徐炎炀-2201210464-Python大作业

Git地址

Mashup-Api-homework: PKUSS-Python (gitee.com)

Juv-Dybala/Mashup-Api-homework: PKUSS-Python (github.com)

选择**技术类大作业**：Programmable Web需求关键词视角下的历史调用记录分析

首先从HIT-ICES/Correted-ProgrammableWeb-dataset: Data Correction and Evolution Analysis of the ProgrammableWeb Service Ecosystem (github.com)中获取数据，其中为参考论文《Data Correction and Evolution Analysis of the ProgrammableWeb Service Ecosystem》开源ProgrammableWeb爬取清洗后的数据集

本作业需要的package

```
# 调用包
import csv
import os
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
from tld import get_tld
```

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：

第一题

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

```
# 第一题选择m-a_edges.csv文件
data1 = pd.read_csv("data/m-a_edges.csv", delimiter="\t")
data1
```

```
data1 = pd.read_csv("data/m-a_edges.csv", delimiter="\t")
data1
```

	source	target
0	Mashup: CouponRoots	/api/coupon
1	Mashup: Raise the Money	/api/nationbuilder
2	Mashup: AnythingToHTML	/api/hpe-haven-ondemand-view-document
3	Mashup: Velocipedia	/api/mapbox
4	Mashup: Api Expert - MyMemory Language ...	/api/mymemory
5	Mashup: Adtegrity	/api/nationbuilder
6	Mashup: Particle Reviews	/api/rotten-tomatoes
7	Mashup: DocuSign for Microsoft Word	/api/docusign-enterprise

12614 rows × 2 columns 在新选项卡中打开

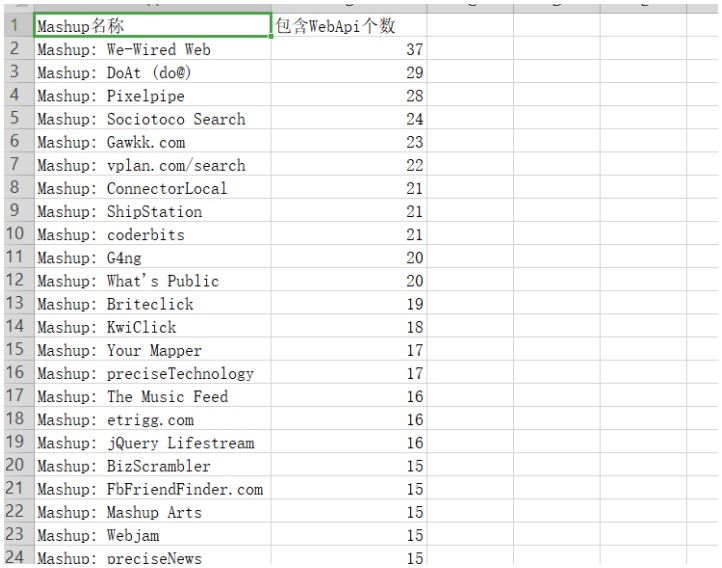
1.1 统计每个Mashup包含的Web API个数

```
group_by = data1.groupby('source')

count_MashupApi = []
for key,values in group_by:
    count_MashupApi.append([key,len(values.values)])

count_MashupApi.sort(key=lambda x:x[1],reverse=True)

# 写入csv
out = open("./problem1/每个Mashup包含WebApi个数.csv", "w", encoding='utf-8-sig', newline='')
write = csv.writer(out,delimiter=',')
write.writerow(["Mashup名称","包含WebApi个数"])
for i,item in enumerate(count_MashupApi):
    if i<20:
        plt.bar(item[0],item[1])
    write.writerow([item[0],item[1]])
f = pd.read_csv("./problem1/每个Mashup包含WebApi个数.csv", delimiter="\t")
print(f)

# 画统计图
plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.rcParams["axes.unicode_minus"] = False
plt.title("Mashup包含Web Api个数(top 20)")
plt.xlabel("Mashup名称")
plt.ylabel("WebApi个数")
plt.xticks(rotation=-90)
plt.savefig("./problem1/每个Mashup包含WebApi个数.png",bbox_inches='tight')
plt.show()
plt.close()
```

