实验二 需求排序

一、小组成员与得分分配

彭玉	181860073	33%
赫龙飞	181860034	33%
黄思明	181860030	34%

二、实验目的

对开源项目VSCode爬虫获取需求，为软件需求进行优先级排序。

三、实验方法

本次实验先爬取github中VScode的相关带有标签的issue上千条，接着通过人工对VScode进行等级排序，将缺陷等级分为5级（Highest, High, Medium, Low, Lowest），对各种标签进行分类、打分，汇总每条issue的得分并进行等级分类。

四、实验过程

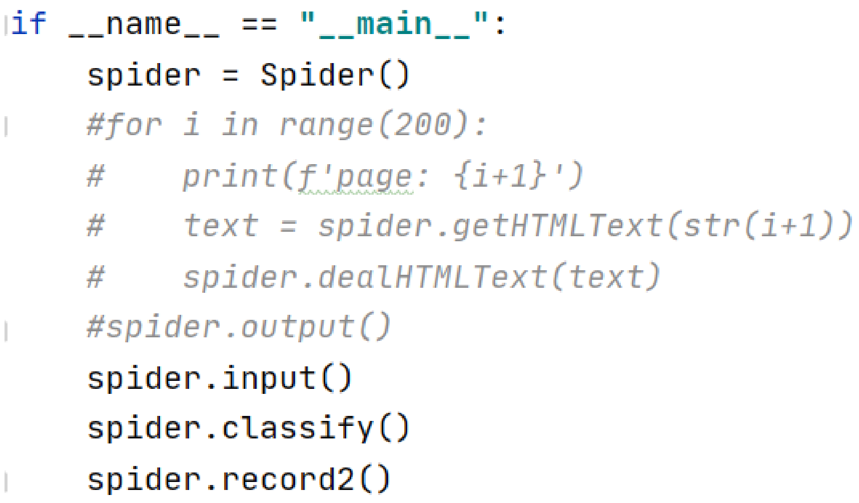
4.1 确定项目

选定开源项目Visual Studio Code。

4.2 数据获取

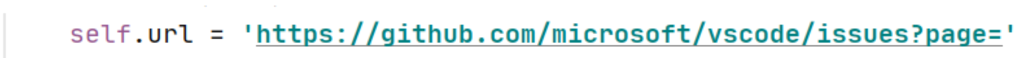
爬虫抓取了Visual Studio Code 的 Issue共4109条用户的需求问答。

主函数：注释部分为爬取阶段，爬取数据结束后，将爬取得到的数据输出到文件中，之后无需再进行数据爬取，只需读取文件中数据进行计算。



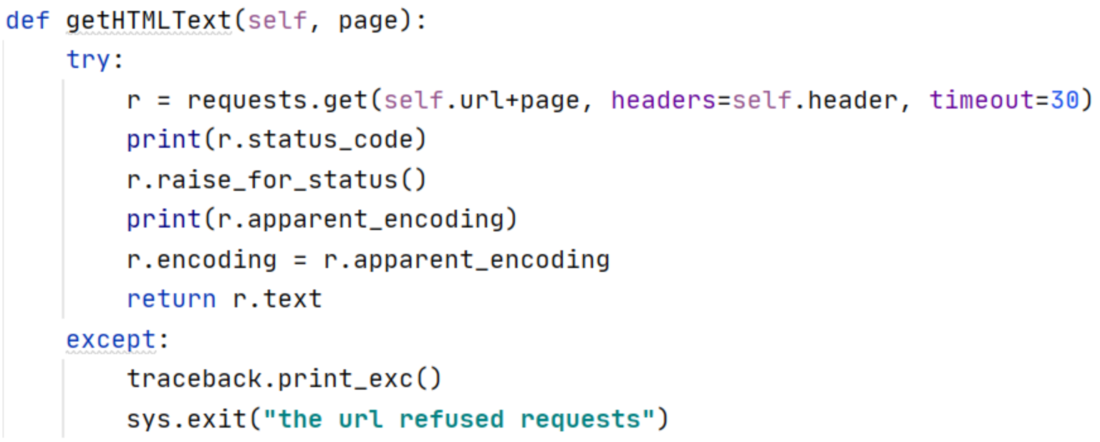
```
if __name__ == "__main__":
    spider = Spider()
    #for i in range(200):
    #     print(f'page: {i+1}')
    #     text = spider.getHTMLText(str(i+1))
    #     spider.dealHTMLText(text)
    #spider.output()
    spider.input()
    spider.classify()
    spider.record2()
```

