# 科研总进度

利用异质图神经网络进行开发者推荐并利用辅助信息辅助开发

# 科研思路简述

在软件的更新过程中，Bugs和issues会不断的出现，当被给一个issue请求时，需要推荐一些合适的开发者去解决这个issue。在软件维护和演化这个方向，许多方法已经被提出来推荐开发者，最经典的是基于协同过滤方法，虽然这种方法能很好地根据开发者与issue的交互推荐开发者，但是也存在交互信息过少、开发者信息过少导致冷启动以及推荐结果质量并不是很高。针对这些问题，我们提出一种利用异质图神经网络进行开发者推荐。我们构建开发者-issue-源代码文件的异质网络，利用最新的GNN技术嵌入传播协同信息来推荐开发者。并为开发者提供一些辅助信息，方便开发者开发，例如：针对issue为开发者推荐源文件、利用开发者画像聚类开发者、issue及源代码历史信息。

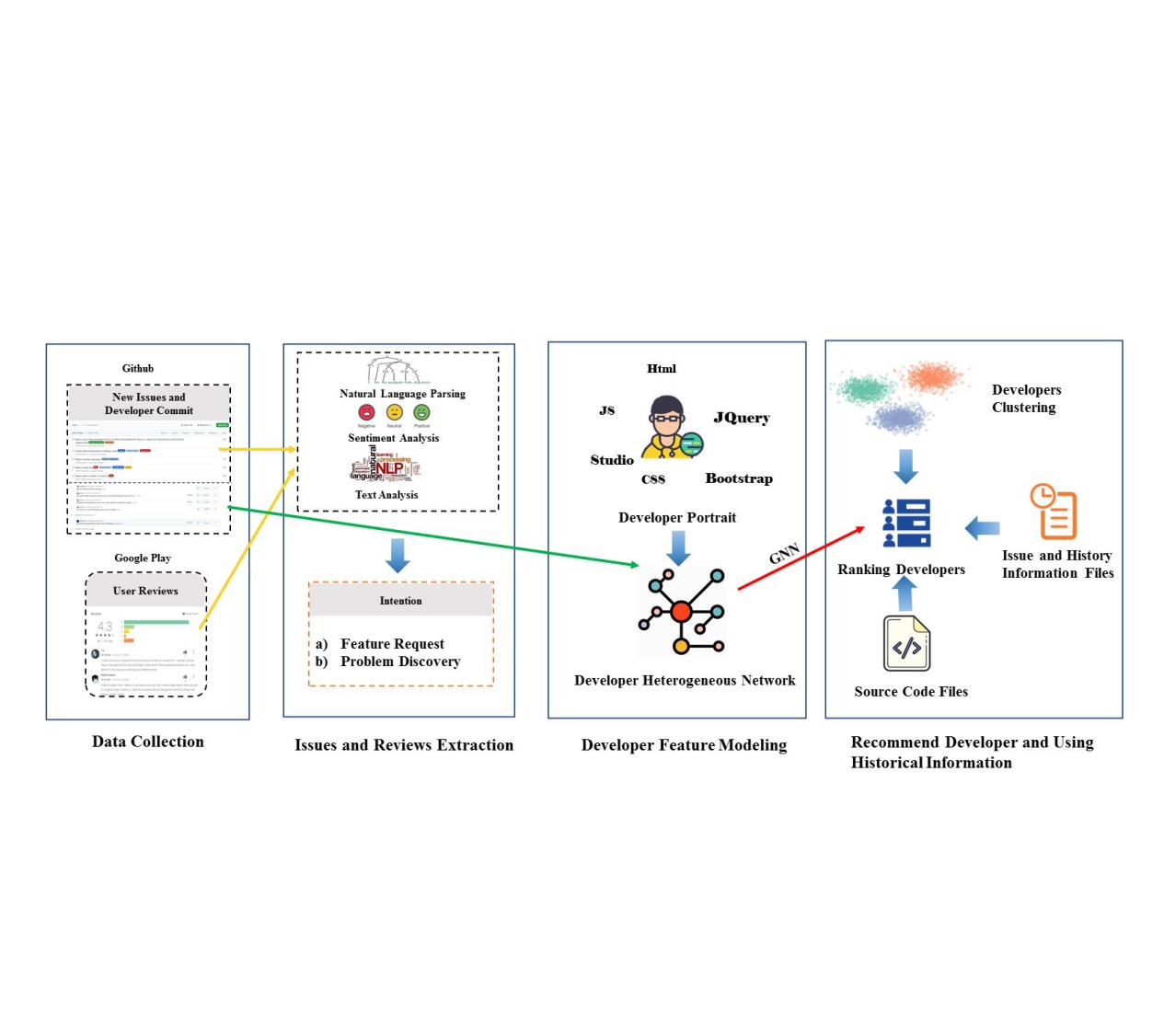
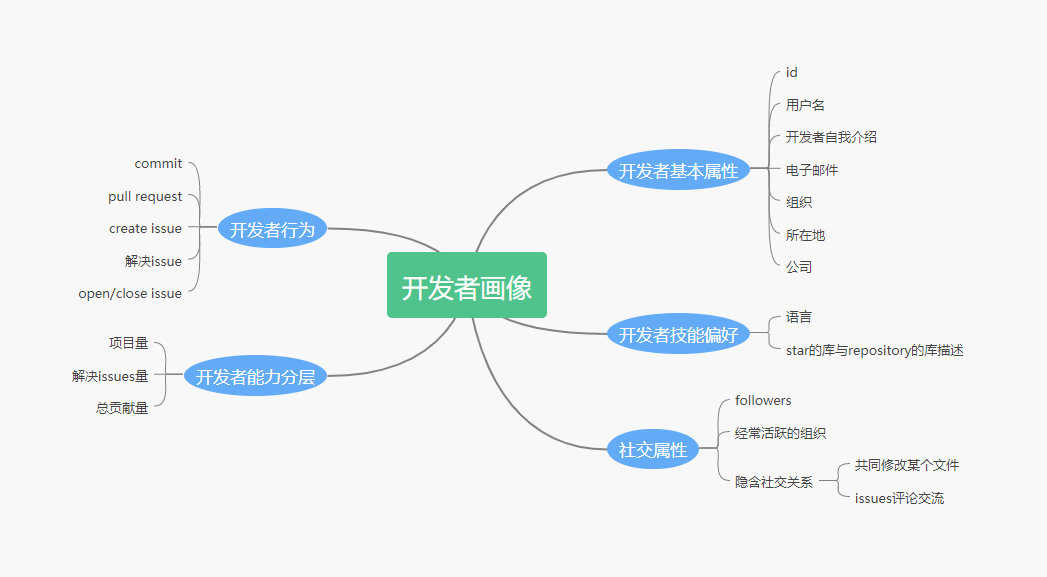


