基于Transformer改进的LightGCN

摘要

本文基于论文《LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation》进行了复现以及改进。随着互联网的发展，有许多之前没有的需求陆续出现，以往的方法逐渐难以解决，推荐系统应运而生。当前有许多基于GCN设计的推荐系统，但其中大部分都是直接继承了GCN中的操作，并没有对其进行细致分析，文章作者经过分析后发现GCN中有许多对于推荐系统不必要的操作，于是乎作者对其重新设计提出了LightGCN。而在本文中，基于GCN的特性对LightGCN进行了改进，针对GCN层数过多会导致过平滑，不能设置过多的层这一限制，给出了解决方案。最终结果表明该改进的有效的，提高了模型的性能。

关键词：推荐系统；GCN；Transformer

1 引言

当前互联网发展非常迅猛，我们的生活越来越离不开它，伴随着的是快速增长的用户规模和商品数量。要对庞大的用户群体精准投送他们喜欢的内容，这对运营者来说无疑是一个挑战。推荐系统就是为了解决这一个问题。推荐系统的核心是预测用户是否会与一个项目进行交互，例如，点击、评级、购买，以及其他形式的交互。协同过滤专注于利用过去的用户-项目交互来实现预测，依旧是十分有效的方法。在推荐系统中，常用的方法是用embedding来代表用户或者是物品等所需要推荐的东西，并基于它们进行预测。随着研究的深入人们发现，使用历史信息作为输入能更好的构建embedding的质量。随着图神经网络的引入，为推荐系统的发展引入了新的思路，许多模型都基于GCN构建，其引入结构信息，在训练完毕后仍然能保持结构不变的特性让GCN在推荐系统中有十分优秀的表现。推动了推荐系统的进一步发展。但同时，GCN在推荐系统中的应用仍然存在着许多的限制，推荐系统的数据十分的有特点，具有极其稀疏的特性，而GCN并不是依据这个特点进行设计的，这也就导致在GCN中采用的操作直接应用在推荐系统中可能会带来一些负面效果。原文中作者就是基于这一思想，对GCN进行了彻底的消融分析，发现当去除权重矩阵与激活函数后，模型的性能得到了提升，基于这个结果，作者重新设计GCN,提出了LightGCN，在Gowalla等数据集中有很好的效果。但LightGCN依旧有缺陷，在本文中针对GCN在层数比较多时会发生的过平滑现象提出了改进方案，由于LightGCN在信息聚合部分只是去除了权重矩阵与激活函数，本质上还是一个图卷积神经网络，GCN具有的特性LightGCN也同样具备，这也就导致LightGCN信息聚合的层数不能太多，而这也就导致节点难以利用那些距离它较远的节点，数据未能得到充分的利用，基于这个想法，本文对LightGCN进行了改进，提出了LightGCN*。

2 相关工作

2.1 协同过滤

协同过滤基于用户之间或物品之间的相似性来进行推荐。协同过滤的基本思想是，如果一个用户或物品在某些方面与其他用户相似，那么他们在其他方面可能也是相似的。协同过滤主要有两种方式：

(1) 用户协同过滤：用户协同过滤基于用户的历史行为和喜好进行推荐。它的核心思想是寻找与目标用户兴趣相似的其他用户，然后将这些用户喜欢的物品推荐给目标用户。用户协同过滤可以通过计算用户之间的相似度矩阵来实现。

(2) 物品协同过滤：物品协同过滤主要关注物品之间的相似性。它通过计算物品之间的相似度，为用户推荐那些与其历史喜好中的物品相似的其他物品。物品协同过滤通常更适用于大规模的物品集合，因为物品之间的相似度计算可以在离线进行。