爬取网站:



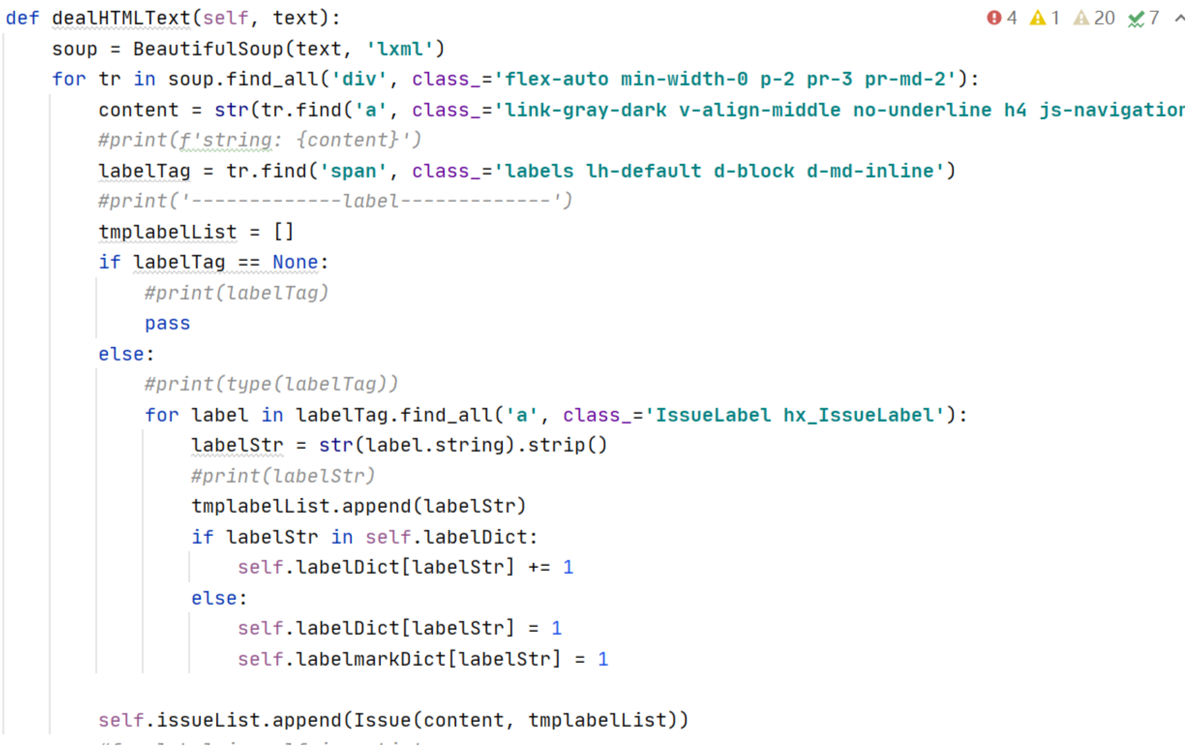
```
self.url = 'https://github.com/microsoft/vscode/issues?page='
```

获取html源代码:



```
def getHTMLText(self, page):
    try:
        r = requests.get(self.url+page, headers=self.header, timeout=30)
        print(r.status_code)
        r.raise_for_status()
        print(r.apparent_encoding)
        r.encoding = r.apparent_encoding
        return r.text
    except:
        traceback.print_exc()
        sys.exit("the url refused requests")
```

