Poster

June 11, 2022

0.1 Importing Required Libraries

```
[ ]: import argparse #used to take values from command line interface
     from selenium import webdriver as wb  #used for websrapping using visually␣
      ↪opening the browser
     import time        #handles the time data
     import os          #handles minor operating system functions
     import glob        # used for getting the files in order
     import shutil      # used for moving and renaming files properly
```

0.1.1 Function to scrape title and url data from google scholar

```
[ ]: def get_title_urls(query: str,thresh: int) -> list:

         ''' query :: takes the word for the search
             thresh :: is the number of top options to keep
             returns a list of urls of top results'''

         driver = wb.Chrome("./chromedriver.exe")
         # driver.minimize_window()  # Uncomment to minimize the window
         driver.maximize_window()    # Comment it out to avoid the window maximizing
         driver.get('https://scholar.google.com/')      #get function lets the␣
      ↪browser driver to get to the url specified as the string
         time.sleep(2)                                   #a little delay to let the␣
      ↪elements load on the webpage
         driver.find_element_by_xpath('//*[@id="gs_hdr_tsi"]').send_keys(query)      ␣
      ↪  #finding the textbox element to input the query
         time.sleep(3)
         driver.find_element_by_xpath('//*[@id="gs_hdr_tsb"]/span/span[1]').click() ␣
      ↪  #clicking the search button for the search
         time.sleep(2)
         main_elem = driver.find_elements_by_class_name('gs_rt')                      ␣
      ↪  #finding the element that contains the titles and the corresponding urls
         title_list= []                          #list that stores the titles
         url_list=[]                             #list that stores the urls
         for elem in main_elem:
```

```
        title_list.append(elem.text)     #getting the text property of the␣
↪element to get the title and appending the title to the title list
        hr = elem.find_element_by_tag_name('a')      #getting the link␣
↪associated with the element
        url_list.append(hr.get_attribute('href'))    #now getting the href or␣
↪url linked with the element
    driver.close()                                   #closing the driver
    try:
        return title_list[:thresh],url_list[:thresh] #returning only the␣
↪limited vlaues as per the requirement limit i.e. threshold value
    except:
        print("[--] Threshold value is either exceeding or falling behind")
        print(f"[==] Length value of the list is {len(url_list)}")
        return title_list,url_list                  #if the threshold value is␣
↪higher than the number of paper available
```

0.1.2 Function to download pdfs from sci-hub

```
[ ]: def sci_hub_download(url: str,down_time = 15) -> bool:
         ''' url : string holding the web address
             down_time : this integer holds the number of seconds the browser need␣
     ↪to wait till the pdf download completes
             returns either True or False if downloads thd pdf or not '''
         driver = wb.Chrome()                  #creating the driver
         driver.maximize_window()              #maximizing the windows for visualization␣
     ↪purposes
         driver.get('https://sci-hub.mksa.top/')     #getting into the sci-hub␣
     ↪website
         driver.find_element_by_xpath('//*[@id="input"]/form/input[2]').
     ↪send_keys(url)    #finding the search bar and inputting the url into it
         driver.find_element_by_xpath('//*[@id="open"]').click()                 ␣
     ↪     #clicking the button to search for article if available or not
         try:
             time.sleep(5)
             driver.find_element_by_xpath('//*[@id="buttons"]/button').click()   ␣
     ↪     #finding one way to get the save button that downloads the pdf
             time.sleep(down_time)          #giving the browser some time to␣
     ↪download the document
             driver.close()
             return True               #returning true if the download happends
         except:
             print('[-] COULD NOT FIND THE SAVE BUTTON')
             print('[+] TRYING ANOTHER WAY ...')
             try:
                 time.sleep(3)
```

```
            driver.find_element_by_xpath('//*[@id="buttons"]/ul/li[2]/a').
↪click()    #trying another way to get the save button that downloads the pdf
            time.sleep(down_time)
            driver.close()
            return True
        except:
            print("[--] Could not find a way to download the pdf")
            return False                                #returning false if the␣
↪download doesnot happengher than the length of the list then returns the list
```

0.1.3 Function to rename files properly

```
[ ]: def refine_rename(title: str ) -> str:
         ''' Excludes the characters that are declined by the renaming policy of␣
     ↪windows file system'''
         excluder = [":","-",";"]      # some values that is to be excluded
         new_values = []
         for value in list(title):
             if not value in excluder and not value == " ": new_values.append(value)␣
     ↪       #deleting the excluder values excluding the space values
             if value == " " : new_values.append("_")                              ␣
     ↪       #replacing the space values with underscoer(_)
         return "".join(str(item) for item in new_values)                          ␣
     ↪       #converting the new list into a string
```

