Python3 requests 包主要功能模块的动态调用图相似度分析报告

王子成 DZ1833026

组员：韩远磊（MG1833022），王子成（DZ1833026），张鹏（DZ1833034）

分工：张鹏同学负责 AST 实验的设计与实现，王子成同学负责调用图实验的设计与实现，韩远磊同学负责数据流实验的设计与实现。

我完成的工作：本次实验(python3 requests 包主要功能模块的动态调用图相似度分析)的设计、实现、结果的分析等全部工作由我一人单独完成，同时我负责编写相应文档和 PPT。

目录

一 背景、要解决的问题

Python 是一种计算机程序设计语言。是一种动态的、面向对象的解释型脚本语言，最初被设计用于编写自动化脚本(shell)，随着版本的不断更新和语言新功能的添加，越来越多被用于独立的、大型项目的开发。因此项目运行脚本中的调用关系愈发被人们重视，分析调用关系的手段有绘制调用图。

调用图反映了程序函数中的依赖关系，可以通过静态分析代码生成静态调用图或者记录程序运行路径生成动态调用图，静态调用图优点是分析调用关系较为全面，同时可以处理多个函数之间循环调用等复杂问题，然而对动态性执行的代码分析能力较差。由于 Python 的是一种动态的解释型脚本，因此使用动态调用图能够更好地表征关键执行路径的调用关系，同时能够忽略项目中文件数目对调用图分析的影响。

Requests 包是 python 实现的一种用于处理 http 协议的工具包，功能点集中，代码量较为适中，该项目一直在 http://github.com 网站维护，此外该项目的源代码不存在循环调用等较为复杂的问题，同时项目实现的过程中代码能够兼容 python3.x 版本和 python2.x 版本，利于进行分析。

我们通过收集 https://github.com 网站上项目的不同开发版本的关键功能模块调用图，和最新发布的 release 版本生成的关键功能模块动态调用图分别对比相似度。
由于收集到的动态调用图使用 pydot 绘制，实现过程不清晰，直接对比生成的 PNG 格式图片像素比较相似性显然准确性欠佳。因此我们应当适当修改 pycallgraph 项目代码，获得动态调用图的连通图矩阵，并使用余弦三角函数表征不同版本项目之间的相似性。

二 研究问题

收集 github 上 requests 的全部 commit 开发版本和 release 稳定版本源代码，使用 python3 提供的 pycallgraph 工具生成这些项目的调用图，相似性关系可以包括相邻版本之间的相似性关系，不同版本与指定版本之间的相似关系。由于相邻版本之间的相似关系可能无法表现项目开发整体过程的相似性，因此我们最终决定和最新发布版本的动态调用图对比获得相似性关系，通过比较相邻版本分别与最新发布版本的相似性关系（相似性关系折线图相邻点之间梯度），判断相邻版本的相似性。

计算相似性时选择被比较动态调用图和最新发布版本的动态调用图的全部节点的并集，使用并集建立两个连通图矩阵，两个图分别表征两个版本的调用关系，将连通图矩阵每行视为一个向量，根据欧几里得距离计算余弦相似性。
将计算出的全部相似性绘制图像进行表征。