提取相关信息:



```
def dealHTMLText(self, text):
    soup = BeautifulSoup(text, 'lxml')
    for tr in soup.find_all('div', class_='flex-auto min-width-0 p-2 pr-3 pr-md-2'):
        content = str(tr.find('a', class_='link-gray-dark v-align-middle no-underline h4 js-navigatior
        #print(f'string: {content}')
        labelTag = tr.find('span', class_='labels lh-default d-block d-md-inline')
        #print('------------label------------')
        tmplabelList = []
        if labelTag == None:
            #print(labelTag)
            pass
        else:
            #print(type(labelTag))
            for label in labelTag.find_all('a', class_='IssueLabel hx_IssueLabel'):
                labelStr = str(label.string).strip()
                #print(labelStr)
                tmplabelList.append(labelStr)
                if labelStr in self.labelDict:
                    self.labelDict[labelStr] += 1
                else:
                    self.labelDict[labelStr] = 1
                    self.labelmarkDict[labelStr] = 1

        self.issueList.append(Issue(content, tmplabelList))
```

对标签设置分数，具体标签打分见下表:

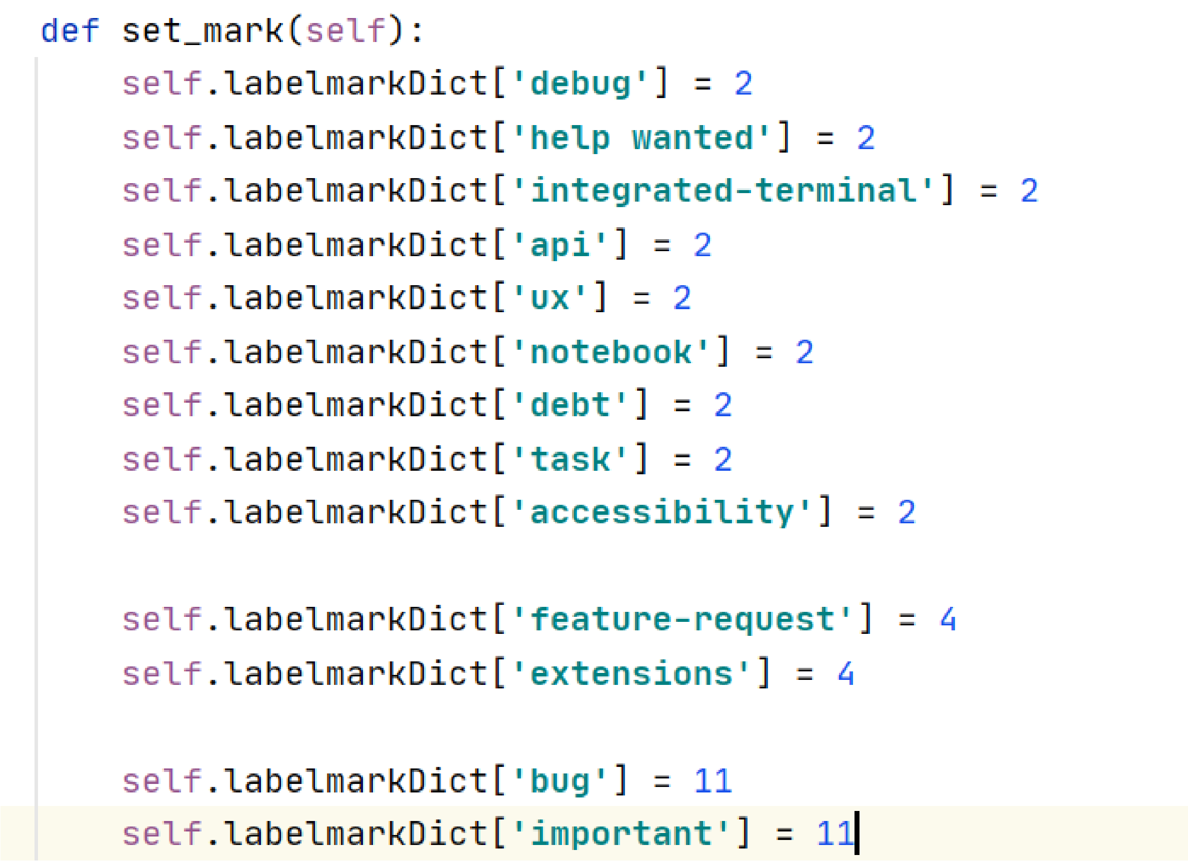
标签	得分
debug	2
help wanted	2
integrated-terminal	2
api	2
ux	2
notebook	2
debt	2
task	2
accessibility	2
feature-request	4
extensions	4
bug	11
important	11

