Assignment 2: Link Analysis & Prediction

Matthew MacEachern
York University
4700 Keele St, Toronto, ON M3J 1P3
Canada
mcmaceac@my.yorku.ca

KEYWORDS

Co-authorship, PageRank, betweenness, link prediction, random predictor, common neighbors, Jaccard coefficient, preferential attachment, Adamic Adar.

1 INTRODUCTION

In this assignment, we were asked to generate several graphs based on the DBLP bibliography data provided for 2005 and 2006 journal articles, and run various algorithms pertaining to link and node importance, as well as link prediction. The bibliography data came in the form of a large JSON file. To accomplish these analyses, the NetworkX library for Python was used. NetworkX already has a rich suite of modules to accomplish these different algorithms that will be discussed throughout this paper, and these modules were utilized as to save time on the analysis process.

2 TEMPORAL GRAPHS

2.1 Graph Generation

In order to generate the graphs needed for the analyses, the JSON Python library was used to parse the data. The JSON file was in the format of tuples (u, v, y) where u and v were individuals who co-authored papers together in a year y. For the purpose of this assignment, only the years 2005 and 2006 were of interest, so when parsing the tuples in the file, if 2005 or 2006 were the y value, the tuple data was added to the appropriate graph.

The graphs generated for this assignment were DBLP2005, DBLP2006, and DBLP2005W. DBLP2005 was a graph that contained edges formed from the tuples with y value 2005. DBLP2006 was a graph that contained edges formed from the tuples with y value 2006. DBLP2005W was DBLP2005 but with edge weights for all authors u and v representing how many times u and v co-authored a paper according to the data. All three graphs were generated using the add_edge(u, v) function of NetworkX. Before adding an edge (u, v) from 2005 to DBLP2005W, however, it was checked if the graph already contained that edge. If it did, the edge weight was incremented by one. If it didn't, the edge was added to the existing graph with edge weight one.

2.2 Graph Attributes

For all three graphs, the number of nodes and edges was very large. For DBLP2005, the number of nodes was 180,227 and the number of edges was 403,026. For DBLP2006, the number of nodes was 201,298 and the number of edges was 465,988. Of course, for the weighted version of the DBLP2005 graph the number of nodes and edges were the same, but with weights added to the edges.

The assignment specified that we were only to work with the giant connected component of the graph, so for completeness, the number of nodes and edges will be reported for those graphs as well. For GCC_DBLP2005, the number of nodes was 106,943 and the number of edges was 300,043. For GCC_DBLP2006, the number of nodes was 123,808 and the number of edges was 356,968. As mentioned previously, the weighted version of the 2005 graph had the same attributes as its unweighted version.

Finally, for the later calculations pertaining to link prediction, the assignment specified to make versions of the 2005 and 2006 graphs that only contained nodes with degree greater than or equal to three. DBLP2005-CORE contained 77,153 nodes and 255,815 edges. DBLP2006-CORE contained 90,173 nodes and 306,757 edges.

3 NODE AND EDGE IMPORTANCE

3.1 PageRank

For all three graphs the PageRank was calculated for all nodes in the graph, and the 50 nodes with the highest PageRank scores were returned. In order to accomplish this, the pagerank(G) function of the NetworkX library was used which was called using the giant connected component version of all three graphs. This function returned a dictionary of nodes with the PageRank of the nodes as the values. In NetworkX, the default alpha value is 0.85, and this was used for all three graphs.

The following are the top 50 nodes and their corresponding PageRank scores for DBLP2005:

1) ('Wen Gao', 0.0001419310322458534 8)
2) ('Xin Li', 0.00013306687252334022)
3) ('Wei Zhang', 0.0001183709570000217 7)
4) ('Chin-Chen Chang', 0.0001122818709619070 5)
5) ('Wei Wang', 0.0001049590537328314 1)
6) ('Guanrong Chen', 0.00010241545494191459)
7) ('Zhaohui Wu', 9.972482112399872e-05)
8) ('Wei-Ying Ma', 9.740727215568839e-05)
9) ('Bin Wang', 9.726301353073486e-05)
10) ('Samuel Pierre', 9.705863952617081e-05)
11) ('Hans-Peter Seidel', 9.630692366059368e-05)
12) ('Francky Catthoor', 9.509687755131548e-05)
13) ('Yong Zhang', 9.502779502456232e-05)
14) ('Fang Liu', 9.291312733184171e-05)
15) ('Witold Pedrycz', 9.15965422498646e-05)

