# 《开源项目分类与聚类》项目报告

报告题目： 开源项目分类与聚类

姓 名： 朱汉伦，王方正

学 号： 10195501411/10195501412

完成日期： 2022/5/25

提供中英文摘要

摘要 [中文]：

Github是当今社会最大的面向开发者的社交平台，对该平台的研究可以有效反映开发者以及其产品这一群体的特征。本次实验将基于阿里云ods层github日志以及用户数据集对不同的开源项目进行特征提取，依据特征对其进行聚类。最后构建模型，以聚类的结果作为分类标签，对模型进行训练，得到一个能对未知领域的开源项目进行分类的模型。

Abstract [英语]

1. 项目概述（阐明该项目的科学和应用价值，以及相关工作进展并描述项目的主要内容）

本次实验对项目进行领域划分可以给出新项目的定位，对于协同推荐，定位搜索有一定价值。

目前对Github用户群体的研究量大于对项目的研究，所以着眼于项目的研究其实现阶段相对较少。

1. 问题描述（问题定义）

如何获取数据，如何对项目进行特征提取，采取怎样的聚类与分类模型以及超参数使结果最优化，以及本次项目存在的问题。

1. 方法（问题解决步骤和实现细节）
2. 特征提取：

根据ods GitHub log数据集的条目来获取特征是本次实验的出发点。根据该数据集的数据说明，比较直接的可以用于特征提取的数据包括repo language和一些文本。两者比较，文本类型的数据量远远大于语言的特征，并且语言类型不能完全反映repo所属的领域，所以我们采用文本作为repo的特征。

(1) 文本特征：

文本当然不能直接拿来使用，但是现阶段对于文本特征提取的研究相当之多，自然语言处理领域的研究尤其多。但是Github的文本有着属于它的独特问题，语义较差。大部分文本都是开发者所输入，文本中很可能包含一些人造词汇，编程语言的语法，甚至是一些URL或是毫无规律的UID，想要精确从中找出正常的语句相当困难。所以本次实验对文本的处理提出了两类方案，一类是提取其最基本的特征例如词频，另一类是按照自然语言处理模型获取文本序列特征。

提取特征之前首先要对数据进行清理。由于Github语义差，所以在词语清洗方面使用了比较严格的筛查。在观察了Github用户用语习惯后，最后决定排除词语长度过段（单个字母）或过长，单个单词包括两个及以上的大写字母的词语。

第一类特征选用tf-idf分数并且使用停用词，保证提取词频的有效性。第二类特征遵从NLP领域的方式，对每组语句截取给定长度，然后根据每个单词生成向量(word to vec)。其一是统计最常见的10000个词语，将这些高频词语生成对应的index，然后使用一个嵌入层(Embedding)将每个词语转换成one-hot向量。其二是依据NLP的基本原理：词语之间的相似性可以通过上下文判断，所以使用n-gram模型加上Laplace平滑获取词语的上下文向量。其三是利用transformer的预训练模型直接输出特征向量。

以上三种方式只是众多方法之一，我们只是选取了自己比较熟悉的方法。

在第一阶段的实验探究中，我们直接使用了文本特征进行聚类然后分类，取得了初步结果。但是这种方式忽视了Github本身作为社交平台的一个根本特征，就是它的社区群体特征。考虑到这一点，我们想出了一个新的方式提取其社区的特征。

1. 社区贡献特征：

最初的设想是找到不同项目之间的社交关系，构建关系网络进行分类，但总归是人类的社交关系更强，并且项目之间的关系难以定义，所以最后选择对用户进行社区发现。

贡献分数向量计算方式：

首先筛选贡献者，对贡献者的社交关系进行社区划分，划分后统计所有类别数N。

对单个项目，计算N个类别对它的贡献分数。每个类别贡献分数是该类别下所有贡献者分数和，再通过一个惩罚函数（缓解登门槛效应）得到结果。单个社区贡献分数计算公式为：

惩罚函数使用：

例如N = 3，第一类有一个开发者提出了issue另一个开发者提出了pr并被merge总计5分，第二类有一个开发者进行了watch共1分，第三类为0分，该项目的贡献分数向量是（两个结果对应两种惩罚函数）：

(2.236, 1.0, 0.0) , (1.792, 0.693, 0.0)