协同过滤能够为用户提供个性化的推荐，通过利用用户的历史行为能够很好的预测用户的兴趣，所以协同过滤广泛应用在在现在的推荐系统中，是一个十分有效的方法。

2.2 GCN

GCN即图卷积神经网络，是由 [2]提出的一种能有效处理图结构数据的方法，其每经过一层都会聚合与自身相连节点的信息，图神经网络与其余模型相比最大的特点就是在训练后仍然能保持自身结构，即能将节点之间的结构信息也带入模型的训练中。而在推荐系统之中，用户与用户之间、物品与物品之间还有用户与物品之间都有可能具有不同的关系，所以在推荐系统当中使用GCN是非常合适的选择。 GCN每一层节点数据聚合主要有三个步骤：

(1) 将自己和与自身相连的节点的信息加和

(2) 随后对结果乘以权重矩阵对其进行线性变换

(3) 最后使用激活函数

GCN的原理与CNN很类似，在CNN中，CNN充当着滤波器的存在，从时域转到频域后卷积操作在频域是乘法，在深度学习中CNN学习适合模型的滤波器从而使模型获得更好的效果。而GCN也是类似的，所以作者才会称之为图卷积神经网络。在GCN中，实际上是将节点之间的连接所构成的邻接矩阵的特征向量作为傅里叶变换的基，并对由特征值构成的对角矩阵进行滤波。而步骤（2）（3）中的线性变换和激活函数则是通过多项式来近似滤波函数。但GCN仍然存在许多问题，首先是GCN的层数不可以太深，因为GCN从随机游走的角度看的话，在深度较深之后会导致节点的信息只有各个节点的度所决定，这时模型就丢失了结构信息，就会导致过平滑的现象。另外，通过多项式来近似滤波函数的方法在有些时候不一定合适，不一定能带来很好的效果。总的来说，GCN是处理图结构数据中一个非常有效的工具，但其本身还有许多不完善的地方，需要根据应用场景对其进行改进，从而让GCN表现出更好的效果。本文中复现的LightGCN就是基于以上的问题针对推荐系统这个应用环境修改了GCN的结构使其更加适合用于推荐系统。 LightGCN基于对NGCF [3]做消融

分析的结果对GCN进行了简化，去除了线性变换和激活函数这两个对最终结果没有正面效果的操作，其次，为了更好的表示各个节点的信息，LightGCN将所有层的结果都考虑到最终的预测当中，事实证明这提升了整体的预测性能。在此基础之上，为了实现更好的结果，还对LightGCN做出了改动。具体来说，首先对LightGCN进行分析，由于GCN在层数过多时会出现过平滑的现象，这就导致在LightGCN中不可能堆叠很多层，但如果层数太少，那每个节点能注意到其他节点的数量就会减少，不能考虑到所有节点。为了解决这个问题，在本次复现中在LightGCN开始聚合之前加入了Transformer，特比的，这个Transformer是去除位置编码的，因为在LightGCN中节点之间的顺序并不是信息，加入Transformer的目的是为了利用其对全局的特征提取能力，通过在进入LightGCN前加入Transformer，能让模型关注到原先关注不到的节点，从而让模型得到更全面的信息，随后再进入LightGCN当中自然能学习到更有用的信息，最终结果证明加入Transformer是有效的。本次复现的创新点主要有：

- 在输入LightGCN之前加入了Transformer，让模型能充分关注到各个节点的信息，即使有些节点与自己距离比较远。
- 改变了Transformer的结构，去除了位置编码，使其适合在LightGCN中应用。

2.3 NGCF

NGCF [3]是一个表现非常优秀的基于GCN设计的网络，是一种用于协同过滤的推荐模型，旨在学习用户和物品之间的复杂交互模式，以进行更准确地进行推荐。NGCF能在推荐系统的任务中表现出很好的结果。 NGCF聚合时的操作是：

$$e_u^{k+1} = \sigma(W_1 e_u^k + \sum_{i \in N_u} \frac{1}{\sqrt{|N_u||N_i|}}(W_1 e_i^k + W_2(e_i^k \odot e_u^k))$$

$$e_i^{k+1} = \sigma(W_1 e_i^k + \sum_{i \in N_i} \frac{1}{\sqrt{|N_u||N_i|}}(W_1 e_u^k + W_2(e_u^k \odot e_i^k))$$

