需求获取与分类实验报告

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，使用特定技术获取潜在需求，并进行分类。

3. 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，既使用爬虫抓取了数据，又从信息源人工获取了需求；需求分析上，有对爬虫抓取的数据进行可视化分析和人工分析两种方法；需求分类则进行了人工分类。

4. 实验结果及效果分析

4.1 确定项目

本次实验，我们选择的项目是 VSCode。

4.2 明确信息源

本次实验，我们的信息源为 VSCode 的 GitHub Issue，以及 Stack Overflow。

4.3 数据获取

除此之外，我们还从 VSCode 的 GitHub Issue，以及 Stack Overflow 获取了一些具体的需求，并进行归纳整理，以待后续分析和分类。

使用 Python 调用 Github API 获取了 vscode 项目仓库下以 feature-request 为 tag 的 issues。

一条 issue 的数据结构如下：

title	description	createdAt	state	number	comments
Issue 标题	描述	创建时间	状态	Issue 标号	Issue 下的评论

Github REST API 是 github 官方提供的用于获取 Github 上数据的 API, 链接为 https://developer.github.com/v3/

爬虫代码调用了 PyGithub(https://github.com/PyGithub/PyGithub)，它对上述的 Github REST API 封装成了 Python 的一个库。

爬虫先获取了 Github 上的”microsoft/vscode” 仓库,然后调用 get_issues 方法获取以”feature-request”为 tag 的所有状态的 issues,因为 tag 为 feature-request 的 issue 最适合用

来作为我们需求获取的源数据。

由于 Github 的 API 有调用的速率限制,所以需要在爬取数据时需要加入速率控制的方法。

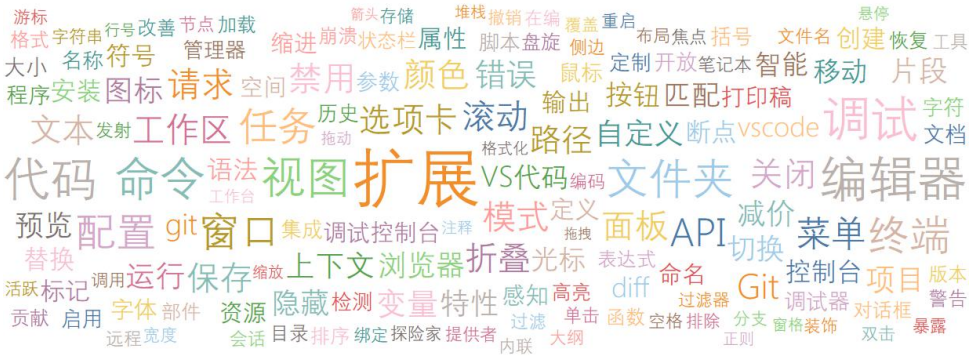
最终爬虫爬取了共 13621 条 issue 数据,存储在 data_with_comments.json 文件中，然后通过 analysis.py 调用有道云的翻译 api 对所有 issue 的 title 翻译成中文。

