



Project

Silvio Peroni

✉ silvio.peroni@unibo.it  0000-0003-0530-4305  @essepuntato

Computational Thinking and Programming (A.Y. 2017/2018)

Second Cycle Degree in Digital Humanities and Digital Knowledge

Alma Mater Studiorum - Università di Bologna



Creative Commons Attribution 4.0 International License

Scholarly Search Engine (SSE)

The name of the project is *Scholarly Search Engine*

It is a software that takes in input a file in a particular format (CSV) and allows one to run particular search operations

The input CSV file describes some metadata of existing scholarly articles published in a few academic journals

You need a group

The project must be implemented by a group of people

You need to

- form groups of at least 3 and at most 4 people
- choose a name for the group (yes: a name) - it will be used to publish the ranks of the best-performing projects (more info later)
- communicate the name of the group and its members (including their emails) to me by sending an email at silvio.peroni@unibo.it

All groups must be ready by next Monday (11 December)

Project stub

```
from <group_name> import *

class ScholarlySearchEngine():
    def __init__(self, source_csv_file_path):
        self.data = process_data(source_csv_file_path)

    def search(self, query, col, is_number, partial_res=None):
        result_data = do_search(self.data, query, col, is_number, partial_res)
        return result_data

    def pretty_print(self, result_data):
        list_of_strings = do_pretty_print(self.data, result_data)
        return list_of_strings

    def publication_tree(self, author):
        pub_tree = do_publication_tree(self.data, author)
        return pub_tree

    def top_ten_authors(self):
        list_of_tuples = do_top_ten_authors(self.data)
        return list_of_tuples

    def coauthor_network(self, author):
        coauth_graph = do_coauthor_network(self.data, author)
        return coauth_graph
```

Your output

You have to develop a Python file named as your group, where spaces are substituted by underscores and the whole file is lowercase

E.g.: group *Best group ever*, file `best_group_ever.py`

The import statement must specify the module implemented by the file

E.g.: `from best_group_ever import *`

Each group has to implement the six functions that have been highlighted in red in the previous slide

An example of input will be provided for testing the software

process_data

```
def process_data(source_csv_file_path)
```

It takes in input a comma separated value (CSV) file, and return a data structure containing all the data included in the CSV in some form

The data can be preprocessed, changed according to some empirical rule, ordered in a certain way, etc.

These data will be automatically provided in input to the other functions

Example of CSV data

title	authors	doi	abstract	categories	keywords	journal	volume	references	figures	tables	year
The appropriation of GitHub for curation	Yu, Wu; Na, Wang; Jessica, Kropczynski; John M., Carroll	10.7717/peerj-cs.134	GitHub is a widely used online collaborative software [...]	Human-Computer Interaction; Software Engineering	Curation; GitHub; Appropriation	PeerJ Computer Science	3	36	1	1	2017
...
...

authors (list of strings): *Given Name 1, Family Name 1; Given Name 2, Family Name 2; ...*

categories (list of strings): *category 1; category 2; ...*

keywords (list of strings): *keyword 1; keyword 2; ...*

do_search

```
def do_search(data, query, col, is_number, partial_res=None)
```

data: the data returned by `process_data`

query: the textual query (type: string)

col: the name of the column of the data in which to search (type: string)

is_number: says if the content of the column (and, implicitly, of the query) should be interpreted as numbers instead of simple strings (type: boolean)

partial_res: if not `None`, they are a reduced set of data on which the query should be applied

Return a selection of the data processed before which
answer to the query

Kinds of query

The query is a string that actually can contains a quite complex set of operations, according to the following ones:

- boolean operator: and, or, not
 <tokens 1> <operator> <tokens 2>
- comparison operator: <, >, <=, >=
 <operator> <tokens>

Examples:

- World Wide Web and not Artificial Intelligence: search for the field containing “*World Wide Web*” and not containing “*Artificial Intelligence*”
- < 2016 or > 2017: search for the field containing a value lesser than “2016” or greater than “2017”

Wild cards and rules

Multiple wildcards “*” can be used, e.g. “*World*Web*” looks for all the string that matches with the word “*World*” followed by zero or more characters, followed by the word “*Web*” (examples: “*World Wide Web*”, “*World Spider Web*”, etc.)

All searches are case insensitive – e.g. searching for “*World*” will match also strings that contain “*world*”

It must be possible to execute another `do_search` on the partial data returned by a previous execution of the same function

do_pretty_print

`do_pretty_print(data, result_data)`

`data`: the data returned by `process_data`

`result_data`: the set of data returned by a previous run of the function `do_search`

Return a list of strings for each of the result contained in `result_data` according to the following format:

<Family Name 1> <Given Name Initials 1>, [...], <Family Name N> <Given Name Initials N>. (<year>). <title>. <journal> <volume>. <https://doi.org/<doi>>

E.g.: Wu Y, Wang N, Kropczynski J, Carroll JM. (2017) The appropriation of GitHub for curation. PeerJ Computer Science 3. <https://doi.org/10.7717/peerj-cs.134>

do_publication_tree

`do_publication_tree(data, author)`

data: the data returned by `process_data`

author: the complete name (i.e. given name and family name) of the author for which one wants to retrieve its publication tree

Return all the publications written by such author described by the following tree:

- root node: author name
- children of root node: as many node as many years of author's publication
- children of an year node: the result of the `do_pretty_print` function applied to the particular article written by the author and published in that year

do_top_ten_authors

```
do_top_ten_authors(data)
```

data: the data returned by `process_data`

Return a list of tuples with two items each, where the first item is a string representing the author full name and the second item is the number of papers it has published

The tuple in the list are sorted in descending order, starting from the author with most publication

do_coauthor_network

`do_coauthor_network(data, author)`

data: the data returned by `process_data`

author: the complete name (i.e. given name and family name) of the author for which one wants to retrieve its coauthor network

Return a graph having a central node for the input author and a series of additional connected nodes for each other person who co-authored a paper with the input author, where the weight of the arc connecting the input author with his co-author is the number of paper they have co-authored together

Submission

The project, i.e. the implementation of the six functions including any additional ancillary function developed, must be included in a single file named as your group, e.g.

`best_group_ever.py`

The file must be sent by email to me at silvio.peroni@unibo.it

Submission: 2 days before the exam session (the whole group must attend the session) – e.g. send the project the 21st of January for discussing it on the session of the 23rd of January

Your project will be compared with the others in terms of efficiency (i.e. time needed for addressing specific tasks)

Evaluation

Maximum score = c-score + e-score + o-score = 16

All projects will run on a large CSV file

Correctness of the result: c-score ≤ 4

Efficiency of the software: e-score ≤ 4 ; projects ranked according to the time spent for addressing various tasks

1. e-score = 4
2. e-score = 3
3. e-score = 2



Oral colloquium: $-8 \leq \text{o-score} \leq 8$; it is a personal score, each member of the group has its own

At least 2 for passing the exam, otherwise an additional effort is required (new function)

END

Project

Silvio Peroni

✉ silvio.peroni@unibo.it  0000-0003-0530-4305  @essepuntato

Computational Thinking and Programming (A.Y. 2017/2018)

Second Cycle Degree in Digital Humanities and Digital Knowledge

Alma Mater Studiorum - Università di Bologna