Web Crawler Implementation Report

پیاده سازی و استخراج مقالات از وبسایت Google Scholar

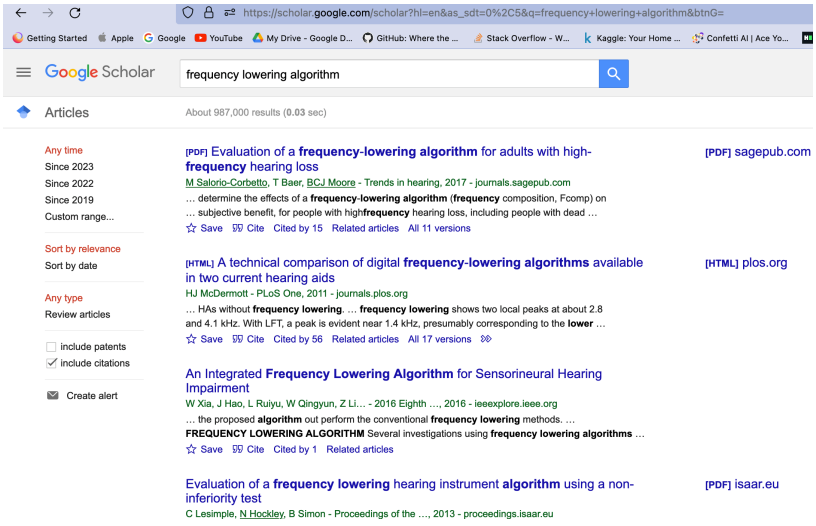
Mohammad Esfandiyar
Instructor’s Name:Dr .chitra Dadkhah
February 18, 2023

گزارش هفته ی اول و دوم (Feb 4-Feb 7)

توضیحات اولیه:

از آنجایی که هدف در وهله ی اول پیاده سازی یک crawler برای جستجو مقالات مورد نظر بر اساس کلید واژه ها،نویسنده، عنوان و ...بود من در دو هفته ی اخیر سه پیاده سازی مختلف انجام دادم و همانطور که مراحل implementation پیش میرفت متوجه شدم که کدام عملی تر و بهتر است و در این مدت ایده های جدیدی را نیز اعمال کردم.که در ادامه به آن ها می‌پردازیم.

گزارش را با یک توضیح کلی پیرامون قابلیت های crawler تا به اینجا شروع می‌کنم.برای شروع کار، crawler از ما لینک صفحه ای از google scholar را که ما به صورت کلی در مورد موضوع مورد نظرمان سرچ کردیم دریافت می‌کند.در وهله ی دوم با متغیر p-key تمام کلماتی که ما می‌خواهیم crawler در مقالات اختصاصا به دنبال آن ها برگردد را تعریف می‌کنیم . برای حل مشکل ،بودن کلمات تکراری و یا اضافه و پر تکرار در زمینه پژوهشی مورد نظر که ممکن است باعث شود نتیجه ی سرچ ما خیلی جامع و بی فایده باشد متغیری دیگر با نام n-key تعریف شده تا اگر دیده شد که این کلمات خیلی در مقاله مورد نظر پر تکرار هستند، آن مقاله را پیشنهاد ندهد.به شکلی که می‌توانیم دقیقا کلمات مورد نظری که می‌خواهیم در مقاله موجود باشد و کلماتی که نمی‌خواهیم در مقاله به آن ها اشاره شود را برای خودمان مشخص و شخصی سازی کنیم.




```
### The start page's URL
start_url = 'https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=frequency+lowering+algorithm&btnG='

### p_key and n
p_key = ['wdrc', 'dynamic range compression', 'hearing aid', 'speech',
         'noise cancellation', 'noise reduction', 'feedback cancellation',
         'sound', 'hearing loss']
n_key = ['imagery', 'image', 'visual', 'video', 'optic', 'opto', 'quantum',
         'photon']

### Google Scholar Crawler, Class Spider
myCrawler = Spider(start_url, p_key, n_key, page=10)
```