使用惩罚函数是因为等门槛效应，这是一种心理学效应，通常一个人对一个repo做出贡献后，他很可能会继续对其做出贡献。为了防止特征单个维度得分过高，所以使用惩罚函数压制分数的线性增长。另外一个原因是每个repo的平均贡献者数量不会太多，所以贡献分数向量大概率是稀疏的向量，所以使用惩罚函数可以起到一个弱归一化的作用。

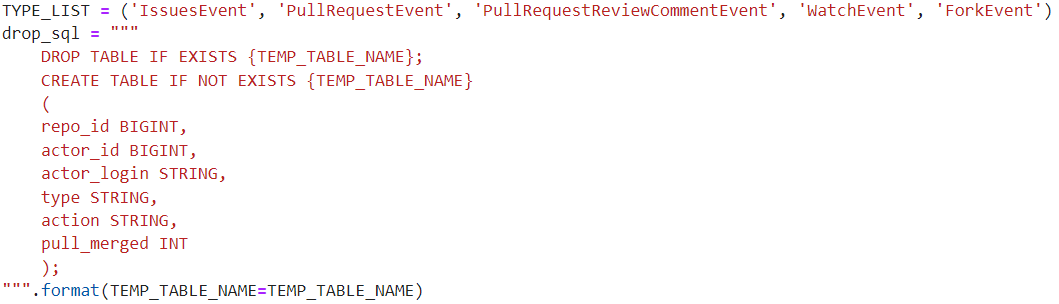
1. 复合特征：

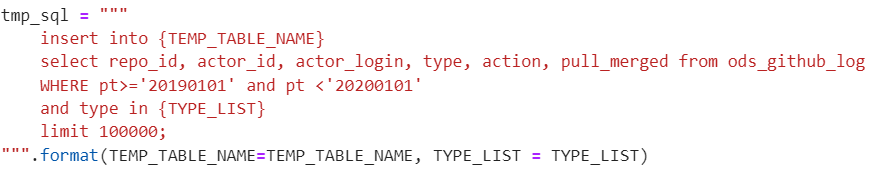
直接将两种特征拼接起来明显是一种不科学的方法，因为没有考虑度量的差异。比如tf-idf值几乎不超过1，而贡献向量的数值大多超过了1，所以两者之间的数值需要一个平衡。我们在拼接两种向量时加入另一个标量w，获取复合特征时把w纳入计算：

这样在聚类时可以将w看做超参数，调整w使聚类效果达到最优。

1. 数据获取：

首先从日志数据集拉取十万条数据并创建临时表，以便于提取文本以及社区关系。

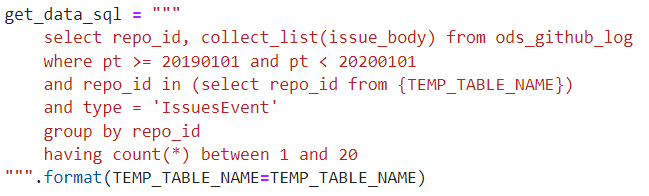




根据日志数据集的操作类型，我们认为IssuesEvent, PullRequestEvent等五种操作类型为有效的贡献操作类型。同时收集repo和贡献者的基本信息。

总计获取34327个repo，13246名贡献者。

1. 文本获取：



提取所有IssuesEvent的操作，获取其中的文本和其对应的repo，最终有24287个repo能找到文本数据。每个repo提取其前十个Issue的文本作为最后的特征，因为部分热门repo的Issue数量会非常多。

1. 贡献分数向量获取：

每种贡献类型的分数：

Issue weight = 1.0

Open Issue weight = 2.0

Pull request not merged weight = 2.0

Pull request merged weight = 5.0

Pull request review comment weight = 4.0

Watch weight = 1.0

Fork weight = 2.0

用户社交网络建设：

根据用户的following关系，建立社交网络。

Following关系可以从ods github users里获得，但该数据集本身信息不全，之前拉取的13246个用户中只有1490个用户在该数据集中有记录，信息损失非常大，所以我们采用其它方式获取剩余的用户信息。