输出到控制台

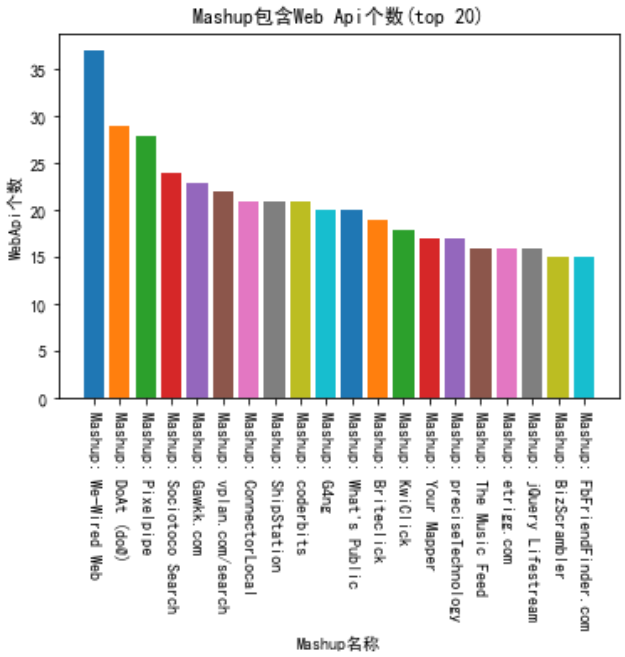
```
                    Mashup名称,包含WebApi个数
0                Mashup: We-Wired Web,37
1                  Mashup: DoAt (do@),29
2                   Mashup: Pixelpipe,28
3             Mashup: Sociotoco Search,24
4                    Mashup: Gawkk.com,23
...                                   ...
5676  Mashup: Website Traffic Estimator,1
5677         Mashup: Website-Grader.com,1
5678                     Mashup: Websta,1
5679                Mashup: Wectar Gems,1
```

写入到csv文件中



	Mashup名称	包含WebApi个数
2	Mashup: We-Wired Web	37
3	Mashup: DoAt (do@)	29
4	Mashup: Pixelpipe	28
5	Mashup: Sociotoco Search	24
6	Mashup: Gawkk.com	23
7	Mashup: vplan.com/search	22
8	Mashup: ConnectorLocal	21
9	Mashup: ShipStation	21
10	Mashup: coderbits	21
11	Mashup: G4ng	20
12	Mashup: What's Public	20
13	Mashup: Briteclick	19
14	Mashup: KwiClick	18
15	Mashup: Your Mapper	17
16	Mashup: preciseTechnology	17
17	Mashup: The Music Feed	16
18	Mashup: etrigg.com	16
19	Mashup: jQuery Lifestream	16
20	Mashup: BizScrambler	15
21	Mashup: FbFriendFinder.com	15
22	Mashup: Mashup Arts	15
23	Mashup: Webjam	15
24	Mashup: preciseNews	15

画统计图



由统计图可得，We-Wired Web Mashup包含着最多得Web Api，多达37个；绝大多数的Mashup包含的Mashup数均小于20个。

1.2 统计每个Web API被使用的次数

```
target = data1['target'].value_counts()
count_WebApiUse = list(target.items())

count_WebApiUse.sort(key=lambda x:x[1],reverse=True)

# 写入csv
out = open("./problem1/WebApi使用次数.csv", "w", encoding='utf-8-sig', newline='')
write = csv.writer(out,delimiter='\t')
write.writerow(["url","WebApi使用次数"])
for i,item in enumerate(count_WebApiUse):
    if i<20:
        plt.bar(item[0],item[1])
    write.writerow([item[0],item[1]])
f = pd.read_csv("./problem1/WebApi使用次数.csv", delimiter="\t")
print(f)

# 画统计图
plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.rcParams["axes.unicode_minus"] = False
plt.title("WebApi使用次数(top20)")
plt.xlabel("url")
plt.ylabel("WebApi使用次数")
plt.xticks(rotation=-90)
plt.savefig("./problem1/WebApi使用次数.png",bbox_inches='tight')
plt.show()
plt.close()
```

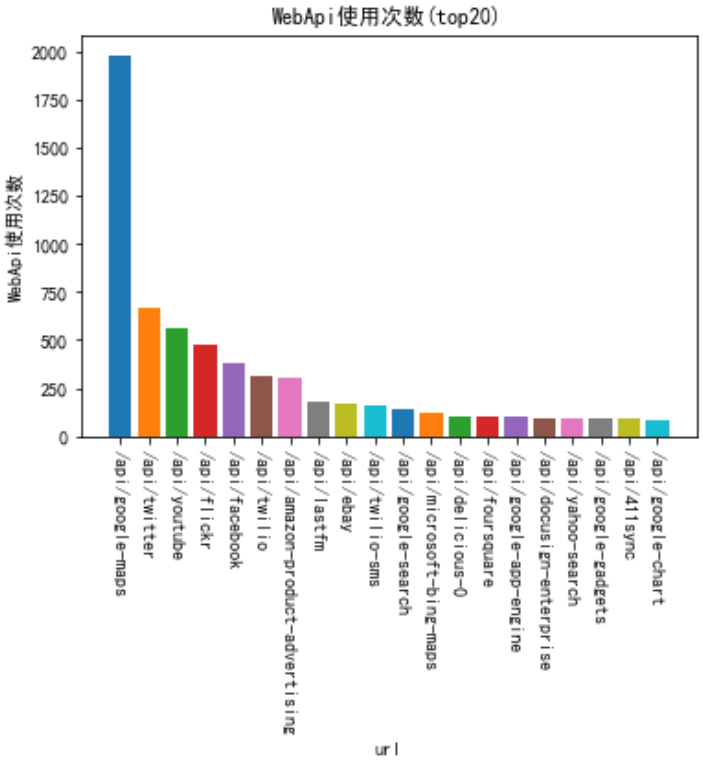
输出到控制台

	url	WebApi使用次数
0	/api/google-maps	1984
1	/api/twitter	671
2	/api/youtube	562
3	/api/flickr	474
4	/api/facebook	381
...	...	...
1503	/api/revel-systems	1
1504	/api/lightspeed-retail	1
1505	/api/microsoft-cognitive-services-emotion	1
1506	/api/tixik	1

