รายงาน

เรื่อง Google Scholar

จัดทำโดย

นายภีมภัช พจน์สุนทร 6209650081

เสนอ

อ.ดร.วสิศ ลิ้มประเสริฐ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา DSI200

การเขียนโปรแกรมเพื่อวิเคราะห์ข้อมูล

(DSI200 Data Analytics Programming)

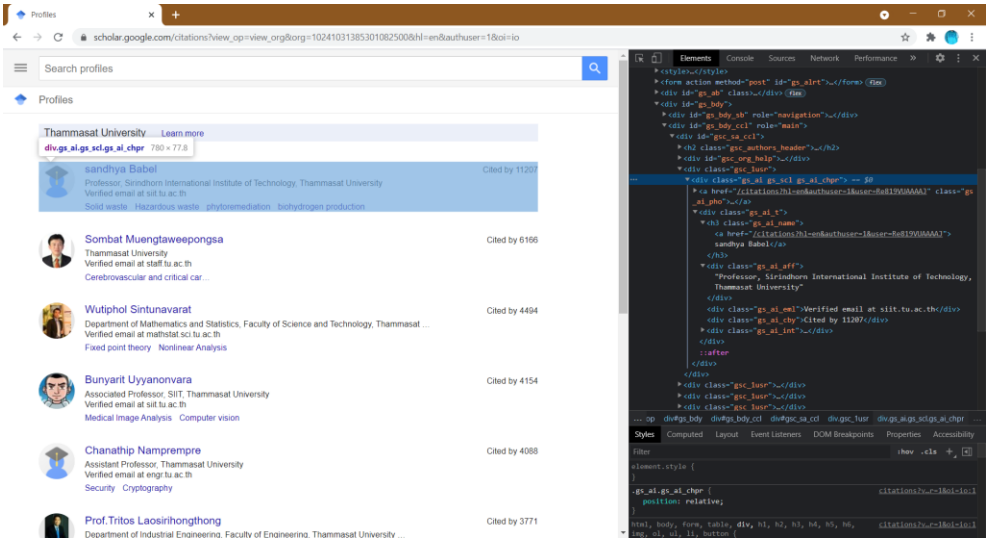
ภาคเรียนที่ 2 ปีการศึกษา 2563

มหาวิทยาลัยธรรมศาสตร์ ศูนย์รังสิต

สารบัญ

โครงสร้างของหน้าเว็บ Google Scholar

1. หน้าแสดง Author ที่เกี่ยวข้องกับมหาวิทยาลัยธรรมศาสตร์

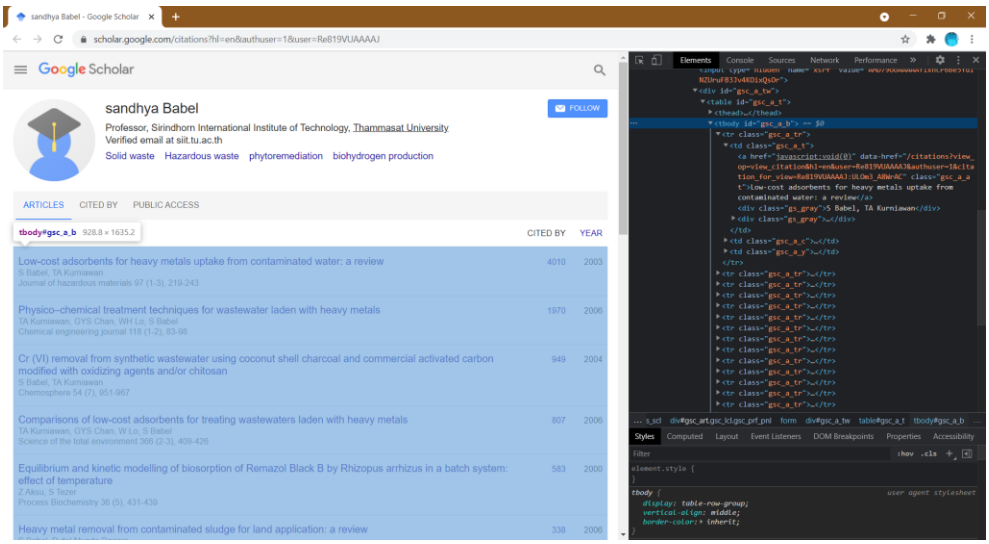


ภาพที่ 1

ที่มา: https://scholar.google.com/citations?view_op=view_org&hl=en&org=10241031385301082500

โดยจะเห็นได้ว่า ข้อมูล user_id, name, affiliation จะอยู่ใน tag ที่ได้ทำการมาร์คไว้ ซึ่งจะเห็นได้ว่าข้อมูลใน tag นั้น จะมี tag a href ที่ทำเก็บข้อมูล user_id อยู่ จึงใช้ส่วนนี้ในการดึงข้อมูล และข้อมูลที่อยู่ใน tag ข้างในอีกทีนั้น ยังมีข้อมูล name และ affiliation ด้วย จึงทำการดึงข้อมูลตรงส่วนนี้มาทำเป็นตาราง Author Table และจะทำการเก็บ a href แยกไว้อีก เพื่อนำไปใช้หา Paper ของ Author แต่ละท่านต่อ

