Applied Social Network Analysis

Under the supervision of Dr Sanjay Kumar Pandey,

Dept. of Mathematical Sciences, IIT (BHU) Varanasi.

-Submitted By

Sree Rupa Harshitha Karamcheti (17123005)

& Mitisha Pandey (17123007)

Dept. of Mathematics and Computing,

IIT (BHU) Varanasi.

CONTENTS

Abstract: In this report we presented the network analysis of Social connections using the NetworkX library. We worked on network analysis and why we might model phenomena as networks, the concept of connectivity and network robustness. Then we explored ways of measuring the importance or centrality of a node in a network. Then finally worked on the evolution of networks over time and covered models of network generation and the link prediction problem.

Key-words: Networks, Graph, Clustering, NetworkX, Social Network Analysis (SNA), Centrality, Link Prediction, Preferential Attachment, Small World Model

Softwares/ libraries used: Jupyter notebook, NetworkX

Introduction: Network is a set of nodes with interconnections (edges). Many complex structures can be modelled by networks. For instances there are transportation networks, biology networks, social networks, financial networks, etc. By understanding the structure of networks, we can answer questions for complex phenomena and deal with very complex relationships. Networks allow us to make those complex phenomena simpler by representing them as a network and then using certain computations that allow us to answer that.

Social Network Analysis (SNA) is a tool for mapping networks and identifying the pattern of a network. SNA's defining point is therefore based on the formation of relationships. "Social media, such as Twitter and Facebook, plays a critical role in disaster management by propagating emergency information to a disaster-affected community. It ranks as the fourth most popular source for accessing emergency information", J. Kim & M. Hastak[4].

SNA can be used by a group as a process of "learning and understanding the (formal and informal) networks that operate in a given field", Hovland, 2007 [3]. This extensive form of mind-mapping allows the group not only to identify networks, but also to highlight the patterns of information exchange within the network. Although networks are often created to pass information from one individual to another, and over time the content is also shared with a wider network, it also gradually grows to take in other outside contact and networks. An SNA focuses on the structure of the relationships that weave between people and organisations within a network.

Within every network there are starting points and branching points. Each individual who forms a network, whether it us within a team or an organisation, will exchange information with others who share the same or common beliefs and ideas. They will be likely to share information with friends, partners and relatives if they find the information interesting. Maximising the appeal of this information can increase traffic to the sites or pages of the distributing organisation.

Bipartite Graphs:

A bipartite graph, also called a bigraph, is a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent. A bipartite graph is a special case of a k-partite graph with k=2.

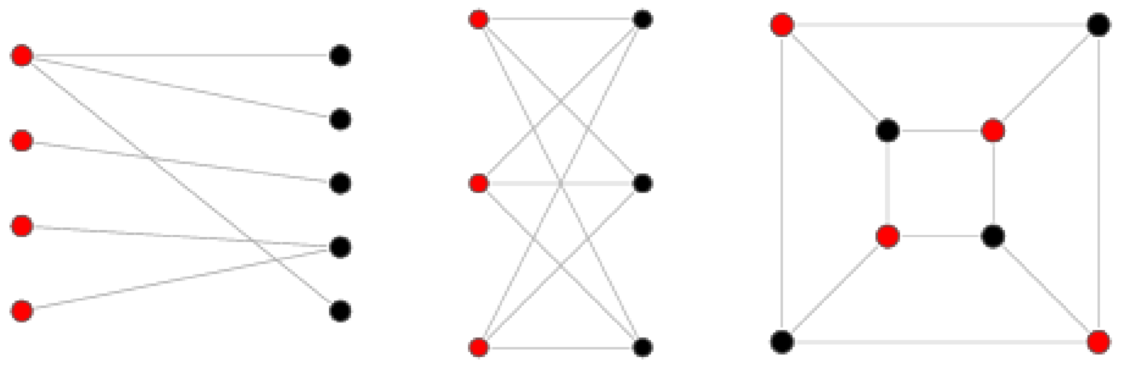


Figure 1:Examples of Bipartite Graphs

NetworkX

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. NetworkX provides data structures for graphs (or networks) along with graph algorithms, generators, and drawing tools. The structure of NetworkX can be seen by the organization of its source code. The package provides classes for graph objects, generators to create standard graphs, IO routines for reading in existing datasets, algorithms to analyse the resulting networks and some basic drawing tools.