NGCF通过引入神经图网络的思想，增强了协同过滤模型对用户和物品之间复杂关系的建模能力，相比传统的协同过滤算法，具有更好的性能。虽然NGCF具有十分优秀的性能，但可以发现在NGCF中有许多操作都是直接继承自GCN中，为了研究这些操作是否真正能对模型的性能有正面作用，作者对NGCF中的各个操作做了消融分析，具体来说作者研究了去除线性变换、去除激活函数和两者都去除对模型性能的影响，结果发现去除激活函数后模型性能有所降低，但去除线性变换与两者都去除的性能相比原模型都有不小的提升，其中两者都去除后的性能提升最大。这表明线性变换和激活函数这两个操作在NGCF应用在推荐系统时并不能带来正面效果。这也许是由于推荐系统数据的稀疏性导致的。此部分对课题内容相关的工作进行简要的分类概括与描述，二级标题中的内容为示意，可按照行文内容进行增删与更改，若二级标题无法对描述内容进行概括，可自行增加三级标题，后面内容同样如此，引文的bib文件统一粘贴到**refs.bib**中并采用如下引用方式 [2]。

3 本文方法

3.1 本文方法概述

LightGCN是作者根据推荐系统的特性重新设计的GCN。具体来说，作者首先研究了GCN各个步骤在推荐系统中的效果，对其做了消融分析。结果发现，当把线性变换和激活函数都去掉之后模型表现得最好，这表明由于推荐系统的特殊性和数据的稀疏性，使用多项式近似的滤波器已经不适用了。为了让GCN能更好的适应推荐系统这一应用环境，作者提出了LightGCN，在这之中作者对GCN进行了精简，删除了不必要的操作并且融合了各层的结果让GCN很好的适应了推荐系统的环境。此部分对本文将要复现的工作进行概述，图的插入如图 1所示：

3.2 信息聚合

在LightGCN中，首先对用户和物品都做一个Embedding，即每一个用户或物品都有一个向量来代表它们。然后作为输入进入到LightGCN中进行节点间信息的聚合，基于上述对GCN的消融研究，在LightGCN中作者直接删去了线性变换与激活函数，因为这两个步骤在推荐系统中对结果并没有正面作用，并且还会加大收敛的难度。每个节点的信息聚合方式如下：

$$e_u^{(k+1)} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_i^k, e_i^{(k+1)} = \sum_{i \in N_i} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_u^k$$

可以发现，LightGCN中删去了线性变换和激活函数，只是简单的将所有与自身相连节点的信息加和后归一化，这个操作不仅在推荐系统中能提升性能，而且减少了计算资源的消耗，同时由于整个模型只有第一层的Embedding有可训练参数，这让模型能很容易就能收敛。之所以删掉线性变换和激活函数能提升模型的效果，也许是因为推荐系统中数据的稀疏性，导致过于复杂的滤波器没有足够的数据让其训练出足够好的效果，相反的，最简单的操作反而是最有效的。在聚合节点的同时，LightGCN会保存每一层中每一个节点的信息，在完成所有层的聚合之后，对这些记录下来的信息按层取均值，作为最终代表对应节点的向量。每一层的信息融合如下：

$$e_u = \sum_{k=0}^{K} a_k e_u^{(k)}, e_i = \sum_{k=0}^{K} a_k e_i^{(k)}$$

在论文中，作者如何处理每一层的数据进行了探讨，发现取均值时就能得到很好的效果，而取其他值或将其设计为可训练参数并没有带来性能的提升。这里是LightGCN里GCN中没有的操作，将所有层得到的节点信息都能进入最终结果中，提高了模型的性能。这里将所有层都考虑到最终层的操作与Head2Toe[2]中的操作有点类似，都是想充分利用每一层的信息作为最终预测的输入。但在Head2Tooe中是将各层拼接起来，在LightGCN中是直接取均值。或许在LightGCN中可以参考Head2Toe的方法让模型有进一步的提升。

