见的负采样策略包括以下几种：

1. 随机负采样：从未发生过交互的用户-物品对中随机选择一些作为负样本。这种策略简单直接，但可能会选择到一些不合适的负样本。
2. 热门负采样：根据物品的流行度进行负采样。流行度越高的物品被选中的概率越大。这种策略可以增加模型对热门物品的学习能力。
3. 稀有负采样：根据物品的稀有度进行负采样。稀有度越高的物品被选中的概率越大。这种策略可以增加模型对稀有物品的学习能力。
4. 硬负采样：根据模型的预测结果选择负样本。具体来说，模型首先对所有用户-物品对进行预测，然后选择预测分数最高的一些作为负样本。这种策略可以增加模型对难以预测的负样本的学习能力。
5. 对抗负采样：通过对抗生成网络（GAN）来生成负样本。生成网络试图生成与正样本相似但又不同的负样本，而判别网络则试图区分正样本和负样本。通过对抗训练，生成网络可以逐渐生成更具挑战性的负样本，从而提高模型的学习能力。 这些负采样策略可以根据具体的应用场景和需求进行选择和调整，以提高推荐模型的性能和效果。

5 实验结果分析

LightGCN是一种用于协同过滤的推荐模型，相比于传统的GCN模型，LightGCN通过简化模型结构和参数，取得了更好的推荐准确性。根据实验结果，在所有情况下，LightGCN 都大幅优于 NGCF。 例如，在 Gowalla 上，NGCF 论文中报告的最高召回率为 0.1570，而LightGCN 在 4 层设置下可以达到 0.1830，高出 16.56%。 平均而言，三个数据集的召回率提升为16.52%，ndcg提升为16.87%，相当显著。 LightGCN的优势主要体现在以下几个方面： 1. 更容易训练：LightGCN通过去除GCN中的非线性激活函数和权重矩阵等冗余参数，简化了模型结构，使得训练过程更加高效和稳定。 2. 更好的泛化能力：LightGCN通过层次组合的方式，将每一层的嵌入向量加权求和得到最终的表示，能够更好地捕捉节点之间的关系，提高了模型的泛化能力。 3. 更有效的推荐：实验结果表明，LightGCN在推荐任务上的准确性明显优于其他方法，包括传统的矩阵分解模型和基于GCN的推荐模型。

总的来说，LightGCN通过简化模型结构和参数，提高了协同过滤任务的推荐准确性，并且具有更好的训练效果和泛化能力。这些结果对于未来推荐模型的发展具有启示作用，并且可以在其他领域的图数据分析中进行进一步探索和应用

Dataset		Gowalla		Yelp2018		Amazon-Book	
Layer #	Method	recall	ndcg	recall	ndcg	recall	ndcg
1 Layer	NGCF	0.1556	0.1315	0.0543	0.0442	0.0313	0.0241
	LightGCN	0.1755(+12.79%)	0.1492(+13.46%)	0.0631(+16.20%)	0.0515(+16.51%)	0.0384(+22.68%)	0.0298(+23.65%)
2 Layers	NGCF	0.1547	0.1307	0.0566	0.0465	0.0330	0.0254
	LightGCN	0.1777(+14.84%)	0.1524(+16.60%)	0.0622(+9.89%)	0.0504(+8.38%)	0.0411(+24.54%)	0.0315(+24.02%)
3 Layers	NGCF	0.1569	0.1327	0.0579	0.0477	0.0337	0.0261
	LightGCN	0.1823(+16.19%)	0.1555(+17.18%)	0.0639(+10.38%)	0.0525(+10.06%)	0.0410(+21.66%)	0.0318(+21.84%)
4 Layers	NGCF	0.1570	0.1327	0.0566	0.0461	0.0344	0.0263
	LightGCN	0.1830(+16.56%)	0.1550(+16.80%)	0.0649(+14.58%)	0.530(+15.02%)	0.0406(+17.92%)	0.0313(+18.92%)

图2. 实验结果示意

6 总结与展望

本篇论文是关于简化和增强图卷积网络（GCN）用于推荐系统的研究论文。该论文提出了一个名为LightGCN的模型，旨在简化GCN模型并提高其在推荐任务中的效果。论文中指出，在LightGCN中，作者放弃了GCN中的特征变换和非线性激活这两种标准操作。在层次组合方面，LightGCN构造了一个节点的最终嵌入，作为其在所有层上嵌入的加权和，证明了这对于控制过度平滑非常有帮助。研究人员通过实验证明，LightGCN具有简单易训练、更好的泛化能力和更高的有效性等优点。他们认为，LightGCN的启发性见解对未来推荐系统模型的发展具有启发性，并且在实际应用中，基于图的模型在推荐系统中变得越来越重要。该论文的主要贡献在于提出了一种简化和增强GCN的模型LightGCN，并通过实验证明了其在推荐系统中的优越性能，为推荐系统领域的研究和实际应用提供了有益的启示。

参考文献

[1] He X, Deng K, Wang X, et al. Lightgcn: Simplifying and powering graph convolution network for recommendation[C]//Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. 2020: 639-64.