Fig. 1.流程概要

我们提出一个模型HWDR-DPW，Fig 1是我们为issue推荐开发者的模型流程。首先，我们分别从Google Play以及Github收集用户评论以及issue request相关信息，然后一方面，我们利用SURF模型对评论的意图进行分类，SURF是一个成熟的评论意图分类java应用，能够帮助我们找到一些往往包含一些bug或者软件有待改进的功能的建议的评论，而这些在App维护过程中，往往很重要，所以我们收集到问题建模中。另一方面，issue经过NLP技术预处理（分词、去噪声词以及标点符号、主干提取等）。接下来收集软件项目的开发者信息进行开发者画像的绘制，开发者画像为开发者管理人员了解开发者详细信息提供了极大帮助，同时也为我们下一步构建开发者异质网络提供了开发者交互信息。接下来构建开发者异质交互网络，开发者画像和开发者异质网络都属于开发者特征建模，在模型构建好后，我们利用GNN技术进行开发者排序推荐。开发者能力对于我们开发者推荐不是关键点，我们的推荐主要考虑的是开发者个性化开发习惯。但是同时又会存在一个问题，如果推荐的是初级开发者，他们由于对软件项目的不熟悉，所以在解决issue过程中，会遇到一些问题，所以我们从三个方面为开发者开发提供补充信息，包括针对issue为开发者推荐源文件、利用开发者画像聚类开发者、issue及源代码历史信息。这些能很好地帮助开发者了解软件项目，同时方便与相同开发习惯或者基本信息的开发者和解决过类似issue和源代码文件的开发者进行技术交流。

* 1. **开发者画像**

设D={}表示开发者集合，我们定义开发者画像为DP(d)={}，例如我们假设开发者画像为DP(d) = {: “开发者基本信息”,: “开发者技能偏好”, : “开发者能力分层” , : “开发者行为”, : “社交属性”}，其中={“id”，”用户名”，“自我介绍”，“电子邮箱”， “组织”， “所在地”， “公司”}，={“编程语言”， ”存储库描述”}，={“项目量”， ”解决的issues量”，”总贡献量”}，={“commit” ， ”pull request”，“create/open/close issue” }，其中={“followers”， ”经常活跃的组织”， ”issues评论交流”，“共同修改文件”}。

### Fig. 2. 开发者画像

根据项目量、解决issues量、总commit量加权得分为开发者能力，我们假设

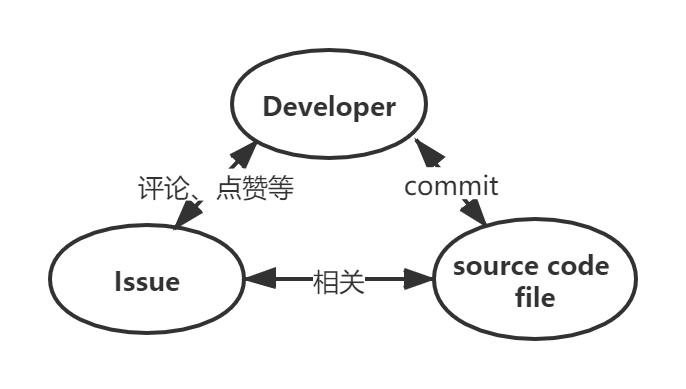
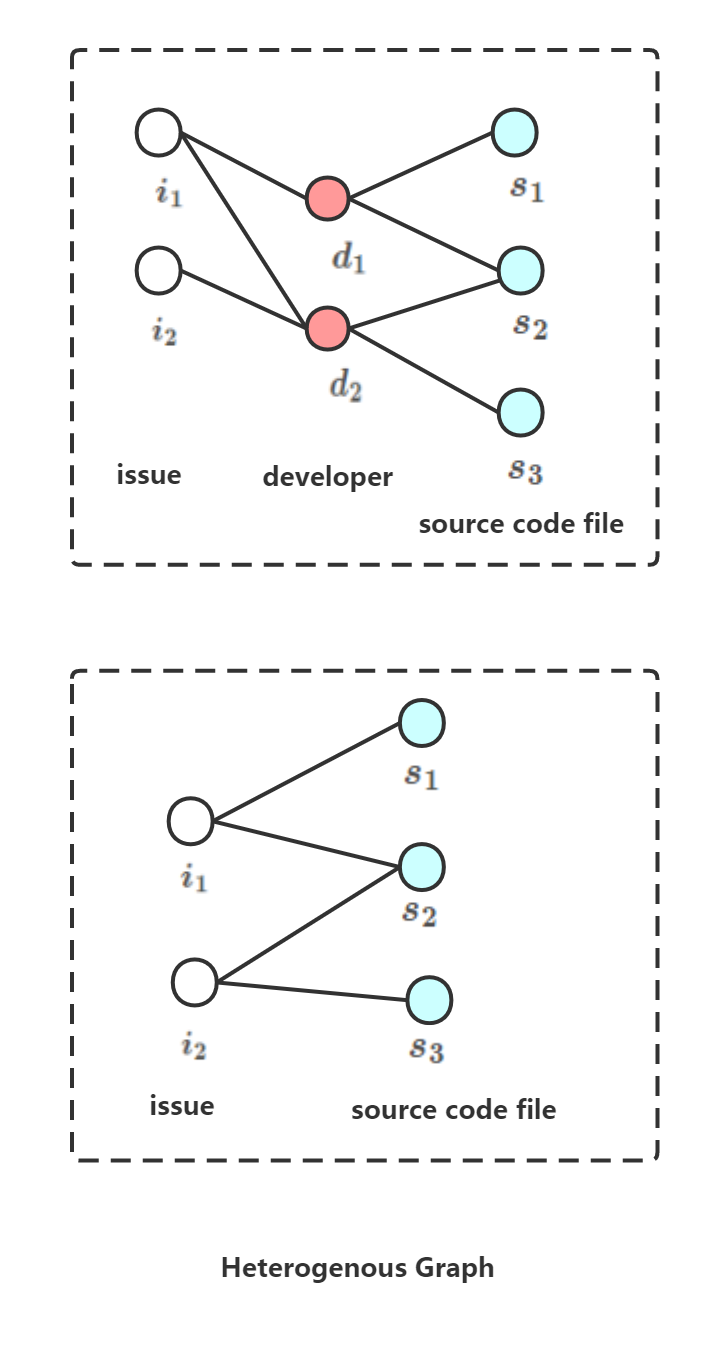
我们将大于ability\_score>0.7的定义为高级开发者，ability\_score>0.4的定义为中级开发者，剩下的为初级开发者。然后将开发者基本信息、开发者能力分层、开发者技能偏好绘制成词云。