写入到csv文件中

	url	WebApi使用次数
2	/api/google-maps	1984
3	/api/twitter	671
4	/api/youtube	562
5	/api/flickr	474
6	/api/facebook	381
7	/api/twilio	311
8	/api/amazon-product-advertis	304
9	/api/lastfm	181
10	/api/ebay	174
11	/api/twilio-sms	156
12	/api/google-search	139
13	/api/microsoft-bing-maps	117
14	/api/delicious-0	106
15	/api/foursquare	104
16	/api/google-app-engine	102
17	/api/docusign-enterprise	97
18	/api/yahoo-search	95
19	/api/google-gadgets	95
20	/api/411sync	93
21	/api/google-chart	84
22	/api/geonames	79
23	/api/yahoo-maps	78
24	/api/google-adsense	77

画统计图



由统计图可得，Web Api中google-maps被使用的次数最高，多达1984次，接近第二名twitter的3倍。这和我们的生活常识较为符合，大多常见的App均需要地理信息，故谷歌地图Api的使用较为频繁。在统计中，绝大多数Api的使用次数少于100次，仅在特定场合发挥自己的作用。

1.3 统计Web API提供商（URL网址）发布Web API的个数

```
Dic = dict()
def solve(path):
    with open(path,"r") as f:
        li = json.load(f)
        for dic in li:
            from_api = dic["from_api"]
            url = from_api["url"]
            visit_status = dic["visit_status"]
            for visit_statu in visit_status:
                visit_url = str(visit_statu['visit_url'])
                visit_url = visit_url.strip()
                try:
                    result = get_tld(visit_url, as_object=True,fix_protocol=True).fld
                    if result in Dic:
                        Dic[result].add(url)
                    else:
                        Dic[result]=set()
                        Dic[result].add(url)
                except Exception as e:
                    pass
for root, dirs, files in os.walk("./data/raw/accessibility/api_accessibility"):
    for file in files:
        solve(os.path.join(root, file))
dic = {}
```

```
for it in Dic:
    dic[it]=len(Dic[it])
count_UrlProvideApi = list(dic.items())

count_UrlProvideApi.sort(key=lambda x:x[1],reverse=True)

# 写入csv文件
with open("./problem1/Web API提供商（URL网址）发布Web API的个数.csv", "w", encoding='utf-8-sig', newline='') as out:
    write = csv.writer(out, delimiter=",")
    write.writerow(["Web API提供商", "次数"])
    for i, it in enumerate(count_UrlProvideApi):
        if i<20:
            plt.bar(it[0],it[1])
        write.writerow([it[0], it[1]])
f = pd.read_csv("./problem1/Web API提供商（URL网址）发布Web API的个数.csv", delimiter="\t")
print(f)

# 画统计图
plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.rcParams["axes.unicode_minus"] = False
plt.title("Web API提供商（URL网址）发布Web API的个数(top20)")
plt.xlabel("Web API提供商")
plt.ylabel("提供商发布WebApi个数")
plt.xticks(rotation=-90)
plt.savefig("./problem1/Web API提供商（URL网址）发布Web API的个数.png",bbox_inches='tight')
plt.show()
plt.close()
```

