Programming Language HW2 Report
Regular Expression with Python

資訊109 F74054067 柳孟芸

▷執行環境：

VS Code 作為 Python 3.7.1 64-bit 的開發環境

▷執行方法：

◎Question 1

$ python3 question_1.py

◎Question 2

$ python3 question_2.py

▷程式碼解說：

◎Question 1

question_1.py

```python
import urllib.request
import re
import matplotlib.pyplot as plt
from matplotlib.ticker import FormatStrFormatter

author = "Ian+Goodfellow"
url = "https://arxiv.org/search/?query=" + author + "&searchtype=author"
content = urllib.request.urlopen(url)
html_str = content.read().decode('utf-8')
# number of the paper
num_pattern = 'title is-clearfix[\s\S]*?:'
num_result = re.findall(num_pattern,html_str)
for i in num_result:
    number = i.split("title is-clearfix\">\n    \n        Showing 1&ndash;50 of ")[1].split(" results for author:")[0].strip()
start = int(int(number)/50)
x = []
y = []
print("Input Author: [" + author + "]")
for page in range(0,start+1) :
    url2="https://arxiv.org/search/?query="+author+"&searchtype=author&order=-announced_date_first&size=50&abstracts=show&start="+str(page*50)
    content2 = urllib.request.urlopen(url2)
    html_str2 = content2.read().decode('utf-8')
    # check if it is equal to the input
    check_flag = False
    check_pattern = 'Authors:[\s\S]*?originally announced[\s\S]*?</p>'
    check_result = re.findall(check_pattern,html_str2)
    for check in check_result:
        name_pattern = 'a href=\"[\s\S]*?\">[\s\S]*?</a>'
        name_result = re.findall(name_pattern,check)
        for name in name_result:
            Name = name.split("\">")[1].split("</a>")[0].strip()
            if(Name.lower() == author.replace('+',' ').lower()):
                check_flag = True
    if(check_flag==True):
        # announced year
        year_pattern = 'originally announced[\s\S]*?</p>'
        year_result = re.findall(year_pattern,check)
        for year in year_result:
            Year = year.split("originally announced</span>")[1].split("\n      \n    </p>")[0].strip()
            x.append(Year[-5:-1])
            X = list(set(x))
            X.sort()
        check_flag = False
for x_axis in X:
    counter = x.count(x_axis)
    y.append(counter)

# plot the bar-plot
ax = plt.figure().gca()
ymajorFormatter = FormatStrFormatter('%d') #設置y軸標籤文本的格式
ax.yaxis.set_major_formatter(ymajorFormatter)
plt.ylabel('The number of papers been published')
plt.title("Input Author: [" + author + "]")
plt.bar(X,y)
plt.show()
```

先抓取input author的paper數，
之後用於更改url2規則，
以確保將所有搜尋到的result都整合

檢查所有paper的作者有無完全符合input author，
若有，則將check_flag設為True，
即代表會計算此paper

if check_flag==True，
則繼續parse此paper的
announced year，
並存入x list中(X list則為之後
bar plot的x-label)，
計算過後再重新reset
check_flag，
y list 則為在某年中有多少篇
paper的次數(y-label)

畫出bar-plot

◎Question 2

question_2.py

```python
import urllib.request
import re
import matplotlib.pyplot as plt
from matplotlib.ticker import FormatStrFormatter

author = "Ian+Goodfellow"
url = "https://arxiv.org/search/?query=" + author + "&searchtype=author"
content = urllib.request.urlopen(url)
html_str = content.read().decode('utf-8')
# number of the paper
num_pattern = 'title is-clearfix[\s\S]*?:'
num_result = re.findall(num_pattern,html_str)
for i in num_result:
    number = i.split("title is-clearfix\">\n    \n        Showing 1&ndash;50 of ")[1].split(" results for author:")[0].strip()
    start = int(int(number)/50)
    co_author = []
    print("Input Author: [" + author + "]")
    for page in range(0,start+1) :
        url2="https://arxiv.org/search/?query="+author+"&searchtype=author&order=-announced_date_first&size=50&abstracts=show&start="+str(page*50)
        content2 = urllib.request.urlopen(url2)
        html_str2 = content2.read().decode('utf-8')
        # check if it is equal to the input
        check_flag = False
        check_pattern = 'Authors:[\s\S]*?originally announced[\s\S]*?</p>'
        check_result = re.findall(check_pattern,html_str2)
        for check in check_result:
            name_pattern = 'a href=\"[\s\S]*?\">[\s\S]*?</a>'
            name_result = re.findall(name_pattern,check)
            for name in name_result:
                Name = name.split("\">")[1].split("</a>")[0].strip()
                if(Name.lower() == author.replace('+',' ').lower()):
                    check_flag = True
        if(check_flag==True):
            # co-author
            author_pattern = 'Authors:</span>[\s\S]*?</p>'
            author_result = re.findall(author_pattern,check)
            for l in author_result:
                href = l.split("Authors:</span>\n      \n      ")[1].split("\n      \n    </p>")[0].strip()
                #獲取超連結<a>和</a>之間名字
                coauthor_pattern = '<a .*?>(.*?)</a>'
                coauthor_texts = re.findall(coauthor_pattern, href, re.S|re.M)
                for t in coauthor_texts:
                    co_author.append(t.strip())
                    Co_author = list(set(co_author))
                    Co_author.sort()
            check_flag = False
    for name in Co_author:
        times = co_author.count(name)
        if name == author.replace('+',' ') :  #去掉自己
            continue
        else:   # print the answer
            print('[' + name + ']: ' + str(times) + ' times' )
```