2. หน้าข้อมูลภายในของ Author

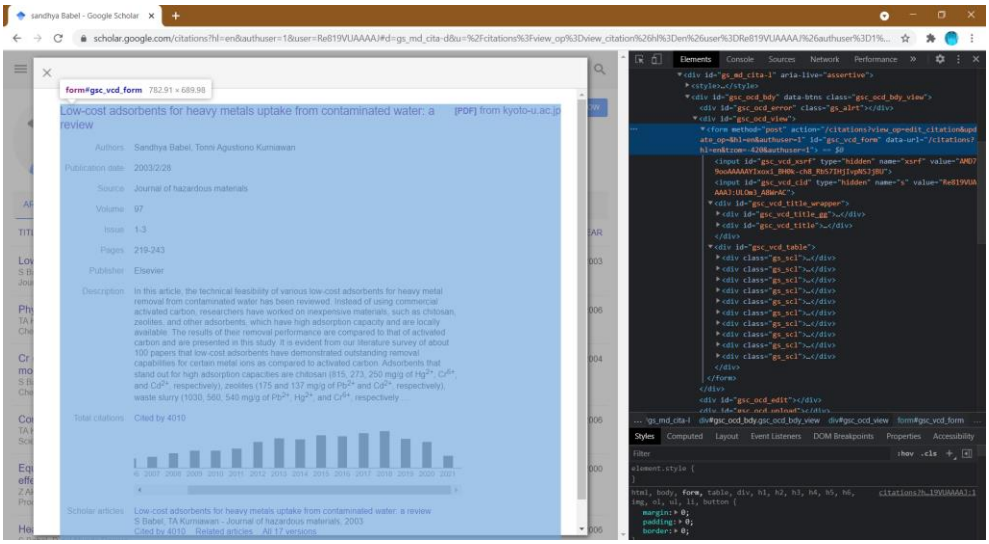


ภาพที่ 2

ที่มา: https://scholar.google.com/citations?hl=en&user=Re819VUAAAAJ

จากภาพจะเห็นได้ว่า Author ที่ชื่อ sandhya Babel มีข้อมูล Paper อยู่ใน tag ที่ได้ทำการมาร์คไว้ ซึ่งข้อมูลภายในแต่ละ Paper นั้นจะไม่สามารถดึงได้จากหน้านี้ จำเป็นต้องทำการเข้าไปในหน้าของตัว Paper ซึ่งวิธีการเข้าไปในหน้าของตัว Paper นั้นจะทำการดึง tag a href มาใช้งาน และใช้ selenium ในการเข้าและใช้ BeautifulSoup ในการเก็บข้อมูล

3. หน้าแสดงข้อมูล Paper



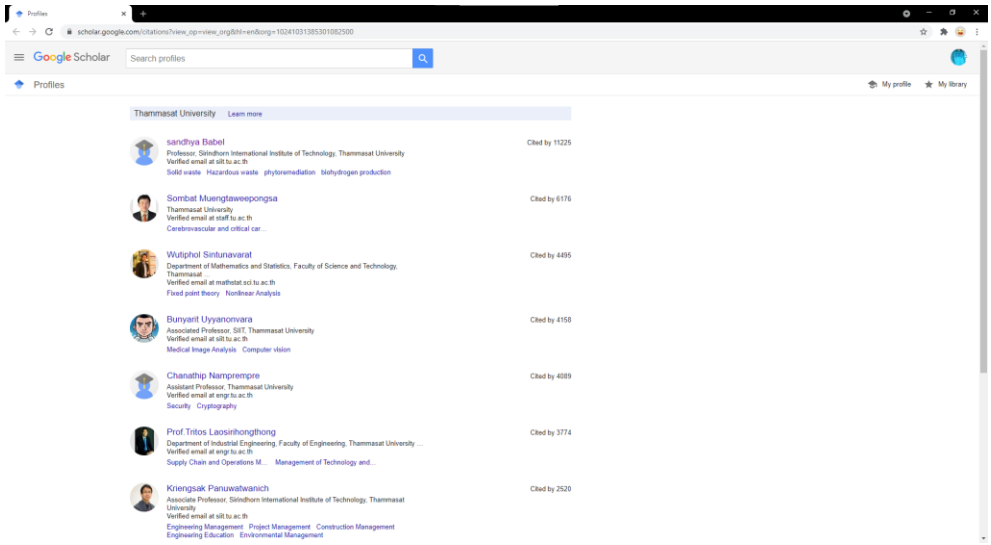
ภาพที่ 3

ที่มา: https://scholar.google.com/citations?hl=en&user=Re819VUAAAAJ

จากภาพ จะเห็นได้ว่าข้อมูลของ Paper ทั้งหมด จะอยู่ใน tag form ซึ่งข้อมูลที่จะนำไปทำ Paper Table มีดังนี้ title, author, publication date, description และ total citations ซึ่งข้อมูล title จะอยู่ใน tag ที่มี id = gsc_vcd_title_wrapper และข้อมูลที่เหลือจะอยู่ใน tag ที่มี id = gsc_vcd_table ก็จะทำการดึงข้อมูลเหล่านี้มาทำ Peper Table

