Fall 2018 PIC 16 Data Project Report

Zixuan Wang, *University of California, Los Angeles*
Shuyuan Rong, *University of California, Los Angeles*

Abstract

Nowadays, big data has triggered continuously heated discussion. There exist plenty of applications utilizing optimized datasets to analyze an existing trend or phenomenon in the real world. Data visualization has become one of the most significant tasks for data scientists by directly displaying the data result in a pleasing and convincing method. Considering the importance in high performance of Networkx library in python, our data project visualization is based on Networkx and we are trying to figure out other Python libraries other than Networkx to avoid wasting too much time to handle sorting sequence and extra visualization tools. We evaluate the advantages and disadvantages when sorting, analyzing, outputting in Python Networkx, igraph and matplotlib to help to make our choice.

1. Introduction

We choose the Citation-netwrok V1 dataset in https://aminer.org/citation. Choosing the Citation-network V1 is because it is well sorted, and every feature of the book is well-organized. The data sets containing large amount of information towards 629,814 papers and has over 632,752 citation relationships and thus it is assured that the reliability of conclusion or networks derived from analysis of this data set is relatively high. In the meantime, the data size comparing to others available on this resource website is not so challenging and does not require advanced computer system to run the project. Considering our design of the project, the performance using Python Regex and Networkx is not up to our expectation due to the large amount of time on running datasets, sorting and visiting sorted results when analyzing different features. To solve such problems and to better visualize the graph, we are looking for the effectiveness of using lists, Collection library, Matplotlib in Python library.

In the dataset we choose, for every paper, it contains: papers titles, paper authors, published year, publication venue, indexes id of this paper, the id of references of this paper and the abstract of the paper. From information provided above, we intend to extract information from reference numbers and authors to get conclusions about the importance of the paper and author by analyzing the rates papers have been referenced and the collaborations between authors who ranked as highest productive authors. By visualizing the network graph between the top first paper that is referenced at most times and other high ranked papers that reference it, the huge referencing network behind this paper is displayed clearly and directly. By visualization the five papers with top highest five PageRank ratio and three papers reference them, we are informed with some connections about several important papers. In the end, by outputting the collaborations between productive authors, we obtain information except papers. During this process, we use the build in PageRank function, which is our main sorting algorithm to conclude the importance of the papers based on their references.

2. NetworkX

NetworkX is a Python library especially built for graphs and networks, which is appropriate for our data project. Our project is focusing on the citation and collaboration network of the data set. NetworkX is suitable for such large real-world graphs due to its efficiency and scalability. We would use the classes for graphs and digraphs for our project, more specifically, graphs for the paper reference part and digraphs for the author part. We would also combine NetworkX with Matplotlib library to produce our output image. Basically, we are using NetworkX to manipulate our large lists of nodes as well as the corresponding edges and Matplotlib to make the output images more readable. We would firstly form the networks by Networkx and visualize the networks more beautiful by using various features of matplotlib.

3. Data Clean

We use the data set Citation-network V1, which contains 629814 papers and more than 632752 citation relationships. The dataset is organized into 629814 blocks, with each block allocated for a paper. Every block has each line beginning with a specific prefix that indicates an attribute of the paper. We are using the lines of paper titles, authors, and references. The line of paper titles starts with the prefix #*; the line of authors starts with the prefix #@; the line of references starts with the prefix #%.

3.1. Construct lists for our needed paper attributes

We construct three global two-dimensional lists for paper titles, author, and references, respectively. After reading all lines of the data file, we look into the data line by line. For each line of the data file, we check the prefix to see if the line represents the attributes we want. If the line starts with #*, we add the line as a list to our global paper title list; If the line starts with #@, we add the line as a list to our global author list; If the line starts with #%, we add the line as a list to our

global reference list. Now, we have our three global data lists for paper titles, paper authors, and paper references, separately. The index of each list is the index of the blocks in our data file. The length of the resulted two-dimensional lists is the same as the number of blocks in the data file and each element is a list for the specific attribute for that block.

3.2. The global variable: the total number of the blocks

We also apply a global variable to represent the total number of the blocks (papers) in the data file to make it convenient to access the three global paper attribute list because the index of each global list is exactly the same as the index of the blocks in the data file. In addition, this global variable can also be used for further evaluation of the data, for example, to extract the top 1 percent of the information we sorted.

4. Graphs of paper reference

We would output two graphs along with some related information for this part and we are making this part into a single function to be called in the main function.