先抓取input author的paper數，
之後用於更改url2規則，
以確保將所有搜尋到的result都整合

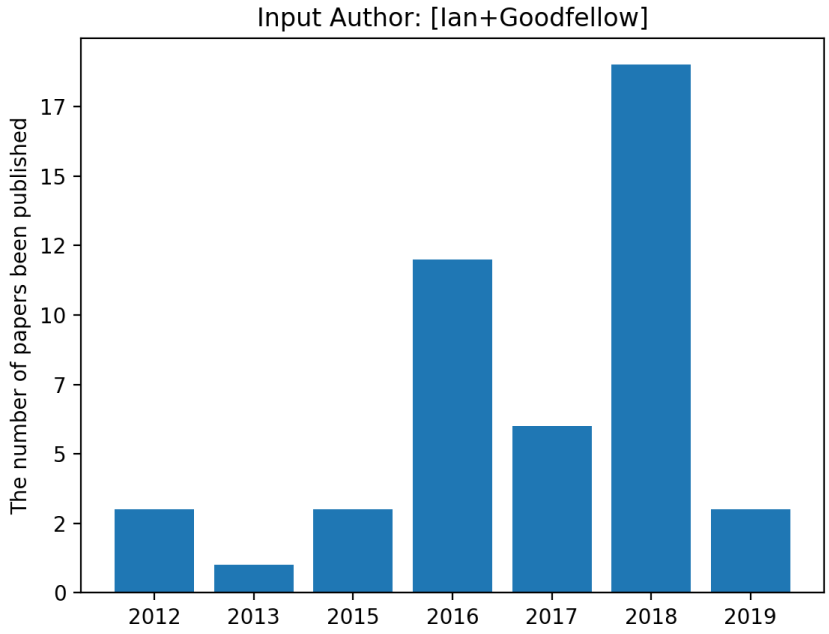
檢查所有paper的作者有無完全符合input author，
若有，則將check_flag設為True，
即代表會計算此paper

if check_flag==True，
則繼續parse此paper的co-author，
並存入co_author list中，
Co_author為sort過後的list，
計算過後再重新reset check_flag