ขั้นตอนการรวบรวมข้อมูล

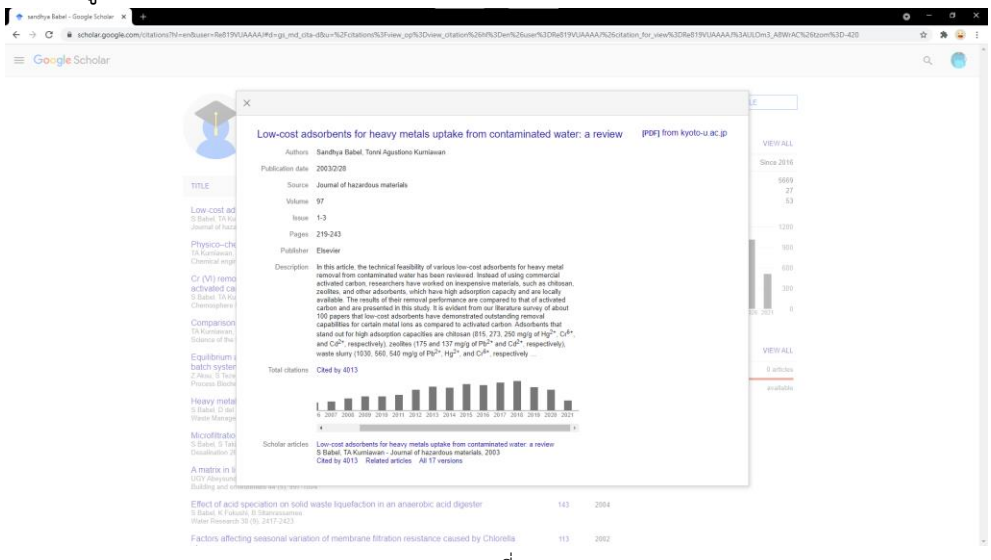
ขั้นตอนที่ 1 เข้าสู่เว็บไซต์ที่เป็นนักวิชาการที่สังกัดมหาวิทยาลัยธรรมศาสตร์ ทำการรวบรวมข้อมูลต่าง ๆ ที่ใช้ ได้แก่ user_ID (ID ของผู้ใช้), name (ชื่อของนักวิชาการ), affiliation (สังกัดที่อยู่)



ภาพที่ 4

ที่มา: https://scholar.google.com/citations?view_op=view_org&hl=en&org=10241031385301082500

ขั้นตอนที่ 2 ทำการเปิดเข้าไปในหน้าของนักวิชาการที่สังกัดมหาวิทยาลัยธรรมศาสตร์ และเปิดเข้าไปในเอกสารวิจัยต่าง ๆ และทำการรวบรวมข้อมูลต่าง ๆ ได้แก่ title (ชื่อของเอกสารวิจัย), authors (ชื่อของผู้ที่เขียนเอกสารวิจัย), publication_date (วันที่ปล่อยเอกสารวิจัย), description (คำอธิบายของเอกสารวิจัย), cite_by (จำนวนคนที่เข้ามาดูเอกสารวิจัย)



ภาพที่ 5

ที่มา: https://scholar.google.com/citations?hl=en&user=Re819VUAAAAJ

ขั้นตอนการทำงานของโค้ด

ขั้นตอนในการเก็บข้อมูลใน Google Scholar จะแบ่งออกเป็น 3 ส่วนใหญ่ ๆ คือ 1. ขั้นตอนการรวบรวมข้อมูล Author Table และรวบรวมเว็บไซต์ของ Author ทุกคน 2. ขั้นตอนการรวบรวมเว็บไซต์ของ Paper ทุกงานของ Author ทุกคน 3. ขั้นตอนการรวบรวมข้อมูล Paper Table ทุกงาน โดยในแต่ละขั้นตอนใหญ่ ๆ จะมีขั้นตอนย่อย ๆ ดังต่อไปนี้

1. ขั้นตอนการรวบรวมข้อมูล Author Table และรวบรวมเว็บไซต์ของ Author ทุกคน

1.1. ขั้นตอนการ import library ที่ใช้ในการรวบรวมข้อมูล

```
import gspread
import numpy as np
import pandas as pd
import requests

from bs4 import BeautifulSoup
from oauth2client.service_account import ServiceAccountCredentials
```

1.2. ทำการกำหนด url ของเว็บไซต์เริ่มต้น

```
 url =
"https://scholar.google.com/citations?view_op=view_org&hl=en&org=102410313853
01082500"
```

1.3. ทำการสร้าง list แต่ละตัวแปร มาเก็บข้อมูลต่าง ๆ ใน Author Table และเว็บไซต์ของ Author ทุกคน

```
user_ID = []
name = []
affiliation = []

links = []
newPage = url
```

1.4. ทำการเข้าลูป 30 ครั้ง โดย 30 ครั้งมาจากจำนวนหน้าทั้งหมดที่เกี่ยวข้องกับมหาวิทยาลัยธรรมศาสตร์ จากนั้นทำการ requests เข้าหน้าเว็บ และทำการดึงข้อมูลด้วย BeautifulSoup และทำการเข้าลูปอีกรอบ โดยลูปนี้ไว้ใช้เก็บข้อมูล เว็บไซต์ user_ID และ name ของ Author จากนั้นเมื่อออกลูปจะเข้าอีกลูปหนึ่ง โดยลูปนี้ใช้สำหรับเก็บข้อมูล affiliation จากนั้นจะออกจากลูปอีกรอบและทำการหาปุ่ม button ที่อยู่ใน html เพื่อที่จะดึงเว็บไซต์อีกหน้าหนึ่งมา จากนั้นจะทำการเข้าลูป ซึ่งเว็บไซต์หน้าต่อไปที่ดึงมานั้นจะต้องทำการแปลงข้อมูล เช่น ตัวอักษร \\ จะต้องลบทิ้ง ตัว x26 จะแทนที่ด้วย & และ x3d จะแทนที่ด้วย = เป็นต้น จากนั้นจะเก็บเว็บไซต์อีกหน้าไว้ในตัวแปร newPage และวนลูปไปเรื่อย ๆ เพื่อดึงข้อมูล Author ทุกคน

```
for i in range(30):
    page = requests.get(newPage)
    soup = BeautifulSoup(page.content, 'lxml')
    for find in soup.find_all('a', href=True):
        if find.text:
            link = "https://scholar.google.com/" + find['href']
            if "/citations?hl=en&user=" in link:
                links.append(link)
                user_ID.append(find['href'][22:])
                name.append(find.text)

    for word in soup.find_all("div", {"class": "gs_ai_aff"}):
        affiliation.append(word.text)

    btn_onclick_list = [a.get('onclick') for a in soup.find_all('button')]
    for click in btn_onclick_list:
        if click is not None:
            click = click.replace("window.location=", "")
            click = click.replace("'", "")
            click = click.replace("\\", "")
            click = click.replace("x26", "&")
            click = click.replace("x3d", "=")
            click = click.replace('&oe=ASCII;', "")
            newPage = "https://scholar.google.com"+click
```

