ENGR 212 Programming Practice
Mini Project 5
April 25, 2016 (Due: May 9, 2016)

Publications of faculty members at a university are usually published on university web pages. As an example, publications of SEHIR's CS faculty members are published at their profile pages (e.g., see Ahmet Bulut's publications at http://cs.sehir.edu.tr/en/profile/6/Ahmet-Bulut/). However, searching for particular publications is not usually possible on such web pages. In this project, you are going to develop a publication search engine (similar to Google Scholar) that will allow searching for publications of CS faculty members with some optional filters. Details regarding the requirements are as follows:

1. Your program will have a graphical user interface (GUI) which will look like as shown in Figure 1. Details about how it should work are provided below.

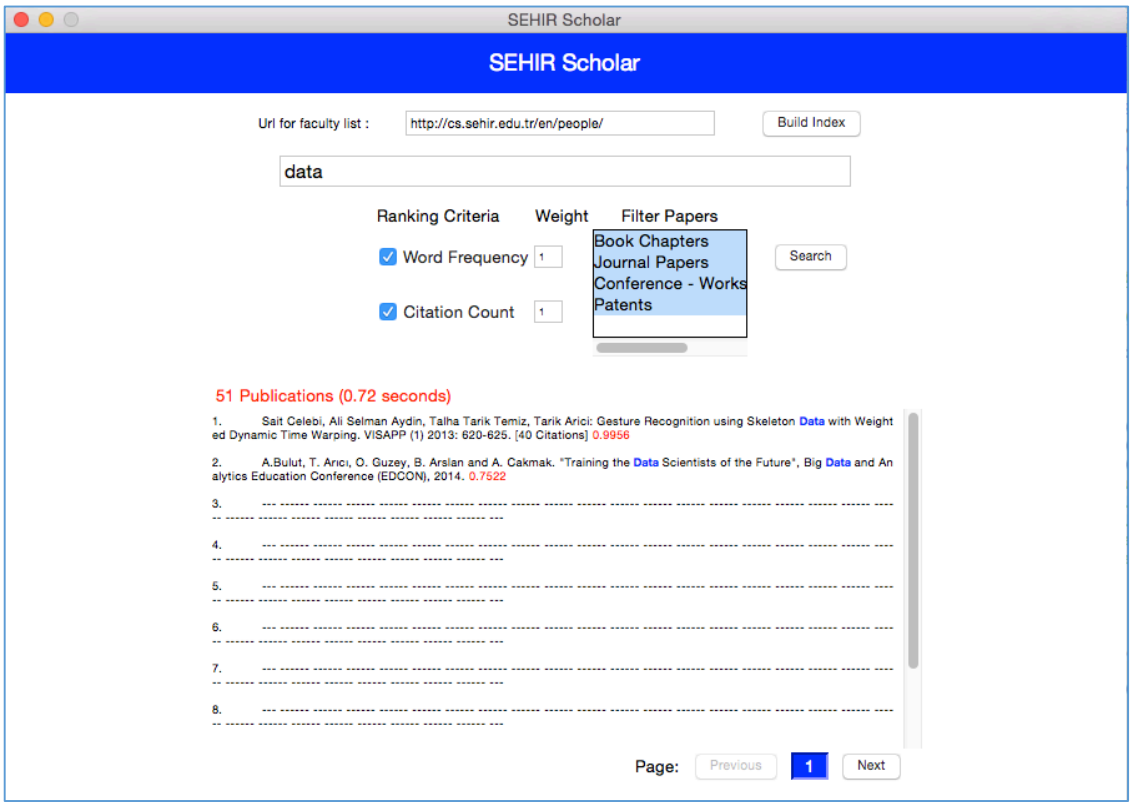

Figure 1

- The user will provide a URL to a web page that lists CS faculty members (e.g., http://cs.sehir.edu.tr/en/people/). Then the user will click on "Build Index" button.
- Then, your tool will process the web page at the provided url using urllib2 and BeautifulSoup, follow the links to the profile pages of each individual faculty member. On each faculty profile page, you need to extract publications of the faculty member, put them into your database (i.e., index). You need to design a database using shelve module. You will decide what dictionaries you will maintain, and what their structures would be. You may want to see mysearchengine.py as an example, but you would have to come up with a modified design to solve the current problem. Note that you are indexing individual publications rather than pages.
- Next, the user will setup the search settings as follows:
    - *Keywords:* In a box, the user will provide one or more search keywords.
    - *Ranking:* You are going to offer two ranking measures that users can combine or use individually.
        - Frequency-based measure: You are going to write a method that will compute a kind of frequency-based score. More specifically, let's say that a user has searched with three

keywords, "word1 word2 word3". Assume that a paper p contains 3 occurrences of word1, 2 occurrences of word2, and 5 occurrences of word3. Then, the content-based score of p is 3 x 2 x 5 = 30. You need to call the normalization function (copy from mysearchengine.py) to normalize the scores at the end (higher is better).

