Project 1

1st Daniel Giordanno Magaña Cruz
Data Engineering Major
Universidad Politécnica de Yucatán
Mérida, México
al1609073@upy.edu.mx

2nd Didier Omar Gamboa Angulo
Data Engineering Major
Universidad Politécnica de Yucatán.)
Mérida, México
didier.gamboa@upy.edu.mx

***Abstract*—This document is an analysis of the network called wiki vote, in which we could detect that there are different communities, in total there are 24 communities, also, we can appreciate their distribution of their degree distribution, we could also appreciate the size of the network their number of nodes as well as the shortest path to get from one node to another, also detect which node is closest to others, the importance of each node both in its betweenness and in its closeness centrality.**

***Index Terms*—Wiki vote, epinions, slashdot, wikipedia.**

I. Introduction

In this work, you will be able to appreciate the importance of this network. This Wikipedia votes network contains all the voting data from Wikipedia to from the start of Wikipedia until January 2008. This network is undirected. The nodes in the network represent users of Wikipedia voting so that a user can become an administrator and the users Links represent a user who voted on another user so that this user can become an administrator. Perhaps, you must be wondering what is the difference between a normal user and an administrator. It turns out that Wikipedia is a free encyclopedia written collaboratively by volunteers from around the world. A small part of Wikipedia contributors are administrators. Those administrators are users who have access to additional technical features that help maintain this network. So in order for a user to become an administrator, a Wikipedia Community admin request is issued through a public discussion or vote deciding who to promote to the admin. So this network seemed very important to me since it is a vote and here we can see these different communities obviously all this can be seen once the analysis has been done and what I could find in this network is that there are quite a few numbers of communities.
These data from this network were used in two important articles in the first article, basically those data were used to observe the relationships between positive and negative ratings on sites such as epinions in which you can rate both articles and reviewers on discussion sites. As the Slashdot social network, where users can label other users as friends and enemies, their focus, that is, in the first article, is to adapt and expand theories of social psychology to analyze these types of signed networks as arise in social computing applications. All this allows us to characterize the differences between the observed and predicted configurations, positive and negative links on social networks. Contrasts can also be made between different theories to make inferences about how links are used in particular social computing applications. In addition, this article explains that they use two types of signed network theories: there is equilibrium theory and State theory. balance theory can be seen as a model of likes and dislikes and as for the context of epinions which is the page that they used this on, a positive link can mean it is my friend but it can also mean i think it has more status higher than me, in the same way a negative link can mean he is my enemy or I think he has a lower state than me, comparing the two theories one can get an idea of how differences between state and equilibrium arise.

The second article is about prediction of signs, that is, suppose we have a network and in that network we have different nodes, as well as links that are the relationships between them and in each link there is a different sign, whether positive or negative. In this case they used 3 different datasets, the first was from epinions, which is a social network, so to speak, in which you can express trust or distrust of a user, then there is slashdot where you can designate a user as a friend or enemy to indicate their approval or disapproval of any comment, the third is the Wikipedia voting network, where you can vote for or against promoting a user to be an administrator. In order to do all that, they used a logistic regression type classifier, in which basically they can see if something is yes or if something is no, that is, there are only two probabilities yes or no so that in this way they can predict the sign either positive or negative, but nevertheless at the time of doing all that, they realized that this was a bit like link prediction. Apart from that, when comparing their results, they used two theories; the first is the theory of structural equilibrium, based on common principles such as: my friend's friend is my friend, my friend's enemy is my enemy, the friend of my enemy is my enemy, and perhaps less convincingly the enemy of my enemy is my friend. Then after that they used another theory is Guha's, in which in this theory a positive edge (x, y) means that x considers y to be higher than the other while a negative order (x, y) means that x considers that y has a lower state. And then basically using these two theories was how they could classify what kind of what sign the border could have.

In addition, in order to be able to do all this work, it was necessary to use Google Colab, which basically serves to run Jupyter Notebooks, since it allows you to have 12 GB of RAM

Identify applicable funding agency here. If none, delete this.

and 72 GB of hard disk space so that you can upload your Notebook and your files. I decided to do it in this way, since my computer was not supporting the network, due to its size and it was taking too long to traverse this network, to do some initial analysis like the average shortest path, it took me about 3 hours on my computer, for so I decided to switch to Google Colab and it was a little easier to program it. On the other hand, I didn't know how to use Google Colab and that was a little bit difficult, but I managed to do that job and it was accomplished. This is a visualization of the graph:

```
[11] nx.draw(wiki)
     plt.show()
```

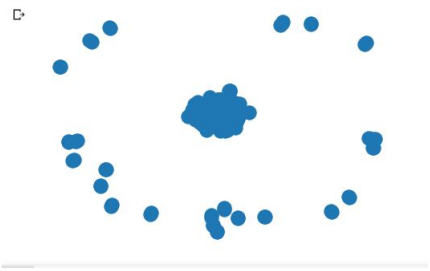

Fig. 1. Visualization of the network

II. Network Characteristics

The network features used here are as follows:

- Number of nodes
- Number of links
- Clustering Coefficient
- Average Shortest Path
- Diameter
- Center
- Length of center (number of elements in the center)

All these characteristics of the network can be seen in the following dataframe:

	num_nodes	num_links	clust_coef	avg_shortest_path	diameter	nodes_center	num_elements_center
0	7115	100762	0.140898	3.122281	8	[11, 178, 127]	3

Fig. 2. Dataframe of Network Characteristics

These are the screenshots of the code used below:

III. Centrality Measures

The measure used in this networks are the following:

- Degree centrality
- Minimum degree centrality
- Maximum degree centrality
- Pagerank of all nodes
- Betweenness
- Minimum Betweenness
- Minimum Betweenness
- Closeness
- Minimum Closeness
- Maximum Closeness

```
[7] network = [wiki]
    to_df_1 = []
    for i in network:
        num_nodes = nx.number_of_nodes(i)
        num_links = nx.number_of_edges(i)
        #density = nx.density(i)
        clust_coef = nx.average_clustering(i)
        #readius = nx.radius(i)
        N_sample = 1000
        subgraph = wiki.subgraph(list(max(nx.connected_components(wiki), key=len)))
        Nodes_sample = random.sample(subgraph.nodes, N_sample+200)
        Graph_sample = wiki.subgraph(Nodes_sample)
        avg_sht_path = average_path_length_sample(Graph_sample, 1000)
        #eccentricity = ecce(Graph_sample, 1000)
        diameter = diam(Graph_sample, 1000)
        center = cen(Graph_sample, 1000)
        elementos_cen = len(center)
        #mean_deg = statistics.mean(degree_cent)
        lis_of_them = [num_nodes, num_links, clust_coef, avg_sht_path,
                       diameter, center, elementos_cen]
        to_df_1.append(lis_of_them)
    net_features = pd.DataFrame(data=to_df_1, columns=['num_nodes', 'num_links',
                                                       'clust_coef', 'avg_shortest_path',
                                                       'diameter', 'nodes_center',
                                                       'num_elements_center'])
```

Fig. 3. Code of Network Characteristics

	min_deg	max_deg	pagerank	min_betweenness	max_betweenness	min_closeness	max_closeness
0	10	999	{'30': 0.0001519420295970706, '1412': 0.000270...	10	999	10	999

Fig. 4. Dataframe

All these meausres of the network can be seen in the following dataframe:
The code used below is the following:

```
[9] network = [wiki]
    to_df_2 = []
    for i in network:
        degree_cent = nx.degree_centrality(i)
        max_deg = max(degree_cent)
        min_deg = min(degree_cent)
        pagerank = nx.pagerank(i)
        betweenness = nx.betweenness_centrality(i)
        max_betweenness = max(betweenness)
        min_betweenness = min(betweenness)
        closeness = nx.closeness_centrality(i)
        max_closeness = max(closeness)
        min_closeness = min(closeness)
        lis_of_them1 = [min_deg, max_deg, pagerank,
                        min_betweenness,
                        max_betweenness, min_closeness,
                        max_closeness]
        to_df_2.append(lis_of_them1)

    cent_measures = pd.DataFrame(data=to_df_2,
                                 columns=['min_deg',
                                          'max_deg',
                                          'pagerank',
                                          'min_betweenness',
                                          'max_betweenness', 'min_closeness',
                                          'max_closeness'])
```

Fig. 5. Code of Network Characteristics

Basically the betweenness centrality tell us how important an actor is, therefore here, the node with highest betweenness centrality is the following: The node with the highess closen-

```
betweenness = nx.centrality.betweenness_centrality(wiki)
sor1 = sorted(betweenness.items(), key=lambda x: x[1],reverse=True)[0:1]
print('The node who has the highest closeness centrality is {}'.format(max(sor1)))

The node who has the highest closeness centrality is ('2565', 0.06125752063855017)
```

Fig. 6. Node with highest betweenness

ness centrality is the following:
In other words, this node is the one who is closest to the other nodes.