1.5. จากนั้นจะทำการแปลง list ของ links ที่เก็บมา แปลงเป็น numpy array และจากนั้นแปลง numpy array ให้เป็น pandas DataFrame ซึ่งเมื่อแปลงทุกอย่างแล้ว จะทำการแปลง DataFrame ให้เป็นไฟล์ csv ที่ชื่อ All Link.csv โดยจะไม่ใช้ index ในการเก็บข้อมูล โดย All link ไว้ใช้ในการหาเว็บไซต์ของ Paper ทุกงาน

```
links = np.array(links)
linkFile = pd.DataFrame(data=links)
linkFile.to_csv("All Link.csv", index=False)
```

1.6. จากนั้นทำการแปลงข้อมูลต่าง ๆ ที่ต้องอยู่ใน Author Table เป็น numpy array และทำการสร้างเป็น pandas DataFrame ซึ่งเนื่องจากให้ data=[user_ID, name, affiliation] ทำให้เก็บข้อมูลแบบเป็น row ไม่ได้เรียงลงมาเป็น column จึงใช้ .T เพื่อกลับตาราง หลังจากนั้นตั้งชื่อแต่ละ column และทำเป็นไฟล์ csv ที่ชื่อ Author Table.csv ซึ่งไม่ใช้ index ในการทำ csv

```
user_ID = np.array(user_ID)
name = np.array(name)
affiliation = np.array(affiliation)
authorTable = pd.DataFrame(data=[user_ID, name, affiliation])
authorTable = authorTable.T
authorTable.columns = ['user_ID', 'name', 'affiliation']
authorTable.to_csv("Author Table.csv", index=False)
```

1.7. ทำการเชื่อมต่อกับ Google Spreadsheets โดยเข้าถึง sheet ที่ชื่อ Author Table และทำการอ่านไฟล์ Author Table.csv ที่ได้สร้างไว้ก่อนหน้านี้ และทำการ import ไฟล์เข้าไปใน Google Spreadsheets

```
scope = ["https://spreadsheets.google.com/feeds",
'https://www.googleapis.com/auth/spreadsheets',
         "https://www.googleapis.com/auth/drive.file",
"https://www.googleapis.com/auth/drive"]

credentials =
ServiceAccountCredentials.from_json_keyfile_name('../client_secret.json',
scope)
client = gspread.authorize(credentials)

spreadsheet = client.open('Author Table')

with open('Author Table.csv', 'r', encoding='iso-8859-1') as file_obj:
    content = file_obj.read()
    client.import_csv(spreadsheet.id, data=content)
```

2. ขั้นตอนการรวบรวมเว็บไซต์ของ Paper ทุกงานของ Author ทุกคน

2.1. ขั้นตอนการ import library ที่ใช้ในการรวบรวมข้อมูล

```
import pandas as pd
import numpy as np
import time

from selenium import webdriver
from bs4 import BeautifulSoup
```

2.2. ทำการอ่านข้อมูลจากไฟล์ All Link.csv จากนั้นทำการแปลงเป็น numpy array และทำการแปลงเป็น list ในลำดับต่อไป จากนั้นทำการแปลง list ให้เป็น มิติเดียว

```
links = pd.read_csv("All Link.csv")
links = links.to_numpy()
links = links.tolist()
links = [j for sub in links for j in sub]
```

2.3. ทำการกำหนด PATH ของไฟล์ chromedriver.exe และกำหนด driver เป็น Chrome โดยให้ดูจาก PATH ที่กำหนด

```
PATH = "C:\Program Files (x86)\chromedriver.exe"
driver = webdriver.Chrome(PATH)
```

2.4. ทำการสร้าง list ตัวแปร paperLinks ไว้สำหรับเก็บเว็บไซต์ของ Paper ทุกงาน

```
paperLinks = []
```

2.5. ทำการเข้าลูปเพื่อเก็บเว็บไซต์ของ Paper ทุกงาน โดยจะเข้าทุกเว็บไซต์ของ Author ที่ได้เก็บรวบรวมมาก่อนหน้า โดยใช้ selenium ในการเข้าถึงแต่ละหน้าเว็บ ซึ่งแต่ละจุด จะมีการใช้คำสั่ง time.sleep เพื่อที่จะให้การเข้าถึงแต่ละเว็บไม่รวดเร็วจนเกินไป เพราะถ้าเร็วเกินไป ตัว Google Scholar จะ detect ว่าเป็น bot ทำให้ไม่สามารถเก็บข้อมูลต่อได้ หลังจากเข้าถึงหน้าเว็บของ Author แต่ละคนแล้ว จะทำการเข้าลูป 10 ครั้ง เพื่อกดปุ่ม Show more เพราะการจะเก็บรวบรวมข้อมูลนั้น ถ้าข้อมูลไม่แสดงออกมา ตัว selenium จะไม่สามารถดึงข้อมูลที่ซ่อนไว้ได้ จึงต้องมีการวนลูปกดปุ่ม Show more ก่อน หลังจากนั้น จะให้ page เก็บหน้า html ของหน้า paper นั้น ๆ และใช้ BeautifulSoup ในการอ่านหน้านั้นต่อ หลังจากนั้นทำการเก็บ url ของแต่ละ Paper ทำซ้ำวนไปจนหมด

```
for link in links:
    driver.get(link)
    time.sleep(1)

    for j in range(10):
        try:

driver.find_element_by_xpath("//button[@id='gsc_bpf_more']").click()
            time.sleep(0.5)
        except:
            continue
    time.sleep(1)

    page = driver.page_source
    soup = BeautifulSoup(page, 'lxml')
    for find in soup.find_all('a'):
        find = str(find)
        if "data-href" in find:
            split = find.split("href=\"")
            paper = split[1]
            paper = paper.replace('oe=ASCII&amp;', '')
            paper = paper.replace('?', '%3F')
            paper = paper.replace('=', '%3D')
            paper = paper.replace('&amp;', '%26')
            paper = paper.replace(':', '%3A')
            paper = paper.replace("/", "#d=gs_md_cita-d&u=%2F")
            paper = paper.replace('"', "")
            paperLinks.append(link+paper)
```