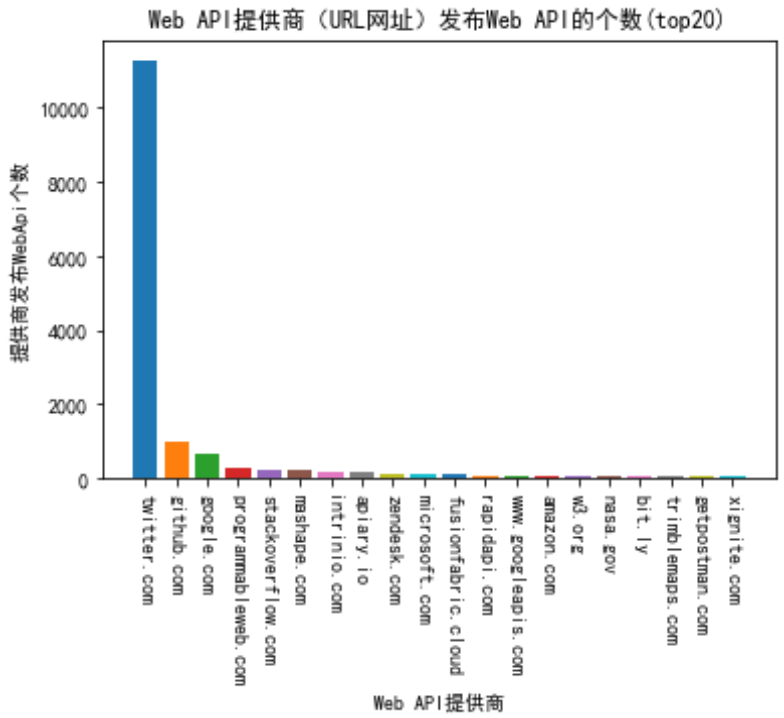
输出到控制台

```
                Web API提供商     次数
0               twitter.com  11268
1                github.com   1005
2                google.com    687
3      programmableweb.com    293
4        stackoverflow.com    221
...                    ...    ...
15817        cloudyrec.com      1
15818        seattle20.com      1
15819         geekwire.com      1
15820      transloadit.com      1
15821           lymbix.com      1
```

写入到csv文件

	Web API提供商	次数
1	twitter.com	11268
2	github.com	1005
3	google.com	687
4	programmableweb.com	293
5	stackoverflow.com	221
6	mashape.com	221
7	intrinio.com	195
8	apiary.io	188
9	zendesk.com	145
10	microsoft.com	137
11	fusionfabric.cloud	102
12	rapidapi.com	90
13	www.googleapis.com	88
14	amazon.com	84
15	w3.org	76
16	nasa.gov	73
17	bit.ly	70
18	trimblemaps.com	69
19	getpostman.com	68
20	xignite.com	61
21	alk.com	60
22	atlassian.net	60

画统计图



在统计图中，十分直观地体现出twitter.com发布了最多的Web Api，多达11268个，是第二名github.com的十多倍。推特作为在世界上使用范围非常广的社交软件，发布的Web Api可以很好地方便用户的生活的各个方面。此外，大部分URL网址发布的Web Api个数少于100个，这些URL大多仅在特定领域提供服务。

第二题

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。

```
# 第二题自己处理出Mashup和Web Api对应的csv文件，并读取Mashup_nodes_estimator.csv文件
out = open("./problem2/MashupAndWebApi.csv", "w", encoding='utf-8-sig', newline='')
write = csv.writer(out,delimiter=',')
write.writerow(["category", "title","url"])

data2 = pd.read_csv("data/Mashup_nodes_estimator.csv", delimiter="\t")
group_by = data2.groupby('c')
for key, df in group_by:
    dic = {}
    for name in df['name']:
        for it in data1[data1['source'] == name]['target'].values:
            write.writerow([key,name, it])
out.close()
```

自己处理得到的csv文件

	category	title	url
1	3D	Mashup: Printr	/api/printr
2	3D	Mashup: Virtual Earth 3D Flight Simulator	/api/microsoft-bing-maps
3	3D	Mashup: English Lake District	/api/amazon-product-advertising
4	3D	Mashup: English Lake District	/api/google-adsense
5	3D	Mashup: English Lake District	/api/google-custom-search
6	3D	Mashup: English Lake District	/api/google-earth
7	3D	Mashup: FotoViewr	/api/flickr
8	3D	Mashup: FotoViewr	/api/smugmug
9	3D	Mashup: FotoViewr	/api/google-gadgets
10	3D	Mashup: FotoViewr	/api/facebook
11	3D	Mashup: Gaiagi Driver - 3D Driving Simulator	/api/google-maps
12	3D	Mashup: Gaiagi Driver - 3D Driving Simulator	/api/microsoft-bing-maps
13	3D	Mashup: Gaiagi Driver - 3D Driving Simulator	/api/google-earth
14	3D	Mashup: 360 Cities	/api/google-maps
15	3D	Mashup: Czech 3Dtour	/api/google-maps
16	3D	Mashup: Virtual Earth 3D Scenes	/api/microsoft-bing-maps
17	3D	Mashup: Virtual Earth 3D Scenes	/api/microsoft-bing-maps
18	3D	Mashup: UFOMaps 3D	/api/poly9-freeearth
19	3D	Mashup: flickrvision 3D	/api/flickr
20	3D	Mashup: flickrvision 3D	/api/poly9-freeearth

