**Logo

Description automatically generated**

**社会计算**

**期末项目报告**

|  |  |
| --- | --- |
| **项目名称** | 开源世界中组织影响力的探究与实践 |
| **指导教师** | 钱卫宁 |
| **小组成员** | 陈越、吴佳威 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2022 | 年 | 5 | 月 | 27 | 日 |

## 

|  |
| --- |
| **一、引言** |
| 开源世界是一个由千万级开发者共同参与构成的世界，蕴含着丰富的社会关系。其进一步演化出的组织、仓库等实体对于开源世界也存在重要意义。其中，对组织进行深入探究分析，不但可以相对准确的判断组织的创新战略、技术能力、经济基础等，还可在繁杂的关系网络中，进一步以此研究结果为“桥梁”，帮助了解组织间的关系以及更多群组动态。  本项目主要分为两个部分：  1.全局组织影响力探究与实践   * 基于PageRank算法的组织影响力排名 * 基于度中心性算法的组织影响力排名 * 度幂律分布的研究与验证   2.头部组织影响力探究与实践   * 基于“小世界”的头部组织关系探究 * 基于Louvain算法的组织社区发现 * 头部组织关联模式挖掘 |
| **二、问题描述**  在开源世界中，各种极具影响力的开源项目往往会隶属于一些开源组织。例如，Apache软件基金会（Apache Software Foundation，简称为ASF）是一个专门为Apache 的团体运作开源软件项目提供支持的非盈利性组织。在Apache旗下，有着大量涉及各个领域的优秀的开源项目。通过对研究的组织，能够帮助我们了解开源社区的发展动态，发现具有发展前景的组织和优秀项目，促进开源社区的健康发展。  本文就开源世界中的组织提出问题，展开探索，进行研究，主要进行了组织影响力分析，以及组织间关联关系和关联强度的分析。基于GitHub数据集，我们期望从海量数据中发现具有影响力的组织，其影响力包括组织对开源社区的贡献能力、组织旗下开源项目的影响力和活跃度等。并且，通过对组织间关联关系的研究，我们期望能够对组织的社区结构进行划分，并挖掘组织之间一对一、一对多、多对一和多对多的关联关系和关联强度。 |
| **三、探究方法**  **1．构建组织间关系图**  **1.1数据清洗**：  我们从ods\_github\_log的所有历史数据中提取有用的信息。首先提取了所有加入了某个组织的用户的信息，基于此可进一步提取出组织的信息和组织的成员信息。然后提取了所有的满足条件的PR信息，条件为提PR的用户须属于某个组织，提的PR的项目仓库须属于某个组织。提取后的组织数据记录数量在百万级，PR数据记录数量在千万级。提取数据后得到以下org信息表：  表一：用户-组织表。保存用户（属于某个组织的用户）的组织信息。  表二：PR表。保存PR的信息。  表三：组织-组织表。基于表一和表二得到表三，保存PR的源组织和目标组织  信息。若组织给某组织提PR,即认为两个组织间存在贡献关系。  **1.2组织间关系图的构建：**  一个结点代表一个组织，结点间是否存在有向边由组织间是否存在PR代决  定。将提PR的用户所属于的组织作为始结点，而仓库所属于的组织作为终结  点，若一个组织与另一个组织存在历史PR，即相当于两个组织间存在一条有向  边。构建的组织间关系图的缩略图如下所示：  A picture containing schematic  Description automatically generated  **2.** **全局组织影响力的探究与实践**  **2.1 基于度中心性算法的组织影响力排名**  **实现思路：**  度中心性衡量网络中一个节点与所有其它节点相联系的程度。在组织关系图中，组织的出度一定程度上反映组织在开源社区中的贡献度，而组织的入度则能够反映属于此组织的项目的活跃度与影响力。  **实验结果**：  分别选取入度、出度前十名组织绘制统计图，结果如下：  **Chart, bar chart  Description automatically generatedChart  Description automatically generated**  **2.2 基于PageRank算法的组织影响力排名**  **实现思路**：  借鉴MapReduce中键值对转换的思想，基于SQL实现PageRank算法。  在SQL中使用表的转换模拟Map阶段中键值对的转换，使用Group By关键字和  分组聚合函数模拟Reduce阶段中对Rank值的统计，最终生成Rank表保存迭代结果。  