三 实验设计

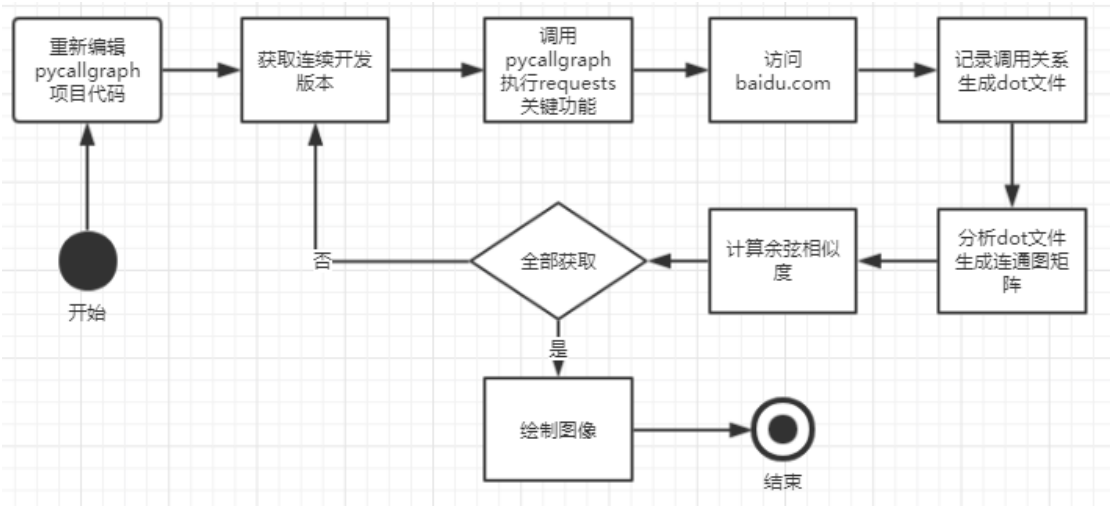
3.1 实验对象

Python requests 项目是 python 社区注明的 http 协议操作工具包，多数数据挖掘工作的开始都是从使用 requests 包获取网页开始的，该项目在 https://github.com 上有 36000 余 Star 和 6665 个 fork，共有 528 位代码贡献者，是该社区的热门项目之一，具有广泛性和代表性，并且维护了 requests 的全部 commit 开发版本（超过 5000 次 commit）和 release 稳定版本源代码，对于连续的 commit 版本可以直接使用 git reset 指令进行切换，而连续的 release 版本的压缩文件都可以在网站上下载且具有类似链接地址（仅版本号不同），因此编写简单脚本即可下载并解压缩全部 release 版本代码。

3.2 总体实验步骤设计

分析 pycallgraph 源码，将用于生成图片的 dot 格式文件进行修改，去除其中多余信息，仅保留 from 节点->to 节点 等关键信息。调用被修改的 pycallgraph 包，执行 requests 包中获取 https://baidu.com 网站（为保证该网站准确可达，且访问速度足够快），pycallgraph 记录 requests 执行关键功能模块的调用关系，生成 dot 格式文件并保存到指定文件夹。

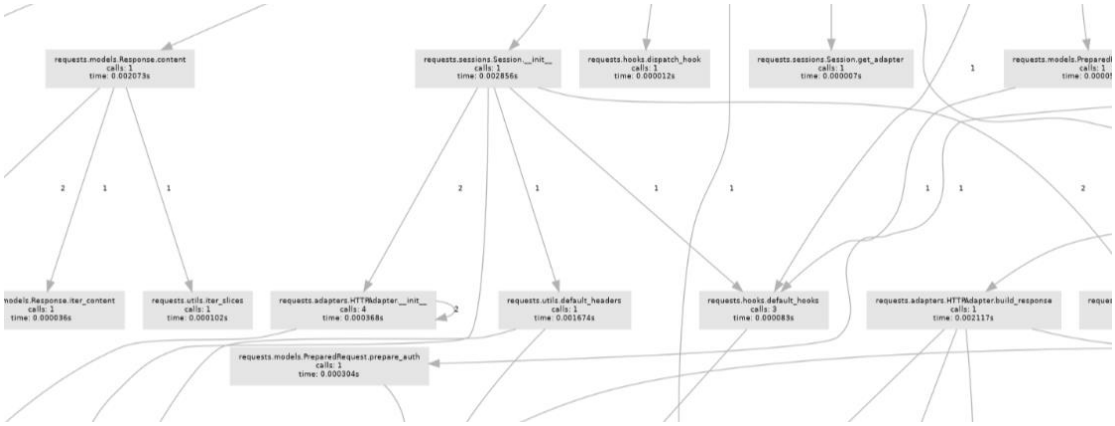
使用脚本读取生成的全部 dot 文件，综合最新发布版本生成表征调用关系的连通图矩阵并计算获得矩阵与最新发布版本矩阵的余弦相似度，收集全部余弦相似度进行分析。流程如图所示:



