Github 热门项目的分析与预测

李明，郭腾

1 引言

社交协作编程平台是一种将开发人员聚集在一起，并支持开源软件的生态系统。例如来自 Apache Foundation 的许多项目就是由多个志愿者共同支持的。GitHub 是一种流行的存储库托管服务平台，它允许多个开发人员对同一个代码做出更新并在软件开发上进行协作，这将会大大提高开发团队的效率。更重要的是，许多成功和受欢迎的项目现在都在 GitHub 平台上进行托管，例如 Homebrew 和 Docker。现在，许多开发者都通过 Github 来寻找热门项目，追逐开源社区的潮流。

2 问题描述

在本文，我们对开源软件如何变得流行这件事感兴趣。我们希望使用一些技术，例如 degree centrality、PageRank 等来分析热门项目热门的原因，同时选取一些合适的特征来划分热门于非热门的项目。我们最终的目标是思考是否可以通过建模去预测一个项目是否是受欢迎的，这将减少开发者大量的寻找热门项目的时间。

3 方法

3.1 目标

我们首先基于一些网络分析方法 (degree centrality 和 pageRank) 交叉比较每个事件构建的图的计算结果并得出一个这些前 10000 项目之间的相似性。接下来，我们将会进一步观察这些排名靠前的项目，探索是否有某个属性是预测项目受欢迎程度的较好的测量指标。

机器学习算法是一个预测 GitHub 项目是否受欢迎的更复杂更准确的方法。本文的目标是基于旧项目事件数据，预测一个项目的流行度。如果这个方法效果很好，我们不需要知道 PullRequest 数 (PullRequest 在后面的内容中将会被证明是划分热门项目最优的特征) 就可以预测这个项目是否受欢迎并且值得关注的。

3.2 数据集

我们使用 GitHub 存档数据 (http://openinsight.x-lab.info/superset/sqllab/) 作为我们的数据源。我们收集 2020 年 1 月 1 日至 2020 年 1 月 6 日之间所有公开的 “Fork”、“Watch”、“Pull Request” 和 “Comment on Issues” 事件来支持我们的图分析。我们还讨论用 logistic regression 分类得到哪个 GitHub 项目很受欢迎。对于这一部分，我们收集从 2020 年 1 月 1 日到 2020 年 1 月 12 日的数据。我们在可用数据上选择了更大的时间窗口来训练我们的预测模型并实现达到较高的训练精度和测试精度。表 1 总结了 4 个图的一些基本数据。由于 watch 和 fork 相对于其他的操作要简单一些，所以它们的图会覆盖更多的项目，所以我们可以从途中看到这两个图的数据要比其他两个操作高处一个数量级。

3.3 数据处理方法

我们采用 igraph 包来进行构图以及执行相关的算法，igraph 是以最大的节点号码进行构图。我们以项目为节点，如果这一天至少有一个用户对两个项目都做了同一种 Event (PullRequest，Watch，Fork，IssueComment

图	顶点数量	连接数量
Watch	273854	52284333
Fork	145922	9998595
IssueComment	75728	746445
PullRequest	82538	1858879

表 1: 图数据的大小

中一种），则视这两个项目在这种 Event 对应的网络里是有连接的（无向图）。然而，Github 上的项目数巨大 (超过四千万个)，如果直接进行构图，将会造成内存的崩溃。我们首先删除掉孤立的节点，得到剩余的至少有一条边的节点的集合，计作 S。接着，由于 igraph 是根据最大节点的号码来构图的，我们建立一个映射：

$$f : S \to \{1, 2, \cdots, |S|\},$$

对于 $x_i, x_j \in S$，如果 $x_i < x_j$ 则 $f(x_i) < f(x_j)$，因此 f 是一个双射，可以完整的保存信息。例如，集合 $\{2, 5, 40\}$ 会被映射成 $\{1, 2, 3\}$，$f(2) = 1, f(5) = 2, f(40) = 3$。在算法执行上，我们首先建立一个字典（dic）保存映射 f。我们使用集合 $\{1, 2, \cdots, |S|\}$ 进行构图与执行图算法。完成算法之后，我们利用 $n_i = f^{-1}(i), i = 1, 2, \cdots, |S|$ 得到相应项目 id 。