16) ('Wei Liu', 9.088044455802495e-05)
17) ('Alberto L. Sangiovanni-Vincentelli', 9.062998434173112e-05)
18) ('Licheng Jiao', 9.05403900028196e-05)
19) ('Yan Zhang', 8.964011202963374e-05)
20) ('Xiang Li', 8.924949886585828e-05)
21) ('Mario Piattini', 8.770301687049541e-05)
22) ('Yvon Savaria', 8.759316637739886e-05)
23) ('Yan Li', 8.757045759012321e-05)
24) ('David J. Evans', 8.646019165514893e-05)
25) ('Wei Zhao', 8.631862736191034e-05)
26) ('Lei Li', 8.521097914198232e-05)
27) ('Minglu Li', 8.385477457657977e-05)
28) ('Jing Li', 8.364797920281224e-05)
29) ('Qing Li', 8.346925817780213e-05)
30) ('Xianlong Hong', 8.280253482108303e-05)
31) ('Jun Li', 8.255650675543822e-05)
32) ('Mario Gerla', 8.151790992165259e-05)
33) ('Yu Chen', 8.098964142814074e-05)
34) ('Yang Xu', 8.09434257470208e-05)
35) ('Tharam S. Dillon', 8.027406115457522e-05)
36) ('Jiannong Cao', 7.996882944317615e-05)
37) ('Elisa Bertino', 7.975921070453301e-05)
38) ('Yan Wang', 7.953052894162082e-05)
39) ('Philip S. Yu', 7.904077202276819e-05)
40) ('Azzedine Boukerche', 7.863772796602966e-05)
41) ('Hong Zhang', 7.85409528646126e-05)
42) ('Ajith Abraham', 7.845321957796965e-05)
43) ('Mani B. Srivastava', 7.835090136234614e-05)
44) ('Ying Liu', 7.826418075094077e-05)
45) ('Luc J. Van Gool', 7.797561722585576e-05)
46) ('Ying Chen', 7.787685123048909e-05)
47) ('Tao Zhang', 7.721288842944204e-05)
48) ('Donald F. Towsley', 7.712843687402164e-05)
49) ('Jun Liu', 7.671917801147298e-05)
50) ('C. C. Jay Kuo', 7.625128869758412e-05)

The following are the top 50 nodes and their corresponding PageRank scores for DBLP2006:

1) ('Wen Gao', 0.0001240131307163933)
2) ('Hui Li', 0.00012330323078582666)
3) ('Chin-Chen Chang', 0.00012155789113614979)
4) ('Xin Li', 0.00011816042342745642)
5) ('Yong Zhang', 0.0001143310055167495)
6) ('Wei Wang', 0.00011418939306766574)
7) ('Qing Li', 0.00011370789902479061)
8) ('Yan Zhang', 0.00010717607486426265)
9) ('Wei Liu', 0.0001071526598924388 6)
10) ('Wei Zhang', 0.00010684051247426161)
11) ('Li Zhang', 9.897328971913329e-05)
12) ('Yu Zhang', 9.855469285395953e-05)
13) ('Ying Li', 9.544758462291252e-05)
14) ('Jing Li', 9.488459658482036e-05)
15) ('Xiaodong Wang', 9.468680055242121e-05)
16) ('Yan Li', 9.368813789090802e-05)
17) ('Wei Li', 9.254352922287345e-05)
18) ('Ying Liu', 8.948725663577388e-05)
19) ('Yong Yu', 8.929031168044772e-05)
20) ('Francky Catthoor', 8.892675086632877e-05)
21) ('Jian Li', 8.725189260558142e-05)
22) ('Wei-Ying Ma', 8.717464224186694e-05)
23) ('Hai Jin', 8.666108402959187e-05)
24) ('Jian Wang', 8.523852018029668e-05)
25) ('Laurence Tianruo Yang', 8.413681197740772e-05)
26) ('Samuel Pierre', 8.38788358404156e-05)
27) ('Lei Li', 8.327246374174173e-05)
28) ('Hao Wang', 8.259999955237506e-05)
29) ('Yang Yang', 8.24824755443084e-05)
30) ('Xin Chen', 8.149464984371827e-05)
31) ('Fang Liu', 8.141584215923697e-05)
32) ('Lei Zhang', 8.126625653572789e-05)
33) ('Masayuki Murata', 8.110350909961691e-05)
34) ('Tao Zhang', 8.043902864329782e-05)
35) ('Gang Wang', 8.018410512901535e-05)
36) ('Hong Zhang', 7.973926909893575e-05)
37) ('Yong Wang', 7.967609034166206e-05)
38) ('Bin Li', 7.808916142631806e-05)
39) ('Wei Zhao', 7.698812599228651e-05)
40) ('Jing Wang', 7.671448876053602e-05)
41) ('Gang Li', 7.656909818744382e-05)
42) ('Tao Li', 7.635290669780309e-05)
43) ('Mario Gerla', 7.552434294885485e-05)
44) ('Liang Zhang', 7.516285920433626e-05)
45) ('Jun Zhang', 7.469378332758827e-05)
46) ('Wei Xu', 7.454573262707774e-05)
47) ('C. C. Jay Kuo', 7.35703627521852e-05)
48) ('Luc J. Van Gool', 7.233009163938588e-05)
49) ('Elisa Bertino', 7.174076615227388e-05)
50) ('Yan Chen', 7.146691397294093e-05)