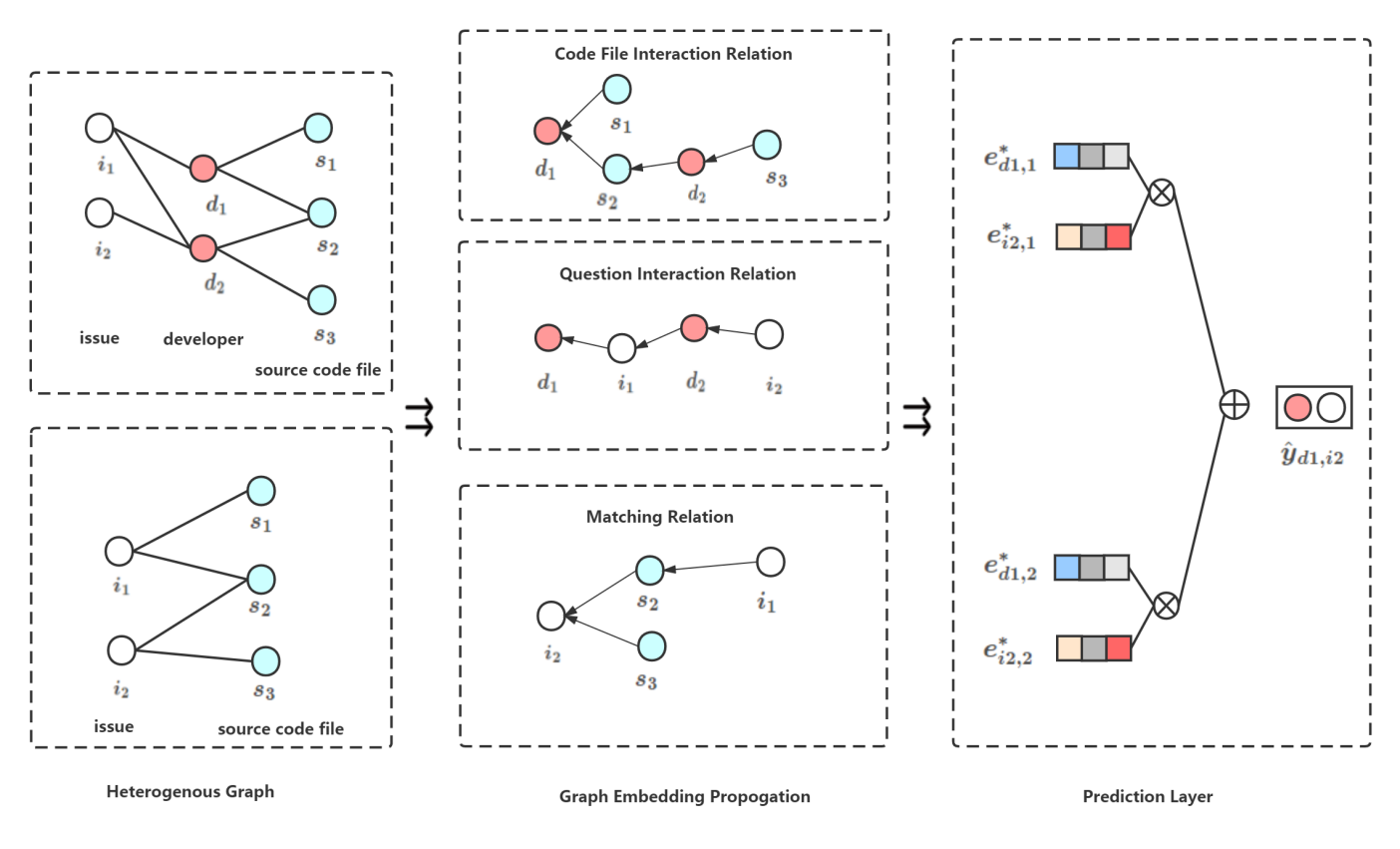
Fig. 2是开发者画像，我认为算是一种比较详细用户特征建模，可以与简历作用相当，包含了开发者详细的特征信息。开发者画像主要包含一些开发者基本信息、开发者技能偏好、社交属性、开发者开发行为、开发者能力分层。可以作为推荐开发者辅助信息，既可以为构建开发者异质网络提供交互信息，又可以为开发者管理加深对开发者基本信息、兴趣偏好、社交属性等各方面了解。

* 1. **异质网络推荐开发者**

开发者异质网络主要提取的是开发者画像开发者行为数据、开发者技能偏好以及社交属性。Fig.3.是一个开发者-issue-源代码文件连接的异质网络图，右边显示的是开发者与issue交互行为、开发者与源代码文件交互以及issue与源代码文件的匹配关系进而构建开发者异质网络结构。我们着重考虑以下开发者行为：开发者对于刚出的issue可以浏览、评论、点赞，表达了一种开发者对于该issue的一种兴趣或偏好。开发者可以提交自己commit，来修改或提交源代码文件，表达开发者一种历史行为信息，隐含开发者能力或开发偏好。一个issue里出现的bug或者功能更新可能与多个源代码文件有关，我们可以通过源代码文件的介绍信息、标识符或者注释与issue进行词匹配，通过这个可以找到issue的相似issue，这很好地解决了issue推荐冷启动问题，当一个新的issue到来时，没有与开发者交互信息，我们通过issue与源代码文件匹配，再通过源代码文件与开发者交互层传递信息，最后推荐开发者。

