Kaanan Kharwa
Section 03

Lab 7 Report

Implementation Overview

My PageRank implementation follows the implementation described in class. To be time efficient, my implementation utilizes NumPy arrays to represent an adjacency matrix of the graph. The threshold varies per dataset, usually around .01 for smaller datasets and .0001 for the larger ones. The value .01 was chosen through trial and error, it produced a low difference in page rank for the smaller datasets. The value for .0001 was chosen a similar way. The only data set that deviated from this was the SLASHDOT-ZOO data set which had an epsilon value of .000000001. For the d value, .85 was chosen since this is usually the standard value of d. For SNAP datasets, the option was added to run the code using a sparse matrix. This allows for faster runtimes on larger datasets. A sparse matrix implementation was only run on the SLASHDOT-ZOO dataset.

Algorithm

The algorithm for my implementation of PageRank is described in this paragraph. Initially, a graph would be created from the input file. There are two separate functions for parsing the files, one for the small datasets and one for the SNAP datasets. Once the graph was created, it would be turned into an adjacency matrix. Each row in the adjacency matrix represented a node in the graph, so I then divided each row by the number of outgoing edges of that node. After that, I multiplied each value in the matrix by d. Then, I added a “sink” to the matrix which was populated with the value of 1/n. The adjacency matrix was then transposed in order to have the correct dimensions for a dot product. I then initialized the initial page rank vector (p_k) and populated it with the value 1/n. Additionally, I added the value (1-d) at the end of the initial page rank vector. I then created the new page rank vector (p_{k+1}) by taking the dot product of the adjacency matrix and p_k. This process of calculating the new page rank was repeated until the change in page ranks was below a certain threshold.

Results

Karate Dataset:

The Karate Dataset was run with a threshold value of .01. A sparse vector was not used.

Output:

```
Time to process read in data and build graph:
0.06275701522827148
Time to compute PageRank for each node in graph:
0.0008678436279296875
Number of iterations until convergence: 3
Threshold value: 0.01

PageRanks:
```

```
34   with PageRank: 0.11758453785403047
1    with PageRank: 0.10687711084303511
33   with PageRank: 0.08194610453261164
3    with PageRank: 0.058014270655679816
2    with PageRank: 0.05727559487442982
4    with PageRank: 0.035618632345494616
32   with PageRank: 0.03393704704201558
6    with PageRank: 0.030124988899314066
7    with PageRank: 0.030124988899314066
24   with PageRank: 0.02996989706733388
9    with PageRank: 0.02640877765544152
30   with PageRank: 0.02524176700367647
14   with PageRank: 0.02513597238923781
28   with PageRank: 0.023973062142565362
11   with PageRank: 0.022278960253480733
5    with PageRank: 0.022278960253480733
25   with PageRank: 0.022278696129493465
26   with PageRank: 0.022041350209354577
31   with PageRank: 0.02167379159858388
8    with PageRank: 0.021508141677091846
29   with PageRank: 0.018302989081222767
17   with PageRank: 0.01783407373366013
20   with PageRank: 0.016183587840626703
27   with PageRank: 0.014160124931917211
13   with PageRank: 0.01332480516088814
18   with PageRank: 0.012555757128480732
22   with PageRank: 0.012555757128480732
10   with PageRank: 0.012323232860157952
15   with PageRank: 0.011962192350898692
16   with PageRank: 0.011962192350898692
19   with PageRank: 0.011962192350898692
21   with PageRank: 0.011962192350898692
23   with PageRank: 0.011962192350898692
12   with PageRank: 0.00865605805440666
```

Observations:
Since the karate dataset represents friendships between members at a karate club at a US university, the page rank could be interpreted as how popular a member is within the club. I believe that PageRank did indeed find the correct ranking as members 1 and 34 have the most connections, so them having the highest page rank makes sense.

Dolphin Dataset
The dolphin dataset was run with an epsilon value of .01, a sparse vector was not used.