در جلسه ای که با خانوم علمی داشتیم، ایشان فرمودند که یک سری کلمات و اصطلاحات مهم تر از باقی عبارات هستند و باید به دنبال یک سیستم برای تشخیص این موضوع باشیم .من برای این موضوع یک سیستم امتیاز بندی طراحی کردم که اگر برای مثال آن کلمه مورد نظر در عنوان مقاله دیده شد، الویت اول و اگر در متن باشد الویت دوم به آن اختصاص پیدا کند و اگر در جای دیگری از مقاله ظاهر شد مانند بقیه عبارات با آن رفتار شود .

```
class Spider:

    def __init__(self, url,
                 p_key=[],
                 n_key=[],
                 score_level=0,
                 key_score={'p': 1, 'n': -3, 'p_none': 1, 'n_none': -1, 'none': -5},
                 weighting={'title': 1.5, 'content': 1},
                 page=5,
                 parser='html.parser',
                 googleScholarURL="http://scholar.google.com.tw"):
```

در قسمت بعد قسمتی طراحی شد تا اگر crawler ما توسط گوگل و یا captcha بلاک شد در همان ابتدا به ما اعلام کند تا ما تا اخر اجرای برنامه صبر نکنیم و بار دیگر با یک ip دیکر تلاش کنیم تا نتیجه ی نهایی کامل به ما نمایش داده شود.

```
page_links = soup.select('div[id="gs_nml"] a')
if not page_links:
    logger.info('1.Google robot check might ban you from crawling!!')
    logger.info('2.You might not crawl the page of google scholar')
```

همان طور که در قسمت قبل به آن اشاره کردم برای بررسی اهمیت هر کلمه با کلمه های دیگر و الویت بندی کردن آن ها یک سیستم جداگانه طراحی شده که در ادامه می‌توانید مشاهده کنید که چگونه کلمات را ابتدا به صورت جداگانه در title بررسی و امتیاز آن ها را ارزیابی می‌کنیم.

```
### Check keywords in titles and contents
### Evaluate the score of titles and contents by keywords
f_title, t_score = ParseOutTitle(result['b_title'], self.p_key, self.n_key, self.key_score)
result['f_title'] = f_title
content, c_score = ParseOutContent(result['content'], self.p_key, self.n_key, self.key_score)
result['require'], result['score'] = self.__requireThesis(t_score, c_score)
```

تغییرات اعمال شده جهت عملکرد موثر و خروجی بهتر:

بر اساس آخرین صحبتی که با خانوم علمی در جلسه داشتم، پیشنهاد دادم که دیتای استخراج شده را با فرمت json ذخیره سازی کنیم چون بر روی کاغذ و تجربه های قبلی خودم این کار انتقال داده را راحت تر می‌سازد.اما در پیاده سازی متوجه شدم برای ذخیره سازی و تبدیل دیتای json به دیتای نهایی که برای کاربر قابل فهم تر باشد ،برنامه به مشکل می‌خورد و باعث Request های زیادی در هنگام اجرای برنامه می‌شود و ممکن

است با توجه به مسائل امنیتی سایت google scholar (که در ادامه بیشتر توضیح میدهم)و زمان محدود جهت استخراج داده تحت شرایطی (وابسته به محتویات صفحه ی آن مقاله)روند استخراج مقالات را نیز مختل کند به همین دلیل این بخش را را به دو قسمت تقسیم کردم.یک :تمام اطلاعات حاصل از crawl، شامل عنوان مقاله و در دسترس بودن یا نبودن نسخه html و pdf برای دانلود به صورت یک جدول با فرمت [title,year,url] به صورت CSV(comma separated values) تحت به کاربر نشان داده می‌شود.در مرحله دوم نسخه ی pdf و یا HTML آن مقالات را در صورت وجود داشتن دانلود و در folder های جداگانه ذخیره می‌کند.