- Citation count-based measure: You will use the citation count of papers as a ranking measure. Again, you need to normalize the scores as stated above.
- In order to combine the above two scores, you need to take into account the weights that the user will provide (the default weight should be 1 for each measure). You may want to see mysearchengine.py for a similar method to adapt for your needs.

- *Paper Type Filter*: The user should be able to search in one or multiple particular categories of papers. In a listbox, you should provide user with a number of available paper types. This listbox should be automatically populated (and sorted alphabetically) from the pages that you crawl, and should allow multiple selection. By default, all types should be selected.
- If the user does not do one or more of the following, your program should generate a warning message with a proper text:
  - provide at least one keyword,
  - choose at least one ranking measure,
  - provide the weights if multiple ranking measure is selected,
  - choose at least one paper category

- Next, the user will click on "Search" button. Your program should search your database for papers that contain all the provided keywords filtered and ranked according to the above settings.
  - At the top, you should let the user know how many papers matched his/her query, and how long the search took.
  - Search results should be paginated with 10 results on each page (except the last one which may have less). A scrollbar should be used if 10 search results do not fit into the visible part of the search result area. You should have pagination buttons at the bottom (e.g., Previous and Next) as shown in Figure 1 to navigate between different pages. Between these buttons, there should be a label showing the current page number (which is 1 by default).
  - The matched keywords should be made bold and colored (e.g., blue) for instant recognition of matched keywords and their locations on result pages.
  - Search results should be numbered starting from 1. For each paper in the search result, you should provide its citation count as well. The computed ranking score should be also listed next to the publication as shown in Figure 1.

Can you provide any further pointers that may be helpful?:

- You **should not** import mysearchengine.py. You should transfer the parts that you would need or modify into your own code file.
- You may use Text widget for the search result area. You may want to use tag_add, tag_modify, etc. methods of Text widget for color and font configuration of particular regions on the search results (see http://stackoverflow.com/questions/14786507/how-to-change-the-color-of-certain-words-in-the-tkinter-text-widget).
- To measure the elapsed time, you may see the following: http://stackoverflow.com/questions/3620943/measuring-elapsed-time-with-the-time-module

Warnings:

- **Do not** talk to your classmates on project topics when you are implementing your projects. **Do not** show or email your code to others. **Do not** work together if you are not in the same group. If you need help, talk to your TAs or myself, not to your classmates. If somebody asks you for help, explain

them the lecture slides, but do not explain any project related topic or solution. Any similarity in your source codes will have **serious** consequences for both parties.

- Carefully read the project document, and pay special attention to sentences that involve "**should**", "**should not**", "**do not**", and other underlined/bold font statements.
- If you use code from a resource (web site, book, etc.), make sure that you reference those resource at the top of your source code file in the form of comments. You should give details of which part of your code is from what resource. Failing to do so **may result in** plagiarism investigation.
- Even if you work as a group of two students, each member of the team should know every line of the code well. Hence, it is **important** to understand all the details in your submitted code. You may be interviewed about any part of your code.

How and when do I submit my project? :

- Projects may be done individually or as a small group of two students (doing it individually is recommended for best learning experience). If you are doing it as a group, only **one** of the members should submit the project. File name will tell us group members (Please see the next item for details).
- Submit your own code in a **single** Python file. Name your code file with your and your partner's first and last names (see below for naming).
  - If your team members are Deniz Barış and Ahmet Çalışkan, then name your code file as deniz_baris_ahmet_caliskan.py (Do **not** use any Turkish characters in file name).
  - If you are doing the project alone, then name it with your name and last name similar to the above naming scheme.
  - Those who **do not** follow the above naming conventions **will get -5 off** of their grade.
- Submit it online on LMS (Go to the Assignments Tab) by **17:00 on Monday, May 9, 2016 (Please note the Monday submission, it is not on Tuesday)**.

  Late Submission Policy:

  - -10%: Submissions between 17:01 – 18:00 on the due date
  - -20%: Submissions between 18:01 – midnight (00:00) on the due date
  - -30%: Submissions which are 24 hour late.
  - -50%: Submissions which are 48 hours late.
  - Submission more than 48 hours late will not be accepted.

Grading Criteria? :

Code Organization			Functionality					
Meaningful variable names (%3)	Classes and objects used (%4)	Sufficient commenting (%4)	Compiles? (20)	GUI Design (10)	Reading papers, building the index (20)	Filtering out papers based on paper category (10)	Searching with single or multiple keywords and showing results properly in pages (20)	Ranking properly (10)

- Interview evaluation
  - Your grade from interview will be between 0 and 1, and it will be used as a coefficient to compute your final grade. For instance, if your initial grade was 80 before the interview, and your interview grade is 0.5, then your final grade will be 80*0.5 = 40. Not showing up for the interview appointment will **result in** grade 0.

Have further questions? :

- Please contact your TAs (Jareth or Dogukan are focusing on projects. You may want to talk to them first, but you may talk to Ali and Bekir as well) if you have further questions. If you need help with anything, please use the office hours of your TAs and the instructor to get help. **Do not walk in randomly (especially on the last day) into your TAs' or the instructor's offices. Make an appointment first. This is important. Your TAs have other responsibilities. Please respect their personal schedules!**