算法步骤如下：  初始化Rank表，表中每一项保存一个vertex（顶点）的信息，顶点Rank初值设为1.0，同时记录vertex的出度。  初始化edge（路径）表，表中每一项保存一条有向边的信息。  一次迭代如下：  1、map操作，输出weight（权重）表。  Rank表的Rank值除以出度得到权重，再与edge表进行Join操作，得到每条有向边的权重值，输出为weight表。  2、reduce操作，更新Rank表。  将weight表按有向边的头结点进行Group By操作，对权重进行聚合求和，求得头结点新的Rank值，并更新Rank表。  **实验结果**：  选取迭代前10名，结果如下    对于所有组织而言，中心性算法的排序结果与我们平常的认知相符，排名靠  前的基本都是如Apache、Google这样的一些大组织。根据组织度的幂律分布，  头部组织是开源世界的重要组成部分，具有较高的研究价值。  **3.** **头部组织影响力的探究与实践**  **3.1 基于“小世界”的头部组织关系探究**  **实现思路**：   * 一个组织视为一个节点； * 若组织A对组织B有贡献，即认为节点A与B间存在直接路径由A至B； * 随机抽取一对组织A和B，使用搜索算法，求A至B的最短路径，作为一次采集的样本（BFS算法求解非带权图单源最短路径问题）； * 多次随机抽取组织对，采集一定数量的样本，研究最短路径长度的分布，计算均值等分布特征；   **实验结果：**  对于头部组织而言，头部组织之间大概率为一跳、二跳或三跳邻居，头部组织之间的联系是相当紧密的。抽取样本的统计结果如下图所示：  横坐标：任意两个结点间的距离(跳数)；  纵坐标：两个结点间的距离(跳数)为n的频数；  **Chart, histogram  Description automatically generated**  **3.2 基于Louvain算法的组织社区发现**  **实现思路**：  我们使用基于模块度的社区发现算法，期望根据头部组织间的链接关系，对  头部组织进行合理的社区划分；  （1）初始时将每个顶点当作一个社区；  （2）依次将每个顶点与之相邻的顶点合并，计算模块度增益，选择能使模  块度增量最大(贪婪思想)的社区标签；  **实验结果：**  将实验结果可视化展示如下图所示：  Icon  Description automatically generated  **3.3 头部组织关联模式挖掘**  **实现思路**：  开源项目中有很多类别，从不同角度看具有不同的体系结构，例如AI领域、  云原生领域、前端UI组件等等，也可从语言特性分类等等。我们选择提取项目  的描述信息(repo\_description)进行聚类，自动生成一个类别体系。我们期望能  够通过聚类，将项目划分为不同的类别，每个类别中存在着不同的组织。然后基  于项目仓库聚类自动生成的类别体系，把聚类结果中的每一类视为一个事务项，  在每个事务项中挖掘频繁项，生成关联规则。  **项目仓库聚类：**  将仓库描述文本信息转化为词向量，采取词频统计、tf-idf方法等提取文  本特征，然后进行PCA降维，在低维空间中进行项目的聚类操作。为达到较为理  想的聚类效果，在聚类过程中，应特别注意词向量的编码，以及聚类算法的选择。  **聚类实验结果：**  DBSCAN聚类结果: K-Means聚类结果:  **Chart, scatter chart  Description automatically generatedChart, scatter chart  Description automatically generated**  我们选取了两个类别作词云图进行实验结果可视化，从中我们可以看到，Cluster-1的关键词为library，Cluster-2的关键词为framework;  Cluster-1： Cluster-2：      **组织所有的仓库占比统计：**  不同组织在不同的cluster类中，的影响力是不同的，根据不同类中组织的  出现情况，能够判断组织在类别中的比重，并衡量他在此类别中的贡献。并基于  聚类的结果，进行频繁项的挖掘，和关联规则的生成。统计结果示例如下图所示：  类别示例一： 类别示例二：  **Chart, pie chart  Description automatically generated** **Chart, pie chart  Description automatically generated**  **基于Apriori算法的关联分析：**  Apriori算法是关联规则挖掘的经典的算法。它利用逐层搜索的迭代方法找  出数据库中项集的关系，以形成规则。根据聚类结果，我们将每一类作为一个事  务项，进行频繁项集的挖掘，产生满足最小支持度和最小置信度强度的关联规则。  