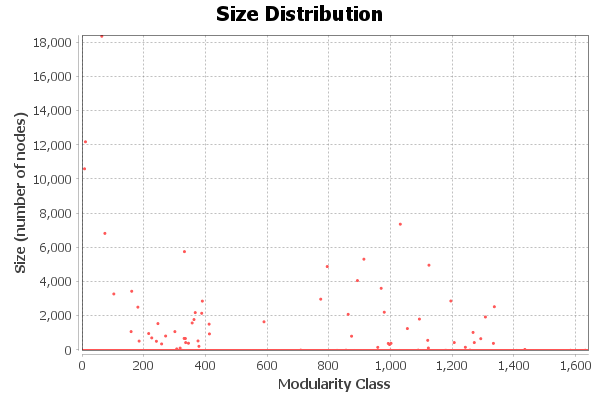
PyGithub是Python中面向github信息的第三方包，我们可以通过该API向github发出request。当然大部分平台对于频繁的request都有限制，PyGithub给游客用户的限制是每日60条，如果使用已有用户的token可以提升到每日5000条request。剩余11756名用户的信息一共花了2天用3个账号来完全获取，因为要获取一个用户的following信息事实上需要2次request。

在获取跟随关系时可能遇到异常点，包括有的用户可能已经注销，有的用户可能更改了ID，这些用户只能作为孤立点考虑。另外一个就是交际花，有的用户的关注数远远超过一般用户，这种时候要考虑异常用户带来的噪声，我们采用的方式是设置一个阈值，关注数超过这个数值的用户我们不纳入考虑或者只提取部分信息。

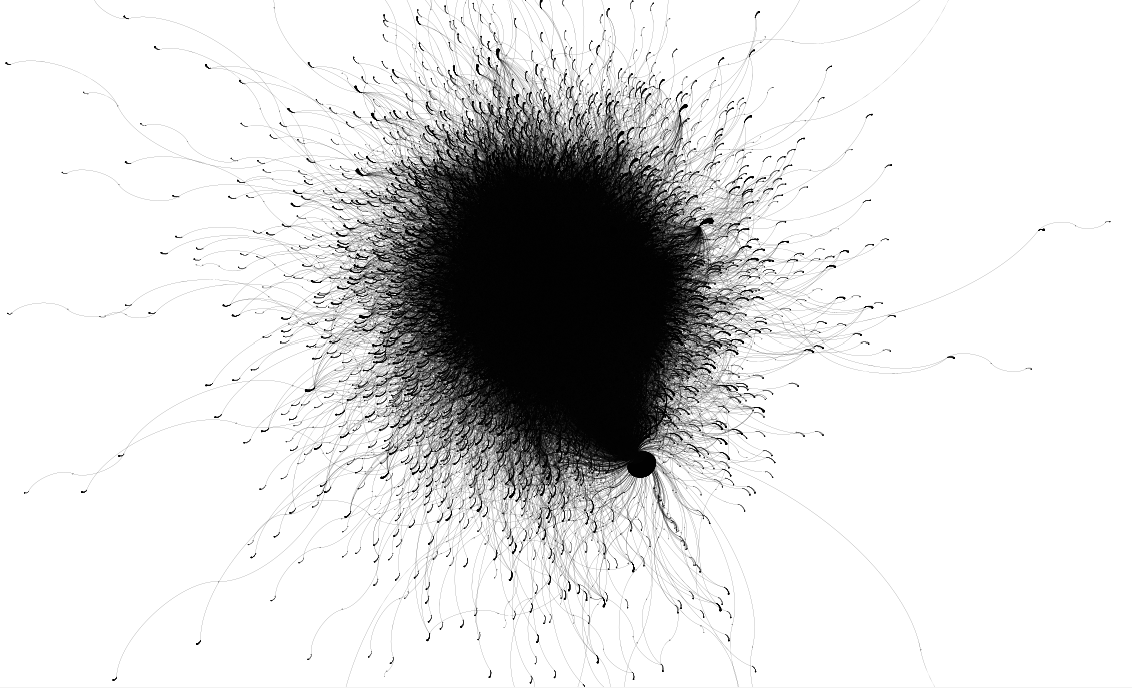
最后是建立网络，本次实验考虑的节点是当前的13246名用户和他们的following，总计十五万个节点，二十三万条边，主要是考虑到可能不同贡献者之间有共同的following，他们之间的follow关系可能非常稀疏。不去继续递归地获取following信息是因为工作量太大，需要提交的request数目大幅增加，并且社交网络规模再次增加，没有必要。