2.6. หลังจากวนลูปเสร็จ จะทำการออกจาก selenium

```
driver.quit()
```

2.7. ทำการแปลง paperLinks ให้เป็น numpy array และทำการแปลงเป็น pandas DataFrame จากนั้นทำการทำเป็นไฟล์ csv ที่ชื่อ All Paper Link.csv โดยไม่เก็บ index ไปในไฟล์ csv ด้วย

```
paperLinks = np.array(paperLinks)
paperFile = pd.DataFrame(data=paperLinks)
paperFile.to_csv("All Paper Link.csv", index=False)
```

3. ขั้นตอนการรวบรวมข้อมูล Paper Table ทุกงาน

3.1. ขั้นตอนการ import library ที่ใช้ในการรวบรวมข้อมูล

```
import time
import gspread
import numpy as np
import pandas as pd

from bs4 import BeautifulSoup
from oauth2client.service_account import ServiceAccountCredentials
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as ec
from selenium.webdriver.support.wait import WebDriverWait
```

3.2. ทำการอ่านข้อมูลจากไฟล์ All Paper Link.csv จากนั้นทำการแปลงเป็น numpy array และทำการแปลงเป็น list ในลำดับต่อไป จากนั้นทำการแปลง list ให้เป็น มิติเดียว

```
paperLinks = pd.read_csv("All Paper Link.csv")
paperLinks = paperLinks.to_numpy()
paperLinks = paperLinks.tolist()
paperLinks = [j for sub in paperLinks for j in sub]
```

3.3. ทำการสร้างตัวแปรประเภท list มาเก็บข้อมูลในตาราง Paper Table

```
titles = []
authors = []
publication_date = []
description = []
cite_by = []
```

3.4. ทำการกำหนด PATH ของไฟล์ chromedriver.exe และกำหนด driver เป็น Chrome โดยให้ดูจาก PATH ที่กำหนด

```
PATH = "C:\Program Files (x86)\chromedriver.exe"
driver = webdriver.Chrome(PATH)
```

3.5. ก่อนเข้าลูปทำการกำหนด countLink ไว้เท่ากับ 0 ไว้ใช้นำรอบของ link ว่าทำไปกี่รอบแล้ว หลังจากนั้นทำการเข้าลูปทุก Paper ที่ได้หาไว้ก่อนหน้านี้ ต่อมาทำการเพิ่มค่า countLink 1 และแสดง link ที่กำลังดึงข้อมูลอยู่ จากนั้นใช้ selenium เข้าหน้าเว็บ และ set time.sleep ไว้ เพื่อที่จะไม่ให้ทำงานเร็วเกินไป จากนั้นทำการค้นหาตำแหน่งของข้อมูลที่ต้องการ และทำการเก็บหน้า html ใน page_source และทำการใช้ BeautifulSoup ในการอ่านต่อ จากนั้นทำการวนลูปเพื่อเก็บ title ของ Paper และแสดงชื่อ title หลังจากออกลูปนั้นมา ทำการสร้าง list field และ value เพื่อเก็บ field และ value ที่อยู่ในหน้า html และสร้างตัวแปร cite ไว้สำหรับ เก็บค่า Cited by ที่อยู่ในข้อมูล ต่อมาทำการเข้าลูปหาแต่ละ field และเข้าอีกลูปเพื่อหาแต่ละ value และเข้าอีกลูปเพื่อหาค่า Cited by หลังจากหาข้อมูลเสร็จแล้ว ทำการเข้าลูปอีกลูป เพื่อที่จะได้เก็บข้อมูลแต่ละประเภท เช่น authors publication_date description cite_by และในแต่ละตอนที่ทำการเก็บค่า จะทำการแสดงข้อมูลนั้น ๆ ด้วย และหลังจากออกลูปแล้ว จะทำการเช็คจำนวนของข้อมูลในแต่ละตัวกับค่า countLink โดยมีไว้สำหรับข้อมูลใน Paper นั้น ๆ ไม่มีข้อมูลบางอย่าง และจากนั้น วนทำไปเรื่อย ๆ จนจบ เมื่อจบแล้วจะทำการปิด selenium

```
countLink = 0
for link in paperLinks:
    countLink += 1
    print(link)
    driver.get(link)
    time.sleep(1)

    try:
        WebDriverWait(driver, 2).until(ec.presence_of_element_located((By.ID,
"gsc_vcd_form")))
        time.sleep(1)
        page_source = driver.page_source
        soup = BeautifulSoup(page_source, 'lxml')

        for find in soup.find_all("div", {"id": "gsc_vcd_title"}):
            titles.append(find.text)
            print(find.text)

        field = []
        value = []
        cite = '-'
        for find in soup.find_all("div", {"class": "gsc_vcd_field"}):
            field.append(find.text)
        for find in soup.find_all("div", {"class": "gsc_vcd_value", "style":
None}):
            value.append(find.text)
        for find in soup.find_all('a', href=True):
            if find.text:
                if "Cited by" in find.text:
                    cite = find.text[9:]
                    break

        count = 0
        for count in range(len(field)):
            if field[count] == "Authors":
```

```
                print(value[count])
                authors.append(value[count])
            elif field[count] == "Publication date":
                print(value[count])
                publication_date.append(value[count])
            elif field[count] == "Description":
                print(value[count])
                description.append(value[count])
            elif field[count] == "Total citations":
                print(cite)
                cite_by.append(cite)
            count += 1

        if len(authors) < countLink:
            authors.append("-")
        if len(publication_date) < countLink:
            publication_date.append("-")
        if len(description) < countLink:
            description.append("-")
        if len(cite_by) < countLink:
            cite_by.append("-")

        print(len(authors))
        print(len(publication_date))
        print(len(description))
        print(len(cite_by))
    except:
        continue

    time.sleep(1)

driver.quit()
```