支持度（support）：  support(A=>B) = P(A∪B)，表示A和B同时出现的概率。  置信度（confidence）：  confidence(A=>B) = support(A∪B) / support(A)，表示A和B同时出现  的概率占A出现概率的比值。置信度一定程度上能够反映A与B之间的关联强度。  **实验结果：**  我们使用Apriori算法进行频繁项集的挖掘，并进行关联规则的生成结果示例  如下图所示，图中包括了组织之间一对一、一对多、多对一和多对多的关联关系  和关联强度(置信度)。  **Table  Description automatically generated** |
| **四、分析与评价**  **1. 探究结果分析与评价**  **1.1 组织影响力与其市值及研发投入间的关系**  我们结合组织在开源世界中的影响力与其市值[1]及研发投入[2]进行分析，两者存在一定的相关关系，组织的综合能力与其在开源社区的影响力正相关。  **A picture containing text, receipt  Description automatically generated** **Table  Description automatically generated**  **1.2 Pagerank算法评价**  由于GitHub数据集数据量大，难以在本地运行，因此我们才尝试使用SQL  语言实现PageRank算法。使用SQL实现的PageRank算法能够直接利用线上的计  算资源完成迭代。并且借鉴了MapReduce中键值对转换的思想，在SQL中用表的  转换模拟键值对的转换，最终生成Rank迭代结果。在线上运行此算法，处理百  万级的数据需要的时间代价大概为分钟级。但是，SQL语言的表达能力是有限的，  面对权重泄露等迭代问题难以进行处理，并且也难以判断迭代是否收敛，简易实  现的迭代算法只能设置固定的迭代次数。  **1.3 组织度中心性“二八定律”**  我们对结点的入度和出度进行统计，并且发现，相对于二八定律，组织图中度的分布更加集中，大部分的PR都由小部分的组织(头部组织)提出，因此，头部组织是为开源社区作贡献的主要力量。这也是我们在对所有组织进行影响力排名的基础上，进一步对头部组织的社区结构和相关关系展开探究的原因。（我们称Google、Microsoft一类在开源社区中具有显著影响力的组织为头部组织）  **度幂律分布的研究与验证:**  横坐标：代表度的数量为前百分之n的组织，取值0-100；  纵坐标：度的数量；  入度的分布： 出度的分布：  Chart  Description automatically generatedChart  Description automatically generated |
| **1.4 头部组织影响力分析**  此处我们基于上一小节中对于头部组织影响力的探究过程及结果进行进一步分析与思考，得到如下结果：   * 基于“小世界”的头部组织关系探究：头部组织之间大概率为一跳、二跳或三跳邻居，头部组织之间的联系是相当紧密的 * 基于Louvain算法的组织社区发现：头部组织作为一个互相紧密联系的整体而存在，很难将其割裂划分。并且，其作为开源社区的重要贡献力量，将开源社区中大大小小的组织联系在一起 * 头部组织关联模式挖掘：先将仓库依照其描述信息聚类，再根据聚类结果挖掘其背后头部组织相关关系强弱   由此我们进一步验证了结论——头部组织是为开源社区作贡献的主要力量；  但是，在使用Louvain算法的组织社区发现过程中，我们发现很难对头部组织进  行划分，头部组织之间联系十分紧密，这一点在“小世界”的头部组织关系探究  结果中可见一斑，头部组织往往作为一个整体而存在。为了对比头部组织和一般  影响力组织的社区结构，作以下可视化图进行比较：  头部组织社区结构：  Icon  Description automatically generated  头部组织+一般组织社区结构：  A picture containing vector graphics  Description automatically generated  一般组织社区结构：  A picture containing diagram  Description automatically generated  从以上图中我们可以看出，头部组织之间联系十分紧密，作为一个整体而存在。  随着一般组织的加入，不同组织之间才更容易体现出多种多样的社区结构。  并且，从Louvain的迭代过程我们可以看出，Louvain算法对点多边少的图较为  友好。但是在实际情况中，对于所有组织的关系图，可能是点多边少的，但是对  于互相之间联系紧密的头部组织的关系图，呈现出边多点少的状态，因此  Louvain算法的迭代效率也不高。  **1.5 项目仓库聚类**  聚类的实验结果显示，将保存文本信息的词向量编码为词频时难以将仓库分  开来，而编码为布尔值时能更好地体现仓库的类别和聚集分布状态，项目的聚结  果比较如下：  词频词向量的DBSCAN聚类结果； 布尔值词向量的K-Means聚类结果；  Chart, scatter chart  Description automatically generated  因此，聚类的方法对于最终的结果影响是非常大的，并且，对于DBSCAN聚  类算法，它不需要我们明确类别数量，能够把具有足够高密度的区域划分为簇，  并能够发现任意形状的簇。通过比较DBSCAN与K-means的聚类效果，我们可以  看出，DBSCAN具有明显优势，聚类结果更加合理。  但是，聚类作为无监督学习，聚类的结果不一定是我们所期望的。我们期望  项目描述信息之间的相似性能够反映项目之间的相似性，但事实上基于词向量的  聚类并不一定能够正确地反映项目的类别。并且，不同项目描述所使用的语言也  可能不同，即使项目本身相似，但分别使用英语和汉语来描述的项目也很难被聚成一类。  **2. 探究过程回顾与评价**  **2.1 探究思路回顾**  在确定好想要探究的方向后，面对如此大规模的数据量，我们首先希望对数据进行一些’quick-and-dirty’的操作，从而对其产生一些粗略的了解和认识。于是我们对全局范围内的所有组织分别使用度中心性算法及PageRank算法进行影响力的初步排名，并调研了组织的市值排名及研发投入榜，发现组织的综合能力与其在开源社区的影响力呈正相关。进而，我们思考推测，是否排名靠前、影响力较高的头部组织间也存在一定联系？它们对开源世界又起到了怎样的作用？  为了弄清我们的疑惑，我们将探究范围缩小至头部组织进行进一步的深入探究与实践，分别对其进行基于“小世界”的关系探究、基于Louvain算法的社区发现以及基于Apriori算法的关联分析，我们发现：头部组织之间相互紧密联系，并且将开源社区中大大小小的组织联系在一起，对开源世界产生重要贡献及影响力。  **2.2 成长与反思**  回顾整个项目，我们最大的收获有两点：   1. 我们收获到了真实场景、大规模数据集开发的体验。本次项目，我们小组两位同学都是第一次接触真实世界场景下的数据，也是第一次接触如此大规模数据的开发与处理，这与我们之前使用的“玩具数据集”形成了鲜明的对比，让我们更真实的体验到了处理实际问题数据集是怎样的感受。 2. 我们深刻认识到：做研究要始终以解决实际问题为目标，在实际问题里找研究的“点”，绝不能“拎着锤子找钉子”，跟在别人后面做研究。在完成项目的过程中，我们发现在课堂里学习的理论上的知识与实际场景中解决问题所需要的知识和经验可能并不相同，因此若想要做出有实际价值的研究，未来也应该选择更多可以解决实际问题的项目进行实践与锻炼。 |
| **五、结论**  本项目我们针对开源世界中组织影响力进行探究与实践，得到如下结论：  1.在全局组织范围内，组织的综合能力与其在开源社区的影响力正相关，其中头部组织是开源社区的重要贡献力量。  2.在头部组织范围内，组织之间大概率为一跳、二跳或三跳邻居，头部组织之间相互紧密联系，它们作为一个互相紧密联系的整体而存在，很难将其割裂划分。并且，头部组织将开源社区中大大小小的组织联系在一起，对开源世界有着重要的贡献及影响力。 |
|  |
| **六、参考文献**  [1] Blondel V D, Guillaume J L, Lambiotte R, et al. Fast unfolding of communities in large networks[J]. Journal of statistical mechanics: theory and experiment, 2008, 2008(10): P10008.  [2] https://www.163.com/dy/article/GUFOR6OM051181GK.html  [3] http://www.199it.com/archives/1366091.html |
| **附录**  **1.分工**  **陈越：**数据清洗、算法实现、头部组织影响力探究与分析、期末项目报告撰写、期末项目汇报展示；  **吴佳威：**全局组织影响力探究与分析、期末项目报告撰写；  **2.进度**  **第5-8周：**数据集探究、技术调研、研究方向确定 |
| **第9-12周：**项目落实完成  **第13周：**优化部分项目细节  **第14周：**准备汇报展示、撰写项目报告  **3.Git仓库**  https://gitea.shuishan.net.cn/10195501438/Social-Computing-Project.git |