北京大学

Programmable Web 需求关键词视角下的历史调用记录分析

姓　　名：孙笑宇

学　　号：2201210649

院　　系：软件与微电子学院

授课老师：褚伟杰

Github: https://github.com/Stellasxy/-Python-sxy/tree/main

二〇二二年十一月二十九日

题目概述

越来越多的软件开发工程师倾向于使用 Web 上的各种 API 服务来开发应用并创造价值。Facebook、Google、Amazon 等企业，在面向服务架构（SOA）、面向服务计算（SOC）等方法的引领下，纷纷通过向用户发布 Web API 开放自己的各种信息和数据服务，以获取额外的经济效益； Twitter、Yahoo Search、Google Maps 等一大批极具商业价值的 Web API 服务应运而生。 随着 Web API 数量的增加，通过调用和合成它们可以产生新的 Web 应用和增值服务，例如 Mashup 应用。在此背景下， Web API 生态系统开始形成。Web API 生态系统是由 Web API 服务，服务提供商， 服务消费者以及它们之间存在的关联关系（引用、合作或竞争等）构成的开放复杂系统。ProgrammableWeb.com、apis.guru，RapidAPI 等一批打造服务生态系统的平台受到关注。

Web API（Application Programming Interface）是网络应用程序接口，是一个 预先定义的函数或 SaaS(Software as a Service)软件，供应用程序与开发人员调用。Web API 包含了广泛的功能，网络应用通过 API 接口，可以实现搜索服务、存储 服务、计算服务和信息服务等能力，利用这些能力开发出功能更强大的 Web 应用。

由多个功能不同的 Web API 和多个互联网资源通过调用和合成的 Web 应用 称为 Mashup。Mashup 是一种重量级的 Web 服务，由 SaaS、SOA 和 Web 2.0 等 多种技术组成，能满足更加复杂的需求场景。 Mashup 作为一种新型的 Web 应用开发方式吸引了产业界的极大关注，如 Google、百度、腾讯、Yahoo 和 IBM 等公司都提供了各自的服务接口和开发平台 。 由于 Mashup 集成了许多的功能服务，使得 Mashup 可以在各式各样的应用中 得到发挥,可以同时向外界提供多种服务。以下列出了常见的几类 Mashup。

对 Programmable Web 数据集中 Mashup- API 的历史调研记录进行统计分析，给出分析过程和结果。包括：

1. （统计 Mashup 中的包含 Web API 个数、Web API 被使用的次数和 Web API 提供商发布 Web API 的个数）分析数据集中 Mashup- API 的历史调研记录，统计如下信息 1）每个 Mashup 包含的 Web API 个数 2）每个 Web API 被使用的次数 3）Web API 提供商（URL 网址）发布 Web API 的个数。
2. 从需求关键词视角，分析在不同标注 Tag 或者 Category 类别中， 编程开发人员的组合需求（Mashup）与该需求所调用的服务（Web API）的关联情况。
3. 从非功能视角（例如 API 之间的兼容性，Web API 不同的服务接口协议 REST、RPC），对功能类别相等的若干 Web API 进行分析（ Web API Tag 标注值 或者 Web API 所属 Category 类别相等）。在 Tag 取值相等 或者 Category 取值相等的 Web API 集合内，统计 API 被使用的情况，尝试从非功能视角进行原因分析（例如 API 之间的兼容性，Web API 不同的服务接口协议 REST、RPC） 注：API 之间的兼容性是指 历史上共同被调用的情况。如果有过共同调用的记录，说明这两个 API 兼容

一、数据集中 Mashup- API 的历史调研记录分析

1.1 统计每个 Mashup 包含的 Web API 个数

首先，我们 https://github.com/HIT-ICES/Correted-ProgrammableWeb-dataset 中获取开源的 ProgrammableWeb 爬取清洗后的数据集 m-a_edges.csv。观察该 csv 文件，发现其存在两列，有对应的 source、target 列标签，每行有 Mashup 和其对应的 WebAPI。

	A	B	C	D	E	F	G
1	sourcetarget						
2	Mashup: CouponRoots/api/coupon						
3	Mashup: Raise the Money/api/nationbuilder						
4	Mashup: AnythingToHTML/api/hpe-haven-ondemand-view-document						
5	Mashup: Velocipedia/api/mapbox						
6	Mashup: Api Expert - MyMemory Language Translator/api/mymemory						
7	Mashup: Adtegrity/api/nationbuilder						