Finally, the following are the top 50 nodes and their corresponding PageRank scores for DBLP2005W:

1) ('Wen Gao', 0.0001754250096223227)
2) ('Chin-Chen Chang', 0.0001516923685712731)
3) ('Wei-Ying Ma', 0.0001349627094632 8825)
4) ('Xin Li', 0.0001276743456190983)
5) ('Licheng Jiao', 0.00012409820637811528)
6) ('Francky Catthoor', 0.00012203863886629682)
7) ('H. Vincent Poor', 0.0001186404503799 3307)
8) ('Zhaohui Wu', 0.0001177337645127 6846)
9) ('Hans-Peter Seidel', 0.00011673838658232987)
10) ('Xianlong Hong', 0.0001137287534809 1101)
11) ('Mario Piattini', 0.0001088283395730 0276)
12) ('Wei Zhang', 0.0001078646399535 0345)
13) ('Tharam S. Dillon', 0.00010781923811267758)
14) ('Minglu Li', 0.0001076818858932 3182)
15) ('Samuel Pierre', 0.0001039655314442 3084)
16) ('Witold Pedrycz', 0.0001015944513526654)
17) ('Guanrong Chen', 0.0001010745772316 9957)
18) ('Wei Liu', 0.0001005183033975 7155)
19) ('Hai Jin', 9.96192529502357e-05)
20) ('David J. Evans', 9.82236321100603e-05)
21) ('Mario Gerla', 9.726039688501413e-05)
22) ('Bin Wang', 9.711535643782988e-05)
23) ('Wei Wang', 9.709528216965252e-05)
24) ('Alberto L. Sangiovanni-Vincentelli', 9.578750689175142e-05)
25) ('Jinxiang Dong', 9.51190975428636e-05)
26) ('Kaushik Roy', 9.345487238418834e-05)
27) ('Heung-Yeung Shum', 9.270821429321068e-05)
28) ('Hong Shen', 9.231102805061213e-05)
29) ('Donald F. Towsley', 9.200393513542262e-05)
30) ('Yong Zhang', 9.053833770711202e-05)
31) ('Norman C. Beaulieu', 9.002159341561067e-05)
32) ('Mahmut T. Kandemir', 8.996536817072493e-05)
33) ('Philip S. Yu', 8.992507300252977e-05)
34) ('Azzedine Boukerche', 8.933477999468098e-05)
35) ('Fang Liu', 8.925815517962778e-05)
36) ('Mani B. Srivastava', 8.910878779941834e-05)
37) ('Ajith Abraham', 8.86335422475728e-05)
38) ('Yvon Savaria', 8.840407055017915e-05)
39) ('Tak-Wai Chan', 8.838628114385348e-05)
40) ('Elisa Bertino', 8.728486393662994e-05)
41) ('Yan Li', 8.727019103745125e-05)
42) ('Jing Li', 8.700450401104184e-05)
43) ('Yan Zhang', 8.691057882308395e-05)
44) ('Xiang Li', 8.645051719824402e-05)
45) ('Jiannong Cao', 8.59880649092033e-05)
46) ('Lei Li', 8.555454806100371e-05)
47) ('Tao Zhang', 8.554798500111325e-05)