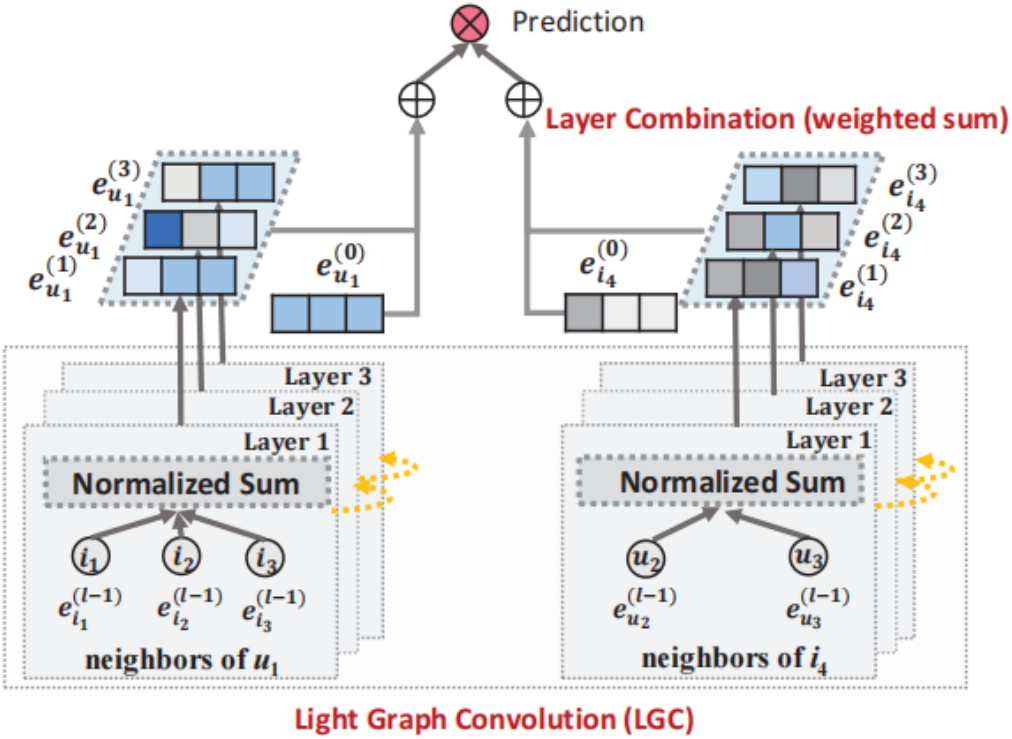

图 1. LightGCN

LightGCN结构如图 1所示，模型最终输出的是用户u与各个物品之间的得分。

$$s_{ui} = e_u^T e_i$$

最后排序后推荐得分最高的物品，这个得分就是对应用户的向量与对应物品的向量之间的点积，相应的，在LightGCN中希望通过梯度下降，使与用户u有连接的节点i得分越大越好，与用户u没有连接的节点j得分越低越好。为了实现这个目标，作者使用了BPR损失来完成这一目标。

3.3 损失函数

为了让与用户相连的物品得分更高，不与用户相连的物品得分更低，LightGCN采用BPR Loss作为损失函数来进行梯度下降。 BPR Loss如下所示：

$$L_{BPR} = -\sum_{(u,i,j)\in D} ln\sigma(f_u(i) - f_u(j))$$

其中i为与用户u相连的物品，j为不与用户相连的物品。使用过BPR Loss，就能通过梯度下降调整每一个节点的信息让其更好的表达其对应的信息。除此之外，为了避免过拟合的发生，作者还加入了L2正则化，控制参数的取值，从而让与用户相连的节点i得分变高，与用户没有连接的节点j得分变低，实现在推荐系统中更加精确，更加高性能的预测。

