Development of a program for determining the Hirsch index using Web of Science and Scopus pages (by Research Id and Author Id)

F R Khakimov¹, O S Kharitonova²⁻⁵, V V Bronskaya¹³, T V Ignashina³ and D S Balzamov⁴

¹Department of Information Systems, Kazan Federal University, 35 Kremlyovskaya Street, Kazan 420008, Russian Federation
²Department of Chemical Process Engineering, Kazan National Research Technological University, 68 Karl Marx Street, Kazan 420105, Russian Federation
³Department of Chemical Technology of Petroleum and Gas Processing, Kazan National Research Technological University, 68 Karl Marx Street, Kazan 420015, Russian Federation
⁴Department of Power Supply of Enterprises and Energy Resource Saving Technologies, Kazan State Power Engineering University, 51 Krasnoselskaya Street, Kazan 420066, Russian Federation

olga.220499@mail.ru

Abstract. The analysis of Web of Science, Scopus sites to extract the Hirsch index is carried out. Programs to extract the Hirsch index using Scopus and Web of Science pages by Research Id and Author Id are created. A program that is a set of scripts in Python for the implementation of the automatic Hirsch index determination on Web of Science and Scopus sites was developed in this work. Collecting the data of scientists about their h-index which will help in assessing their scientific productivity is possible using this program.

In the modern world any information can be accessed from any web browser. In addition the amount of information is rapidly growing increasing many times. The question - how to collect and analyze data that are kept in store on the Internet, for example, universities often need to analyze the performance of scientific activities of their employees using the Hirsch index - comes in the first place.

Identifying this important information is extracting, structuring and collecting data from a website. Unlike human robot is difficult to disassemble information among advertising, text, links [1].

The primary problems which a bot implementing the web scraping process is faced with are fuzzy structure of web pages that is easy to perceive information by human and protection against bots including captcha, hidden data fields in the document, analysis of user behavior, blocking IP addresses, user authentication, checking request headers. In this case it is used Web scraping that is extraction of data from websites. This process is automated that is executed by a program. Web scraping consists of stages: download web-page, document parsing, data collection and subsequent analysis of results.
Popular Python libraries such as Beautiful Soup, Requests, Scrapy and Selenium are used for extracting information from known scientometric data [1]. Unlike others we have chosen the approach using the Grab web scraping tool.

“Scopus is the largest common database containing annotations and information about the citation of peer-reviewed scientific literature with tools for tracking, analyzing and visualizing data” [2].

“Web of Science is an Internet search platform that unites abstractive databases of publications in scientific journals and patents including databases that take into account the mutual citation of publications” [3-5].

“The Hirsch index (or h-index) is a quantitative characteristic of productivity of a scientist, a group of scientists, a scientific organization or a country as a whole based on the number of publications and the number of citations of these publications” [6-8].

The existing solutions were considered, the structure of websites was analyzed, the program was developed, the program results were analyzed. In the course of this work a program that is a set of scripts in Python for the implementation of the automatic Hirsch index determination on Web of Science and Scopus sites was developed. It is possible to collect the data of scientists about their h-index which will help in assessing their scientific productivity using this program.

The program consists of modules each of which solves its own problem. A component diagram indicates their interaction (Figure 1).

![Figure 1. Component diagram.](image)

The output.py component is responsible for performing operations on the ScrappedData database using the sqlite3 library interface.

The scopus.py, wos.py, and teachers.py components realise methods to extract data from Scopus, Web of Science pages. The program entry point is the program.py component in which the main methods of the three components specified earlier are started up.

The functionality of working with the database is represented as the DataCtx class.

```python
class DataCtx:
    __conn = connect('ScrappedData.db')
    __cursor = None

def __init__(self):
    self.rowcount = 0
    self.__cursor = self.__conn.cursor()
```

![Figure 2. DataCtx class.](image)

This class has fields: database connection and cursor. SQL query execution methods - as arguments the query string and its parameters, returning the number of changed strings (Figure 3). These methods create, clean and delete tables, paste, update, and delete strings.
The data extraction and returning the list of results - the select () method (Figure 4).

```python
def select(self, query: str, *params):
    self.__cursor.execute(query, *params)
    rows = self.__cursor.fetchall()
    self.rowcount = len(rows)
    return rows
```

**Figure 4.** Select () method.

A file using the save_to_file () method, in the input of which is fed the way to the file that makes it possible to upload all the records of the table to the text (Figure 5). If there is no table with results yet, a warning about it is printed.

```python
def save_to_file(self, path: str):
    try:
        with open(path, 'w', encoding='utf-8') as out:
            for row in self.select("select * from teachers");
                out.write('|'.join([str(i) for i in row]) + '\n')
    except OperationalError:
        remove(path)
        print('There is no table in the database yet!'
            'Try again after creating the table.')
    else:
        print("All data has been successfully saved to the file.")
```