3.3 脚本文件设计

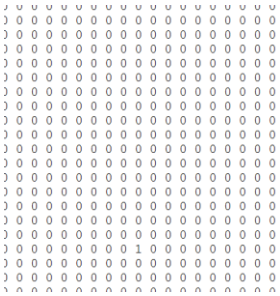
Callgraph.py：输入为待分析的 requests 项目版本，调用重新编辑过的 pycallgraph 包，执

行 requests 获取 https://baidu.com 网页，生成 dot 格式文件并保存在指定目录中。
cmpMatrix1.py ：获取全部连续 release 之间的 dot 格式文件，与最新发布版本 dot 文件对比，生成连通图矩阵计算全部余弦相似度。并将相关数据保存到硬盘中。
cmpMatrix2.py ：获取全部连续 commit 之间的 dot 格式文件，与最新发布版本 dot 文件对比，生成连通图矩阵计算全部余弦相似度。并将相关数据保存到硬盘中。
op.sh：编写 shell 文件分别对每个 requests 的 release 版本执行 callgraph.py 脚本。
opsolve.sh：编写 shell 文件使 git 在连续的 commit 之间切换并在每个版本切换完成后执行 callgraph 脚本。
draw1.py：读取连续 release 之间的相似度结果文件，绘制折线图和 y=kx 函数坐标图。
draw2.py：读取连续 commit 之间的相似度结果文件，绘制折线图和 y=kx 函数坐标图。
Graphviz.py：pycallgraph 中负责生成 dot 文件并绘图的脚本，重新编辑其中的 generate_node 和 generate_edge 等函数本项目所需格式的 dot 文件。



Commitresult.txt：连续开发版本之间结果
Result.txt：连续的发布版本之间结果

3.4 连通图矩阵设计



$$Matrixfuncs = latestfuncs \cup currentfuncs$$

连通图矩阵的节点是读取被测试版本和最新发布版本的主要功能模块调用图的全部节点（即函数）的并集，分别生成两个矩阵，分别表征两个版本的函数调用关系，通过分析可见此类矩阵属于稀疏矩阵。如图所示

3.5 评价标准设计

利用三角函数中余弦函数来计算相似性，原理如下：
将矩阵压缩成向量，计算余弦复杂度
X=(x_0，x_1，x_2,⋯，x_n)
Y=(y_0，y_1，y_2,⋯，y_n)

$$\cos\theta = \frac{\sum_{i=0}^{n}(x_i \times y_i)}{\sqrt{\sum_{i=0}^{n} x_i^2} \times \sqrt{\sum_{i=0}^{n} y_i^2}}$$

$\cos\theta$ *值*越大相似性越高。

y=kx 斜率计算 $k = \frac{\cos\theta}{\theta}$，函数图像越靠近 y 轴相似性越高。

3.6 研究问题设计

RQ1.连续的 Commit 间，模块 动态调用图 的变化可能有多大？
观察根据 commit 之间相似度的折线图中相邻点的斜率能够进行判断，根据整体折线图分析相邻点之间变化趋势，根据 y=kx 函数图像观测连续 commit 中较大变化的情况。
RQ2.连续的发布版本间，模块 动态调用图 的变化可能有多大？
根据相邻发布版本之间的相似度折线图观察相邻点的斜率能够看出
Bug-fix 版本间，模块 动态调用图 的变化可能有多大？
定位到指定的 bug-fix 版本，观察折线图两侧的相邻点斜率，查看变化情况。
功能版本间，模块 动态调用图 的变化可能有多大？
定位到指定功能版本，观察折线图两侧的相邻点斜率，查看变化情况，也可以根据 y=kx 函数图像不同函数簇的汇聚情况查看功能版本之间的变化情况。
RQ3.在什么情况下，相近的两次 Commit 会使得模块 动态调用图 发生大的变化？
RQ4.开发过程中，模块 动态调用图 有没有可能发生连续的大的变化？
具体分析详见实验结果部分。