48) ('Chao-Tung Yang', 8.520155293730393e-05)
49) ('Hsinchun Chen', 8.47942994614094e-05)
50) ('Katsumi Tanaka', 8.444588662514061e-05)

Looking at the results, the PageRank function seems to indicate that Wen Gao is the most important author in all three versions of the graph. After some research, it turns out that Dr. Gao has had his papers cited thousands of times according to Google Scholar, and received four national awards of science and technology achievement, including one in 2005. This could indicate that after receiving a prestigious award, Dr. Gao received more attention in the academic community and co-authored a larger number of papers in 2005 and 2006 as a result.

It is also interesting to note that the top 50 authors in all three versions of the graphs seem to be exclusively of Asian decent based on their names. This could indicate that there might be a larger amount of activity in terms of academic papers for Computer Science in this part of the world than there is in the rest of the world. This could also be since the Asian community may co-author papers only with other authors in the Asian community which would lead to more importance being in this community, like a spider-trap in the traditional example used for PageRank. This, however, is only speculative and could be due to other factors.

3.2 Edge Betweenness Centrality

To calculate the top 20 edge betweenness centrality scores for all three graphs, NetworkX's edge_betweenness_centrality(G) function was used. The edge betweenness centrality is defined to be the number of shortest paths that pass through an edge. This of course involves calculating all pairs shortest paths. In the case of the dataset used for this assignment, the number of possible pairs of nodes is very large for the graphs generated and calculating the edge betweenness centrality using these graphs would take a very long time. After leaving the script running for nearly 18 hours (for a single graph!), I decided to make use of a parameter in the edge_betweenness_centrality(G) function: k. When this k parameter is set, the function only considers a subset of k nodes to use the edge betweenness centrality calculation. The k parameter was set to 1000 and the calculation time was reduced to 30 minutes for each graph.

The following are the results for DBLP2005:

1) ('Xin Li', 'Farshad Fotouhi')
2) ('Xin Li', 'Philip S. Yu')
3) ('Philip S. Yu', 'Ming-Syan Chen')
4) ('Yi-Bing Lin', 'Yuguang Fang')
5) ('Xin Li', 'Leen-Kiat Soh')
6) ('Hua Wang', 'Francky Catthoor')
7) ('Xin Li', 'Jian Wang')
8) ('Yang Xu', 'Xin Li')
9) ('Xin Li', 'Joseph S. Valacich')
10) ('Dan Roth', 'Xin Li')
11) ('Ying Liu', 'Wei-Ying Ma')
12) ('C. C. Jay Kuo', 'Qing Li')
13) ('Peng Li', 'Xin Li')
14) ('Xin Li', 'Christophe Diot')
15) ('Jian Wang', 'Hideki Imai')
16) ('Philip S. Yu', 'Bamshad Mobasher')
17) ('Ying Liu', 'Jorge Civera')
18) ('Ramesh Govindan', 'Xin Li')
19) ('Jan-Ming Ho', 'Ming-Syan Chen')
20) ('Xin Li', 'Hsinchun Chen')

Looking at this list, like the PageRank scores discussed in section 3.1, the resulting edges seem to be primarily from co-authors of Asian descent. This could indicate that overall the Asian community has a bigger presence overall in academic papers from this bibliography. Author Xin Li in particular seems to have an important role in the topology of the graph when looking at edge betweenness scores.

The following are the results for DBLP2005W:

1) ('Xin Li', 'Farshad Fotouhi')
2) ('Hua Wang', 'Francky Catthoor')
3) ('Xin Li', 'Philip S. Yu')
4) ('Yang Xu', 'Xin Li')
5) ('Philip S. Yu', 'Ming-Syan Chen')
6) ('Yi-Bing Lin', 'Yuguang Fang')
7) ('Xin Li', 'Leen-Kiat Soh')
8) ('Jian Wang', 'Hideki Imai')
9) ('Xin Li', 'Jian Wang')
10) ('Dan Roth', 'Xin Li')
11) ('Ramesh Govindan', 'Xin Li')
12) ('Xin Li', 'Joseph S. Valacich')
13) ('Peng Li', 'Xin Li')
14) ('Tao Zhang', 'Yong Zhang')
15) ('Yuan Chen', 'Qing Wang')
16) ('Andrzej Cichocki', 'Fabian J. Theis')
17) ('Xin Li', 'Christophe Diot')
18) ('Jun Liu', 'Mario Gerla')
19) ('Hua Wang', 'Yu Chen')
20) ('Chuan Yi Tang', 'Hsueh-I Lu')