数据处理的第一步是导入所需数据包：

```
import csv
import pandas as pd
from scipy import stats
import re
import matplotlib.pyplot as plt
import json
from  scipy.stats import chi2_contingency
import networkx as nx
```

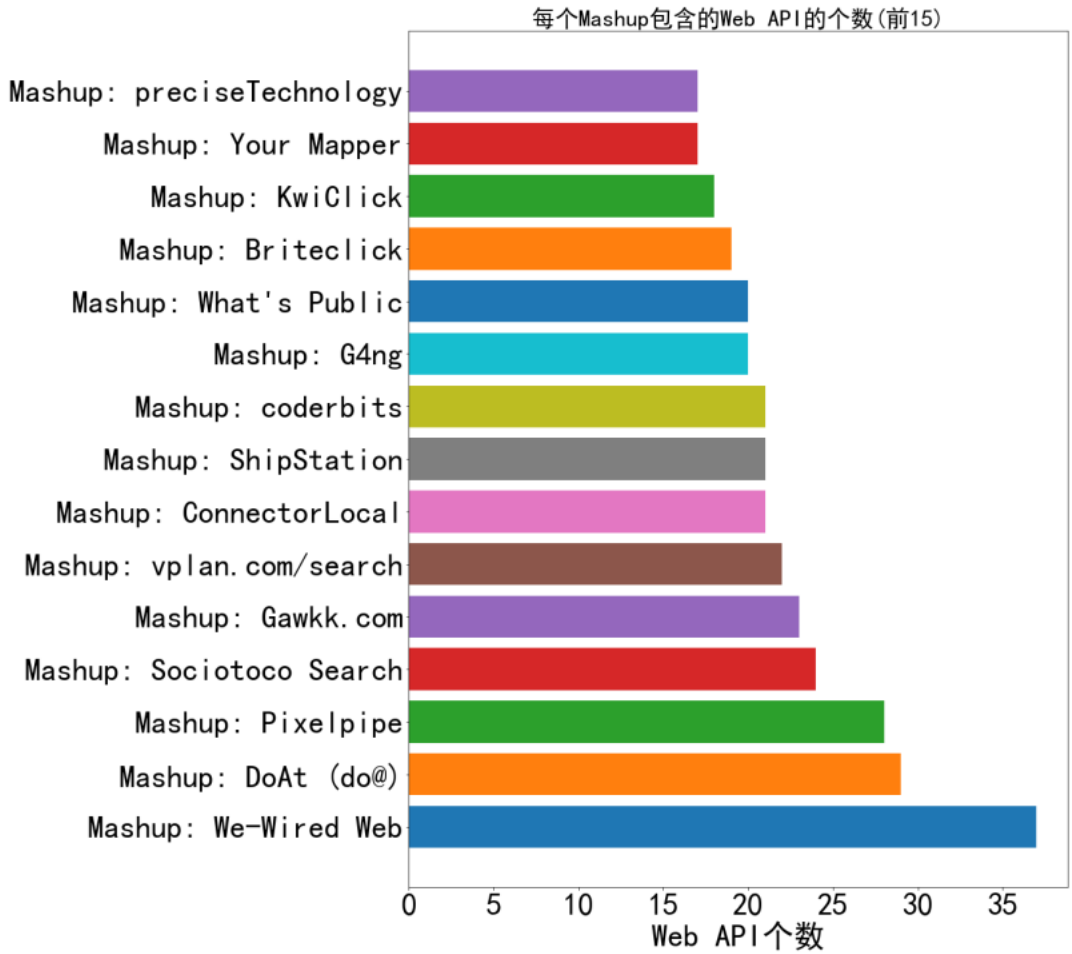
接下来，将数据集根据 source 分组后，按照 WebAPI 数量降序排列，将重新整理后的数据存入 csv 文件，得到：

	A	B	C	D	E
1	MashupWebApi个数				
2	Mashup: We-Wired Web37				
3	Mashup: DoAt (do@)29				
4	Mashup: Pixelpipe28				
5	Mashup: Sociotoco Search24				
6	Mashup: Gawkk.com23				
7	Mashup: vplan.com/search22				