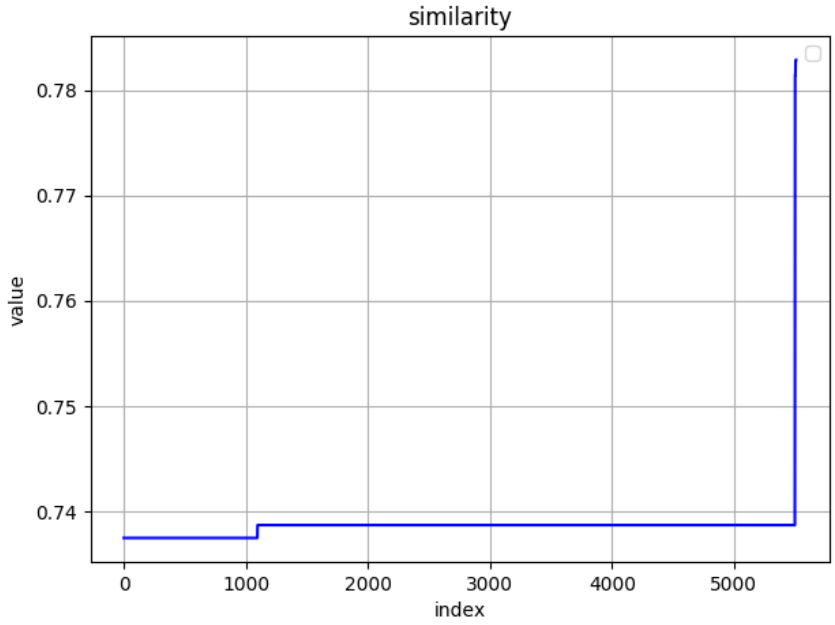
四 实验结果

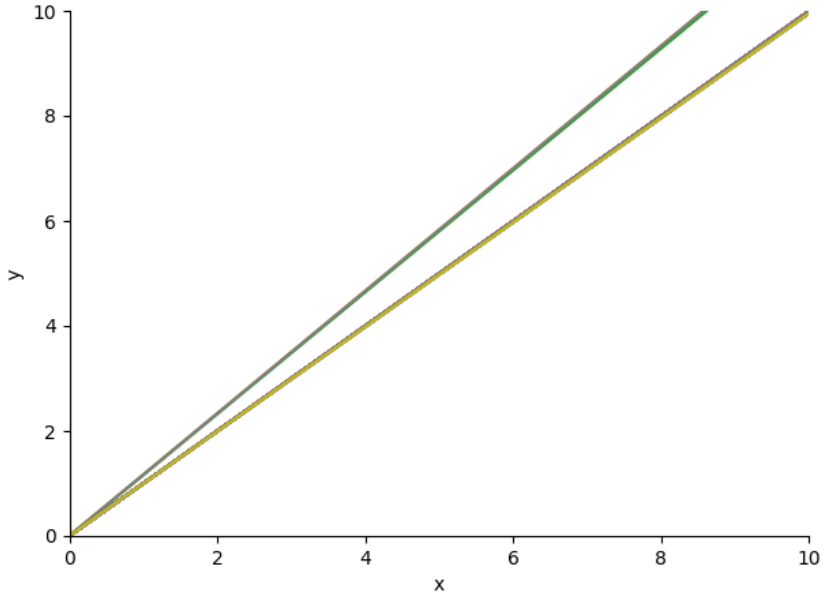
4.1 RQ1.连续的 Commit 间，模块 动态调用图 的变化可能有多大？

共收集了超过 master-branch 超过 5000 次 commit 版本与最新发布版本的关键功能模

块动态调用图，根据收集到的相似度信息绘制一下图像。

在折线图中可以看到 master-branch 分支中从最初的 commit 到最末的 commit 主要功能模块的动态调用图相似度仅发生不到 0.05 的变化，5000 次中绝大部分的相似度基本保持在相同的水平，仅在最后几次的项目改版中调用关系突然发生较为明显的变化。根据 y=kx 函数图像也可以得出以上结论。因此可以推断相邻的 commit 直接除非发生较大的功能模块性改动，否则调用图相似性没有明显变化。