**Figure 5.** Save_to_file () method.

The last method will close the connection after the class object is destroyed (Fig. 6).

```python
def __del__(self):
    self.__conn.close()```

**Figure 6.** Method def_del_ ()

The Web of Science web page parser where the Hirsch index extraction method takes the ResearcherId identifier input and looks for author's publications was considered. Then it goes to the quotation report and returns the Hirsch index (Figure 7).
def get_h_wos(id):
g = Grabc()
g.go('https://apps.webofknowledge.com/WOS_GeneralSearch_input.do')

try:
g.doc.set_input_by_id('_id','value=input1', value=id)
except DataNotfound:
    print("Search is not available! It's necessary to login.")
raise PermissionError
g.doc.set_input_by_id('id=select1', value='AI')
g.submit()

try:
xpath = '//*[@id="view_citation_report_image"]/div/div/a/@href'
href = g.doc.select(xpath).text()
except DataNotfound:
    return 'None'
g.go('https://apps.webofknowledge.com#/? & href')
return g.doc.select('//*[@id="H_INDEX"]').text()

Figure 7. Output of Hirsch index with WoS

A possible case is that the method will return an access error and report an authorization problem, so the script going was run not in the network of the required organization. If the author has no publications, the method returns None string. The scrape_wos () method selects employees who have a profile on Web of Science from the ResearcherId teachers table. Then it passes them to the get_h_wos () method execution to 8 processes. After collecting Hirsch indexes the method updates the table by adding the collected value to the h_index column.

def scrape_wos():
    ctx = DataCtx()
    query = 'select researcher_id from teachers where ' 'researcher_id <> "None"
    teachers_ids = ctx.select(query)

    pool = Pool(8)
    results = pool.map(get_h_wos, teachers_ids)

    query = 'update teachers set h_index = ? ' 'where researcher_id = ?'
    params = [(i, j) for i, j in zip(results, teachers_ids)]
    ctx.execute_many(query, params)

    Figure 8. Scrape_wos() method.

The code of get_h_scopus () method is short and simple. It accepts AuthorId in the input, goes to the author's page, takes the Hirsch index and returns it. If it is not, it returns None (Figure 9).

def get_h_scopus(author_id):
g = Grabc()
url = "https://www.scopus.com/authid/detail.uri?authorId=%s" & author_id
g.go(url, timeout=20)

xpath = '//*[@id=authorDetailsHindex]/div/div[2]/span'
try:
    return g.doc.select(xpath).text()
except DataNotfound:
    return 'None'

Figure 9. Extracting Hirsch Index with Scopus
As in the previous module the second method collects employees' identificator, runs ten processes to work, and updates records in the table. The Hirsch index is updated only where initially was less than or equal to None (Figure 10).

```python
def scrape_scopus():
    ctx = DataCtx()
    query = 'select author_id from teachers where author_id <> "None"'
    teachers_ids = ctx.select(query)

    pool = Pool(10)
    results = pool.map(get_h_scopus, teachers_ids)

    query = 'update teachers set h_index = ? where (h_index < ?' +
             ' or h_index = "None") and author_id = ?'
    params = [(i, i, "") for i, j in zip(results, teachers_ids)]
    ctx.execute_many(query, params)
```

Figure 10. Scrape_scopus() method.

The input point of the program from which three methods for collecting data from the previous three modules are run is in this main module (Figure 11).

```python
if __name__ == '__main__':
    scrape_kfu()
    scrape_wos()
    scrape_scopus()
```

Figure 11. The input point.

Table 1 shows the average time of the module work. Time measurement was performed only for data collection processes from web pages i.e. the time taken to read the input and write the output data was not taken into account.

| Module Name   | Working time without paralleling, sec | Working time with several processes, sec |
|---------------|---------------------------------------|------------------------------------------|
| teachers.py   | 1200                                  | 246                                      |
| wos.py        | -                                     | 980                                      |
| scopus.py     | 220                                   | 22                                       |

Table 1. Module operating time.

Kazan Federal University employees were selected as input data to check the program. Information about each employee is presented in his profile on the official website of the university and is publicly available for viewing. The output data were analyzed and diagrams were constructed to better visualize the results. One of them is shown in figure 12.
Figure 12. Percent ratio of the number of employees with h-index on the departments of one of the institutes of KFU

A program decision as a set of scripts in Python for the implementation of the automatic Hirsch index determination using Web of Science and Scopus sites was developed in the course of this work. You can collect data from scientists about their Hirsch index through the use of this program.

References
[1] R Mitchell 2018 Web Scraping with Python Allyson MacDonald Sebastopol O’Reilly Media, Inc. 280
[2] Scopus — Elsevier in Russia URL: http://elsevierscience.ru/products/scopus/
[3] Web of Science — Wikipedia URL: https://ru.wikipedia.org/wiki/Web_of_Science
[4] T Braun, W Gla’nz and A Schubert 2005 A Hirsch-type index for journals The Scientist 19 8
[5] B Cronin and L Meho 2006 Using the h-index to rank influential information scientists Journal of the American Society for Information Science and Technology 57 1275-8
[6] Web scraping — Wikipedia URL: https://en.wikipedia.org/wiki/Web_scraping
[7] L Egghe 2006 Theory and practise of the g-index Scientometrics 69 131-52
[8] W Gla’nz 2006 On the h-index: a mathematical approach to a new measure of publication activity and citation impact Scientometrics 67 315-21