使用**卡方检验**分析关联情况

```
MashupAndApi_df = pd.read_csv("./problem2/MashupAndWebApi.csv", delimiter=",")
group_by_category = MashupAndApi_df.groupby('category')
columns = set()
indexes = set()
df = pd.DataFrame(data=None)
p_values = []

# 卡方检验
chi_square = open("./problem2/chi_square.csv", "w", encoding='utf-8-sig', newline='')
write = csv.writer(chi_square)
num1 = 0
num2 = 0
# 置信度:取90% 80% 70%
CF = 0.70
write.writerow(["category","卡方检验统计量", r"p-value","自由度","{:.0%}置信度下是否关联".format(CF)])
```

```
for key,value in group_by_category:
    columns = set()
    indexes = set()
    df = pd.DataFrame(data=None)
    for item in value.values:
        columns.add(item[1])
        indexes.add(item[2])
    chi_square_df = pd.DataFrame(data=0,columns=list(indexes),index=list(columns))
    for item in value.values:
        chi_square_df.loc[item[1], item[2]]+=1
    # 卡方检验的 卡方值,p值,自由度,预期频率
    chi2,p,free,exp = chi2_contingency(chi_square_df)
    p_values.append([key,p])
    # 置信度CF时的卡方分位数
    quantile = stats.chi2.ppf(q=CF,df=free)

    # 若卡方值大于分位数，则大于CF的概率二者相关;而1-p值是相关的具体概率,p值越小相关性越高
    # if chi2 >= quantile:
    if p < 1-CF:
        num1 += 1
        print([key,chi2,quantile,True,p])
        write.writerow([key,str(chi2),str(p),str(free),True])
    else:
        num2 += 1
        print([key,chi2,quantile,False,p])
        write.writerow([key,str(chi2),str(p),str(free),False])

    chi_square_df.to_csv("./problem2/chi square/"+key+".csv")
chi_square.close()

p_values.sort(key=lambda x:x[1],reverse=True)
for i,item in enumerate(p_values):
    if i==0:
        plt.bar(item[0],item[1])
    elif p_values[i-1][1]-p_values[i][1]>0.01:
            plt.bar(item[0],item[1])

plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.rcParams["axes.unicode_minus"] = False
plt.title("Mashup与Web Api关联情况")
plt.xlabel("category")
plt.ylabel("卡方检验的p值")
plt.xticks(rotation=-90)
plt.savefig("./problem2/Mashup与WebApi关联情况.png",bbox_inches='tight')
plt.show()
plt.close()

# 扇形图体现不同Category中有无联系的比例（置信度CF）
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
labels = ["True", "False"]
x = [num1, num2]
explode = [0, 0]
title="不同Category中是否有{:.0%}的把握认为Mashup与Web Api有联系分布".format(CF)
plt.pie(x=x, labels=labels, explode=explode, shadow=True, autopct="%0.2f%%")
plt.legend()
plt.title(title)
plt.savefig("./problem2/扇形图.png",bbox_inches='tight')
plt.show()
plt.close()
```

输出各category的列联表（以Database为例）

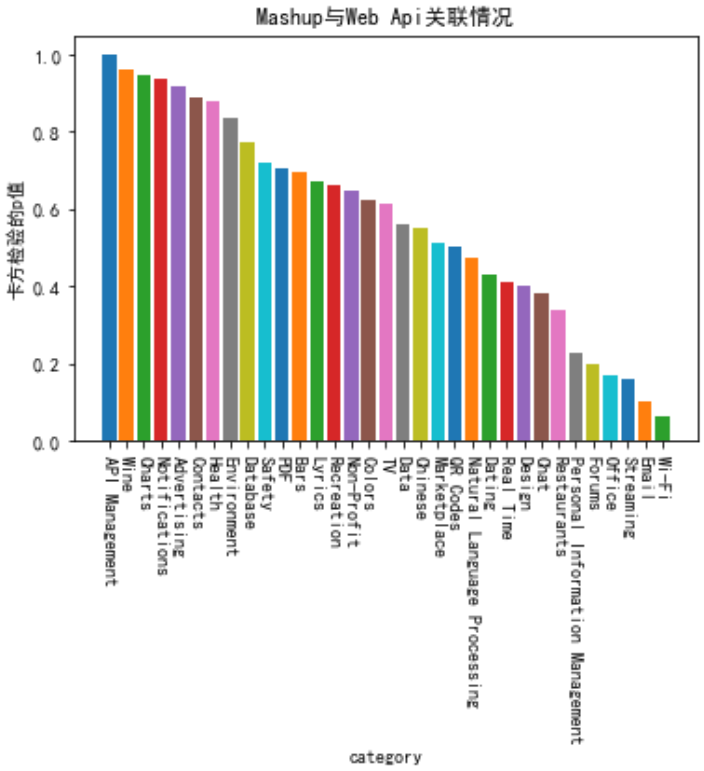
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		/api/github	/api/twitter	/api/amazon-	/api/youtube	/api/mailchimp	/api/twilio	/api/twilio-sms	/api/imsas	/api/notifo	/api/google-maps	/api/baby-	/api/facebook	/api/linkedin
2	Mashup: DBStract.com	0	1	0	0	1	1	1	0	1	1	0	0	0
3	Mashup: Talk Rain	0	0	0	0	0	1	0	0	0	0	0	0	0
4	Mashup: hearSay	0	0	0	0	0	1	0	0	0	0	0	0	0
5	Mashup: Fast Criminal Records	0	0	0	0	0	0	0	1	0	0	0	0	0
6	Mashup: Elseif	1	1	0	1	0	0	0	0	0	0	0	0	0
7	Mashup: PitchVater	0	0	0	0	0	1	0	0	0	0	0	0	0
8	Mashup: SDB Explorer	0	0	1	0	0	0	0	0	0	0	0	0	0
9	Mashup: Baby Name Meanings	0	0	0	0	0	0	0	0	0	0	1	0	0
10	Mashup: CriminalDatabase	0	0	0	0	0	0	0	1	0	0	0	0	0
11	Mashup: Datantify	1	1	0	0	0	0	0	0	0	0	0	1	1