其中issue与源代码相似度连接又或者是已解决的issue中利用源代码解决用到的文件。

****



**（暂时是这样还在实验中）**

**元路径**

**1. developer-review-issue-review-developer**

**2. developer-like-issue-like-developer**

**3. developer-commit-source code-commit-developer**

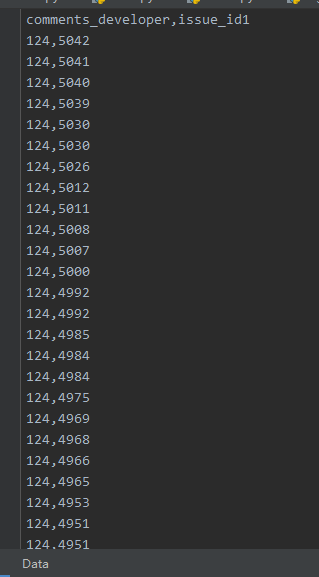
**4. issue- similar-source code**

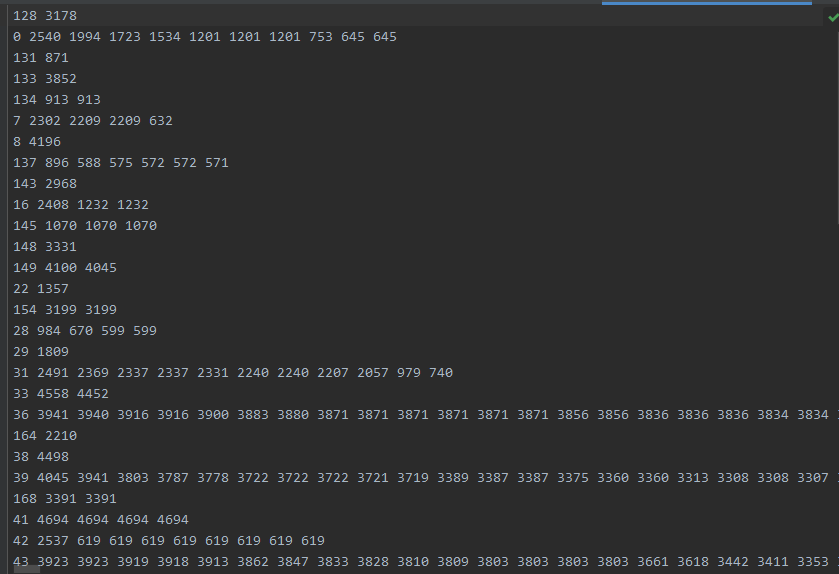
**5. developer- solve-issue**

# 实验

1. 数据预处理

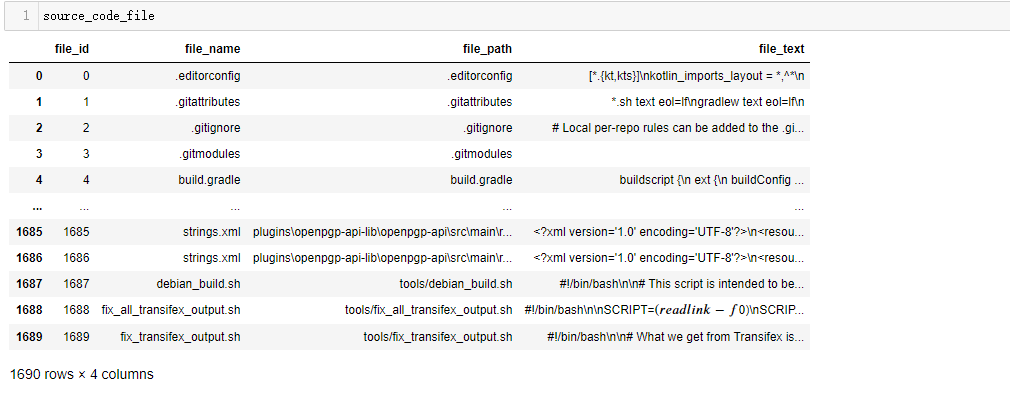
将爬取的有关developer、issue、以及source\_code交互的数据处理成可以导入成图的数据。同时映射到id。

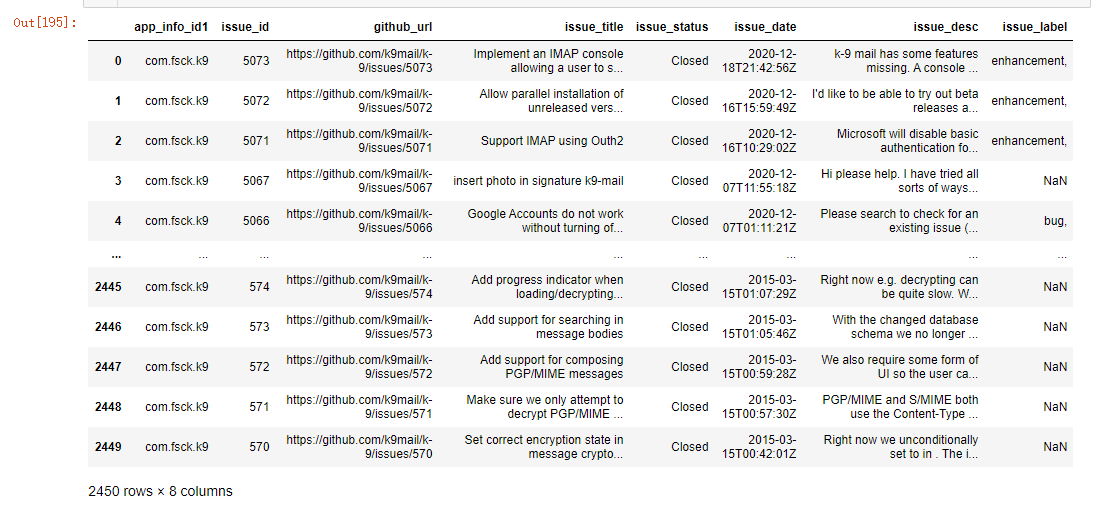




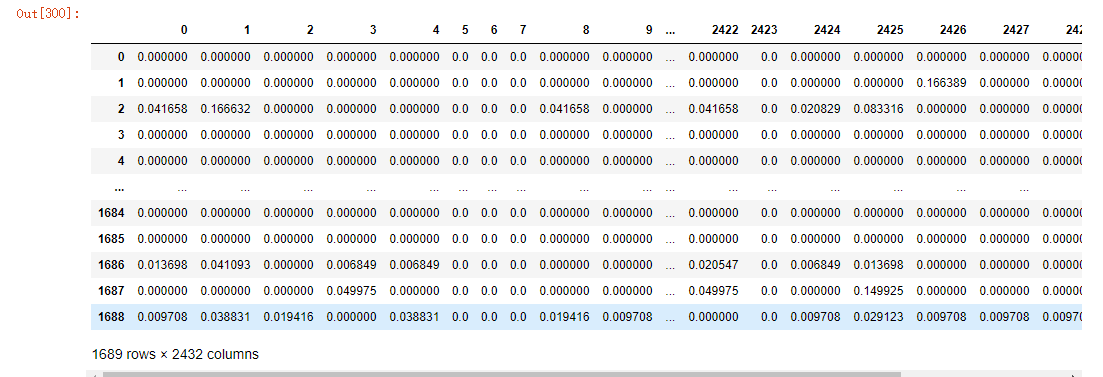
特别的是issue与source\_code匹配关系需要处理一下，提取出源代码文件的源代码和源代码注释与issue的标题和描述进行关联度匹配。







我们需要对描述和源代码NLP预处理包括去掉非英文、转化小写形式、去掉标点符号、去停用词、主干还原词法还原词型。最后计算关联度如下：



1. 数据集：我是用的开发者和issues的数据集，虽然我已经还有source\_code数据集，但是为了简单，先用开发者和issues交互数据集构建二部图交互。训练集用的是69个开发者，与5000多个issues总共3000多交互数据。测试集使用的是为39个开发者做推荐（其中包括新的开发者）推荐issue，总共800多交互数据。主要就是开发者较少。
2. 实验测试的算法模型：打算先用简单模型测试一下，LightGCN属于轻量的GCN算法，还没有用到我的论文里面的算法。不过思路差不多，比之前写的模型简单一点。
3. 实验过程：修改原始论文中一些参数，适当调小。将交互图数据映射成张量图数据，输入到模型中，通过训练集训练，预测出测试集的开发者5个推荐的issues，比较与真实列表的误差。