4 复现细节

4.1 与已有开源代码对比

在本次复现中，我参考了作者给出的源代码，由于本人不是做推荐系统的研究，所以数据处理部分直接使用了源码中的dataloader，并参照了源码中构建模型的方式。本人完成了对LightGCN模型的构建，完成了节点之间信息的聚合，对模型进行训练和预测的代码，并且在LightGCN 的模型当中加入了许多尝试提升性能的操作，例如将各层聚合的比例设置为可训练参数，加入基于Gumbel Softmax的特征选择层，以及参考Head2Toe改变最后聚合各层信息的方式等等。还构建了一个LightGAT想尝试图注意力网络是否同样适用LightGCN的结论，但由于计算资源的问题这个想法未能得到检验。另外，由魏哲巍教授的讲座，还简单尝试了他们的研究成果可否在推荐系统中使用，但结果并不太理想，推荐系统中数据过于稀疏的特性让这些比较复杂的模型都难易发挥他们的作用，有些加入的操作减少了损失但召回率大幅下降了，这表明模型出现了严重的过拟合，归根结底还是因为数据的稀疏性让模型极其容易过拟合，一些复杂的操作可能会适得其反。最后，在众多的尝试当中，唯一对模型性能有提升的是在LightGCN进行信息聚合之前让Embedding中的weight经过一层去除位置编码的Transformer，但由于计算资源的限制，这里的Transformer是分块做的，也就导致有一些节点在Transformer中不能被观测到，同样的，也没办法尝试多层Transformer对LightGCN做信息聚合的影响。在各种尝试中，还对加入了位置编码的Transformer进行了测试，发现最终结果下降了2%的召回率，证明删除位置编码是正确的选择。

4.2 实验环境搭建

此次复现的环境是通过Anaconda构建的虚拟环境，使用了pytorch12.1、python3.10、pandas、jupyter notebook，为了方便开发所有程序都是之间在jupyter notebook上运行。

4.3 创新点

在本次复现中，为了提升模型的性能，对LightGCN增加了一些模块。具体来说，在对用户和物品进行Embedding之后先经过一层Transformer再进行信息聚合。其中Transformer中去除了位置编码以适配模型中的数据构成。在这里，加入Transformer是因为在LightGCN中，为了防止模型发生过平滑，模型聚合的层数就不能做的太深，只有3 4层，这就导致节点不能注意到离自己距离较远节点的信息，即和卷积特点很类似，在聚合操作不够多的情况下不能有效注意到全局的信息，只能关注到局部区域。所以这里引入Transformer就是利用其对全局特征的提取能力，让每个节点都能注意到原本注意不到但很重要的节点的信息。让其能得到全局节点的信息。从结果看，引入Transformer令模型性能得到了提升，这么做是有效的。除此之外，本次复现中还使用了许多方法尝试优化LightGCN，具体尝试如下：

(1) 从如何聚合每一层数据中入手，尝试了包括不同层所占的比例、把每一层所占比例设置为可训练参数还有使用基于Gumbel Softmax的特征选择层来进行层与层之间的聚合。但模型性能并没有得到提升。发现将这里的聚合方式只要不是固定比例最终的训练结果都会大幅下降，也也许是过拟合的原因。另外，参考Head2Toe [1]中的方式，尝试了将各层拼接起来后进行线性变换的方式生成最终的各节点的向量，这里很明显可以发

现损失比原方法是降低了的，但是召回率却没有随着损失降低而提高，这表明这个方法的导致了模型的过拟合，在这里并不适用。同时，本次复现还测试了Transformer的摆放位置对模型新能的影响，发现当Transformer摆放在聚合完成之后时，模型性能会有细微的下降，这也许是因为Transformer不能进入GCN的信息聚合当中，影响了节点得到的信息。