計算每個co-author出現過幾次，
並去去掉input author自己，
之後印出來

▷執行結果：

◎Question 1



◎Question 2

```
liumengyunde-MacBook-Pro:HW2_F74054067 newmileou$ python3 question_2.py
Input Author: [Ian+Goodfellow]
[Aaron Courville]: 2 times
[Abhibhav Garg]: 1 times
[Alan Yuille]: 1 times
[Alec Radford]: 1 times
[Aleksander Madry]: 1 times
[Alex Kurakin]: 1 times
[Alexander Matyasko]: 1 times
[Alexey Kurakin]: 7 times
[Alireza Makhzani]: 1 times
[Ananthram Swami]: 1 times
[Andrew Harp]: 1 times
[Andrew M. Dai]: 3 times
[Andy Chu]: 1 times
[Andy Davis]: 1 times
[Anish Athalye]: 1 times
[Arnaud Bergeron]: 1 times
[Ashish Agarwal]: 1 times
[Augustus Odena]: 5 times
[Aurko Roy]: 2 times
[Balaji Lakshminarayanan]: 1 times
[Been Kim]: 2 times
[Brendan Frey]: 1 times
[Brian Cheung]: 1 times
[Catherine Olsson]: 4 times
[Chelsea Finn]: 1 times
[Chiyuan Zhang]: 1 times
[Chris Olah]: 1 times
[Christian Szegedy]: 1 times
[Christopher Olah]: 1 times
[Cihang Xie]: 2 times
[Colin Raffel]: 3 times
[Craig Citro]: 1 times
[Dan Boneh]: 2 times
[Dan Mane]: 1 times
[David Andersen]: 1 times
[David Berthelot]: 2 times
[David Warde-Farley]: 1 times
[Derek Murray]: 1 times
[Dimitris Metaxas]: 1 times
[Dimitris Tsipras]: 1 times
[Dumitru Erhan]: 1 times
[Eugene Brevdo]: 1 times
[Fangzhou Liao]: 1 times
[Fartash Faghri]: 2 times
[Florian Tramèr]: 2 times
[Frédéric Bastien]: 1 times
[Gamaleldin F. Elsayed]: 2 times
[Garrison Cottrell]: 1 times
[Geoffrey Irving]: 1 times
[George E. Dahl]: 1 times
[Greg S. Corrado]: 1 times
[H. Brendan McMahan]: 2 times
[Han Zhang]: 1 times
[Harini Kannan]: 1 times
[Ilya Mironov]: 2 times
[Ilya Sutskever]: 1 times
[Jacob Buckman]: 1 times
[James Bergstra]: 1 times
[Jascha Sohl-Dickstein]: 2 times
[Jeffrey Dean]: 1 times
[Jianyu Wang]: 1 times
[Joan Bruna]: 1 times
[Jonas Rauber]: 2 times
[Jonathan Uesato]: 1 times
[Jonathon Shlens]: 2 times
[Josh Levenberg]: 1 times
[Julius Adebayo]: 2 times
[Jun Zhu]: 1 times
[Junjiajia Long]: 1 times
[Justin Gilmer]: 4 times
[Karen Hambardzumyan]: 1 times
[Kunal Talwar]: 3 times
[Li Zhang]: 2 times
[Lukasz Kaiser]: 1 times
[Luke Metz]: 1 times
[Maithra Raghu]: 1 times
[Manjunath Kudlur]: 1 times
[Martin Wattenberg]: 1 times
[Martín Abadi]: 4 times
[Matthieu Devin]: 1 times
[Michael Isard]: 1 times
[Michael Muelly]: 1 times
[Mihaela Rosca]: 1 times
[Ming Liang]: 1 times
[Moritz Hardt]: 1 times
[Motoki Abe]: 1 times
[Navdeep Jaitly]: 1 times
[Nicholas Carlini]: 4 times
[Nicolas Bouchard]: 1 times
[Nicolas Papernot]: 10 times
[Pascal Lamblin]: 1 times
[Patrick McDaniel]: 4 times
[Paul Barham]: 1 times
[Paul Christiano]: 1 times
[Paul Hendricks]: 1 times
[Pieter Abbeel]: 1 times
[Rafal Jozefowicz]: 1 times
[Rajat Monga]: 1 times
[Razvan Pascanu]: 1 times
[Reuben Feinman]: 1 times
[Rob Fergus]: 1 times
[Rujun Long]: 1 times
[Ryan P. Adams]: 1 times
[Ryan Sheatsley]: 1 times
[Samaneh Azadi]: 1 times
[Samuel S. Schoenholz]: 1 times
[Samy Bengio]: 3 times
[Sandy Huang]: 1 times
[Sangxia Huang]: 1 times
[Sanjay Ghemawat]: 1 times
[Seiya Tokui]: 1 times
[Sergey Levine]: 1 times
[Shakir Mohamed]: 1 times
[Sherry Moore]: 1 times
[Shreya Shankar]: 1 times
[Somesh Jha]: 1 times
[Stephan Zheng]: 1 times
[Surya Bhupatiraju]: 1 times
[Takeru Miyato]: 1 times
[Takuya Akiba]: 1 times
[Thomas Leung]: 1 times
[Tianqi Chen]: 1 times
[Tianyu Pang]: 1 times
[Tim Salimans]: 1 times
[Tom B. Brown]: 2 times
[Tom Brown]: 2 times
[Trevor Darrell]: 1 times
[Vahid Behzadan]: 1 times
[Vicki Cheung]: 1 times
[Wieland Brendel]: 1 times
[Willi Gierke]: 1 times
[William Fedus]: 2 times
[Wojciech Zaremba]: 2 times
[Xi Chen]: 1 times
[Xiaolin Hu]: 1 times
[Yan Duan]: 1 times
[Yang Song]: 1 times
[Yangqing Jia]: 1 times
[Yao Qin]: 1 times
[Yao Zhao]: 1 times
[Yash Sharma]: 1 times
[Yerkebulan Berdibekov]: 1 times
[Yi-Lin Juang]: 1 times
[Yinpeng Dong]: 2 times
[Yoshua Bengio]: 3 times
[Yuzhe Zhao]: 1 times
[Z. Berkay Celik]: 1 times
[Zhi Li]: 1 times
[Zhifeng Chen]: 1 times
[Zhishuai Zhang]: 2 times
[Zhonglin Han]: 1 times
[Zhou Ren]: 1 times
[Úlfar Erlingsson]: 2 times
```