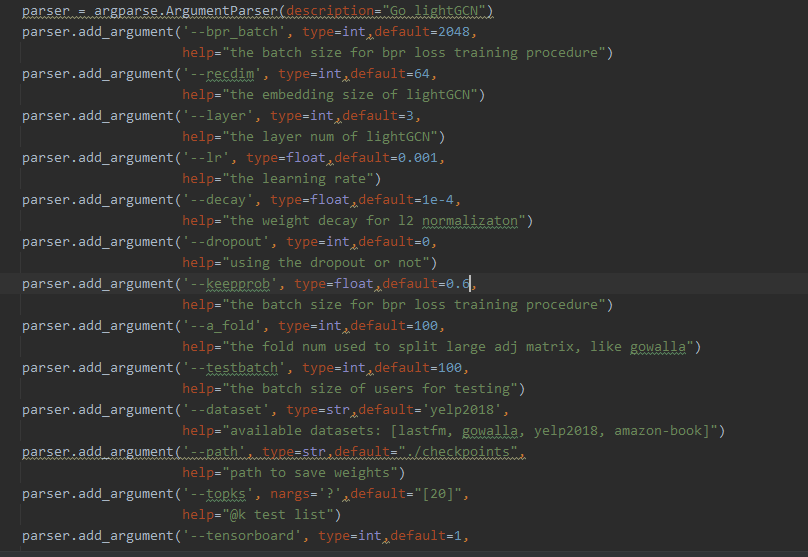


Fig.1. 训练参数

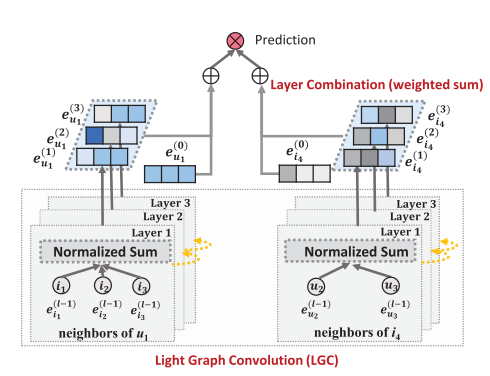


Fig.2. 模型算法

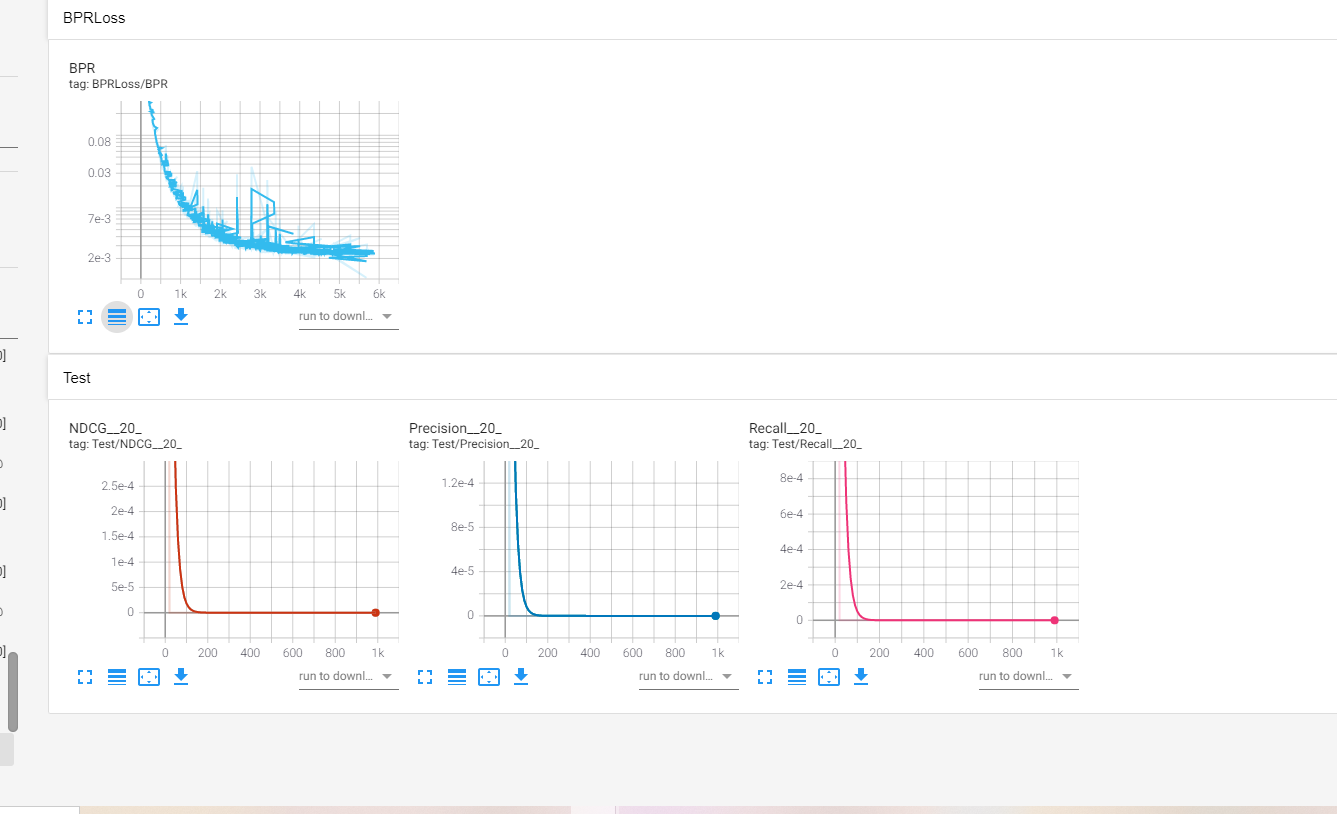


Fig.3. 运行结果指标

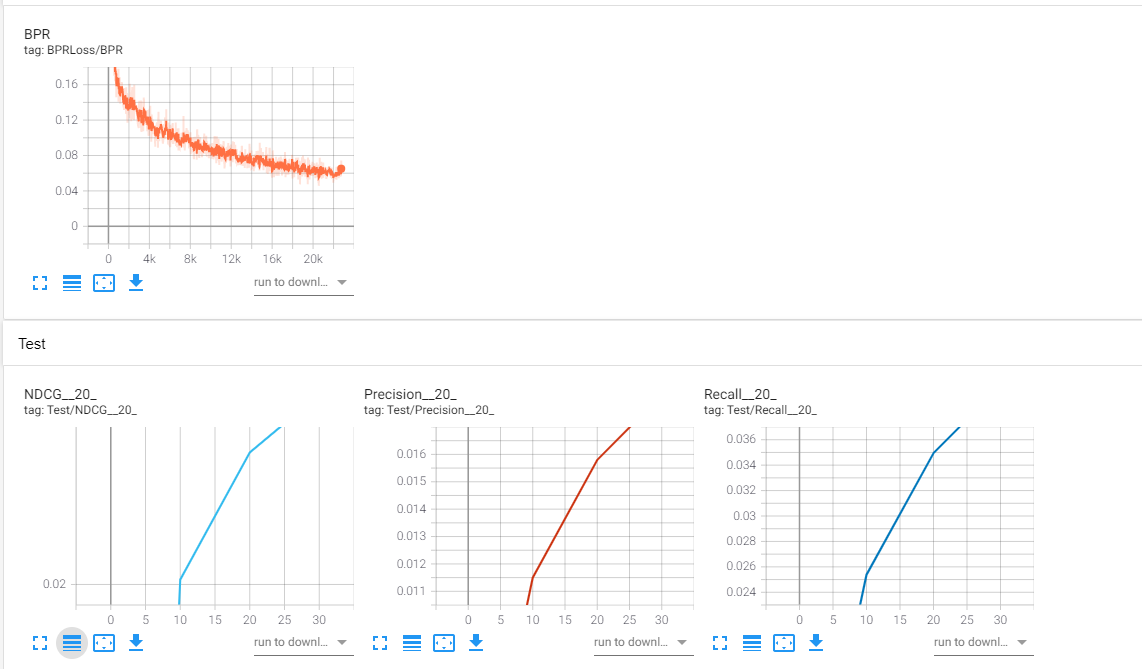


Fig.4. 正常训练结果

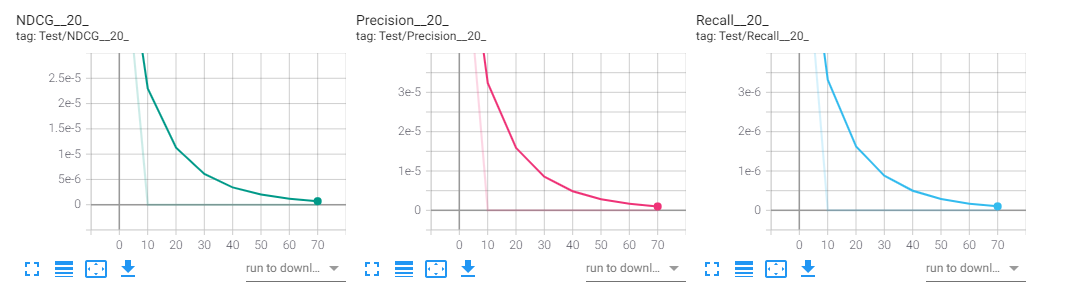
**解决思路：**

**换一个更大的数据集（tensorflow）**

**训练集：58000条交互developer-issues，1063开发者，480000 developer-source\_code**

**测试集：8648条交互developer-issues，579开发者。**

**实验结果：**

****

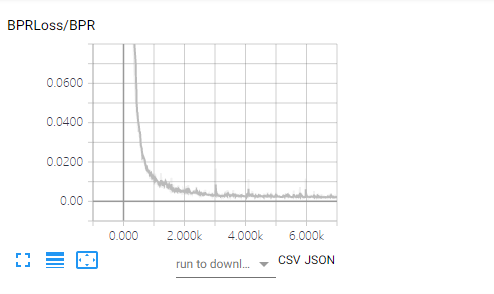
**可以看到相比上面的数据集结果是好了一点，但是趋势还是下降的，我就开始思考是不是模型问题，并不适合我这个场景。**

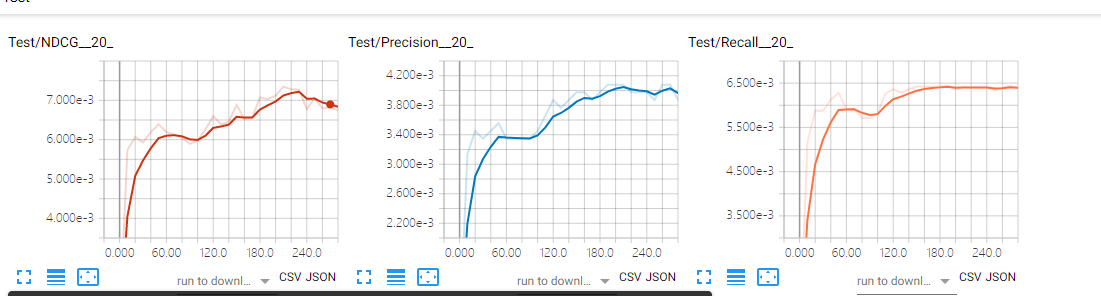