对各列联表进行卡方检验的结果写入csv文件中（以70%置信度为例）

	A	B	C	D	E	F	G
1	category	卡方检验统计量	p-value	自由度	70%置信度下是否关联		
2	3D	108.7777778	0.7595624	120	FALSE		
3	API	26	0.353164933	24	FALSE		
4	API Management	0	1	0	FALSE		
5	Accessibility	0	1	0	FALSE		
6	Accounting	94	0.36565925	90	FALSE		
7	Accounts	0	1	0	FALSE		
8	Activity Streams	0	1	0	FALSE		
9	Addresses	0	1	1	FALSE		
10	Adult	8	0.238103306	6	TRUE		
11	Advertising	126.5833333	0.9178087	150	FALSE		
12	African	0	1	0	FALSE		
13	Aggregation	411.8531136	0.931838088	456	FALSE		
14	Air Travel	187.0666667	0.870705717	210	FALSE		
15	Algorithms	0	1	1	FALSE		
16	Analytics	833.8571429	0.865151211	880	FALSE		
17	Animals	133.4888889	0.471771616	133	FALSE		
18	Application Develop	1304.275225	0.999994577	1536	FALSE		
19	Applications	54	0.255912602	48	TRUE		
20	Art	258.2380952	0.996684348	323	FALSE		
21	Artificial Intelli	3	0.22313016	2	TRUE		
22	Asia	0	1	0	FALSE		
23	Astrology	3	0.22313016	2	TRUE		
24	Astronomy	16.88888889	0.325549938	15	FALSE		
25	Auctions	1047.209546	1	1517	FALSE		
26	Audio	204.125	0.312533366	195	FALSE		
27	Australian	28.38888889	0.813055	36	FALSE		
28	Auto	612.4658586	0.851564932	650	FALSE		

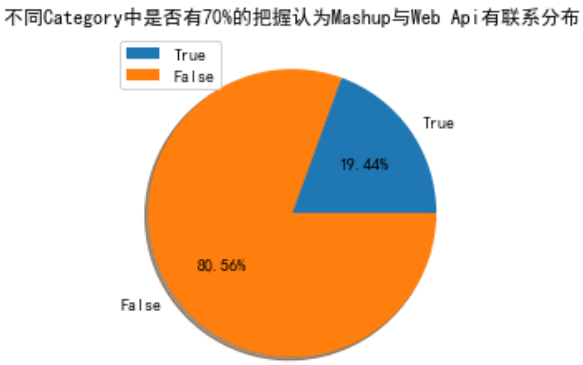
画统计图：Mashup与Web Api的关联情况

（通过卡方检验的p值查看，p值越低关联度越高）

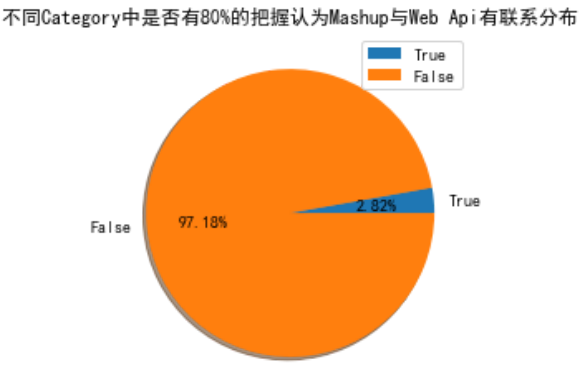


画扇形图记录不同置信度下关联情况的判断

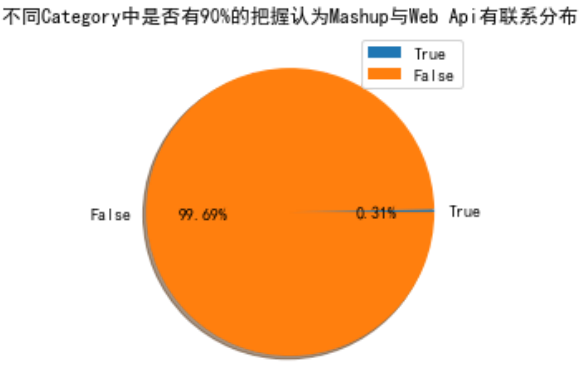
70%置信度



80%置信度



90%置信度



根据扇形统计图可以较为直观地发现在大多数Category中Mashup和Web Api之间的联系并不显著。在90%置信度的情况下仅有0.31%的Category可认为Mashup和Web Api有联系；而在70%置信度的情况下这一比例增长到19.44%，这些Category中Mashup和Web Api相关性较弱。而大多数Category中Mashup和Web Api无明显关联，或是由于数据量太少无法增强置信度。