社区发现使用Louvain算法，单机上在十万级别的网络运行这个算法无论是时间还是空间都不允许。Spark分布式上的Louvain算法时间消耗仍然不太理想。

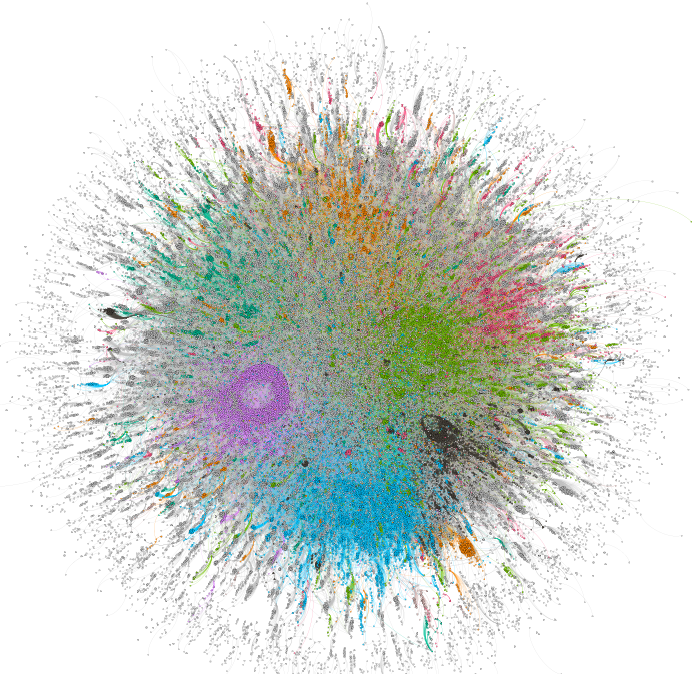
在我搜索社交网络可视化工具时找到了Gephi这一工具，它完美的解决了上述问题。同样使用Louvain算法，Gephi运行的时长只有不到一分钟。最后Gephi算法划分出1638个模块，模块度0.700。所有的模块规模服从长尾分布，只有前100个模块有着有效的规模，后面的几乎都是孤立点所属于的模块。



完成社区发现后，整个社区网络可视化结果如下：



上图是没有分类的网络图，可以看出这个图结构的趋势。因为运行算法时使用了随机初始化，如果算法效果较差的话上图应当是属于正态或者圆形的形状，所以Louvain算法是划分出了部分有效区域的。



上图是带标签的图，灰色部分均是规模较小的社区，有颜色的部分是有效的社区。

需要提及的是无效的社区并不代表这种特征会带来太大的负面效果，他们通常对之后聚类的效果影响较小，比如说一个贡献分数向量只在一个无效社区上的维度上有值，那么它与其它向量计算距离时这个维度上的数值几乎不会影响距离大小的判断。

1. 聚类与分类：
2. 聚类

在第一阶段的尝试中，我们直接使用文本的tf-idf向量进行聚类，使用方法KMeans，距离使用欧氏距离。由于没有聚类的真实标签，我们无法计算TP, TN等值，所以没有使用常规的聚类评价手段，而是使用了分类模型观察分类模型最后在测试集上收敛的准确率来评判聚类结果的好坏。因为模型的训练依赖于训练数据集，训练与测试数据集的优劣会反映在模型上，部分规律性较差的测试用例很难预测正确。但实际上这种评价方式会受到模型本身的影响，并且没有一个数学上的证明，所以合理性有待提高。

第二阶段我们使用了平方根惩罚的贡献向量复合特征进行KMeans聚类，距离仍然使用欧氏距离。在调整了超参数之后的确在模型上取得了比第一阶段更高的准确率，但是聚类的方法仍有问题，欧式距离在高维空间下使不适用的。

最后阶段综合了之前所有的经验，使用内部指标轮廓系数作为评价，距离使用余弦距离，复合特征的惩罚函数使用对数惩罚，聚类方法仍选择KMeans。以轮廓系数为标准调整超参数，最终获取了比较合理的超参。对比三次尝试的轮廓系数指标，第三次的评价远远高于前两次。

1. 分类：

分类所选取的所有特征均来自于文本特征，这样适用于文本分类的模型都能派上用场。

针对两种文本预处理方式，第一种特征我们使用MLP多层感知机作为分类器，第二种特征使用基本的NLP分类模型textCNN以及textRNN。不同阶段使用的不同方法在实验结果中会进行分析。