The weights seem to have changed the edge betweenness rankings of the 2005 graph slightly, but many of the edges in the unweighted version seem to still be present. In particular, the edges that author Xin Li seem to persist through the weighted version.

The following are the results for DBLP2006:

1) ('Alberto L. Sangiovanni-Vincentelli', 'Xi Chen')
2) ('Kai Yang', 'Pawan K. Bhartia')
3) ('Xin Li', 'Philip S. Yu')
4) ('Elisa Bertino', 'Yu Zhang')
5) ('Wen Gao', 'Josef Kittler')
6) ('Jar-Ferr Yang', 'Ming-Ting Sun')
7) ('Lin Chen', 'Gudrun Wagenknecht')
8) ('Burkhard Morgenstern', 'Ming Zhang')
9) ('Bin Li', 'Yao Li')
10) ('Hongji Yang', 'William C. Chu')
11) ('Xiaodong Wang', 'Kai Yang')
12) ('Z. Zhang', 'Gang George Yin')
13) ('Hua Wang', 'Seiichi Shin')
14) ('Jian Xu', 'Jongkyung Kim')
15) ('Xin Li', 'Scott Shenker')
16) ('Rudolf Fleischer', 'Sándor P. Fekete')
17) ('Wei Zhang', 'Qing Li')
18) ('Xin Li', 'Bin Li')
19) ('Wang-Chien Lee', 'Wen-Chih Peng')
20) ('Liang-Gee Chen', 'Chih-Wei Hsu')

The 2006 version of the graph shows similar results to the 2005 version, with Xin Li still having a major presence in the list of

edges. However, there are many new authors that appear in this list that were not in the 2005 or weighted 2005 versions.

4 LINK PREDICTION

4.1 Graph Preparations

For the link prediction problem, the graphs GCC_DBLP2005 and GCC_DBLP2006 were to be used. However, these graphs needed to be modified to only include the nodes that had degree greater than or equal to three. This was completed by simply iterating through the graphs and storing in a list the nodes that had degree greater than or equal to three (using NetworkX's G.degree() function), and then returning the subgraph constructed only using those nodes (using NetworkX's G.subgraph(nodes) function). These graphs were called DBLP2005-CORE and DBLP2006-CORE.

From the DBLP2005-CORE graph a friends-of-friends list needed to be constructed which would contain all the pairs of nodes that were exactly two hops away from each other. This required more than simply having a triple nested loop for each node in the graph. For example, in Figure 1 node 2 is exactly two hops away from node 3, but node 2 is not exactly two hops away from node 0 even though there is a path of length two from node 0 to node 2. In order to get around this, the findFoF(G) function was created. This function iterated over every node u in the DBLP2005-CORE graph. For every node u it would find all the neighbors v of u (using NetworkX's G.neighbors(u) function). Finally, all the neighbors of v which were named w were found. In order to avoid including nodes that were not exactly two hops away as explained previously with the help of Figure 1, there is a check that w != u, w != v and w is not in the set of neighbor nodes of u. If the w node passes all these checks, the pair (u, w) is added to the friends-of-friends list. This friends-of-friends list was the list of candidate edges for the prediction problem. The total size of this list was 987,973 edges.

Next the set of target edges T that would be ideally predicted by the algorithms needed to be constructed. This was the set of edges that were in the DBLP2006-CORE graph but were not in the DBLP2005-CORE graph. This was accomplished by simply iterating through the DBLP2005-CORE edges and checking if they were present in the DBLP2006-CORE graph. If they weren't, they were added to the list. The size of the T list was 252,968 edges.

For all the following link prediction methods (excluding the random predictor), the score for each candidate (friends-of-friends) edge was calculated by iterating through the list of candidate edges and calculating the scores for those edges using the DBLP2005-CORE graph. The results of each of these algorithms can be found in both the 'Link Prediction Excel Sheets' file and the 'Link Prediction Graphs' file.