Output (Top 20):

```
Time to process read in data and build graph:
0.06551718711853027
Time to compute PageRank for each node in graph:
2.193450927734375e-05
Number of iterations until convergence: 1
Threshold value: 0.01

PageRanks:
"Trigger"      with PageRank: 0.05252001582646742
"Jet"          with PageRank: 0.0463447260624679
"Web"          with PageRank: 0.038592229902713764
"Patchback"    with PageRank: 0.0334026323139226
"Scabs"        with PageRank: 0.033174137690266725
"SN63"         with PageRank: 0.031264080901177674
"Grin"         with PageRank: 0.029975212959083924
"Topless"      with PageRank: 0.026602197086068053
"SN4"          with PageRank: 0.025976557045105434
"Beescratch"   with PageRank: 0.0240828213005632
"Stripes"      with PageRank: 0.0231258145975887
"Kringel"      with PageRank: 0.023032834101382484
"SN100"        with PageRank: 0.022783072662104917
"Gallatin"     with PageRank: 0.022407194060419865
"Haecksel"     with PageRank: 0.021504590839268256
"Feather"      with PageRank: 0.020203853046594982
"Ripplefluke" with PageRank: 0.0198012672811059
"Upbang"       with PageRank: 0.018566308243727597
"SN9"          with PageRank: 0.018550481776288225
"DN21"         with PageRank: 0.017951548899129543
```

Observations:
Like the karate dataset, this dataset is about social connections between dolphins, so a page rank could be interpreted as how popular a dolphin is amongst other dolphins. I think that PageRank did a pretty good job finding the correct ranking. Although the ranking is not completely accurate, it is pretty close. Dolphins with a lot of social connections have higher page ranks than dolphins with fewer social connections.

Les Misérables Dataset:

The Les Misérables Dataset was run with an epsilon value of .01, and no sparse vectors were used.

Output (Top 20):

```
Time to process read in data and build graph:
0.06881904602050781
Time to compute PageRank for each node in graph:
0.00042819976806640625
Number of iterations until convergence: 2
Threshold value: 0.01
```

```
PageRanks:
"Valjean"           with PageRank: 0.06660447672629546
"Gavroche"          with PageRank: 0.03666182863226877
"Marius"            with PageRank: 0.0309949137054942
"Javert"            with PageRank: 0.028199609725627316
"Myriel"            with PageRank: 0.026930338943071418
"Fantine"           with PageRank: 0.026300306517109736
"Thenardier"        with PageRank: 0.025888403463039146
"Cosette"           with PageRank: 0.021084042034664886
"Enjolras"          with PageRank: 0.020065892287065245
"MmeThenardier"     with PageRank: 0.018626117876246365
"Bossuet"           with PageRank: 0.017525699587298074
"MlleGillenormand"  with PageRank: 0.017416758334074987
"Babet"             with PageRank: 0.016074775657691043
"Gueulemer"         with PageRank: 0.016074775657691043
"Courfeyrac"        with PageRank: 0.01587159890509527
"Bamatabois"        with PageRank: 0.01585864859337782
"Gillenormand"      with PageRank: 0.01569841288367212
"Claquesous"        with PageRank: 0.01550875644617049
"Eponine"           with PageRank: 0.015176217588354933
```

Observations:
In the Les Misérables dataset, the edges represent instances where characters appear in the same chapter in the novel. Therefore, the page rank can be interpreted as how often a character appears in the novel. I think that PageRank did a good job determining the correct ranking since main characters like Valjean, Marius, Javert, and Fantine have high page ranks.

NCAA-Football Dataset:

The NCAA-Football dataset was run with an epsilon value of .0001 since it was a larger dataset. No sparse matrix was used.