(2) 还写了GAT的代码想尝试在LightGCN中的结论是否适用于GAT但由于计算资源的问题没能进行尝试。

(3) 参考魏哲巍教授的讲座，改变信息聚合时的操作并加深了模型的层数，但仍然出现严重的过拟合现象。

由于推荐系统中数据的特殊性，上述尝试都不能提高模型的性能，只有Transformer通过其对全局特征的提取能力，真正提高了模型的性能。

5 实验结果分析

实验结果如图 2所示，在原模型中，如果与原论文一样训练1000个epoch那么得到的召回率与原文一致，约为18.23%，但如果继续训练至2000个epoch，发现召回率还能进一步提升，取损失最低的一次作为最终的模型，为了保证收敛到最大值，这里训练的轮数都设置的比较大。原模型训练2600个epoch，得到的召回率为18.33%。在这里跟随原论文作者的设置将指标设置为top20，即评分最高的20个内出现用户会选择的物品就算预测成功。为了进一步提升模型的性能，在加入Transformer并且训练2600个epoch之后召回率上升至18.42%，有小范围的提升，这表明通过增加Transformer来加强节点之间的信息交互能有效加强模型的性能，能更好的利用历史信息。

表 1是原方法与改进后方法在不同数据集中的对比，可以发现在Gawalla中本文提出的改进方法会有更好的效果，但Yelp2018中并没有取得更好的效果，这也许是因为Yelp2018的数据过于稀疏，仅通过一层Transformer难以很好的提取特征。

在加入的Transformer中，是去除位置编码的，因为在这个应用场景中节点之间并没有先后次序之分，也就没必要加入位置编码了。为了验证这个操作的有效性，本次复现中对此做了实验，结果发现加入位置编码后最终结果的召回率仅有16.3%，远低于原模型，证明了删去位置编码的必要性。在删去位置编码后，还对使用什么东西代替位置编码能对模型有正面效果进行了思考，并进行了其他尝试。比如对用户和物品都分别用一个向量表示，与位置编码类似的称之为“类别编码”，并分别加入用户和物品中，希望通过告诉模型用户节点与物品节点的不同从而带来更多的信息来增加模型的性能，但实验发现加入之后反而带来了性能的下降，但也不绝对，也许在多层的Transformer中这个编码会产生效果。但总结下来，在只有一层的Transformer中，删去其中的位置编码能让模型获得更好的效果。特别的，由于计算资源的原因此处的Transformer没有添加层归一化，为此在后续的实验中还专门删去了Transformer中MLP后加入层归一化观察效果，结果发现加入后极大的增加了模型收敛速度，但随后损失急剧增大，无法收敛，侧面说明了在这里加入层归一化并不合适，当然，这里与原来相比删去了太多模块，也许是一个特例。在本次复现中这个创新仍然有许多不足之处。由于计算资源不足的原因所以Transformer是分块计算的，这就导致其对模型性能的增强

表 1. 不同数据集中的表现

Dataset		Gawalla		Yelp2018	
Layer	Method	recall	ndcg	recall	ndcg
1 Layer	LightGCN	0.01701	0.01425	0.0551	0.0451
	LightGCN*	0.01687	0.01416	0.0527	0.0433
3 Layer	LightGCN	0.01833	0.01564	0.063	0.0517
	LightGCN*	0.01843	0.01575	0.0626	0.0513

并没有完全发挥出来，并且这里的Transformer只做了一层，也许使用更多层Transformer能更大程度的加强模型的性能。作者针对GCN做了精简和设计设计出的LightGCN，这个想法可否用于其他类型的图神经网络例如GAT呢？这些都是未能取探索到的地方。