查看关联程度较高的Category有Wi-Fi，Email等，这些Category专业性较强，其中的Mashup和Web Api均为了特定目的服务，故关联性较强符合逻辑。

第三题

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

```
# 第三题处理数据得到Web Api之间的兼容情况
link = {}
def getEdge(path):
    with open(path,"r") as f:
```

```
        data = json.load(f)
        for dic in data:
            if dic==None:
                continue
            relation = dic['related_apis']
            l = len(relation)
            for i in range(l):
                if relation[i]==None:
                    continue
                url1 = relation[i]['url']
                if url1 not in link.keys():
                    link[url1] = {}
                for j in range(i+1,l):
                    if relation[j]==None:
                        continue
                    url2 = relation[j]['url']
                    if url2 not in link[url1].keys():
                        link[url1][url2] = 0
                    link[url1][url2] += 1
                    if url2 not in link.keys():
                        link[url2] = {}
                    if url1 not in link[url2].keys():
                        link[url2][url1] = 0
                    link[url2][url1] += 1
getEdge("./data/raw/api_mashup/active_mashups_data.txt")
getEdge("./data/raw/api_mashup/deadpool_mashups_data.txt")
print(link)
```

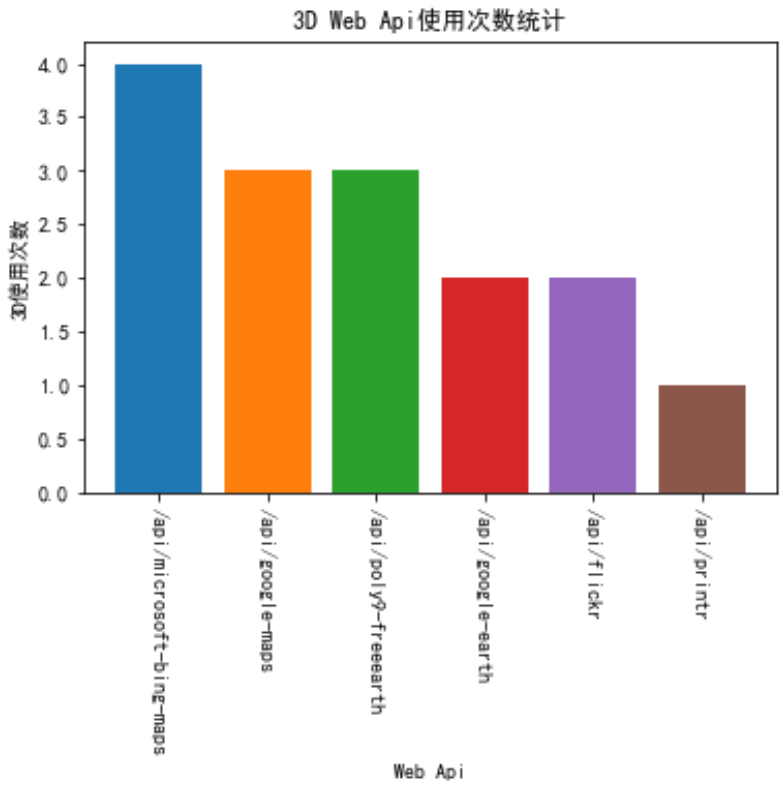
输出字典形态的Web Api之间联系


```
{"/api/yelp-fusion": {"/api/rotten-tomatoes": 1, "/api/unofficial-imdb": 1, "/api/cnet": 2, "/api/google-maps": 39,
"/api/google-custom-search": 2, "/api/foursquare": 13, "/api/easytobook": 1, "/api/google-analytics-managment": 1,
"/api/google-adsense": 3, "/api/mapbox": 1, "/api/amazon-s3": 2, "/api/elasticsearch": 2, "/api/world-weather-online": 1,
"/api/flickr": 15, "/api/twitter": 16, "/api/facebook": 15, "/api/alchemyapi": 2, "/api/google-maps-places": 3,
"/api/facebook-ads": 1, "/api/zillow": 3, "/api/walk-score": 5, "/api/geocoder": 1, "/api/trulia": 1, "/api/factual-v3": 1,
"/api/socrata-open-data": 1, "/api/ordrx": 1, "/api/expedia": 1, "/api/sensis-business-search": 1, "/api/scribd-platform": 1,
 "/api/yahoo-maps": 1, "/api/yahoo-geocoding": 4, "/api/google-cloud-translation": 3, "/api/google-translator-toolkit": 2,
"/api/yahoo-local-search": 8, "/api/citygrid": 5, "/api/google-geocoding": 1, "/api/yipit": 1, "/api/twilio": 3,
"/api/quova": 1, "/api/groupon": 2, "/api/8coupons": 1, "/api/blogger": 2, "/api/delicious-0": 3, "/api/ebay": 3,
"/api/youtube": 13, "/api/lastfm": 5, "/api/google-talk": 1, "/api/google-calendar": 1, "/api/vimeo": 3, "/api/linkedin": 3,
"/api/tumblr": 2, "/api/plurk": 1, "/api/soundcloud": 2, "/api/bitly": 1, "/api/yammer": 1, "/api/diigo": 1,
```