4.1. Initial preparation of the graphs for paper references

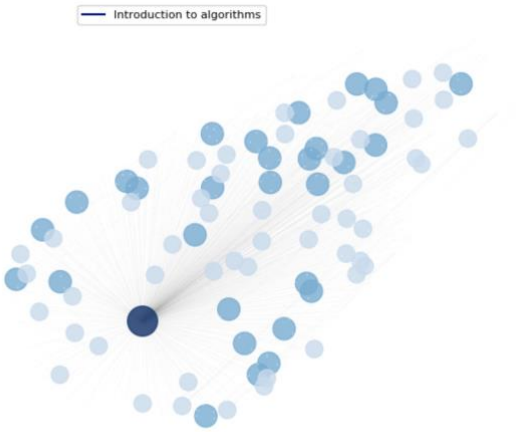
We would firstly further process our lists of references to a list of tuples whose first element is a type of string representing the index of the papers and the second element is a type of string representing the paper that this paper is in reference to. In this way, each paper may have more than one tuple in this list if it references more than one paper. Therefore, this list is actually the edges of our network for the paper references. The nodes or our network for the paper references are all the indexes in the data file. After generating all the nodes and the edges, we can get our digraph, the graph with directions, for the paper reference. In the graph, a paper points to another paper if this paper references that paper.

4.2. Pagerank

PageRank is a function in the NetworkX library for Python which returns the PageRank of the nodes in the graph. We call this function to computes the ranking of the nodes in our digraph. Since PageRank does its computation based on the structure of the incoming links, it ranks the papers in our data set by the times other papers reference it, which is also a measure of its impact. We also create a list for the sorted PageRank values for further use.

4.3. Construct and Visualize the graph for the paper with the highest PageRank value and all the papers that reference it

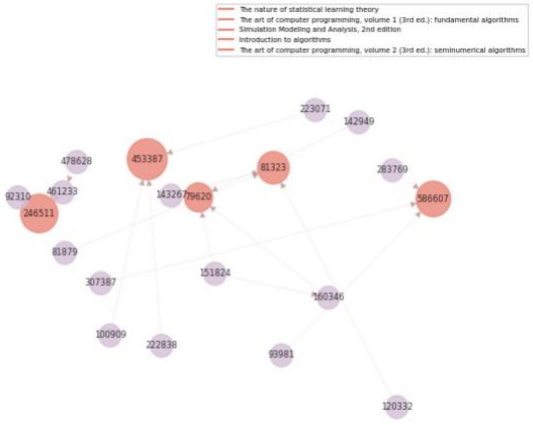
We use the list for the sorted PageRank values to find the paper with the highest PageRank value and get the index of that paper in the data set by looking into the dictionary the PageRank function returns. Then, we use the global reference list to find all the papers that reference the paper with the highest PageRank value and we are able to find all the nodes we would like to plot in the output graph. We also sort the papers that reference the paper with the highest PageRank value by their own PageRank values. We apply the subgraph function in the NetworkX library to get our network. To visualize our graph, we make the size of the node with the highest PageRank value largest, the sizes of the nodes of the top 30 papers ranking by their own PageRank values and referencing the highest PageRank value paper smaller than the largest node, and the sizes of the nodes of the next 50 papers ranking by their own PageRank values and referencing the highest PageRank value paper much smaller, and the sizes of the remaining nodes smallest. We also use the color map to note the importance of papers by the brightness of colors, for example, using dark blue for the relatively larger nodes and light blue for the relatively smaller nodes. We use spring layout for our nodes, make edges narrow and light, create a color legend, and adjust the node transparency as well. Due to the large amount of the data, we would not display the arrows and labels in the graph. We would also print out the titles of the top 30 papers with the highest PageRank values and referencing the highest PageRank values paper. The graph we generated is as follows:



4.4. Construct and Visualize the graph for the paper with the top 5 highest PageRank values and the top 3 highest PageRank value paper for each of the 5 papers

We use the list for the sorted PageRank values to find the top 5 papers with the highest PageRank values and get the indexes of those papers in the data set by looking into the dictionary the PageRank function returns. Then, we use the global reference list combined with the list of the sorted PageRank values to find the top 3 papers with the highest PageRank values for each of those 5 papers, which means we choose 3 papers from every reference list of those 5 papers by their own PageRank values. Now, we are able to find all the nodes we would like to plot in the output graph. The number of nodes should be 5 + 5 * 3 = 20. We apply the

subgraph function in the NetworkX library to get our network. To visualize our graph, we make the size of the node with the top 5 highest PageRank values large, and the sizes of the remaining nodes small. We use color salmon for the 5 nodes, color thistle for the remaining nodes, color rosy brown for the edges. We use random layout for our nodes, make edges narrow and light, create a color legend for the top 5 papers with highest PageRank values, and adjust the node transparency as well. We would display the arrows and labels in this graph. The graph we generated is as follows:



5. Graphs of authors

Graphs of Authors is a graph contains the most productive authors, as they published the highest quantity of papers, either as main authors or as collaborators. As the graph shows, the dark green big nodes denote authors are the top five most productive authors among all 51,000 authors and the light green small nodes denote authors that are among the top 0.5% productive authors that connect to these top five authors. Lines between them illustrate there exists collaborations between two authors and the collaborations can be more than once.