3.4 Logistic 回归

我们执行 Query 来查询不同项目在不同事件的计数，按事件分组，根据项目 ID 在 PullRequest 的计数，标记前 10% 的项目作为有影响的项目，其余项目作为无影响的项目。即标记顶部 10% 的项目为阳性（“1”），另外 90% 的项目为阴性（“0”）。接下来，我们将这些数据随机分成两部分：70% 的数据作为训练数据，30% 数据作为测试数据。对于每个项目，我们检查它是否是在以前的标记数据中被标记为有影响力的项目。logistic regression 的目的是识别有影响力和无影响力的项目。这是一个二元分类问题。逻辑回归是一种广泛使用的为此目的的分类算法。我们使用带 sigmoid 函数的

logistic regression，使用随机梯度下降 (SGD) 来优化模型，这是因为 SGD 非常适用于大型训练数据集。该规则具有以下方程:

$$l(\theta) = \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))$$

我们计算训练正确率和测试正确率。如果模型过拟合，我们应该添加正则化项以避免此问题。我们计划同时使用 L1 和 L2 正则化。随机梯度上升修改为：

L_1:

$$\theta_j = \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)} + \frac{\lambda}{m} sign(\theta_j),$$

L_2:

$$\theta_j = \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)} + \frac{\lambda}{m}\theta_j,$$

其中，m 为项目总数。λ 是正则化调整参数。

4 评价

4.1 Centrality 统计

我们首先可视化了 4 个图顶点的的 centrality 的 log-log 图，如图 1-4 所示，从图中可以看出，Fork 和 Watch centrality 的 log-log 基本符合线性现象，这表明用户倾向于 “Watch” 或 fork 已经很受欢迎的项目。表 2 显示了在所有 4 个网络中的前 10,000 个项目列表之间的余弦相似度。可见 “Fork” 网络和 “PullRequest” 网络的前 10000 个项目的相似度很高。这表明一旦用户 fork 一个项目并且修改代码，他们很可能会提出 PullRequest 以反馈他们修改的内容。

4.2 PageRank 统计

如表 3 展示的，相较于 Centrality degre 的结果，Pagerank 的 4 个网络前 10,000 个项目的列表具有非常显着的相似性。这些结果证实，PageRank 是