再将含 WebAPI 数量最多的前 15 个 Mashup 绘制条形图：

```
for i,element in enumerate(APInum):
    if i<15:
        plt.barh(element[0],element[1])
```

```
    wr.writerow([element[0],element[1]])
    plt.rcParams["font.sans-serif"] = ["SimHei"]
    plt.rcParams["axes.unicode_minus"] = False
    plt.rcParams['font.size'] = 25
    plt.title("每个 Mashup 包含的 Web API 的个数(前 15)")
    plt.xlabel("Web API 个数")
    plt.rcParams['figure.figsize'] = (23, 13)
    plt.savefig("python 期末报告/每个 Mashup 中的包含 Web API 个数.png",bbox_inches='tight')
```



由图中，也可看出包含最多 WebAPI 的 Mashup 是 We-Wire Web。

1.2 统计每个 Web API 被使用的次数

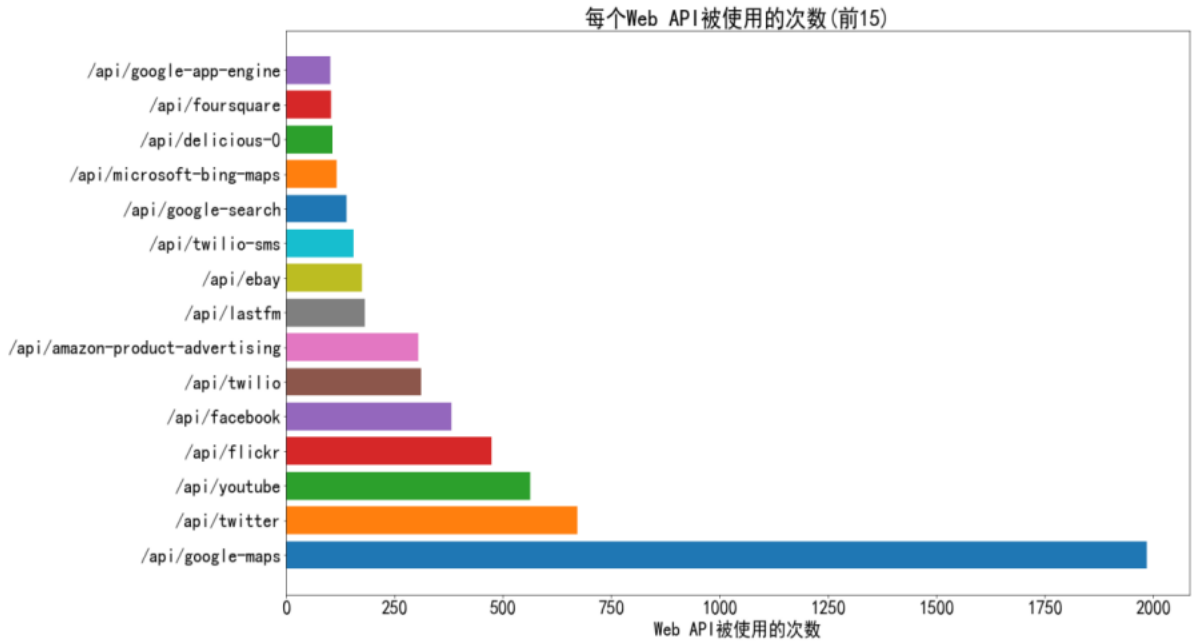
将数据集中 target 进行计数后，将 WebAPI 按照其使用次数降序排列，将重新整理后的数据存入 csv 文件，得到：

	A	B	C	D	E	F
1	urlWebAPI使用次数					
2	/api/google-maps1984					
3	/api/twitter671					
4	/api/youtube562					
5	/api/flickr474					
6	/api/facebook381					
7	/api/twilio311					
8	/api/amazon-product-advertising304					

再将使用次数最多的 15 个 WebAPI 进行可视化，绘制条形图：

```
for i,element in enumerate(WAnum):
    if i<15:
        plt.barh(element[0],element[1])
    wr.writerow([element[0],element[1]])

plt.title("每个 Web API 被使用的次数(前 15)")
plt.xlabel("Web API 被使用的次数")
plt.savefig("python 期末报告/每个 Web API 被使用的次数.png",bbox_inches='tight')
```