فراموش نکنید اگر قصد داشته باشیم در ادامه پردازشی بر روی فایل های مقالات انجام بدهیم با استفاده از نتایج این crawler هم به لینک url هم به فایل html و فایل pdf(در صورت وجود)دسترسی داریم و می‌توانیم مراحل پردازش متن و crawling را از هم جدا کنیم تا مراحل debugging و پردازش در ادامه راحت تر شود.و یا همچنین می‌توانیم هنگام اجرای context vector دوباره آن هارا به json تبدیل کنیم تا پردازش های nlp به صورت جداگانه انجام شود که دیگر مشکلی پیش نیاید.در ادامه می‌توانید تصاویر کد ها و نتایج جدول csv را مشاهده کنید.

```
### Record the required thesis with tag.
### Set result['tag'], result[tag_link] to None.
### If the thesis is required and its tag is [PDF] or [HTML],
### set result['tag'] to 'PDF' or 'HTML' and also record the
### link in result[tag_link]
result['tag'] = None
result['tag_link'] = None
if result['require']:
    tag = block.select('div[class="gs_ggsd"] a')
    if tag:
        tag_link = tag[0]['href']
        tag_text = tag[0].text

        tag_text = ParseOutTag(tag_text)

        result['tag'] = tag_text
        result['tag_link'] = tag_link
```

در قسمت بعد تصاویر دو جدول دیگر با نام های thesisTitle.csv و thesisneedDownload.csv که هر کدام به ترتیب عنوان مقاله ، سال انتشار و نتیجه موفقیت آمیز بودن یا نبودن دانلود را به ما از نتیجه نشان می‌دهد و در جدول تصویر دوم مقالاتی که با شرایط بالا ، جهت دانلود انتخاب شدند را نمایش داده می‌دهد.

```
def ThesisNeedDownload(df):
    ### Write the required thesis [title, year, url] which
    ### need to be downloaded to a DataFrame and output the
    ### DataFrame to a csv file
    df_table = df[df.tag.isnull() == True][df.require == True][['b_title', 'year', 'url']]
    df_table = df_table.sort_values(by=['year'], ascending=[False])
    title = 'ThesisNeedDownload.csv'
    df_table.to_csv(("./CSV/" + title), encoding='utf-8')

def ThesisPDFDownload(df):
    ### Download the PDF file which has the PDF tag
    df_table = df[df.tag == 'PDF'][['f_title', 'year', 'tag_link']]
    for index, row in df_table.iterrows():
        title = str(row['year']) + ' - ' + row['f_title'] + '.pdf'
        url = row['tag_link']
        print (title)
        print (url)
        try:
            res = requests.get(url)
            if (res.status_code == 200):
                with open(("./PDF/" + title), 'wb') as f:
                    print ("Downloading PDF... ")
                    f.write(res.content)
                    df.at[index, 'download'] = True
        except:
            print ("Can not download the PDF file")
```

```
def ThesisHTMLDownload(df):
    ### Download the HTML file which has the HTML tag and output a PDF file
    options = {'page-size': 'A4', 'dpi': 400}
    df_table = df[df.tag == 'HTML'][['f_title', 'year', 'tag_link']]
    for index, row in df_table.iterrows():
        title = str(row['year']) + ' - ' + row['f_title'] + '.pdf'
        url = row['tag_link']
        print (title)
        print (url)
        try:
            pdfkit.from_url(url, ("./HTML/" + title), options=options)
            df.at[index, 'download'] = True
        except:
            print ("Can not download the HTML file")

def Thesis(df):
    ### Write all the required thesis [title, year, download] in csv file
    df_table = df[df.require == True][['b_title', 'year', 'download']]
    df_table = df_table.sort_values(by=['year'], ascending=[False])
    title = 'ThesisTitle.csv'
    df_table.to_csv(("./CSV/" + title), encoding='utf-8')
```

ThesisTitle.csv — Open with Numbers

	b_title	year	download
18	Speech perception and sound-quality rating with an adaptive nonlinear frequency compression algorithm in Mandarin-speaking hearing aid users	2020	False
77	Design and Performance Evaluation of Frequency Compression Algorithm for Marathi Hearing Aid Users	2018	False
0	Evaluation of a frequency-lowering algorithm for adults with high-frequency hearing loss	2017	False
14	Piecewise-Linear Frequency Shifting Algorithm for Frequency Resolution Enhancement in Digital Hearing Aids	2017	False
20	A NEW ALGORITHM FOR FREQUENCY TRANSPOSING IN DIGITAL HEARING AIDS.	2013	False
1	A technical comparison of digital frequency-lowering algorithms available in two current hearing aids	2011	False
48	An algorithm for intelligibility prediction of time–frequency weighted noisy speech	2011	False
13	Nonlinear Frequency Compression Algorithm	2009	False
93	YIN, a fundamental frequency estimator for speech and music	2002	False
5	Experimental study and improvement of frequency lowering algorithm in Chinese digital hearing aids	0	False
11	Fricative phoneme detection using deep neural networks and its comparison to traditional methods	0	False
99	Pitch determination of speech signals: algorithms and devices	0	False