**由于第一步测试的轻量模型的不好，我打算换一个模型，并重新理解思考图神经网络模型，同时也边测试其它类似模型，然后后面也会继续测测带有元路径的异质图神经网络模型。同时，最近也在看ICLR和WWW有关图神经网络的论文，关注最新进展。**

****2021/3/12进度报告****

1. 在检查论文代码运行的时候发现数据集上存在一定的设置缺陷，之前设置训练集的时候以解决的issues为训练集，未解决的issues为测试集。后来训练过程中就发现存在冷启动问题，首先一些新开发者可能只存在于测试集中，训练集中不存在，无法做出很好的预测，其次一些开发者只有几条交互记录，这还需要进一步处理，目前我的做法是过滤掉交互记录少于10的开发者。实验结果有了一些改善如下：

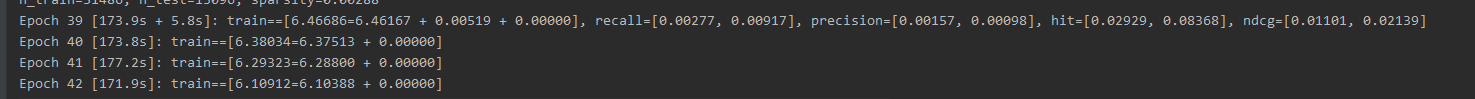
轻量GNN LightGCN：





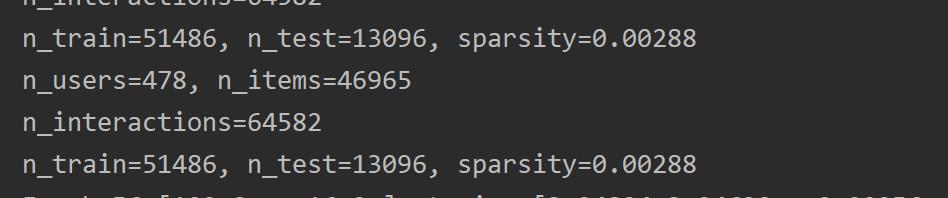
可以看到上图收敛后NDCG可以达到0.008，recall召回可以达到0.007，相比之前训练结果有了改善。

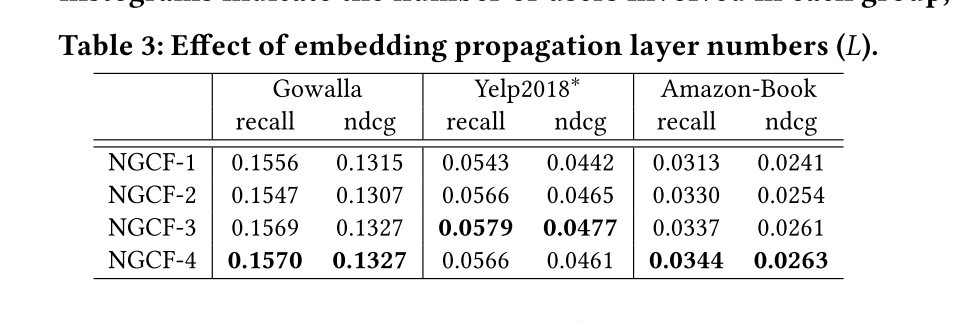
用了图协同过滤算法NGCF:



NDCG可以达到0.02最高，recall召回可以达到0.009.

