Summer 2017 - Undergraduate Research Report
Wongelawit Teka Zewde

Social Network: Collaborative Resistance Project

1. Project Description:

This project expands on the research paper Social Resistance written by Michael P. Friedlander, Nathan Krislock, Ting Kei Pon. It is completed as a summer research project by two undergraduate students Wongelawit Teka Zewde and Clarence Su; with the supervision of Professor Michael P. Friedlander. The aim of the project is to apply the social resistance algorithm provided in the research paper in the research community.

The project, using a modified version of the social resistance algorithm, finds the resistance between co-authors in a graph. The project has three main parts. The first step is harvesting and parsing raw data. Then, the data is used to implement the algorithm from the research paper.

At the end, we were able to find the resitance of an author in the graph of data set from Physics and Math authors from 2010-2017 in the Arxiv. We also produced visual representation of our result.

2. Implementation:

Python

dataHarvestor.py

params: subject, begin_year, end_year

output:

dataHarvestor will harvest paper data from the given subject field that are uploaded to Arvix from begin_year to end_year

shortestPath.py(networkx library)

params:subject, graph index, author index 1, author index 2

Output:

shortestPath will find the shortest path(return a list of node that represent the path) between author 1 and author 2 in their graph

dataParser.py(networkx library)

disjointComponent():

Find all the connected components in a paper data set and output them into seperate file

paperToGraph():

Parse a paper data file into a structure/format(indexing the authors and assigning weight based on the paper) that can be used for computation

iterativePaperToGraph():

Iteratively call paperToGraph() on all the file in a given directory

getDataRepo():

Get the list of subject contained in the current project directory

Julia

toMmatrix.jl
params: subject, graph index
Output:
toMmatrix will transform the edge file corresponding to the given graph index into the M matrix and output it as a .jld file for computation usage

distance.jl
params: subject,matrix_index,author_index1,author_index2,num_edge,num_author
output:
Distance will load the corresponding matrix and compute the distance between given authors

query.jl
params: name1, name2, subject
Output:
Query will search for the given names in the corresponding subject data repo and compute the distance between the given authors

3. Problem encountered

Problem 1:
The same procedure didn't work for the set of data harvested for Computer Science authors in the arxiv. The matrix from this set of data is singular matrix and the procedure doesn't work if the matrix is singular. Assuming the process of construction M matrix won't result in singularity, our primary suspicion is the imprecision of float computation. There are some paper in computer science composed by hundreds of authors. Consequently, this results in a edge weight with small float value. These small float value might lose precision and value during computation and singularize the matrix.

We attempt to fix this issue by trying scale all the edge weight with a factor of 1000. Doing so largely extends the time needed for the computation procedures. We can't even complete the procedures(constructing the M matrix and computation distance between two arbitrary authors) overnight.

Problem 2:

We haven't used github for any project with lot of fragmented data files before. We didn't know that github has issue with handling too much fragmented data files. Therefore, in the last month, when we tried to clean up the github repo to wrap up the project, we had issue updating and deleting files in the repo.

Nevertheless, this has a quick fix. Since we have local copy of all the script and data, we can simply delete the github repo and create a new one.

5. Future work

1. To investigate into what cause the singularity in the CS data set, and have it fixed to make the algorithm more robust
2. To migrate the project into a web application so that it is accessible for other people