0.1.4 Function to place the files in proper directories

```
[ ]: def save_in_download(query : str ,title: str) -> None:
         ''' query : this variable stores the search query and renames the results
             title : this variable stores the title of the pdf downloaded for the␣
     ↪renaming purposes '''
         path = f"C:/Users/avyar/Downloads/{query}"                          ␣
     ↪#creating the path for directory for the query
         if not os.path.exists(path[:-1]):                         #if the directory␣
     ↪does not exists that this creats it
             os.mkdir(path[:-1])
         list_o_files = filter(os.path.isfile, glob.glob("C:/Users/avyar/Downloads/
     ↪"+"*"))       #getting all the files in the folder
         sorted_files = sorted(list_o_files, key=os.path.getmtime)                  ␣
     ↪             #sorting the files according to recently modified
         recent_one = sorted_files[-1]                                              ␣
     ↪             #getting the last files after sorting i.e. the file that is just␣
     ↪downloaded
         recent_one = recent_one.split("\\")[-1]                                    ␣
     ↪ #splitting the text using a \ due to some unorthodox renaming convention␣
     ↪created by the system
```

```
    print(recent_one)
    src_path = f"C:/Users/avyar/Downloads/{recent_one}"                        ␣
↪   #mentioning the path of the new file
    new_src_path = f"C:/Users/avyar/Downloads/{refine_rename(title)}.pdf"     ␣
↪   #mentioning the path of the file after the modification
    os.rename(src_path,new_src_path)                                          ␣
↪   #renaming the file and it's location
    dst_path = f"C:/Users/avyar/Downloads/{query[:-1]}/{refine_rename(title)}.
↪pdf"     #defing the file path to move the file
    shutil.move(new_src_path, dst_path)                                       ␣
↪   #moving the file using some utility functions
```

0.1.5 Function to read queries from the local file "queries.txt"

```
[ ]: def read_query(path: str ) -> list:
         ''' Reads multiple queries as per the text file mentioned in the path'''
         with open(path, 'r') as f:queries = f.readlines()             # reading the␣
     ↪file that contains the queries
         return [query for query in queries if query[0] != "#"]        # returning␣
     ↪the list of the queries
```

0.1.6 Function to move all query directories to the "./depostion" directory

```
[ ]: def move_final(queries : list):
         ''' this function moves the files to the main directory of submission from␣
     ↪the download folder'''
         down_path = "C:/Users/avyar/Downloads/"                  #specifying the␣
     ↪download path as a string
         depo_path = "./deposition/"                              #specifying the new␣
     ↪directory path to move the pdf files
         for query in queries:
             shutil.move(down_path+f"{query[:-1]}",depo_path+f"{query[:-1]}")     ␣
     ↪#moving the files from download to the required location
```

0.1.7 Defining main parameters and run

```
[ ]: if __name__ == "__main__":
         parser = argparse.ArgumentParser(description= "give thershold value for the␣
     ↪number of papers")                #using argument parser for cli input
         parser.add_argument("--thresh",
                             type = int,
                             help = "this param holds the threshold value of maximum␣
     ↪value for the available papers")    #creating args for the threshold numeric␣
     ↪value
         parser.add_argument("--down_time",
                             type = int,
                             help = "this param holds the threshold value of maximum␣
     ↪value for the available papers")    #creating ars for the download time delay
         args = parser.parse_args()                                              #parsing␣
     ↪the values from the cli
         query_list = read_query("./queries.txt")                               #reading the␣
     ↪queries
         thresh = args.thresh                                                   #getting the␣
     ↪threshold values from the parser
         down_time = args.down_time                                             #getting the␣
     ↪down_time values from the parser
         for query in query_list:
             title_list, url_list = get_title_urls(query,thresh)                #getting the␣
     ↪titles and urls from the google scholar webpage
             for title,url in zip(title_list,url_list):                         #iterating␣
     ↪over the list to download the pdfs
                 if sci_hub_download(url,down_time):                            #if the␣
     ↪download occures the file will be placed as per path specified
                     save_in_download(query,title)
         move_final(query_list)                                                 #finally␣
     ↪moving the files from download to the required locations
```