3.6. จากนั้นทำการแปลงข้อมูลต่าง ๆ ที่ต้องอยู่ใน Paper Table เป็น numpy array และทำการสร้างเป็น pandas DataFrame ซึ่งเนื่องจากให้ data=[titles, authors, publication_date] ทำให้เก็บข้อมูลแบบเป็น row ไม่ได้เรียงลงมาเป็น column จึงใช้ .T เพื่อกลับตาราง หลังจากนั้นตั้งชื่อแต่ละ column และทำเป็นไฟล์ csv ที่ชื่อ Paper Table.csv ซึ่งไม่ใช้ index ในการทำ csv

```
titles = np.array(titles)
authors = np.array(authors)
publication_date = np.array(publication_date)
description = np.array(description)
cite_by = np.array(cite_by)
paperTable = pd.DataFrame(data=[titles, authors, publication_date,
description, cite_by])
paperTable = paperTable.T
paperTable.columns = ['title', 'authors', 'publication_date', 'description',
'cite_by']
paperTable.to_csv('Paper Table.csv', index=False)
```

3.7. ทำการเชื่อมต่อกับ Google Spreadsheets โดยเข้าถึง sheet ที่ชื่อ Author Table และทำการอ่านไฟล์ Author Table.csv ที่ได้สร้างไว้ก่อนหน้านี้ และทำการ import ไฟล์เข้าไปใน Google Spreadsheets

```
scope = ["https://spreadsheets.google.com/feeds",
'https://www.googleapis.com/auth/spreadsheets',
         "https://www.googleapis.com/auth/drive.file",
"https://www.googleapis.com/auth/drive"]

credentials =
ServiceAccountCredentials.from_json_keyfile_name('../client_secret.json',
scope)
client = gspread.authorize(credentials)

spreadsheet = client.open('Paper Table')

with open('Paper Table.csv', 'r', encoding='iso-8859-1') as file_obj:
    content = file_obj.read()
    client.import_csv(spreadsheet.id, data=content)
```

ปรับปรุงและแก้ไขโค้ด

การทำ Google Scholar งานนี้นั้น คือ การทำ web scraping ซึ่งใน python จะมี library ที่ช่วยในการทำ web scraping อยู่ ได้แก่ 1. Scrapy 2.bs4 3.selenium โดยโค้ดที่เขียนไปข้างต้นเป็นการผสมระหว่าง bs4 และ selenium ซึ่งใช้งานได้ แต่หลักการในการทำ web scraping ยังคงไม่ดีพอ ยังคงมีหลายจุดที่สามารถพัฒนาได้ โดยมีดังต่อไปนี้

1. ตอนเปลี่ยนหน้า ในโค้ดที่เขียนคือทำการดึง url ของหน้าถัดไป แล้วใช้ selenium เปิด url นั้น แต่สามารถปรับปรุงได้ด้วยการใช้ selenium คลิกที่ปุ่มไปหน้าถัดไปได้เลย ไม่จำเป็นต้องดึง url

2. ตอนสร้างตัวแปรมาเก็บข้อมูล ในโค้ดที่เขียนคือสร้างเป็น list แล้วทำการเก็บข้อมูล และแปลงเป็น numpy และทำเป็น pandas DataFrame แล้วทำเป็นไฟล์ csv แต่สามารถปรับปรุงได้ด้วยการทำเป็น pandas DataFrame ตั้งแต่แรกเลย และทำการเก็บข้อมูล แล้วค่อยทำเป็นไฟล์ csv

3. สามารถใช้แค่ selenium เก็บข้อมูลได้เลย ไม่จำเป็นต้องใช้ทั้ง bs4 และ selenium

โดยหลังจากได้ทำการปรับปรุงโค้ดตามที่ได้เขียนไว้ข้างต้น สามารถเขียนโค้ดได้ดังนี้

```
import pandas as pd
import time
import gspread

from oauth2client.service_account import ServiceAccountCredentials
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as ec

PATH = './chromedriver'
```