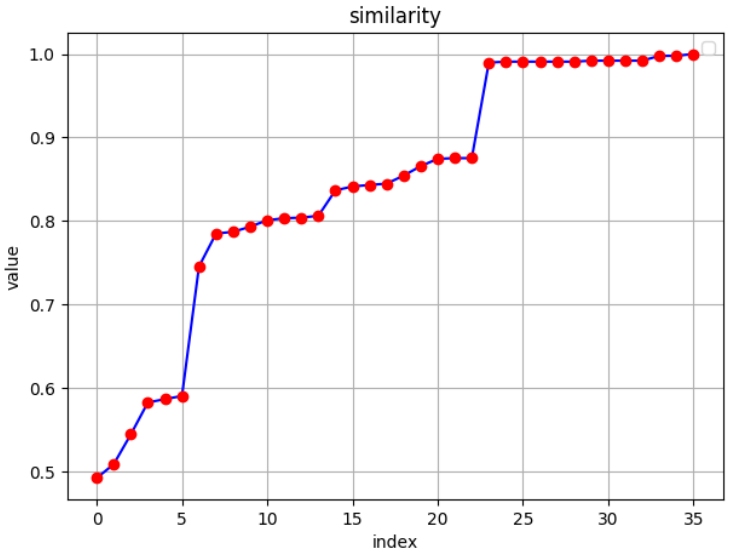
4.2 RQ2.连续的发布版本间，模块 动态调用图 的变化可能有多大？

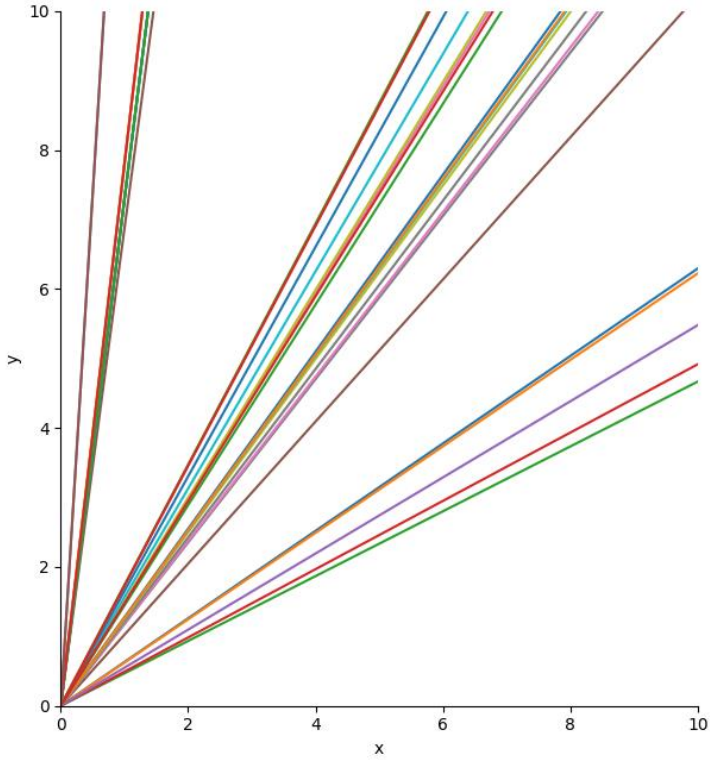
根据折线图显示，连续的发布版本之间主要功能模块动态调用图的变化可能基本限定在 0.05 范围内，然而根据 release 信息可以看出功能版本之间的动态调用关系变化明显比 bug-fix 版本之间调用关系明显，新功能增加后的发布版本和之前发布版本存在较大差异，主要功能模块的调用图相似度可能会超过 0.1。