4.4 需求分析

数据分析分为两个部分：

第一部分，对爬虫抓取的数据进行可视化分析。将抓取的 13621 条 Issue 标题翻译后，以文本文件 titiles_zh.txt 为数据源，制作如下词云图：

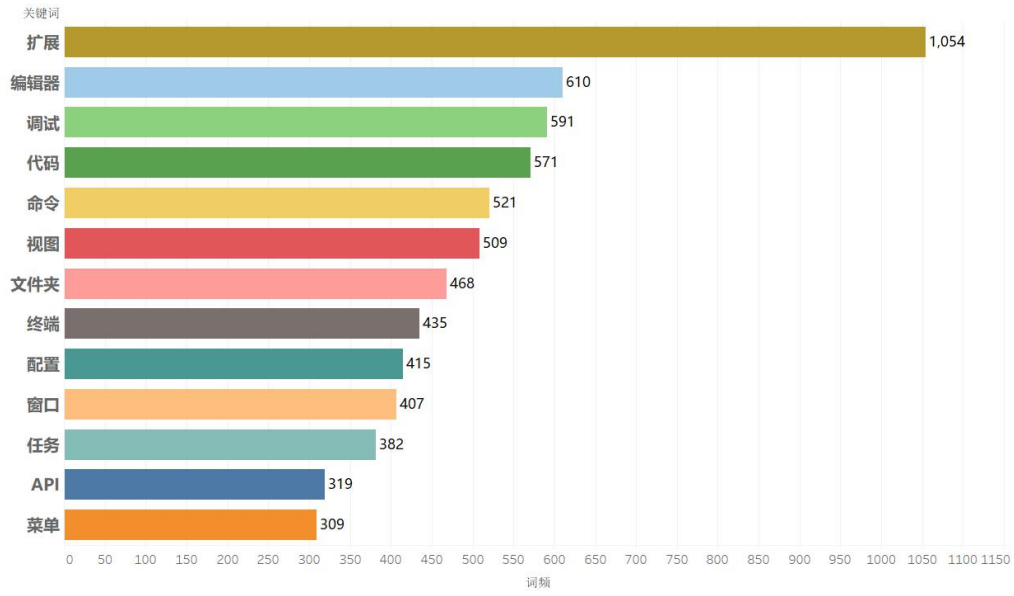
VSCODE需求词云图



可以看出，在 GitHub Issue 中，对 VSCode 的需求分布在多个方面，涉及到了诸如调试、命令、运行、保存等功能需求，也涉及到了扩展、视图、颜色等非功能需求。由此可见，VSCode 的需求是极其多元而丰富的。

再以上述文本为数据源，筛选词频大于 300 的关键词进行分析：

VSCODE需求条形图



可以看到，上述关键词是比较突出的部分需求。其中，“扩展”这一关键词共出现了1054次，占据了极高比重，可能与VSCode一些非常有用的扩展功能有关，这些功能满足了用户对于VSCode的易用性等方面需求。纵观这些突出的需求可以发现，功能需求和非功能需求并重，说明两者在软件需求中都占据着重要位置，而非主从关系。

第二部分，则是从各信息源人工获取需求进行分析。

非功能需求：

（1）“希望能够选择将侧边栏中项目里一些不常用的文件或者文件夹隐藏，便于找到常用的需要的文件”——隐藏侧边栏中某些文件；

（2）“想在编写完代码后能自动将所写的代码自动缩进、对齐等，使其格式美观，便于阅读”——格式化代码；

（3）“在搜索某一变量或者函数等其他内容时希望能够自行选择不在某些文件夹或文件中搜索，以免带来很多不必要的搜索结果”——在搜索期间能选择忽略某些文件夹；

（4）“想要对出现在多个地方的同一个词进行修改或者删掉时，能够直接进行，而不是只能先搜索再替换”——可以选中某个词的多个实例然后进行编辑或删除；

（5）“阅读HTML文本时，希望能对其进行语法高亮显示，同时保持vscode本身仍然是纯文本编辑器”——在文本区域对HTML进行语法高亮显示；

（6）“希望在没有网络的时候也能够进行插件的下载安装”——离线安装插件；

（7）“对于已经安装的插件，希望增加一个导出功能，以便将插件清单分享给好友”——导出已安装的插件列表；

（8）“希望能够直接找到文件中的空格和制表符，以及将空格转换为制表符，或是将制表符转换为空格”——实现空格和制表符的相互转换；

（9）“打开多个窗口时，希望不同的窗口可以使用不同的主题”——同时打开多个不同主题的窗口；

（10）“打开一个新的文件夹时，希望能够不直接覆盖当前窗口内容，以便于编辑”——方便的打开多个文件夹；

（11）“希望在MAC上能够直接使用命令行打开vsc，并且拥有与windows/linux下相同的功能”——在不同平台能有相同的操作体验；

（12）“有时候不小心误操作后无法通过撤销回到之前想要的某个位置，希望能够提供这一功能，可以像虚拟机快照一样记录下某个位置的代码，以便恢复”——操作上的可撤销性，方便回到指定的某个位置；