相比上一个模型有一定的增强。但是我觉得这是我这个数据比较正常的结果了，经过预处理后，数据集又变小了一点点。





由于上面的模型都没用到异质图，而是用的二部图模型做测试，接下来打算研究一下异质图模型的代码测试一下，看看加入其它信息后，结果有没有改善。

1. 论文《基于软件历史库的开发者个性化推荐技术研究与系统实现》

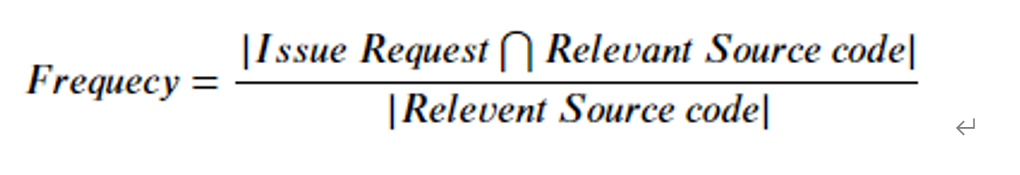
cｏｍｍｉｔ描述，提交cｏｍｍiｔ的开发者姓名和提交时间，修改过相关源码文件路径这三个相关的信息对于推荐比较重要。

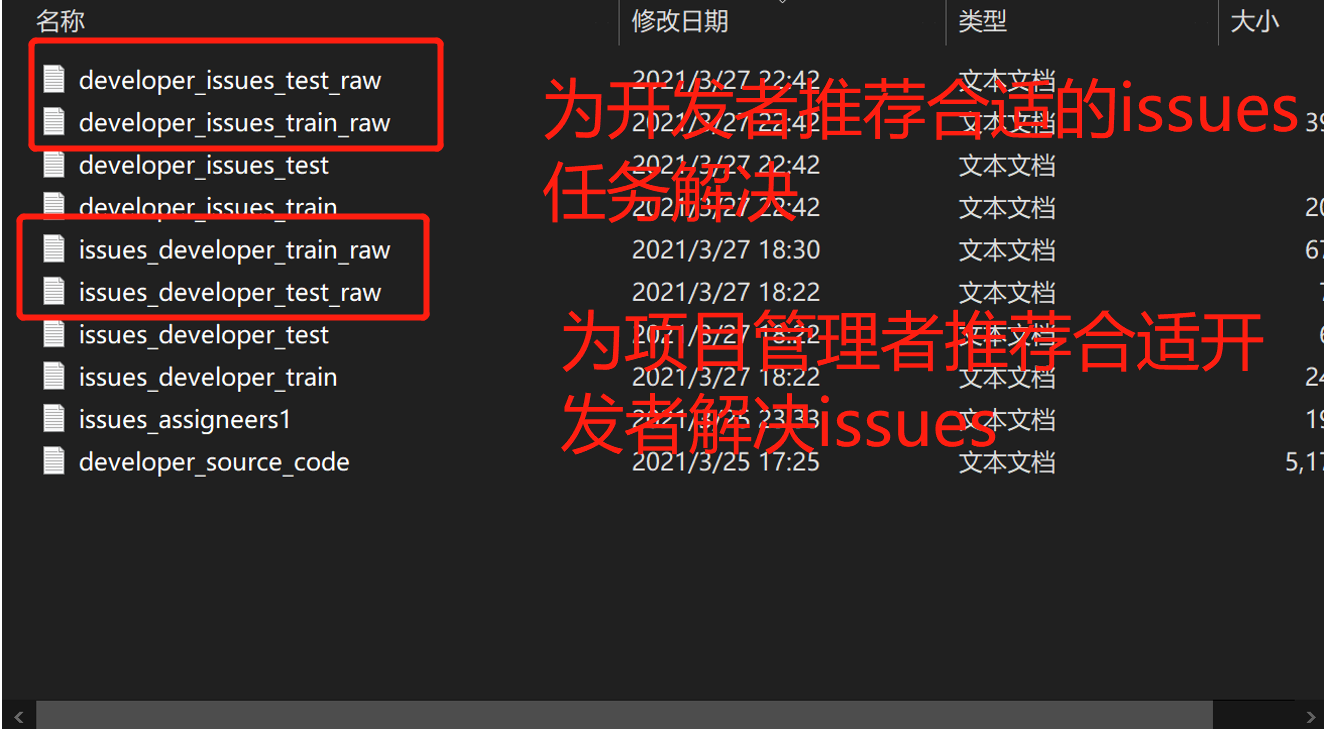
大部分开发者拥有修改以前多次修改过的文件的爱好，并且，以前修改次数越多的文件，他们偏向与修改该文件。所以，应用修改次数作为开发者的排序参因素是有效的.

然后采取的推荐算法是CTM基于主题的协同过滤算法，这种算法相对于小数据还是有优势的，深度学习模型还是比较吃数据训练的。

# 日论文进度汇报

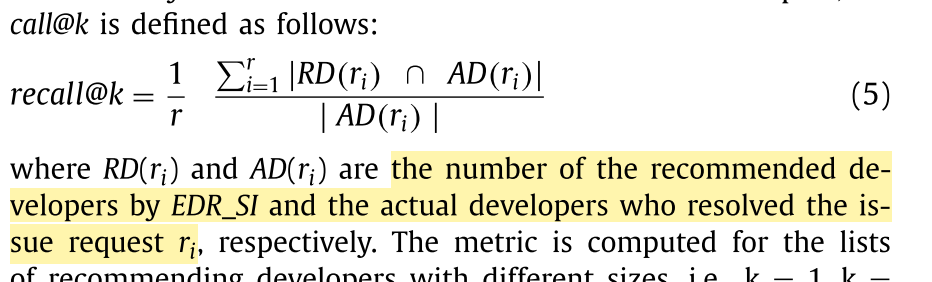
1. 最近的主要时间继续完善数据集，理顺思路，数据集一些处理还是比较画时间的，例如issue与source\_code相关性匹配，例如之前关联度的取值，刚开始取平均值0.05效果并不是很好，发现关联边有几千万条，数据存储较大，经过一定实验发现0.5效果较好，有几百万条，目前还在去掉一些不在issues节点的id关联边记录。（运行的速度比较慢）