从图片中，我们很容易观察到 google-maps 具有远超过其他 url 的 WebAPI 使用次数。

1.3 统计 Web API 提供商（URL 网址）发布 Web API 的个数

首先，利用正则表达式区分存在连接符”-”间隔和不存在连接符的 WebAPI：

```
dt = {}
r1 = re.compile(r"/api/(.*?)\-.*")
r2 = re.compile(r"/api/(.*)")
```

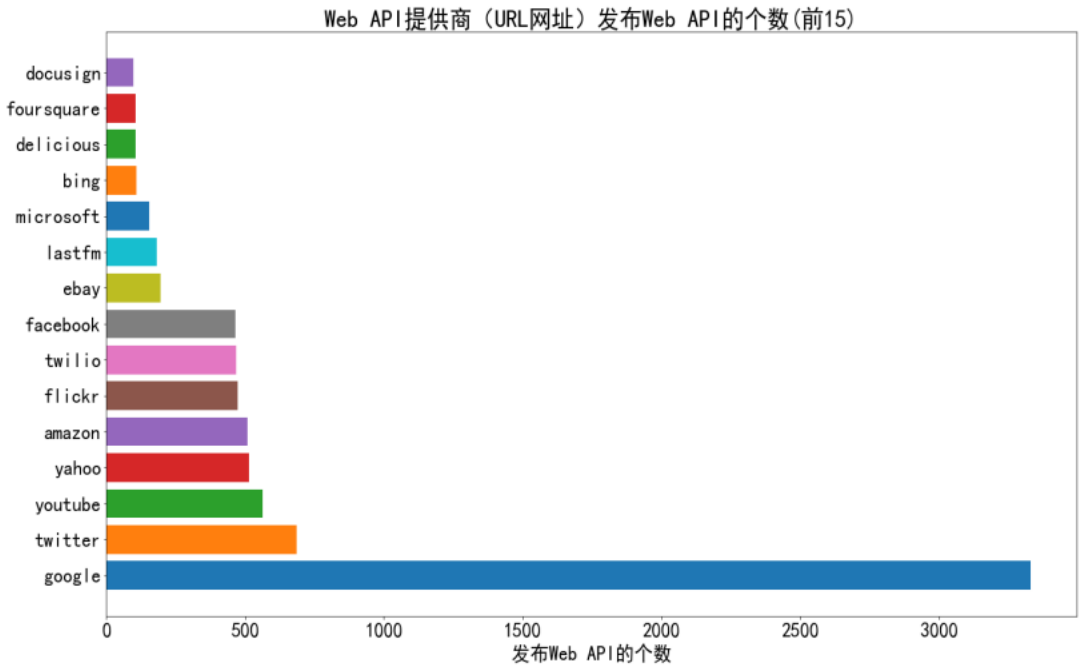
通过遍历，将 WebAPI 提供商及其发布 WebAPI 的个数组成的数据集按照发布 WebAPId 的个数降序排列：

```
for el in df['target']:
    if '-' not in el:
        tp = r2.match(el)
        if tp.group(1) not in dt.keys():
            dt[tp.group(1)] = 0
        dt[tp.group(1)] += 1
    else:
        tp = r1.match(el)
        if tp.group(1) not in dt.keys():
            dt[tp.group(1)] = 0
        dt[tp.group(1)] += 1
pAPI = list(dt.items())
pAPI.sort(key=lambda x:x[1],reverse=True)
```

数据存入 csv 的得到：

	A	B	C	D	E
1	Web API提供	发布Web API的个数			
2	google	3329			
3	twitter	686			
4	youtube	562			
5	yahoo	513			
6	amazon	508			
7	flickr	474			