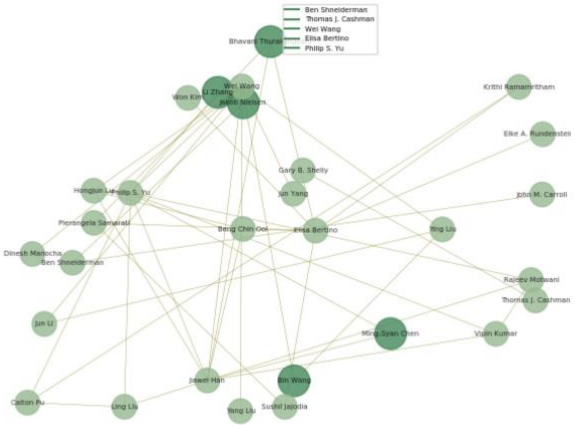
5.1. Sorting data to desired format

Data sorting and visiting for author_sort() function, where we further sort the relations among authors and visualize these relations, is relatively efficient and does not require much work comparing to running time for two graphs of paper reference. From the author_list we achieve in dataclean(), we are able to separate the authors to a list of tuples that each tuple contains two authors that are collaborators of one paper. When separating the string that contains all names of the authors, we choose Regex library in python to detect if the string contains comma, which is a sign that separate two authors. If we do need to separate them, we use string.split function instead of regex. We compare the results using regular expression to separate authors and that with string.split and decide to use the latter one. Here, we save some running time since we only need to consider the cases when there are commas involved in the string and using split instead of obscure regular expression, making the codes clean and straightforward.

5.2. Constructing nodes and edges to include the collaborations among authors

When read the connection between the edge list, I used a for loop to visit elements in the lists of co-authors of a book. Thus, in this list, instead of visiting each nodes step by step and include their relations, we eliminate the running time by half.

In the process of choosing top five productive authors, we choose using Counter methods from Collection library, which can automatically count the reoccurrence of one element in the list and return a sorted special list containing the most common elements at first. It saves us some running time not to visit the large author list. We further choose the most common authors by selecting first 0.5% authors (1/2000 of the total authors). In doing so, we find the authors that are relatively productive and construct correlations between them. Also, when writing author_sort() function, we do not draw a graph and sort from the nodes extracted from subgraph. Instead, we create nodes list and edge list directly and visualize the graph to make it more readable by assigning different node sizes and node colors. We finally draw the graph as follows:



However, we do encounter a problem about the most common authors. The first top four authors we get from the text files are not real name but suffix like “Jr” and “staff”. In order to make sure the reliability of the data, we eliminate

these values out manually by taking the fourth to ninth elements in the list.

6. Conclusion

Networkx is a convenient library to use when handle a large set of data and it does not require comprehensive information of datasets.

6.1. Conclusion about data

From Graph1, we can see that the most important paper, Introduction to algorithms, influences a huge amount of other papers, it also keeps close relationships with other papers that has high page rank ratios. Meanwhile, from the lines connecting with each node, there is an approximate linear relationship between the number of lines and the size of nodes, which is equivalent to the fact that papers with higher page rank ratio has a higher possibility to be referenced. From Graph 2, the referencing relationship of the first few important papers are clear and evident. Some paper has a lot of resources and thus have a higher page rank ratio while for other papers that do not have a lot of resources but still weighted highly since they are referenced by high page rank papers. For Graph 3, we find out that there is a cluster of authors that collaborate with each other a lot due to the density of connections between several nodes. We conclude that they might work in the similar fields or might publish in the same venue. There is also an obscured assumption that authors that more productive might work with other productive authors frequently.

6.2. Improvements

For our projects, we can improve from the following aspects: algorithm and visualization. First, we can reduce the needs of visiting large datasets and think about an easier way to write an algorithm for finding the suitable size data sets. Due to the time limit, the graphs are not visualized perfectly, especially in the first graph, we output over 800 data points and for approximately 700 of them are small white points denotes the relatively unimportant papers that reference Introduction to algorithms. However, most of the points are not tellable and we could find a better way to display these nodes.

6.3. Further Utilization

After sorting our datasets and extract some of the most important information, we can conclude more interesting features based on the existing project we submit. For example, it is easy to find out the relationships of published venues between these most productive authors. It is also easy to eliminate the datasets to smaller ones that classified by the possible fields they are in.

References

[1] Citation Network Dataset. https://aminer.org/citation

[2] NetworkX documentation. https://networkx.github.io/documentation/networkx-1.9/