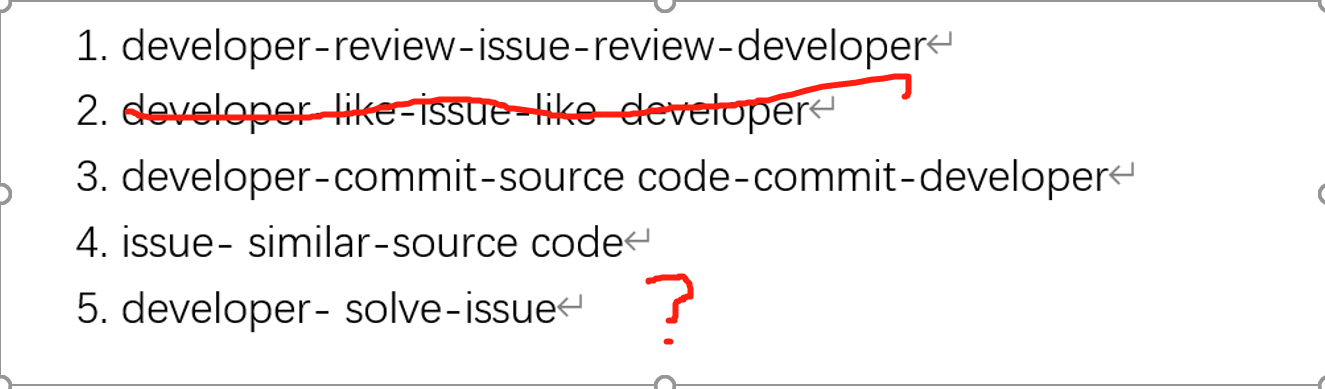


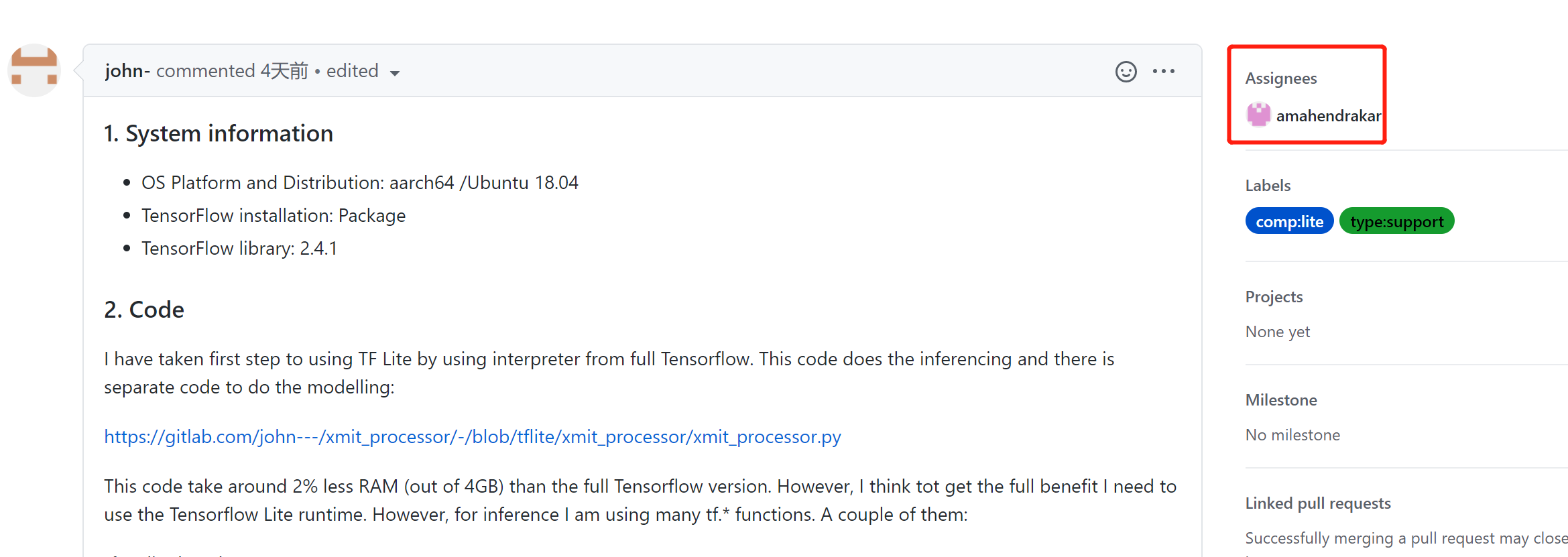


1. 另外在边跑边编写自己的数据集输入的异质图模型的代码，目前在写网络信息的传播，还在边编写边测。在等上面数据集跑好后一起输入到模型里测。
2. 模型的训练目标，主要使用的是开发者评论解决issues，通过观察其实大部分分配给开发者的issues，都不是受让者个人解决的，所以发现github的受让管理人员并不能很好地分配，但是我们也不能忽略已经解决的issues受让者的这些信息，可以作为异质图关系补充，但是我们模型的训练目标还是将评论最为训练的目标，划分为训练集，测试集。但是孙小兵那篇论文，以解决issues作为衡量，具体他没有细说，无法得知。

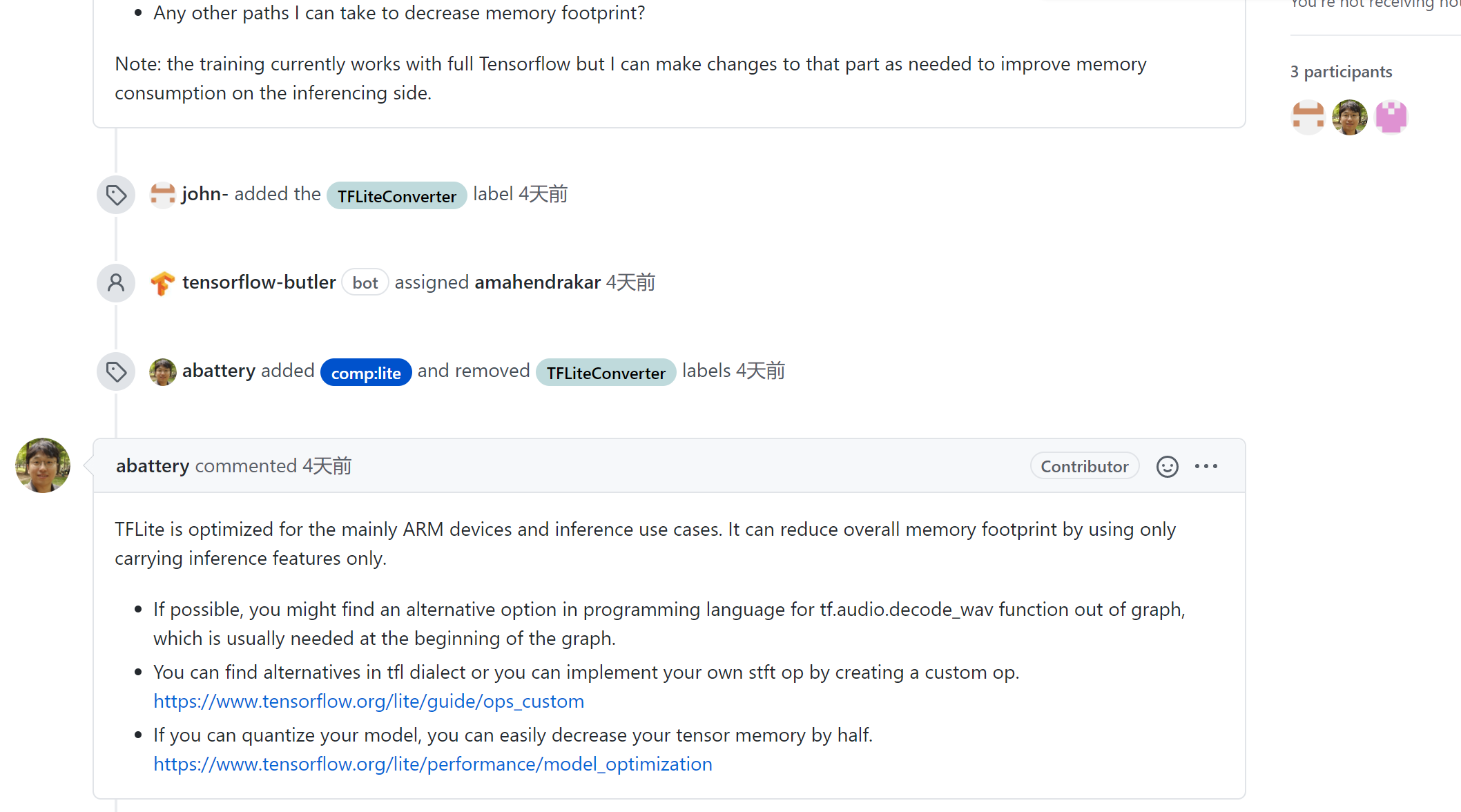


孙小兵的衡量指标





受让者



实际解决问题的人