Output (Top 20):
```
Time to process read in data and build graph: 0.1875622272491455
Time to compute PageRank for each node in graph:
0.00115203857421875
Number of iterations until convergence: 3
Threshold value: 0.0001

PageRanks:
 "Campbell"                  with PageRank: 0.0010274122807017545
 "Arkansas-Pine Bluff"       with PageRank: 0.0010059314954051798
 "Savannah State"            with PageRank: 0.00092577798663325
 "Valparaiso"                with PageRank: 0.0008969298245614036
 "Idaho State"               with PageRank: 0.0008833489974937345
 "Indiana State"             with PageRank: 0.0008120718462823728
 "Western Kentucky"         with PageRank: 0.0008023966165413535
 "Iona"                     with PageRank: 0.0007915674603174604
 "Howard"                   with PageRank: 0.0007633406432748539
 "St. Francis (PA)"         with PageRank: 0.0007548193400167086
 "Wagner"                   with PageRank: 0.0007548193400167085
 "Washington"               with PageRank: 0.0007120352965747704
 "Miami (OH)"               with PageRank: 0.0007050310435178858
 "Alcorn State"             with PageRank: 0.0007020937761069341
 "North Carolina Central"   with PageRank: 0.0006975749031670085
 "Dartmouth"                with PageRank: 0.0006953558137768665
 "Idaho"                    with PageRank: 0.0006874477861319968
 "Mississippi Valley State" with PageRank: 0.0006853174603174604
 "Southern Methodist"       with PageRank: 0.0006841877420824791
 "New Mexico State"         with PageRank: 0.0006834534252297412
```

Observations:
The outgoing edges in the NCAA-Football dataset indicate that the team with the outgoing edge beat the team the other team which the edge is going to. Therefore, page rank can be interpreted as value indicating how often a team wins. I do not think that PageRank did a good job determining the correct ranking since of the top 5 teams, only Indiana State appears with a high page rank.

Wiki Vote
The Wiki Vote dataset was run with an epsilon value of .0001, no sparse matrix was used.

Output (Top 20):
```
Time to process read in data and build graph: 7.074322938919067
Time to compute PageRank for each node in graph:
0.03813910484313965
Number of iterations until convergence: 3
Threshold value: 0.0001

PageRanks:
4037 with PageRank: 0.0026608702971472518
6634 with PageRank: 0.0024829609868888426
15   with PageRank: 0.002271965029934039
2625 with PageRank: 0.0021555522018836294
2398 with PageRank: 0.0017533171297509527
7553 with PageRank: 0.001481827808518657
4191 with PageRank: 0.0014627505479184136
2237 with PageRank: 0.0014071416357701657
5412 with PageRank: 0.0014036098171801312
5254 with PageRank: 0.001362114311428735
7632 with PageRank: 0.0013618673417093048
1297 with PageRank: 0.0013613047734883914
4335 with PageRank: 0.001337869030539303
```

```
2328 with PageRank: 0.001310761659975526
2470 with PageRank: 0.0012840182807221838
737  with PageRank: 0.0012618059406678647
7620 with PageRank: 0.0012588488393050342
2066 with PageRank: 0.0012550840106179348
6832 with PageRank: 0.0012183067533643023
```

Observations:
The edges in the Wiki Vote dataset represents a vote of one user for another, so the page rank could be interpreted as how likely someone would vote for that user. I think that PageRank did a good job determining the correct ranking as users with the most votes such as 4037, 15, 2398, and 1297 all have high page ranks.

Gnutella Dataset

The Gnutella dataset was run with an epsilon value of .0001, no sparse matrix was used.

Output (Top20):

```
Time to process read in data and build graph: 8.330074071884155
Time to compute PageRank for each node in graph:
5.888938903808594e-05
Number of iterations until convergence: 1
Threshold value: 0.0001

PageRanks:
1676 with PageRank: 0.0008403829728439708
876  with PageRank: 0.0008391921308776377
1020 with PageRank: 0.0008241012887180634
226  with PageRank: 0.0008155600773733256
386  with PageRank: 0.0007765908006129568
842  with PageRank: 0.0007745622629185812
222  with PageRank: 0.0007216067525812036
227  with PageRank: 0.0007205391011631114
389  with PageRank: 0.0007194714497450191
271  with PageRank: 0.0007088919947839232
223  with PageRank: 0.000697317682819604
391  with PageRank: 0.0006852244773339049
221  with PageRank: 0.0006789006958575121
230  with PageRank: 0.0006732065549610199
278  with PageRank: 0.0006642204888587431
688  with PageRank: 0.0006618182731680354
225  with PageRank: 0.0006543447132413894
229  with PageRank: 0.0006543447132413894
47   with PageRank: 0.000649273369005451
```