再将发布 WebAPI 个数排名前 15 的 WebAPI 提供商绘制统计条形图：



可以看到，google 是发布 WebAPI 数量最多的提供商。

二、Mashup 与所调用 Web API 的关联情况分析

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	tpurlnamestetoetcoacac												
2	mashuphttp://www.couponroots.comMashup: CouponRoots2015-08-042222-02-222222-02-22CouponsTrueTrue												
3	mashuphttps://raisethemoney.com/Mashup: Raise the Money2014-10-122222-02-222222-02-22FundingTrueTrue												
4	mashuphttps://paraggupta1993.github.io/AnythingToHTML/Mashup: AnythingToHTML2015-05-122222-02-222222-02-22DocumentsTrueTrue												
5	mashuphttp://velocipedia.euMashup: Velocipedia2014-11-042019-07-232222-02-22MappingTrueFalse												
6	mashuphttps://github.com/hyeshik/moves-daily-lifeMashup: Moves Daily Life2015-04-282222-02-222222-02-22FitnessTrueTrue												

首先，我们 https://github.com/HIT-ICES/Correted-ProgrammableWeb-dataset 中获取开源的 ProgrammableWeb 爬取清洗后的数据集 mashup_nodes_estimator.csv。观察该 csv 文件：

根据 github 上注释，我们了解到列标签对应的数据含义：

Column name	Meaning
tp	Type. API or Mashup
url	The URL of an API or Mashup
name	The API or Mashup's title
st	Submit date
et	Corrected dead date
oet	Dead date provided in PW
c	Category
oac	Corrected accessibility
ac	Accessibility provided in PW

根据题目要求，我们根据数据的类别 Category 进行分组，将分组后的类别、Mashup 和 WebAPI 名称写入 csv 文件，如图所示：

	A	B	C	D	E	F	G	H
1	类别MpWA							
2	3DMashup: Printr/api/printr							
3	3DMashup: Virtual Earth 3D Flight Simulator/api/microsoft-bing-maps							
4	3DMashup: English Lake District/api/amazon-product-advertising							
5	3DMashup: English Lake District/api/google-adsense							
6	3DMashup: English Lake District/api/google-custom-search							
7	3DMashup: English Lake District/api/google-earth							
8	3DMashup: FotoViewr/api/flickr							

接下来，我们了解到当有多个个定量变量时，使用相关性检验，而当有多个类别变量时，使用卡方检验其独立性。

卡方检验是在样本量较大时用于列联表分析的一种统计假设检验。简单地说，该检验主要用于检查两个类别变量(列联表的两个维度)在影响检验统计信息(表中的值)方面是否独立。卡方统计量正是衡量观察值和期望值之间存在的偏差，卡方统计量越大即为偏差越大，变量之间相关性高；反之，变量之间相关性低。

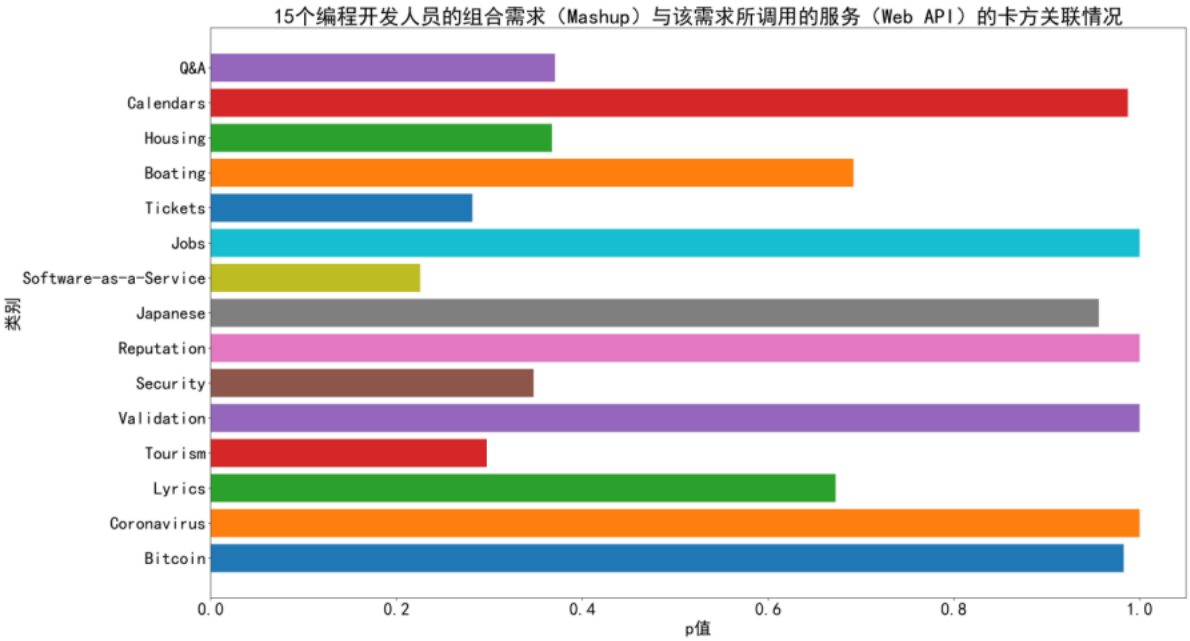
首先，建立卡方矩阵：

```
for k,v in group1:
    cols = set()
    rows = set()
    for el in v.values:
        rows.add(el[1])
        cols.add(el[2])
    Chi = pd.DataFrame(data=0,columns=list(cols),index=list(rows))
    for el in v.values:
        Chi.loc[el[1], el[2]] += 1
```