4.3 Bug-fix 版本间，模块 动态调用图 的变化可能有多大？

根据查看发布版本信息，可以看出 bug-fix 版本间动态调用关系的变化范围基本限定在 0.05，也就是说调用关系变化可能性较小，主要的修改集中在函数内部而不是函数之间。功能版本间，模块 动态调用图 的变化可能有多大？

根据查看 release 信息，可以看出功能版本之间的主要功能的动态调用关系变化较大，甚至能够超过 0.1，由此可见主要功能模块的增加是通过增加新的函数并调用这些函数实现的。





4.4 RQ3.在什么情况下，相近的两次 Commit 会使得模块 动态调用图 发生大的变化？

当开发者视图增加新的功能模块时，会增加函数数量和并调用增加的函数，此时会发生较大变化。

4.5 RQ4.开发过程中，模块 动态调用图 有没有可能发生连续的大的变化？

在 commit 版本中这种可能性非常小，因为 requests 的开发者数量庞大（超过 500 人）而且项目开发需要时间，因此每个连续的 commit 之间基本上都是规模较小的更改，发生连续的大的变化可能性非常小。然而根据上图图像分析，连续的发布版本之间是可能发生较大的变化的，原因可能是每次发布的项目可能是已经添加过全新功能模块并经过测试修改完善的版本，因此主要功能模块的调用关系相似度连续发生较大变化，然而由于 bug-fix 原因发布的新版本相似度变化普遍不大。

五 讨论、工作亮点

1. 针对 python 较强的动态特性，将全部精力集中在主要功能模块的动态调用关系，一方面可以减小调用图的规模，加快相似度分析速度，另一方面绝大多数用户使用这类工具仅使用工具的主要功能模块，集中分析主要功能模块的调用关系的相似度能够帮助广大用户更好的了解项目运行过程中的调用关系。

2. Master-branch 连续 commit 版本之间和最新发布版本主要功能模块的动态调用关系相似度仅位于 0.75-0.79 之间，令人十分困惑。经过仔细查看发现之前版本和最新版本的开发并不位于 master-branch，而是在其他 branch 中开发，待测试通过后直接合并到 master-branch 中，在 master-branch 中提交的 5000 多次 commit 版本主要集中在第 6-10 发布版本之间。

3. Commit 提交信息往往由于开发者的缘故并不明确，我们仅仅能够通过人工的关键字提取分析版本提交的可能性，而想要确认，必须通过调用图或者亲自阅读代码进行确认。

4. 项目的早期版本相似度和最新版本主要功能模块的动态调用关系的相似度十分接近，说明该项目主要功能模块的函数设计十分稳定，多数的功能模块性更新均用于提升性能和健壮性，是经过深思熟虑的成功项目。

5. 除余弦相似性还可以利用其他方式对主要模块的动态调用图进行比较，例如直接比较调用图图像之间的像素相似性，然而我们无法保证相似的调用图拥有相似的调用图图像。

六 结论

程序的调用图对开发人员理解源码有非常大的帮助，特别是面对大型项目，庞杂的代码量，项目文档缺失，第三方代码库，检查实际函数调用关系跟规划的设计是否一致这些复杂问题时，调用图能够帮助项目的开发人员快速的了解一个项目。在 https://github.com 的开源项目中，我们也可以使用主要功能模块的调用图来快速了解该项目中主要功能的实现方法。

利用项目中不同开发版本的主要功能模块动态调用图相似度的比较，我们可以做出以下推断：1）项目开发过程中应当频繁的使用 git 保存项目进度，以便出现问题是及时补救。2）连续的 commit 之间变化主要功能模块的调用图变化通常并不显著，因为在多人协力开发的过程中不同功能的函数及其调用被最初确定，其余的修改集中在函数内部的功能实现上。3）在多人合作开发项目的过程中，添加新的功能模块往往通过增加新的函数和函数之间的调用关系实现，而 bug-fix 往往仅对函数内部功能实现进行维护，因此相邻的发布版本之间往往因为新功能的增加而存在较为显著的调用图相似度变化。