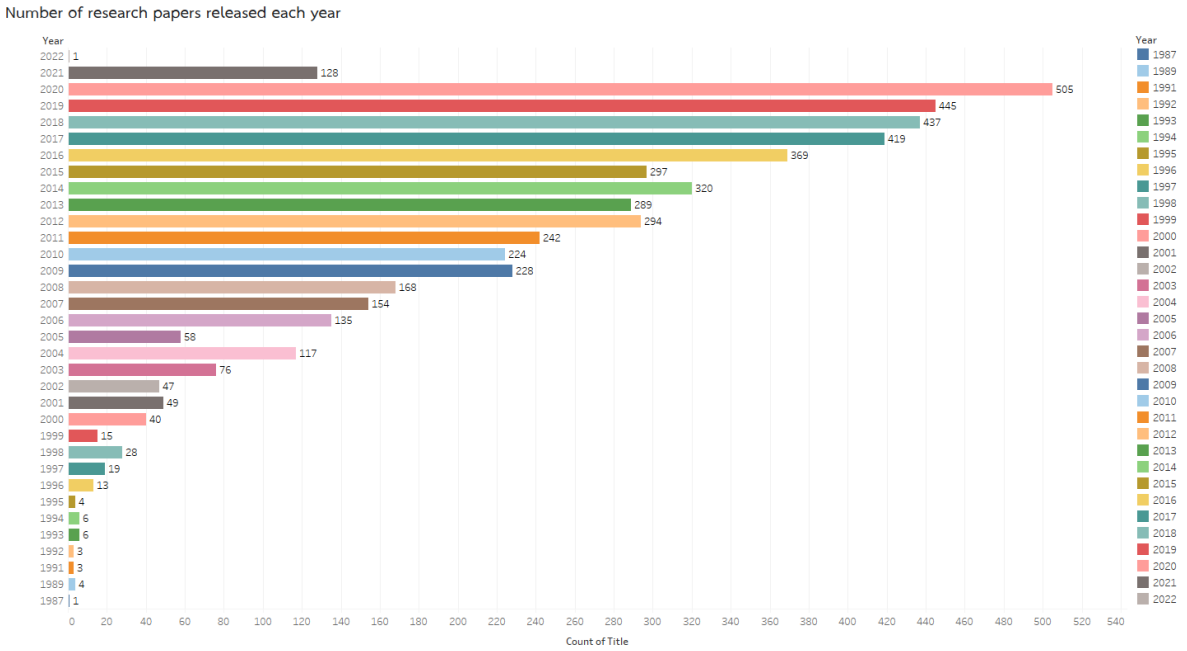
```
driver = webdriver.Chrome(PATH)

driver.get("https://scholar.google.com/citations?view_op=view_org&hl=en&org=1
02410313853010825000")
df = pd.DataFrame({'user_id': [], 'name': [], 'affiliation': []})
df2 = pd.DataFrame({'title': [], 'author': [], 'publication_date': [],
'description': [], 'cite_by': []})
links = []

author = driver.find_element_by_css_selector("#gsc_sa_ccl > div:nth-child(3)
> div > div > h3 > a")
while True:
    for i in driver.find_elements(By.CSS_SELECTOR, "div.gs_ai_t"):
        a = i.find_element_by_css_selector('a')
        user_id = a.get_attribute('href').split('=')[-1]
        author = a.text
        affiliation = i.find_element_by_css_selector('div.gs_ai_aff').text
        df = df.append({'user_id': user_id, 'author': author, 'affiliation':
affiliation}, ignore_index=True)
        links.append(a.get_attribute('href'))
    try:
        element = WebDriverWait(driver,
5).until(ec.element_to_be_clickable((By.XPATH, '//button[@aria-
label="Next"]')))
        element.click()
    except:
        break
    time.sleep(1)

for link in links:
    driver.get(link)
    while True:
        try:
            element = WebDriverWait(driver,
5).until(ec.element_to_be_clickable((By.ID, 'gsc_bpf_more')))
            element.click()
        except:
            break
    for i in driver.find_elements(By.CSS_SELECTOR, "tr.gsc_a_tr"):
        a = i.find_element_by_css_selector('a')
        a.click()
        try:
            element = WebDriverWait(driver,
5).until(ec.presence_of_element_located((By.ID, 'gsc_ocd_view')))
        except:
            continue
        title = driver.find_element_by_id('gsc_vcd_title').text
        author = '-'
        publication_date = '-'
        description = '-'
        cited = '-'
        for j in driver.find_elements(By.ID, 'gsc_vcd_table'):
            field = j.find_element_by_class_name('gsc_vcd_field').text
            if field == 'Authors':
                author = j.find_element_by_class_name('gsc_vcd_value').text
            if field == 'Publication date':
                publication_date =
```

```
j.find_element_by_class_name('gsc_vcd_value').text
            if field == 'Description':
                description =
j.find_element_by_class_name('gsc_vcd_value').text
            if field == 'Total citations':
                cited = j.find_element_by_class_name('gsc_vcd_value').text
        df2 = df2.append({'title': title, 'authors': author,
'publication_date': publication_date,
                          'description': description, 'cite_by': cited},
ignore_index=True)
        driver.find_element_by_id('gs_md_cita-d-x').click()
        time.sleep(1)

driver.quit()
df.to_csv("Author Table.csv", index=False)
df2.to_csv("Paper Table.csv", index=False)

scope = ["https://spreadsheets.google.com/feeds",
'https://www.googleapis.com/auth/spreadsheets',
         "https://www.googleapis.com/auth/drive.file",
"https://www.googleapis.com/auth/drive"]

credentials =
ServiceAccountCredentials.from_json_keyfile_name('../client_secret.json',
scope)
client = gspread.authorize(credentials)

spreadsheet = client.open('Author Table')

with open('Author Table.csv', 'r', encoding='iso-8859-1') as file_obj:
    content = file_obj.read()
    client.import_csv(spreadsheet.id, data=content)

spreadsheet = client.open('Paper Table')

with open('Paper Table.csv', 'r', encoding='iso-8859-1') as file_obj:
    content = file_obj.read()
    client.import_csv(spreadsheet.id, data=content)
```