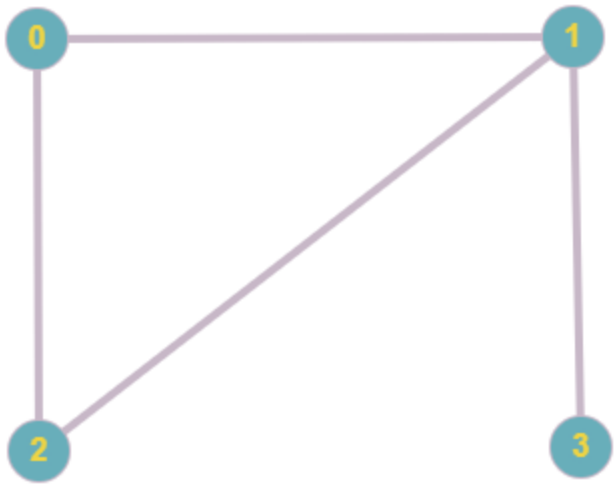

Figure 1. Example graph to illustrate the friends-of-friends method.

4.2 Results

For each of the prediction methods, the precision was calculated at k=10, 20, 50, 100, and 252,968 (|T|). The top k edges were chosen based on their score for one of the particular link prediction methods and then a check occurred to see how many of those k edges were in the target set of edges T. The amount of edges that matched were the true positives. This amount was then divided by k (true positives and false positives) to give the precision at k.

4.2.1 Random Predictor

The random predictor is a simple function in the Python script where edges are chosen at random from the candidate list of edges. The edges selected were then checked against the target edge list T to see if it was present in the edges that would ideally be predicted. Unfortunately for the runs of the program the precision was very low:

- P@10 = 0.0
- P@20 = 0.0
- P@50 = 0.0
- P@100 = 0.0
- P@252968 = 0.0

4.2.2 Common Neighbors

The common neighbors was calculated using NetworkX's common_neighbors(G, e) function where G was the DBLP2005-CORE graph and e was each candidate edge. The results were of course better than the random predictor, but still lower than expected:

- P@10 = 0.0
- P@20 = 0.0
- P@50 = 0.02
- P@100 = 0.04
- P@252968 = 0.01699029126213592

4.2.3 Jaccard Coefficient

The Jaccard coefficient was calculated using NetworkX's jaccard_coefficient(G, e) function where G was the DBLP2005-CORE graph and e was the list of candidate edges to consider. The Jaccard coefficient's precision results were nominally better than the common neighbor results. However, for k=|T| the Jaccard coefficient performed worse than common neighbors:

- P@10 = 0.2
- P@20 = 0.15
- P@50 = 0.08
- P@100 = 0.04
- P@252968 = 0.013701337718604725

4.2.4 Preferential Attachment

The preferential attachment score was calculated using NetworkX's preferential_attachment(G, e) function where G was the DBLP2005-CORE graph and e was the list of candidate edges to consider. The preferential attachment link prediction method performed worse across all values of k when compared to the Jaccard coefficient:

- P@10 = 0.0
- P@20 = 0.05
- P@50 = 0.04
- P@100 = 0.04
- P@252968 = 0.009416210746023213

4.2.4 Adamic Adar

The Adamic Adar score was calculated using NetworkX's adamic_adar_index(G, e) function where G was the DBLP2005-CORE graph and e was the list of candidate edges to consider. The Adamic Adar method did not perform very well for smaller values of k but performed the best out of all of the five methods for k=|T|, which could be indicative that this method is more accurate for larger samples of links.

- P@10 = 0.0
- P@20 = 0.0
- P@50 = 0.04
- P@100 = 0.05
- P@252968 = 0.019120995540937984

5 CONCLUSIONS

Link importance and node importance are both incredibly important measures when looking at the topology of the graph. These measures can give you important insights into which nodes and links are critical to the overall structure of the graph. These measures can also be indicative of very active communities and important figures in those communities, as discussed in section 3.1.

Link prediction is by no means an exact science in the area of information networks, but there are various powerful measures that can give a good approximation as to the links that can form in future versions of a graph. These measures all seem to have different strengths, with some showing that they can more accurately predict larger amounts of links, while others predict smaller numbers of links more accurately.