在4109条issue中，对大多数出现100次以上的标签进行打分，以及对一些重要但是出现次数不多的标签进行打分(例如'important')。对于出现次数很少以及不重要的标签设置为1分；对于软件中较为重要方面出现的需求(debug,help wanted等)，或是基于用户自身体验(ux等) 打2分；对于软件中新功能新特性的需求(feature-request)，以及在旧功能上的拓展(extensions)，打4分；对于软件出现重大错误纰漏(bug)以及用户急需解决重要的问题(important)，打10分；对于每条issue，只要出现了'bug'以及'important'，其缺陷等级至少为high，需要十分重视。另外只要issue存在标签，即不为lowest，标签数量在一定程度上体现了需求的重要程度，因此在之前的打分中，每个标签的分数都至少为1分。

其余缺陷等级则根据具体得分判断即可，缺陷等级分类见下表：

得分	缺陷等级
16及以上	highest
11~15	high
6~10	medium
1~5	low
0	lowest

对标签设置分数，其余标签已设置为1分：

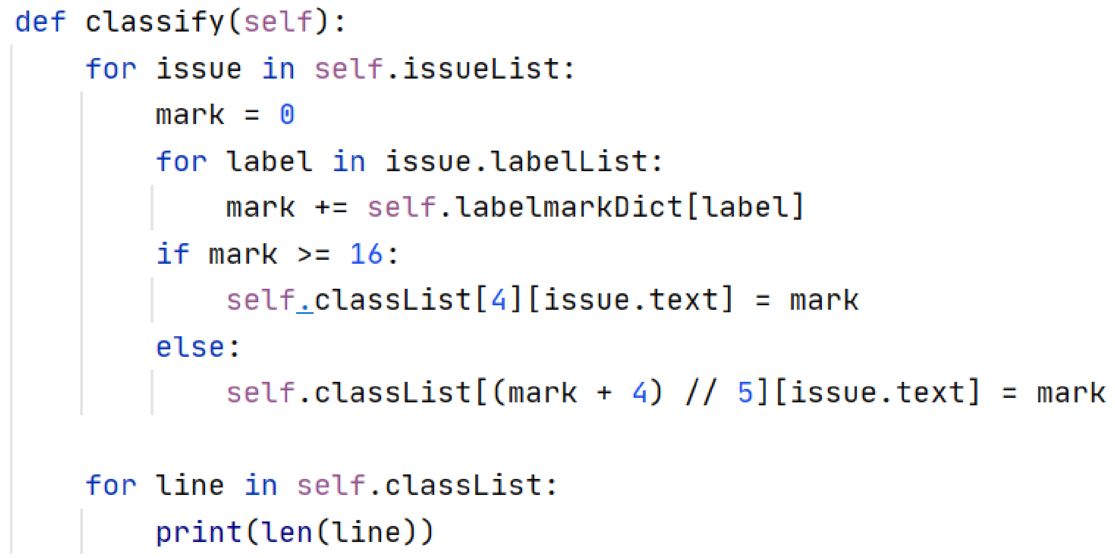


```
def set_mark(self):
    self.labelmarkDict['debug'] = 2
    self.labelmarkDict['help wanted'] = 2
    self.labelmarkDict['integrated-terminal'] = 2
    self.labelmarkDict['api'] = 2
    self.labelmarkDict['ux'] = 2
    self.labelmarkDict['notebook'] = 2
    self.labelmarkDict['debt'] = 2
    self.labelmarkDict['task'] = 2
    self.labelmarkDict['accessibility'] = 2

    self.labelmarkDict['feature-request'] = 4
    self.labelmarkDict['extensions'] = 4

    self.labelmarkDict['bug'] = 11
    self.labelmarkDict['important'] = 11
```