การทำ Visualization

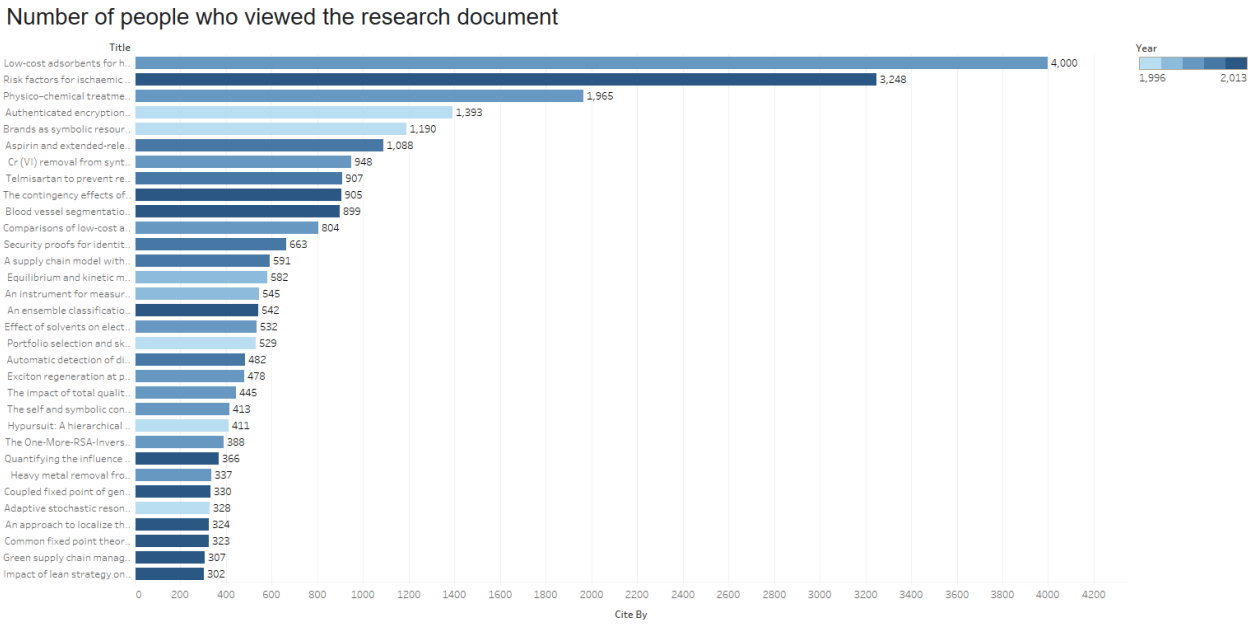
ในขั้นตอนการทำ Visualization จะใช้ Tableau ในการทำ

โดยกราฟแรกจะเป็นกราฟที่แสดงถึงจำนวนของเอกสารวิจัยที่ปล่อยในแต่ละปี โดยจะเรียงจากปีมากสุดไปปีน้อยสุด



ซึ่งจะเห็นได้ว่าในช่วงปี 2020 จะมีเอกสารวิจัยที่ปล่อยออกมามากที่สุด อาจเป็นเพราะเหตุการณ์ COVID-19 ทำให้นักวิชาการหลายท่านมีเวลาในการศึกษาเนื้อหาใหม่ ๆ ทำให้มีการปล่อยเอกสารมาเป็นจำนวนมาก

ต่อมาจะเป็นกราฟที่แสดงถึงจำนวนคนที่เข้าไปดูเอกสารวิจัย โดยจะเรียงจากจำนวนคนที่เข้าชมมากไปน้อย และได้มีการกำหนดให้เรียงสีจากอ่อนไปเข้มโดยดูจากจำนวนปี ยิ่งปีน้อยยิ่งสีอ่อน ยิ่งปีเยอะยิ่งสีเข้ม



จากกราฟ จะได้ว่าเอกสารวิจัยที่ชื่อ Low-cost absorbents มีจำนวนผู้เข้าชมถึง 4,000 คน และมีเอกสารวิจัยชื่อ Risk factors for มีจำนวนผู้เข้าชมรองลงมา มีจำนวน 3,248 คน

Source Code

สามารถดู Source Code ได้ในลิงก์นี้ https://github.com/BrightBct/DSI200-Project.git

สรุป

รายงาน Google Scholar ทำขึ้นเพื่อที่จะได้รวบรวมข้อมูลของนักวิชาการที่อยู่ในสังกัดมหาวิทยาลัยธรรมศาสตร์และเอกสารวิจัยที่นักวิชาการแต่ละท่านได้เขียนขึ้นมา โดยขั้นตอนในการทำแบ่งได้ 2 ขั้นตอนคือ

1. รวบรวมข้อมูลของนักวิชาการที่อยู่ภายในสังกัดมหาวิทยาลัยธรรมศาสตร์ โดยรวบรวม ชื่อ รหัสประจำตัวใน Google Scholar และสังกัดที่นักวิชาการอยู่

2. รวบรวมข้อมูลของเอกสารวิจัยที่นักวิชาการแต่ละท่านได้เขียนขึ้นมา โดนรวบรวม ชื่องานวิจัย ผู้เขียนงานวิจัย วันที่เผยแพร่งานวิจัย รายละเอียดงานวิจัย และจำนวนผู้เข้าชมงานวิจัย

โดยเมื่อทำการรวบรวมข้อมูลแล้ว จะนำข้อมูลดังกล่าวมาทำเป็นกราฟต่าง ๆ เพื่ออธิบายความสัมพันธ์และความเชื่อมโยงของข้อมูล และหวังเป็นอย่างยิ่งว่ารายงานฉบับนี้จะก่อให้เกิดประโยชน์แก่ผู้ที่มาศึกษาต่อไป