ThesisNeedDownload.csv — Open with Numbers

	b_title	year	url
77	Design and Performance Evaluation of Frequency Compression Algorithm for Marathi Hearing Aid Users	2018	https://link.springer.com/chapter/10.1007/978-981-10-5903-2_15
20	A NEW ALGORITHM FOR FREQUENCY TRANSPOSING IN DIGITAL HEARING AIDS.	2013	https://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=0973662X&AN=102891875&h=v%2FRy7Hr7RqI9OirNrfa47IEEqDLjuGReDOde4VWK5
13	Nonlinear Frequency Compression Algorithm	2009	https://leader.pubs.asha.org/doi/full/10.1044/leader.IN3.14042009.4
5	Experimental study and improvement of frequency lowering algorithm in Chinese digital hearing aids	0	http://aps.cpsjournals.org.cn/EN/abstract/abstract47511.shtml
11	Fricative phoneme detection using deep neural networks and its comparison to traditional methods	0	https://publica.fraunhofer.de/entities/publication/87a174af-c5c6-4d1a-a7b7-ea9737f9e45a/details
99	Pitch determination of speech signals: algorithms and devices	0	https://www.google.com/books?hl=en&lr=&id=VfT0CAAAQBAJ&oi=fnd&pg=PA1&dq=frequency+lowering+algorithm&ots=v54zbuaR5z&sig=u3sd3MyZBQxsh86jSWCmZ_7cgeA

چالش ها:

مشکل امنیت و محدودیت زمان که به آن اشاره کردم مربوط به امنیت سایت گوگل اسکولار است که بعد از چند ثانیه سرچ کردن database ،سایت ip کاربر را به عنوان crawler و یا بات(BOT)به دلایل مربوط به امنیت دیتای سایت شناسایی و بلاک می‌کند و بعد از تحقیق متوجه شدم که مهم ترین crawler هایی که جهت جمع آوری دیتا از وبسایت نوشته شده اند از پراکسی برای تغییر ای پی استفاده می‌کنند تا به این مشکل بر نخورند اما من راه حل بهتری را برای این موضوع پیاده سازی کردم. با استفاده از متغیر های موجود در کتاب خانه های پایتون(spider) یک متغیر جدید با نام page تعریف کردم تا شماره صفحه را از user بگیرد تا از همان ابتدا مشخص باشد دیتای چند صفحه از گوگل اسکولار را قرار است استخراج کند.تا بعد از چند ثانیه قبل از بلاک شدن بتواند کاملا دیتای سایت را برای جستجو های سریع و مختصر تر استخراج کند .و اگر قصد جستجو صفحات زیاد تری را داشتیم می‌توانیم به راحتی با استفاده از DHCP سرور خودمان هر ۱۰ ثانیه IP سیستممان را تغییر دهیم و یا به سادگی از vpn جهت تغییر ip استفاده کنیم.

نتیجه و جمع بندی:

تا به اینجا با نتایج و قابلیت های crawler امان آشنا شدیم و خروجی های آن را دیدیم.هفته های آینده را به تحقیق و بررسی پیرامون الگوریتم های tf-idf و bag of words بر روی داده های خروجی crawler اختصاص می‌دهم تا با تبدیل خروجی هر یک از مقالات به فرمت json پپردازش را جهت تولید یک context vector از نتایج انجام دهم. تا پس از آن وارد مرحله ارزیابی مدل خود شویم.
در آخر از شما استاد عزیز و خانوم علمی بسیار تشکر می‌کنم که شرایط بندرو درک کردید و تا به اینجای کار به بنده کمک کردید.باز هم بابت مشکلات پیش آمده عذرخواهی می‌کنم و در صورت وجود مشکل و یا ایده بهتر خوشحال می‌شم به بنده اطلاع دهید.