功能需求：

（1）“提供打开并导航到工作区中的文件的命令”——工作台相关的需求；

（2）“左右移动标签，可以右键单击“文档”选项卡，并且将其移动”——优化工作台标签；

（3）“关闭所有编辑器时但不应该关被固定的选项卡”——工作台上标签部分应该要

做的改进；

（4）“如果可以提供一个重新运行任务的命令，我可以从这里开始调试”——回到上次运行状态的命令；

（5）“在工作空间更改时保留每个窗口的环境变量”——工作台设置；

（6）“固定标签：允许更改固定标签的主题颜色”——工作台中对标签的更改；

（7）“光标指示范围不够明确，不能直接指出父级范围，容易造成错误，希望能得到改进”——改进光标指示范围，是指示范围更全；

（8）“远程 SSH：支持垃圾桶的”——文件管理上要对 SSH 垃圾桶做出改进；

（9）“在保存时，更新远程服务器上的较大文件时，Vscode 需要很多的时间，希望能使用差异化策略来更新远程服务器上的内容（仅发送已更改的部分）”——在远程服务器上快速更新文件内容；

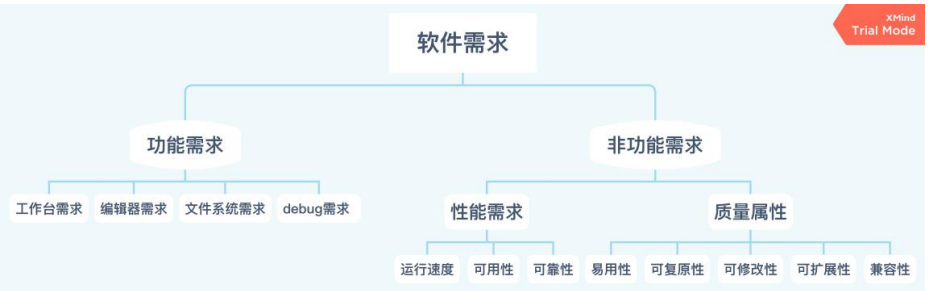
（10）“是否可以先在临时位置更新文件，然后文件完全上传到服务器后替换原始文件。”——保存文件的需求；

（11）“在 debug”悬停底部显示帮助文本/提示，以切换到普通悬停”——在 debug 时对于显示的需求。

4.5 需求分类

这一部分，对从各信息源人工获取的需求进行分类。

首先，我们构建了如下的需求分类图：



其中，对非功能需求做出如下解释：

- 运行速度，指软件在各环节中的运行速度。
- 可用性，指能打开多个窗口、离线安装插件等必要的非功能需求的可使用性。
- 可靠性，指软件正常运行，尽量少发生意外错误的需求。
- 易用性，指在使用方式上和使用过程中软件方便使用的程度。
- 可复原性，指在操作上可撤销以及意外退出自动保存等非功能需求。
- 可修改性，指在安装和卸载软件上的相关需求。
- 可扩展性，指可安装插件等扩展需求。
- 兼容性，指对多种语言的兼容和在多平台运行的兼容性等。

将上一部分人工获取的抽样需求进行分类，获得如下分类结果：

非功能需求：

性能需求：可用性：（6）（7）（9）（10）

质量属性：易用性：（1）（2）（3）（4）（5）（8）

可复原性：（12）

可扩展性：（7）

兼容性：（11）

功能需求：

工作台需求：（1）（2）（3）（5）（6）

编辑器需求：（7）

文件系统需求：（8）（9）（10）

Debug 需求：（4）（11）

5. 结论

VSCode 作为一个功能丰富、受众庞大的知名 IDE，用户对它的需求也是多元而细致的。经过上述实验流程，我们对 VSCode 的需求有了一定的认知，了解了部分用户对于 VSCode 的具体需求，总结出了一个相对完整的分类体系，从而完成了本次实验。