1. 实验结果（验证提出方法的有效性和高效性）

阶段一：

KMeans聚类K = 10，使用tf-idf向量聚类，距离使用欧式距离。

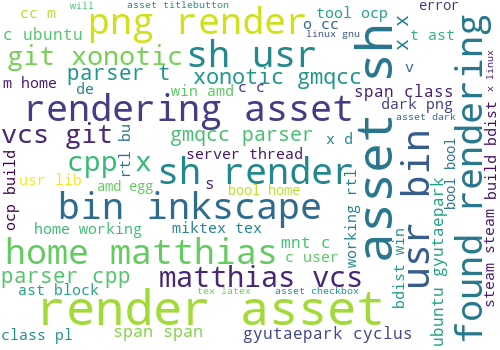
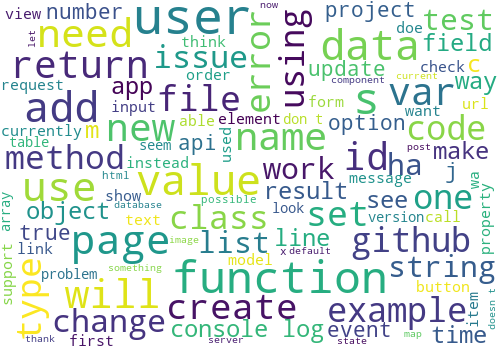
聚类结果轮廓系数：0.021

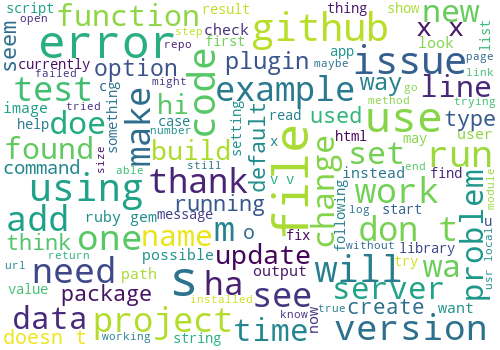
使用数据tf-idf向量，分类器使用MLP。

优化器使用adam，训练5个epoch，使用五折交叉验证，训练集3200条，验证集800条，测试集1000条。

十分类准确率：87.10%

十分类中三个较大类别统计词频生成的词云：



阶段二：

KMeans聚类K = 10， 20，使用平方根损失复合特征，平衡特征权重

w = 1e-02，距离使用欧式距离。

聚类结果轮廓系数：K = 10，0.026；K = 20，0.027

使用数据tf-idf向量，分类器使用MLP。

优化器使用adam，训练5个epoch，训练集19428条，测试集4857条。

十分类准确率：89.66%

二十分类准确率：86.47%

阶段三：

KMeans聚类K = 10，15，20，25，使用对数损失复合特征，平衡特征权重w = 1.0，距离使用余弦距离。

聚类结果轮廓系数：K = 10，0.10；K = 15，0.20；K = 20，0.27；

K = 25，0.16。

使用数据tf-idf向量，分类器使用MLP。

优化器使用adam，训练5个epoch，训练集19428条，测试集4857条。

二十分类准确率：98.44%

使用Embedding编码词向量和bi-gram + Laplace Smooth编码上下文向量。

分类器使用textCNN，textRNN。

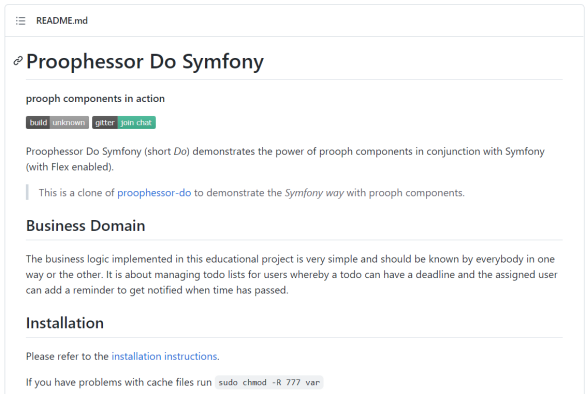
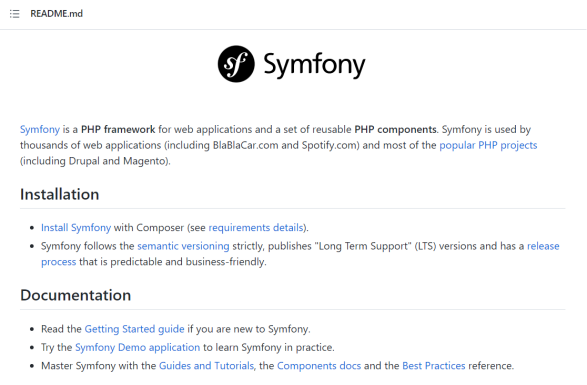
结果较差，仅仅能保证训练集上准确率收敛，测试集上准确率最高30%。