Observations:

The edges in the Gnutella dataset represent one server requesting information from another server. Therefore, the page rank can be interpreted as how popular a server is (how often other servers request from it). I think that PageRank did a good job determining the correct order, since the top 5 servers that information is requested from, appear in the top 6 page ranks.

Slashdot Zoo Dataset:

The Slashdot Zoo dataset was run with an epsilon value of .000000001 and utilized a sparse matrix.

Output (Top 20):

```
Time to process read in data and build graph: 0.6271507740020752
Time to compute PageRank for each node in graph:
0.011313199996948242
Number of iterations until convergence: 9
Threshold value: 1e-09

PageRanks:
371  with PageRank: 1.2927078351021885e-05
459  with PageRank: 1.2927078351021885e-05
480  with PageRank: 1.2927078351021885e-05
583  with PageRank: 1.2927078351021885e-05
940  with PageRank: 1.2927078351021885e-05
1256 with PageRank: 1.2927078351021885e-05
1554 with PageRank: 1.2927078351021885e-05
1669 with PageRank: 1.2927078351021885e-05
1733 with PageRank: 1.2927078351021885e-05
1776 with PageRank: 1.2927078351021885e-05
1838 with PageRank: 1.2927078351021885e-05
2210 with PageRank: 1.2927078351021885e-05
2562 with PageRank: 1.2927078351021885e-05
2962 with PageRank: 1.2927078351021885e-05
3110 with PageRank: 1.2927078351021885e-05
3205 with PageRank: 1.2927078351021885e-05
3220 with PageRank: 1.2927078351021885e-05
3283 with PageRank: 1.2927078351021885e-05
3380 with PageRank: 1.2927078351021885e-05
3550 with PageRank: 1.2927078351021885e-05
```

Observations:

An interesting observation is that all the top page ranks have the same value. Since PageRank was able to converge at a very small value (.000000001), I am fairly confident that it determined the correct order.