我们利用卡方检验对经过处理的数据集进行检验，以此判断 Mashup 与所调用 Web API 的关联情况，我们将检验结果所得到的 p 值，自由度和置信水平 95%情况下的相关性判断写入 csv，得到：

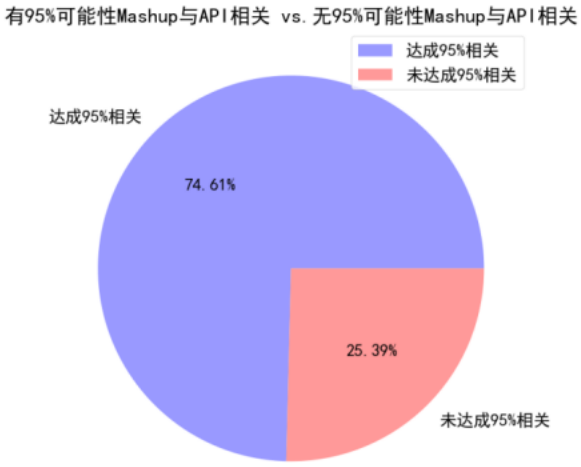
	A	B	C	D	E	F	G
1	类别	p值	df	95%			
2	3D	0.7595624	120	TRUE			
3	API	0.3531649	24	TRUE			
4	API Manage	1	0	FALSE			
5	Accessibi	1	0	FALSE			
6	Accounting	0.3656593	90	TRUE			
7	Accounts	1	0	FALSE			

随机抽取 15 个不同类别的 Mashup 与该需求所调用的服务 Web API，将其经检验所得卡方 p 值绘制条形图：



可以看出随机抽取的数据中大部分组合需求与该需求所调用的服务之间有较大的卡方 p 值，即不能拒绝原假设，故而两者关联性较强。

我们也可以用饼图展示出，所有不同类别下能在 95%的置信水平下认为组合需求与该需求所调用的服务有较强关联性的比例：



可以看出，大部分即 74.61%的类别数据中，Mashup 与其 WebAPI 有 95%的可能性存在相关。

三、从非功能视角统计 WebAPI 被使用的情况

首先，我们 https://github.com/HIT-ICES/Correted-ProgrammableWeb-dataset 中获取开源的 ProgrammableWeb 爬取清洗后的数据集 active_mashups_data.txt 与 deadpool_mashups_data.txt，作为该问题的处理对象。

通过查阅资料，本文选择用无向图结构来反映 WebAPI 之间的兼容性。一个图表是由数个顶点和边组成的。然而，无向图的边是没方向的，即两个相连的顶点可以互相抵达，可以反映顶点之间的兼容性。