计算每个issue分数并分类：



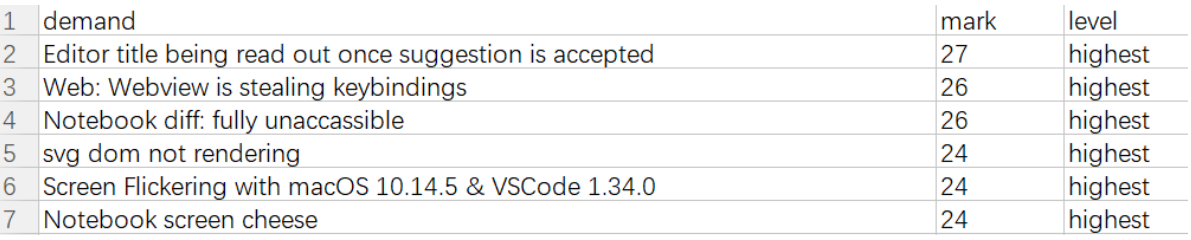
```
def classify(self):
    for issue in self.issueList:
        mark = 0
        for label in issue.labelList:
            mark += self.labelmarkDict[label]
        if mark >= 16:
            self.classList[4][issue.text] = mark
        else:
            self.classList[(mark + 4) // 5][issue.text] = mark

    for line in self.classList:
        print(len(line))
```

4.3 实验数据

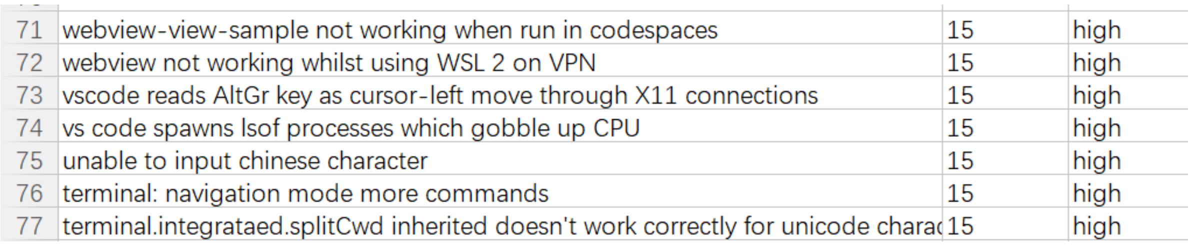
实验结果示例如下：

highest等级：



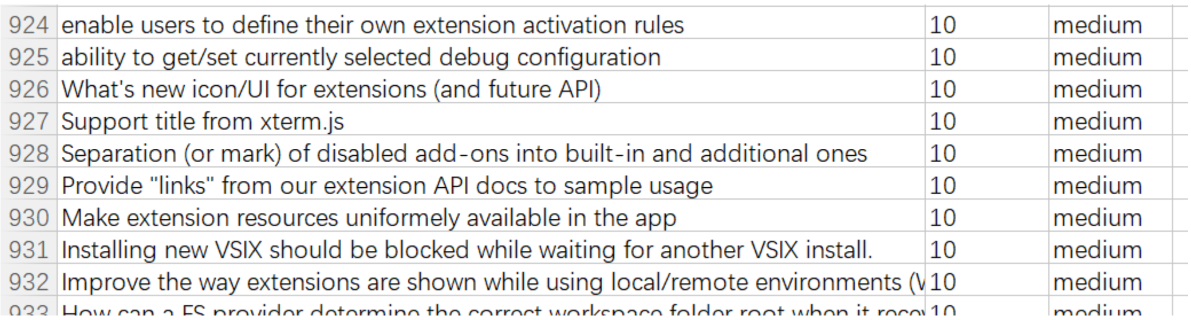
1	demand	mark	level
2	Editor title being read out once suggestion is accepted	27	highest
3	Web: Webview is stealing keybindings	26	highest
4	Notebook diff: fully unaccassible	26	highest
5	svg dom not rendering	24	highest
6	Screen Flickering with macOS 10.14.5 & VSCode 1.34.0	24	highest
7	Notebook screen cheese	24	highest

