需求排序实验报告

1. 小组成员及得分分配
   - 刘国涛，20%
   - 李翰，20%
   - 龚雨彤，20%
   - 陈彦如，20%
   - 周昊棣，20%

2. 实验目的

本次实验的目的是选定一个开源项目，确定可能的信息来源，获取有效信息，对所获取的需求进行优先级排序。

3. 实验数据

爬虫部分：

spider.py 通过调用 Github API 爬取 vscode 仓库下所有 tag 为 feature-request 的 issue，并存储在 new_data.json 中。

数据格式为：

```
{
    "title": 标题,
    "createdAt":issue 创建时间,
    "state":issue 状态，open 或 closed,
    "number":issue 编号(方便查找，与分级算法无关),
    "commentNum":评论数,
    "voteNum":得票数,
}
```

排序部分：

data.py 整理 data.json 中的 reactions，统计其中 positive reactions 的数量，并据此划分排序等级，生成 data.txt 文件。

数据格式为：

[positive reactions count] [level] [title]

4. 实验方法

本次实验在不同环节使用了不同方法。数据获取方面，使用爬虫抓取了数据。需求排序上，通过数学工具进行分析，判断相关性，再根据时间进行区间划分，最后根据点赞量进行排序。

5. 实验结果及分析

5.1 确定项目

本次实验，我们选择的项目是 VSCode。

5.2 明确信息源

本次实验，我们的信息源为 VSCode 的 GitHub Issue。

5.3 数据获取

数据获取首先确定分为三个部分：

第一部分是确定获取数据的方式，方案有直接通过网页爬虫进行获取或通过 Github API 获取。由于网页爬虫需要解析 html 代码并需要实现翻页爬取等，工作量较大，且爬取速度慢，而 Github API 调用方便且由官方提供，爬取数据更为优雅，因此选择后者，并采用 Python 编写，调用对 Github API 封装良好的 PyGithub 库进行数据的爬取。

第二部分是确定需要的数据，这部分需要考虑排序策略所需要用到的数据，因此爬取了所有 feature-request 为 tag 的 issue，每条 issue 的字段如实验数据所示。

第三部分是编写代码，为防止爬取过程中异常中断导致爬取结果丢失，每爬取四百条数据就进行了一次保存，并且处理了 RateLimitExceededException（Github API 有每小时 5000 次查询的次数限制）和超时异常。

5.4 需求排序

需求排序分为两个部分：

第一部分，我们首先定义了一种需求排序方法，即进行等级排序。根据每条 Issue 的 reactions 信息，统计其中 positive reactions 的数量，以此作为确定等级的标准。具体而言，排序分为 5 个等级，需求的优先级依次递减，即：Highest, High, Medium, Low, Lowest。

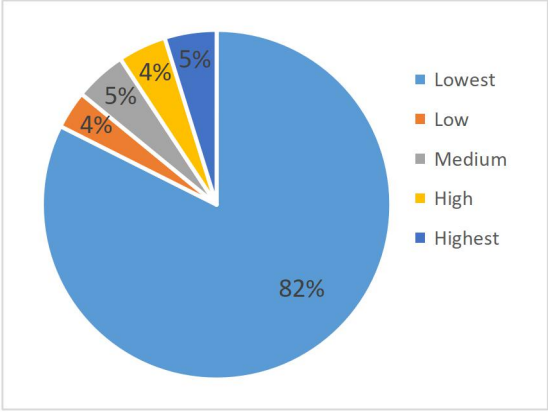
第二部分，起初考虑到 issue 的评论量也可能反映需求等级，计算其与点赞量的皮尔森系数为 0.7452264475098408，可见二者有一定相关性；但是发现数据中存在一些点赞数极低但评论数较高的情况，考虑到评论对需求等级的影响难以估计，斟酌后选择在确定需求等级时不对评论数加以考虑。其次，考虑到各 issue 发布时间不一样，而较早发布的 issue 获得更多反响理所应当（个人觉得浏览量会比新发布的高），所以按时间分成几个区间对其中 issue 分别确定需求等级；而爬取的数据中点赞量分布极不规律，暂未想到较好的处理方法，于是稍加人工处理，统计各区间点赞数分布情况，发现大部分集中在 0~4（或 3,5 等较小的自然数）之间，于是人为将这部分数据定性为“Lowest”，对剩余数据依据点赞量从小到大采用四分位数分别定性“Low”~“Highest”。（包含数据的更具体分析可见 requestLevel.ipynb）

最终排序结果为（共 14049 条需求）：“Lowest”：11577 条；“Low”：494 条；“Medium”：674 条；“High”：629 条；“Highest”：675 条。

5.5 分析排序效果

由于能力有限以及数据分布极不规律、无标签，本次实验并未采取 NLP 等机器学习方法进行分类。故而排序过程中对需求描述、评论数以及评论本身的不考虑可能会对排序结果带来一定误差。

但可以注意到的是，由于数据样本本身较为丰富，且许多需求有大量用户的各方面评价作为排序依据，所以排序得到的结果可以在很大程度上反映用户本身的真实需求。



如图所示，可以发现，排序等级为 Lowest 的需求占据了 82%的比重，而其他 4 个等级

的占比较为平均。由于 VSCode 是一个受众庞大的知名 IDE，虽然用户提出了许多需求，但其中大部分都是不甚重要或紧急的，还受限于用户本身的水平和能力。因此，这个排序后的需求等级分布较为合理，说明排序效果较好。

6. 结论

VSCode 作为一个功能丰富、受众庞大的知名 IDE，用户对它的需求多元而细致，并有着各不相同的重要性认知。经过上述实验流程，我们对 VSCode 的需求进行排序，利用用户本身的反馈信息作为判断依据，从而确定一个需求所对应的等级。由此，达到了为软件需求进行优先级排序的目的，完成了本次实验。