较差结果分析：Github语义较弱，用户输入的文本有大量人造词汇，提取的文本甚至可能是URL，所以文本序列前后上下文关系薄弱，这违背了自然语言处理的一项基本定律：相似词语的上下文通常也比较相似。因此更为简单直接的词语频数效果更好。

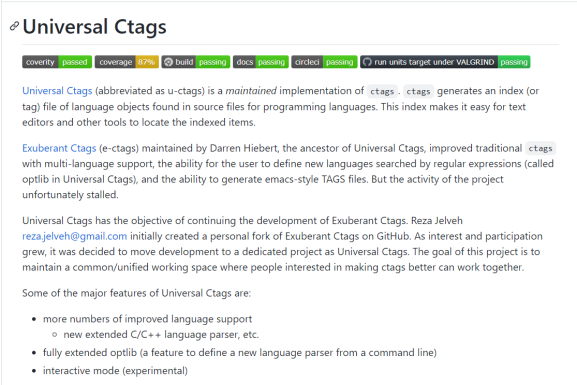
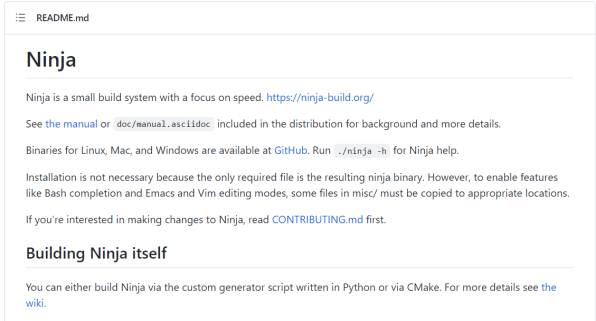
阶段二，三聚类结果示例：

被分到同一较小类别的repo：

两者皆是使用PHP开发的网页前端项目，第一个repo是基于第二个repo开发的。

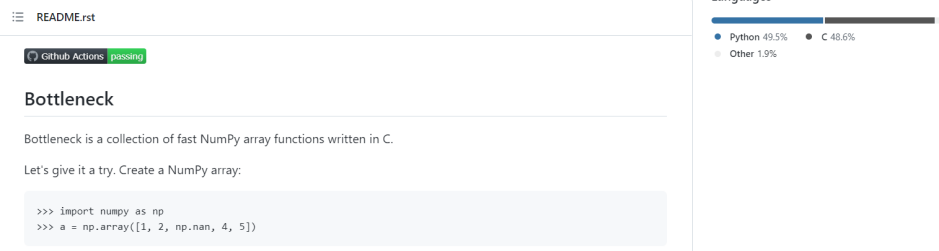
被分到同一较大类别的repo：

两者有一定差别，但都是关于计算机系统底层的项目。

较差结果示例：





其一是用于文档管理，其二是用于计算的项目。

错误可能原因：用户使用相同操作系统比如Linux。其中的命令词语大多是有一定频率的低词语，容易被识别成有效的特征。即便tf-idf向量对这一类词频有限制，但考虑到向量本身比较稀疏，有效的特征也较少，仍然有出错可能性。

结论（对使用的方法可能存在的不足进行分析，以及未来可能的研究方向进行讨论）

项目意义：

对于已经聚好的类别，可以依照词频统计等方式人为对其打上标签。

对于新建立的仓库，提取其自然文本特征（不止有Issue文本），利用分类器得到其领域标签。在用户搜索时可以限定标签搜索，可以有效增加搜索的准确度以及缩小范围，但可能损失召回率。利用同一领域内的项目可以作为推荐参考。

项目流程总结：对仓库进行特征提取，包括文本特征和贡献来源特征，并结合两种特征对其进行聚类。聚类完成后通过聚类结果获取标签，构建分类模型，利用模型对未知标签的仓库进行分类。

项目问题总结：

Github内的文本语义差，存在大量人造词汇，特征提取困难。

训练数据集大小有限，只能在特定的时间和范畴达到最好的效果，对未来会新出现的类别效果较差。

Github用户社区关系获取困难，大量用户没有社交关系，难以构建一个完整且有效的社交网络。部分社交关系获取困难，包括两名用户曾在同一组织的关系等。

对用于分类的仓库有一定要求，不能普及所有的仓库。

组内分工：

10195501411朱汉伦：负责方法构思，部分代码编写，项目汇报。

10195501412王方正：负责方法验证，参数调整，部分代码编写。

水杉码园：

https://gitea.shuishan.net.cn/10195501411/SC2022-10195501411