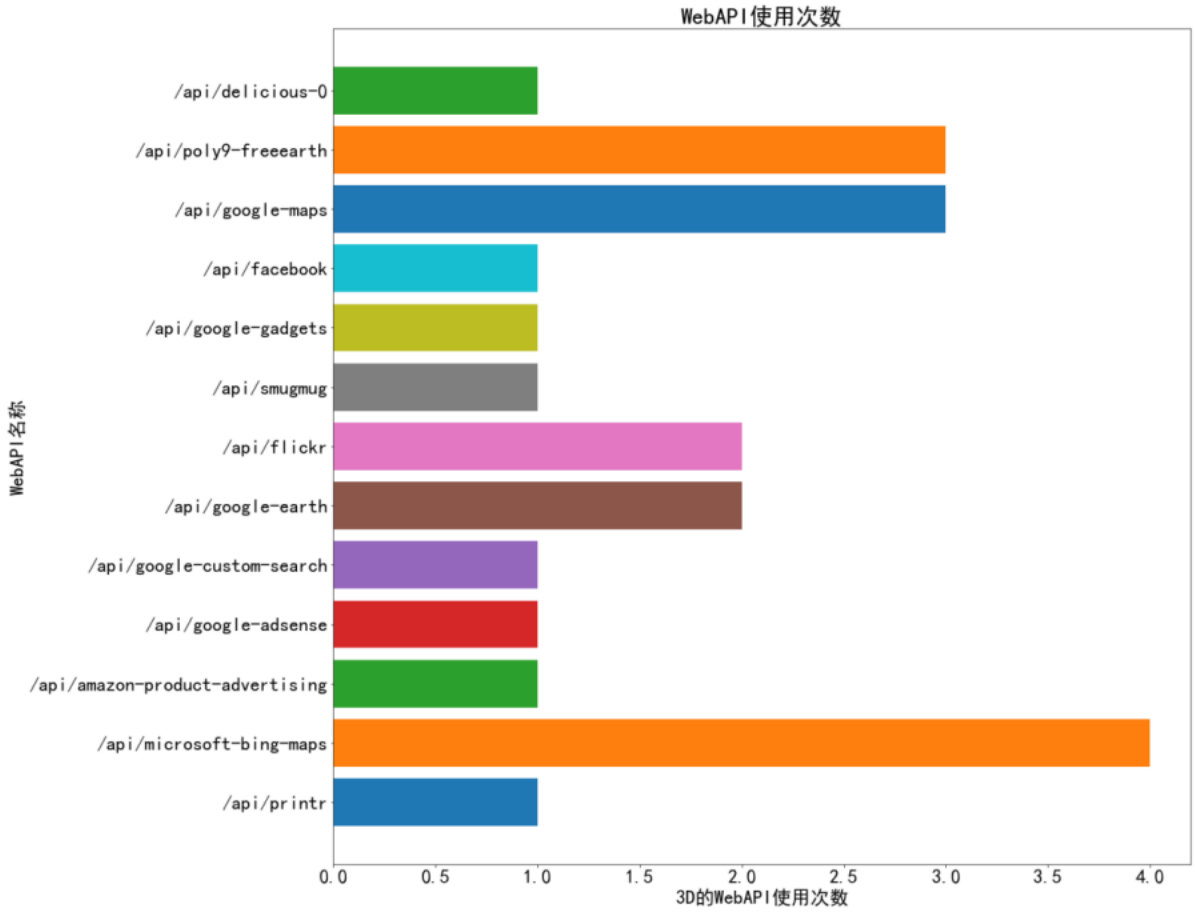
首先，利用函数建立无向图的边：

```
def f(path):
    with open(path,"r") as file:
        data = json.load(file)
        for v in data:
            if v!=None:
                ra = v['related_apis']
                for i in range(len(ra)):
                    if ra[i]!=None:
                        url_ = ra[i]['url']
                        if url_ not in edge.keys():
                            edge[url_]={}
                        for j in range(i+1,len(ra)):
                            if ra[j]!=None:
                                url__ = ra[j]['url']
                                if url__ not in edge[url_].keys():
                                    edge[url_][url__]=0
                                edge[url_][url__]+=1
                                if url__ not in edge.keys():
                                    edge[url__]={}
                                if url_ not in edge[url__].keys():
                                    edge[url__][url_]=0
                                edge[url__][url_]+=1
```