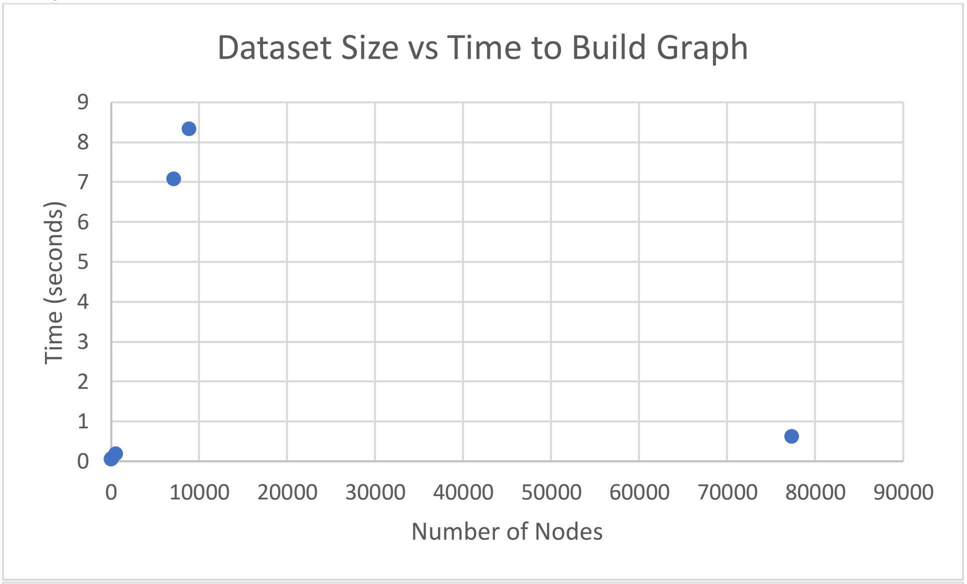
Overall Summary

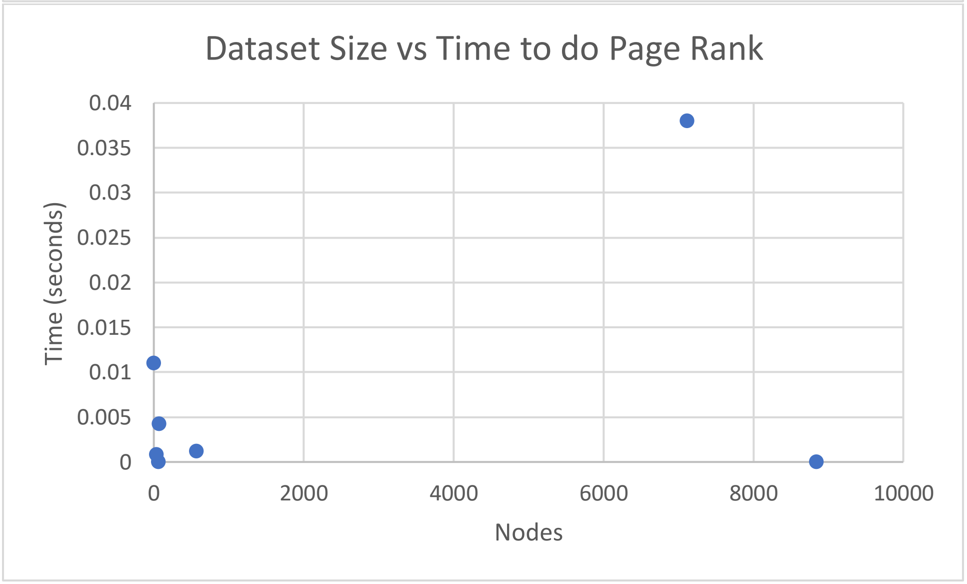
I am confident that PageRank produced the correct ranking for all datasets except the NCAA-Football dataset. This may be because page rank tends to give a higher score to more connected edges (outgoing and incoming), but in the case of the NCAA-Football dataset, we only want to pay attention to outgoing edges. For this reason, I think that PageRank performs better on undirected graphs since both outgoing and incoming edges matter equally.

Performance Evaluation

For most datasets, most of the time spent is in reading the input file and creating the adjacency matrix. The time to compute the page rank is significantly less. This may also be since for most datasets, page rank converged very quickly. The exception to this is the Slashdot Zoo dataset because it was implemented using the sparse matrix.

Graphs:





For the graph showing time taken to build the graph, there seems to be a clear correlation between the number of nodes and the time taken. As the number of nodes increases, the time also increases. The exception of course is the Slashdot Zoo dataset which created the adjacency matrix using a sparse matrix for faster times. For completing PageRank it seems that size of the data set does not play a big role in time taken to complete. Rather, the number of iterations plays a much more significant role.

README

Files:

pageRank.py

- Usage: python3 pageRank.py <file_path> <epsilon_value> <snap_vector>
    - file_path
        - The path to the file with the dataset
    - epsilon_value
        - The value of epsilon to use as a threshold
    - snap_vector
        - Optional parameter, required when data is SNAP file
        - Use either value 0 or 1. 0 is no sparse vector and 1 is sparse vector
            - Use sparse vector on SlashDot Zoo dataset

karate_out.txt

- The output file for the karate.csv dataset

dolphins.out_txt

- The output file for the dolphins.csv dataset

NCAA_football_out.txt

- The output file for the NCAA_football.csv dataset

lesmis_out.txt

- The output file for the lesmis.csv dataset

wiki-Vote_out.txt

- The output file for the wiki-Vote.csv dataset

p2p-Gnutella05_out.txt

- The output file for the p2p-Gnutella05.csv dataset

soc-sign-Slashdot081106_out.txt

- The output file for the soc-sign-Slashdot081106.csv dataset