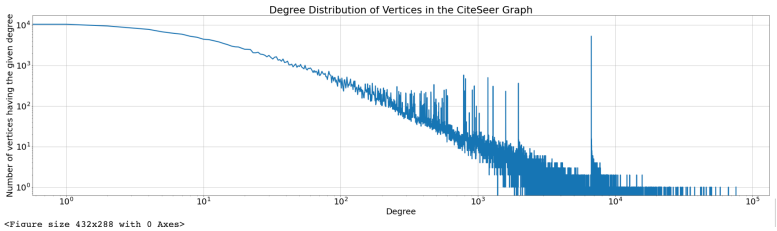

图 1: Watch Graph 的节点 centrality 的 log-log 图

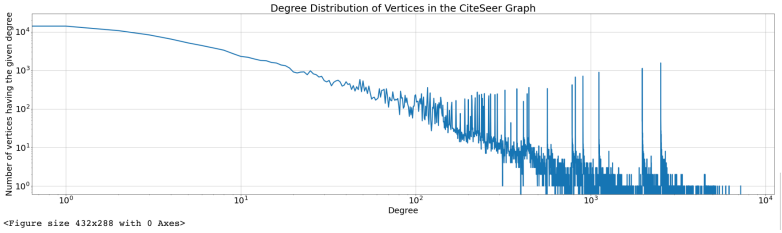

图 2: Fork Graph 的 centrality 的 log-log 图

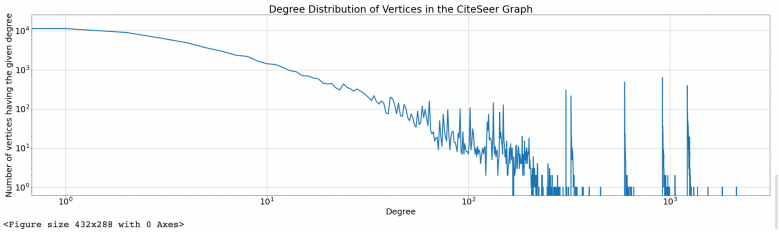

图 3: PullRequest Graph 的节点 centrality 的 log-log 图

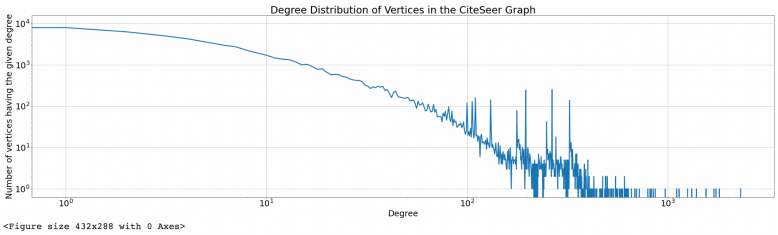

图 4: IssueComment Graph 的 centrality 的 log-log 图

一个更可靠，可以用于识别 GitHub 中流行项目。我们可以看到在 centrality

	Fork	IssueComment	PullRequest	Watch
Fork	1	0.08	0.72	0.14
IssueComment	0.08	1	0.04	0.10
PullRequest	0.72	0.04	1	0.22
Watch	0.14	0.10	0.22	1
Mean(excluding self)	0.31	0.07	0.33	0.15

表 2: 4 个网络中 centrality 结果最高的前 10000 个节点列表之间的余弦相似性

	Fork	IssueComment	PullRequest	Watch
Fork	1	0.22	0.32	0.49
IssueComment	0.22	1	0.35	0.11
PullRequest	0.32	0.35	1	0.44
Watch	0.49	0.11	0.44	1
Mean(excluding self)	0.34	0.23	0.37	0.35

表 3: 4 个网络中 Pagerank 结果最高的前 10000 个节点列表之间的余弦相似性

和 pagerank 的结果中，PullRequest 的结果是最高的，这是符合直觉的，因为如前面提到的，watch 和 fork 这两种操作时非常容易的，用户执行这两种操作并不能表示用户深入了解了这个项目，因此我们使用 PullRequest 作为特征来判断一个项目是否是受欢迎的项目。

4.3 logistic 回归的结果

我们尝试了从 2020 年 1 月 1 日到 2020 年 1 月 6 日和从从 2020 年 1 月 1 日到 2020 年 1 月 12 日。发现应用默认的正则化参数，logistic regression 算法在以下情况下表现更好。表 4 显示了两个时间范围之间的差异。很明显 logistic regression 算法在使用更多的训练数据时表现更好。在我们决定考虑更广泛的时间范围（01/01/2020 到 12/01/2020）之后，下一步是调优训练参数。我们从 α= 1, $\lambda = 3.0\times$, e^{-5}, step offset=1000, decay exponential=0.9 开始. 表 5 描述了调整正则化参数 λ 的结果。结果表明我们没有遇到过拟合问题。测试正确率达到 0.917 是一个不错的结果。

时间段	训练正确率	测试正确率
01/01/2020-06/01/2020	0.82	0.80
01/01/2020-12/01/2020	0.85	0.84

表 4: 两段时间的结果

λ	训练正确率	测试正确率
$3.0 \times e^{-5}$	0.921	0.917
$1.0 \times e^{-5}$	0.915	0.916
1	0.715	0.701
10	0.784	0.776

表 5: 调整超参数 λ

4.4 正确率对比

为了验证模型的有效性,我们在测试集合中根据 ForkEvent,WatchEvent, IssueCommentEvent 的数量，选择前 10% 的项目标注为热门项目，测试这样直接进行预测的正确率,，如表 6 所示。我们可以看到这种直接预测的正确率远低于机器学习模型的预测结果，其中 Watch 的结果最低，这可能是 Watch Graph 中有许多的噪声信息所导致的。另一方面，直接根据数字预测的话，我们无法对数字的大小有一个范围的估计，也就是说，我们需要收集所有的 Event 的数据之后才能进行预测，需要大量时间，因此利用机器学习进行预测可以显著节省时间。

模型	测试正确率
Fork	85.9%
Watch	78.2%
IssueComment	84.4%
机器学习模型	91.7%

表 6: 不同方法的正确率对比

5 相关工作

类似的关于度量影响力的研究方向已经被许多研究者探究过。Cha 等人 [1] 提及了三种影响测量指标的比较：关注、转发和提及。该论文研究了跨主题和时间的用户的影响力的变化。这篇论文的结论是转发 (retweets) 是由推文的内容驱动的，提及 (mentions) 是由用户名驱动的并且关注 (follows) 代表用户的受欢迎程度。