接下来，选取 5 个不同类别的数据，统计其 WebAPI 使用次数，为画无向图做准备。首先以，3D 类别的数据为例：

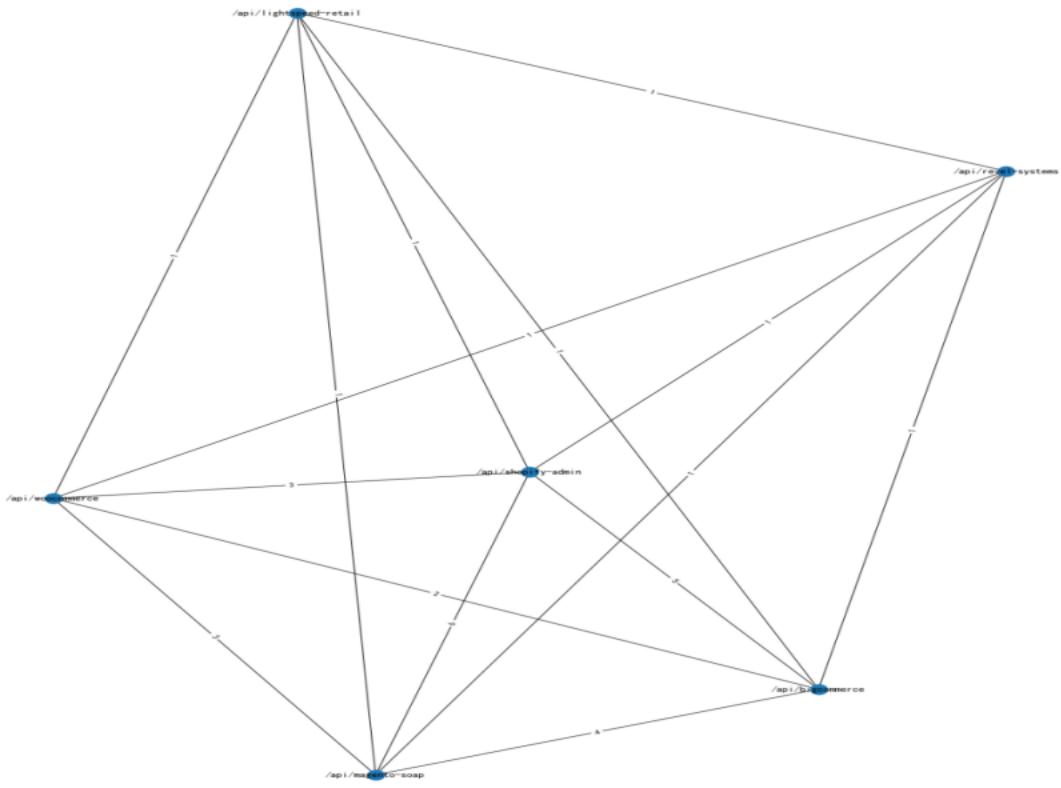
```
num = 0
for k,v in group1:
    if num==5:
```

```
        break
    num+=1
    enum = dict()
    for el in v.values:
        if el[2] not in enum.keys():
            enum[el[2]]=0
        enum[el[2]]+=1
    lt = list(enum.items())
    for i in lt:
        plt.barh(i[0],i[1])
    plt.title("WebAPI 使用次数")
    plt.ylabel("WebAPI 名称")
    plt.xlabel(k+"的 WebAPI 使用次数")
    plt.savefig("python 期末报告/"+k+"使用数.png",bbox_inches='tight')
```



再进行无向图的构建，以 API Management 为例：

```
        F = nx.Graph()
        l = len(lt)
        for i in range(l):
            for j in range(i+1,l):
                a = lt[i][0]
                b = lt[j][0]
                if a in edge.keys() and b in edge[a].keys():
                    F.add_edge(a, b, weight=edge[a][b])
        position = nx.spring_layout(F)
        label = nx.get_edge_attributes(F, 'weight')
        plt.rcParams["font.sans-serif"] = ["SimHei"]
        plt.rcParams["axes.unicode_minus"] = False
        plt.rcParams['figure.figsize'] = (15, 20)
        plt.rcParams['font.size'] = 40
        nx.draw(F, position, with_labels=True)
        nx.draw_networkx_edge_labels(F, position, edge_labels=label,
font_weight='bold',fontsize=50)
        plt.savefig("python 期末报告/"+k+"的无向图.png")
```



可以看出，从非功能视角，对功能类别相等的若干 Web API 进行分析，在 Category 取值相等的 Web API 集合内，以 API Management 为例，上图中 api/shopify-admin 和 api/magento-soap 之间有较高的兼容性。