IV. Degree Distribution and Models of Networks

This is the degree distribution of this network is the following

```
[23] clos_cen = nx.closeness_centrality(wiki)

     sor = sorted(clos_cen.items(), key=lambda x: x[1],reverse=True)[0:1]
     print('The airpot who has the highest closeness centrality is {}'.format(max(sor)))

     The airpot who has the highest closeness centrality is ('2565', 0.48741490125142045)
```

Fig. 7. Node with highest closenness

```
[25] plt.hist(nx.degree_histogram(wiki))
     plt.show()
```

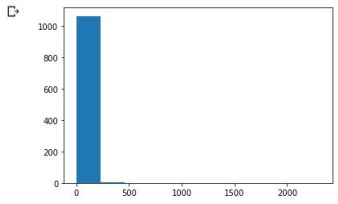

Fig. 8. Degree Distribution

V. Community Detection

In order of doing the community analysis I had to use Gephi. I computed the granularity,which is of 0.429, in order of then visualazing the different communities that are there. I found that there are 24 different communities in my network, but the most important ones are the following:

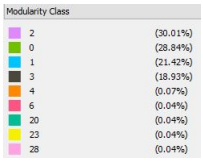

Fig. 9. Most important communities

Therefore this is the visualization of the communities:

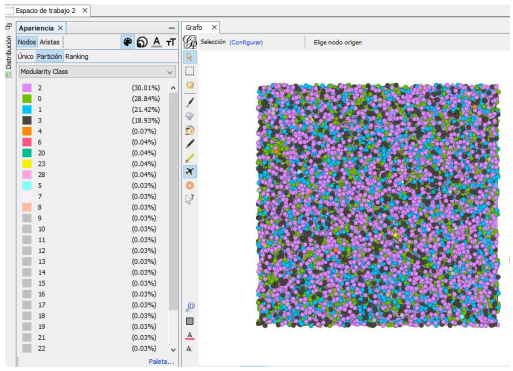

Fig. 10. Visualization of the communities

VI. Conclusion

It could be verified that this network itself is quite large, although it is not immensely large, but nevertheless, since it has 7115 nodes and 100762 links, the network can be considered as a medium network. Apart from that its clustering coefficient is 0.1408 and the average shortest path to get from one node to another is 3.50, the diameter of this network is 8, the nodes that are in the center of the list of nodes are 4 In this case, according to the function, this number can vary because the function randomly selects 1000 nodes and thus calculates the different functions. Another characteristic feature of this network is that this network was not connected, therefore we had to get its maximum number of components and from that we had to get each measure, instead of using the Networkx library. In this case, we have to take out the different components that were in this network and from that go on getting what we wanted to get, for example if we wanted to get the average shortest path, if we wanted to get the distance or if we wanted to get another metric and that's how I was doing For each sample of nodes, all this was done within the same for and then nested to a data frame. What we can check at the beginning is that this network could be easily visualized, immediately we could appreciate the node that has the greatest central closeness and in this case it is node 2565 and we could also discover which has the greatest number of centrality betweenness and it is the same node: 2565. Only their numbers are going to change, we could see that the most important nodes in this network are 5: the nodes 199, 998, 996, 995 and 994 because these are the ones with the highest degree number after We could appreciate their distribution by seeing their distribution of each one and something very important in this is that we were able to get their different communities, although we got the communities with Gephi and not with Python. We could see that doing it with Gephi, our network looked quite large, the visualization that Gephi gives you is a very clogged visualization, we can find that there are 24 different communities and that the granularity in this case is 0.429, generally the community the community the largest in this case is the two is represented in purple and there are 38.01 percent, then the one with the least are those from 9 to 24, all of which have 0.03 percent. An important fact in this is that the beginning when I downloaded this network and decided to use it, the official page told me that this network was a direct network, but after the analysis I concluded that this network is not directed.

References

[1] "Overview of NetworkX," Overview of NetworkX - NetworkX 2.4 documentation, 17-Oct-2019. [Online]. Available: https://networkx.github.io/documentation/networkx-2.4/.

[2] M. Grandjean, "GEPHI - Network visualization tutorial [HD]," Youtube. [Online]. Available: https://www.youtube.com/watch?v=FLiv3xnEepw.

[3] R. Compton, "Community detection and colored plotting in networkx," Community detection and colored plotting in networkx - Ryan Compton. [Online]. Available: http://ryancompton.net/2014/06/16/community-detection-and-colored-plotting-in-networkx/.

[4] M. O. H. A. M. M. E. D. Z. U. H. A. I. R. K. A. D. R. Y. S. E. I. F. E. D. I. N. E. AL-TAIE, PYTHON FOR GRAPH AND NETWORK ANALYSIS. Place of publication not identified: SPRINGER, 2019.

[5] M. E. J. Newman, Networks. Oxford: Oxford University Press, 2019.

[6] M. Tsvetovat and A. Kouznetsov, Social networks analysis for startups: finding connections on the social web. Sebastopol (CA): O'Reilly Media, 2011.

[7] F. Menczer, S. Fortunato, and C. A. Davis, A first course in network science. Cambridge: Cambridge University Press, 2020.

[8] "Wikipedia vote network," SNAP. [Online]. Available: http://snap.stanford.edu/data/wiki-Vote.html.

[9] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," Proceedings of the 28th international conference on Human factors in computing systems - CHI '10, 2010.
[10] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," Proceedings of the 19th international conference on World wide web - WWW '10, 2010.