在网络分析中，“关注” 这个衡量标准，可以用于识别获得大量关注的用户的 degree centrality。在网络中，被提及最多的用户多大多是为名人。在所有三个指标中，最有影响力的人通常是公认的公众人物和网站。在这三个指标衡量的结果里，排名靠前的部分有一定的重叠性。本文该部分提到的分析类似于社区结构。根据该论文的分析，转发和提及这两个指标更相关，但关注与其他指标无关。

类似的，Thung 等人 [3] 研究了 GitHub 项目网络以及开发人员网络来回答这个问题：项目之间和开发人员之间的关系可以有多强？以及最有影响力的项目和开发商是什么？

他们通过连接每一对至少有一个开发者相同的项目构建了一个项目网络，类似的也连接每一对至少有一个开发项目相同的开发者构建了一个开发者网络。通过探究网络的一些经验属性 (empirical properties)，如 PageRank, network diameter, average shortest path and degree centrality，Thung 等人找到了有影响力的项目和开发人员，也发现项目网络和开发者网络的 diameter 和 shortest path 明显小于其他真实网络的相应指标 (例如 Facebook 或 SourceForge)。因此，他们得出结论：GitHub 作为一个用于编程的社交平台，确实使得每个项目之间和每个开发人员之间的距离更近。

在另一项研究中，Dabbish 等人 [2] 采访了 25 个 GitHub 上的 “红人”，发现用户会从其他用户的行为中推断出很多东西。用户通过将这些推论转化为协调编程的有效策略，以提高他们的技术水平和管理他们的声誉。例如，他们发现 “Forks” 或 “Wathces” 是社区里衡量重要性的标准，以及用户倾向于关注社区里已经广受关注的项目。或者，近期操作和项目的操作数量可以反映用户的兴趣和 Commit 的水平。他们的研究显示用户的行为对项目的声望有重要的作用，而这就是本文将要深入研究的内容。

6 结论

基于 “Fork”、“Watch”、“Issue Comments” 和 “PullRequest” 事件进行实验，计算了节点 degree centrality 和 PageRank 作为 4 个网络的两个度量。我们发现相较于 degree centrality，在使用 pagerank 时所有 4 个网络中的 Page Rank 排名最高的前 10000 个项目列表都表现出很高的相似性。此外，“Fork” 和 “Watch” 的 degree 分布网络遵循幂律分布。这表明用户倾向于 “Watch” 或 fork 已经很受欢迎的项目。logistic regression 结果符合我们的预期，证实了特征事件和项目受欢迎程度之间的联系是高度相关的。此外，正则化改进了学习算法，它避免了过拟合并提高了学习精度。通过增加训练数据量解决了欠拟合问题。特征选择、训练数据量和正则化参数调优三个基本因素使机器学习发挥更好的作用

References

[1] Meeyoung Cha et al. “Measuring user influence in twitter: The million follower fallacy”. In: *Proceedings of the international AAAI conference on web and social media*. Vol. 4. 1. 2010, pp. 10–17.

[2] Laura Dabbish et al. “Social coding in GitHub: transparency and collaboration in an open software repository”. In: *Proceedings of the ACM 2012 conference on computer supported cooperative work*. 2012, pp. 1277–1286.

[3] Ferdian Thung et al. “Network structure of social coding in github”. In: *2013 17th European conference on software maintenance and reengineering*. IEEE. 2013, pp. 323–326.

A 附录 1: 分工

李明：完成网络分析部分代码，机器学习部分代码，撰写 ppt，撰写汇报 paper。

郭腾：完成数据爬取部分的代码，撰写 ppt，撰写汇报 paper。

B 附录 2: 进度

2 月 20 日 ~3 月 15 日：搜寻资料，进行开题。3 月 15 日 ~4 月 10 日：阅读相关论文，爬取数据代码。4 月 10 日 ~5 月 1 日：爬取数据代码，完成网络分析。5 月 1 日 ~5 月 10 日：完成机器学习不分代码。5 月 10 日 ~5 月 27 日：撰写 ppt，撰写汇报 paper。