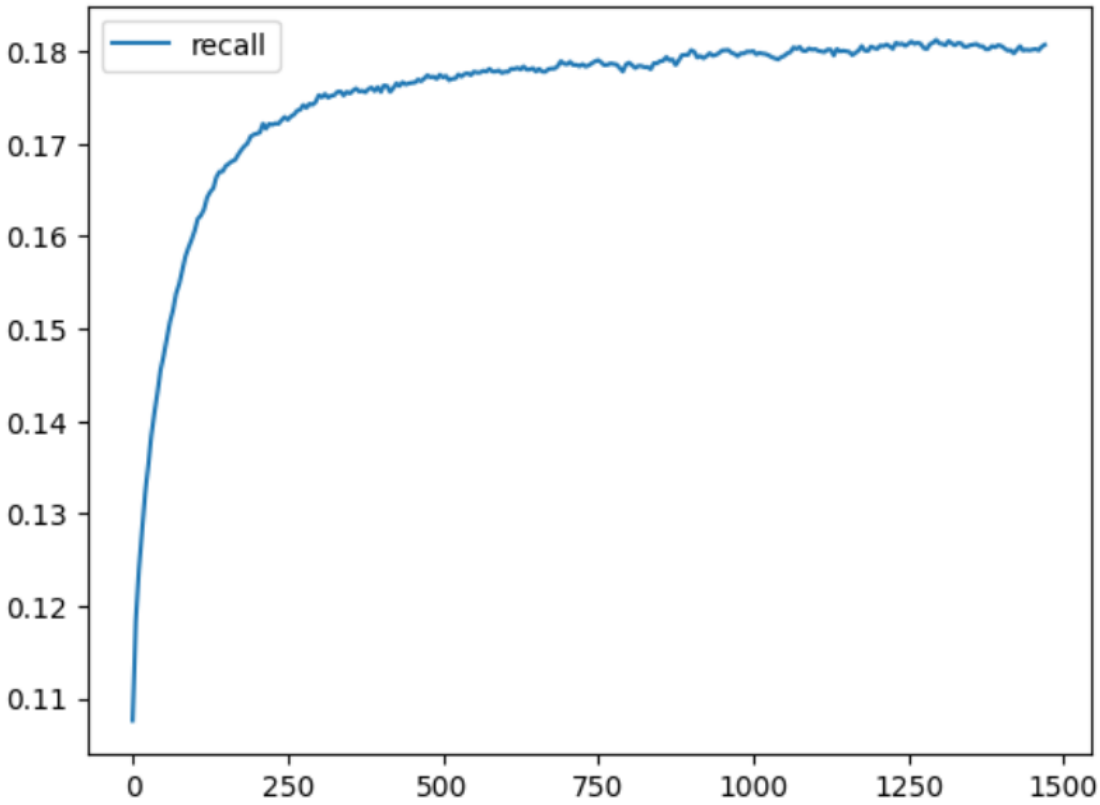

图 2. 实验结果示意

在加入Transformer之余，还对上述的多个想法进行了一一验证，虽然都没能带来性能的进一步提升，但有一个现象十分值得注意。在模仿Head2Toe将融合各层信息的方式从求均值变为拼接后进行线性变换中，实验发现虽然召回率有大幅下降，但损失相比最好的模型居然是更低的，这表明这个方法有着强大的拟合能力，但用在这个场景中会因为拟合能力过强导致发生过拟合的现象，但在推荐系统以外的，数据不那么稀疏的的环境中也许能真正提高模型的性能。

6 总结与展望

在本次复现中，我学习了GCN的原理和结构和它在推荐系统中的应用LightGCN，在这过程中受益颇多，在对源代码的细节进行考究的过程中也逐渐加深了对整个论文的理解。在本次复现中最让我获益的是创新部分，在创新的过程中培养我们的发现新事物的能力，鼓励我们在已有的研究基础上发挥想象力。在这次复现中学习到的知识未来会在我成长的过程中不断帮助我前行。

参考文献

[1] Utku Evci, Vincent Dumoulin, Hugo Larochelle, and Michael C Mozer. Head2toe: Utilizing intermediate representations for better transfer learning. In *International Conference on Machine Learning*, pages 6009–6033. PMLR, 2022.

[2] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[3] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174, 2019.