high等级：



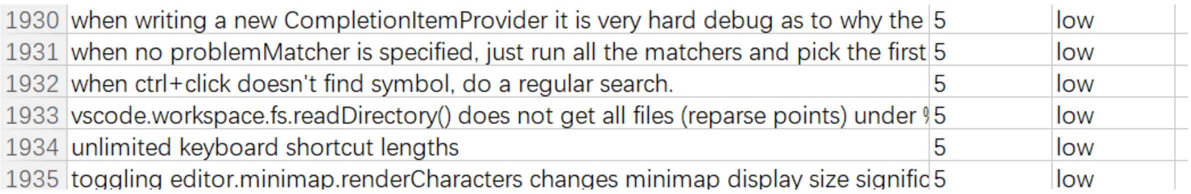
71	webview-view-sample not working when run in codespaces	15	high
72	webview not working whilst using WSL 2 on VPN	15	high
73	vscode reads AltGr key as cursor-left move through X11 connections	15	high
74	vs code spawns lsof processes which gobble up CPU	15	high
75	unable to input chinese character	15	high
76	terminal: navigation mode more commands	15	high
77	terminal.integrataed.splitCwd inherited doesn't work correctly for unicode charac	15	high

medium等级：



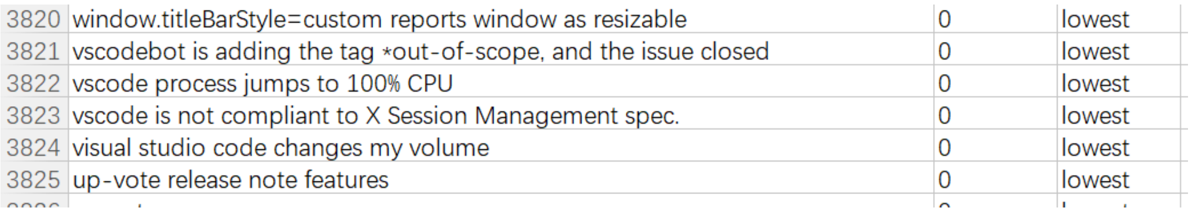
924	enable users to define their own extension activation rules	10	medium
925	ability to get/set currently selected debug configuration	10	medium
926	What's new icon/UI for extensions (and future API)	10	medium
927	Support title from xterm.js	10	medium
928	Separation (or mark) of disabled add-ons into built-in and additional ones	10	medium
929	Provide "links" from our extension API docs to sample usage	10	medium
930	Make extension resources uniformely available in the app	10	medium
931	Installing new VSIX should be blocked while waiting for another VSIX install.	10	medium
932	Improve the way extensions are shown while using local/remote environments (V	10	medium
933	How can a FS provider determine the correct workspace folder root when it recei	10	medium

low等级：



1930	when writing a new CompletionItemProvider it is very hard debug as to why the	5	low
1931	when no problemMatcher is specified, just run all the matchers and pick the first	5	low
1932	when ctrl+click doesn't find symbol, do a regular search.	5	low
1933	vscode.workspace.fs.readDirectory() does not get all files (reparse points) under	5	low
1934	unlimited keyboard shortcut lengths	5	low
1935	toggling editor.minimap.renderCharacters changes minimap display size signific	5	low

lowest等级：



3820	window.titleBarStyle=custom reports window as resizable	0	lowest
3821	vscodebot is adding the tag *out-of-scope, and the issue closed	0	lowest
3822	vscode process jumps to 100% CPU	0	lowest
3823	vscode is not compliant to X Session Management spec.	0	lowest
3824	visual studio code changes my volume	0	lowest
3825	up-vote release note features	0	lowest

统计结果见下表：

缺陷等级	issue个数
highest	268
high	1906
medium	1009
low	857
lowest	69

具体代码和实验数据见附件。

五、实验评价

此次实验满足了需求排序的基本要求，但最后的效果呈现还有许多不足，具体如下：

1、由于只抽取了少量数据，不够全面。

2、排序的标准较为主观，只以标签作为唯一打分标准过于片面，对标签的打分标准也是主观理解，不够严谨。

3、由于本次实验为人工标志打分，以及分数计算为简单的线性相加，缺少严谨性，但又能力有限，还不会使用机器学习中的决策树或神经网络等分类方法和自然语言处理技术。

六、实验总结

本次实验完成了基本任务，但还有很多需要改进的方面。虽然使用了对标志打分后根据得分进行排序分类，避免了人工手动全排序，减轻了一定的工作量，但是在打分标准和排序标准上还不够完善，较为片面。希望能学习更多有关机器学习的知识后，在以后的实验中更有改进。