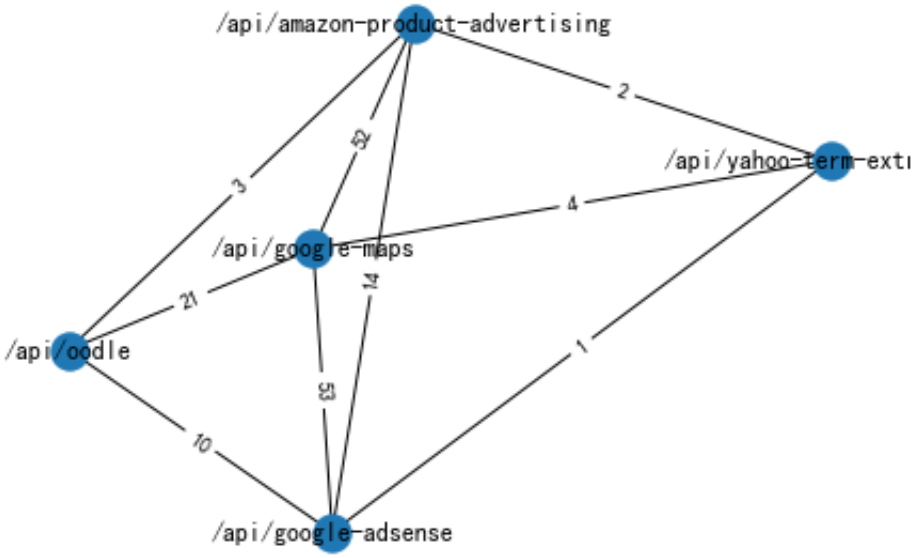

```
# 展示柱状图和关联图各一个到输出台上
show_bar = False
show_graph = False

for key,value in group_by_category:
    num = {}
    for item in value.values:
        if item[2] not in num.keys():
            num[item[2]] = 0
        num[item[2]] += 1
    li = list(num.items())
    li.sort(key=lambda x:x[1],reverse=True)
    li = li[:min(len(li),6)]

    # 对各个Category中Web Api的使用次数进行统计并做统计图
    plt.rcParams["font.sans-serif"] = ["SimHei"]
    plt.rcParams["axes.unicode_minus"] = False
    for item in li:
```

```
        plt.bar(item[0],item[1])
    plt.title(key+" Web Api使用次数统计")
    plt.xlabel("Web Api")
    plt.ylabel(key+"使用次数")
    plt.xticks(rotation=-90)
    plt.savefig("./problem3/Web Api使用次数统计/"+key+" Web Api使用次数统计.png",bbox_inches='tight')
    if show_bar == False:
        plt.show()
        show_bar = True
    plt.close()

    # 使用networkx包对各个Category兼容情况进行构图,以Web Api为结点,协作次数为边
    G = nx.Graph()
    n = len(li)
    for i in range(n):
        for j in range(i+1,n):
            u = li[i][0]
            v = li[j][0]
            if u in link.keys() and v in link[u].keys():
                G.add_edge(u, v, weight=link[u][v])
    pos = nx.spring_layout(G)
    labels = nx.get_edge_attributes(G, 'weight')
    nx.draw(G, pos, with_labels=True)
    nx.draw_networkx_edge_labels(G, pos, edge_labels=labels, font_weight='bold')
    plt.savefig("./problem3/Web Api兼容情况/"+key+" Web Api兼容情况图",bbox_inches='tight')
    if show_graph == False:
        plt.show()
        show_graph = True
    plt.close()
```

各Category中Web Api使用情况统计图（以3D为例）



各Category中Web Api兼容情况图（以Advertising为例）



以Web Api为结点，协作次数为边，得到的Category中Api兼容情况较为直观，可看到在Advertising这个Category中Web Api的兼容情况较好，Web Api之间的联系比较密切，可能是由于在广告中大多需要多方面的投放。有些Category中Web Api的兼容性较差，体现在图上就是结点之间连接关系较为稀疏。