NetworkX provides:

- Tools for the study of the structure and dynamics of social, biological, and infrastructure networks;
- A standard programming interface and graph implementation that is suitable for many applications;
- A rapid development environment for collaborative, multidisciplinary projects;
- An interface to existing numerical algorithms and code written in C, C++, and FORTRAN; and
- The ability to painlessly work with large nonstandard data sets.

With NetworkX you can load and store networks in standard and nonstandard data formats, generate many types of random and classic networks, analyse network structure, build network models, design new network algorithms, draw networks, and much more.

Bipartite Module:

This module provides functions and operations for bipartite graphs. Bipartite graphs **B = (U, V, E)** have two node sets **U, V** and edges in **E** that only connect nodes from opposite sets. It is common in the literature to use a spatial analogy referring to the two node sets as top and bottom nodes.

The bipartite algorithms are not imported into the network namespace at the top level so the easiest way to use them is with:

>>> from networkx.algorithms import bipartite

NetworkX does not have a custom bipartite graph class but the Graph() or DiGraph() classes can be used to represent bipartite graphs.

Clustering Co-efficient

Triadic Closure: It is the tendency for people who share lots of connections, to form a connection themselves to become connected.

Clustering coefficient: Measure the degree to which nodes in a network tend to form 'cluster' or triangle. It is categorised into: Local, Global

Local Clustering Co-efficient: Measures clustering from the point of view of a single node. It is the fraction of pairs of the nodes friends that are friends with each other. Local Clustering co-efficient of a Node A in Graph G:

$$\text{LCC} = \frac{No.of\ pairs\ of\ A's\ Friends\ who\ are\ friends}{No.of\ pairs\ of\ A's\ Friends}$$

For nodes with less than 2 friends, Local Clustering Co-efficient is 0. Local Clustering co-efficient in NetworkX is implemented using network.clustering(Graph, Node) function.

Global Clustering Co-efficient: It is the measure of clustering for all nodes together. It can be calculated in two ways:

1. Average of sum of all Local Clustering co-efficients of the nodes.
2. Also called as Transitivity: $\frac{3*No.of\ closed\ triads}{No.of\ open\ triads}$ where closed triads are triangles formed by 3 nodes, 3 edges and open triads are 3 nodes in which only two edges are present. (Calculated by using network.trasitivity(Graph))

Distance measures

Average distance: It is average of distances between each pair of nodes (sum of distances/no of edges).

Diameter: The maximum of distances between all pairs of nodes.

Eccentricity: The maximum distance of a given node from the other nodes.

Radius: The minimum among all the maximum distances between a node to all other nodes i.e., minimum of the eccentricity of all nodes.

Periphery: Set of nodes in graph whose eccentricity is equal to diameter

Centre of a Graph: The set of nodes that have eccentricity equal to radius.

Connected Undirected Graph: In an Undirected Graph, for every pair of nodes there is a path between them.

Connected Components: A connected component is a subset of nodes such that there are two conditions this set of nodes satisfy.

(i) Every node in the subset has to have a path to every other node in the subset. That's condition number one.
(ii) No other node outside of the subset has a path to any node inside the subset.

Strongly Connected Directed Graph: A direct graph is strongly connected if for every pair of nodes, say U and V, there is a directed path that goes from U to V and another directed path that goes from V to U.

Weakly Connected Directed Graph: A direct graph is weakly connected if when replaced all the directed edge into undirected, it results in connected undirected graph.

Network Robustness

The ability of a network to maintain its general structure, or its general functions when it faces failures or attacks.

Node Connectivity: The minimum number of nodes that you need in order to disconnect a graph or disconnect a particular pair of nodes.

Edge connectivity: The minimum number of edges that you need in order to disconnect the graph, or to disconnect any pair of nodes

Network Centrality

Network Centrality measures are the ones that allows us to find the most important nodes in a particular network. The cases in which we want to use these techniques is, for example, sometimes we're interested in finding who the influential nodes in a social network are, which are the nodes that are very good at disseminating information to many other nodes, or on the other hand, which are the nodes that are good at preventing sort of bad behaviours or epidemics from spreading on a social network? These can be used in other networks as well, so we can use a centrality measures to find hubs in a transportation network or important pages on the web or more generally these measures allow us to find nodes that prevent the network from breaking up. Nodes that if we were to remove them, they would cause the network to fracture and break up into different components.

Types of centrality measures:

- Degree Centrality
- Closeness Centrality
- Betweenness Centrality
- Load Centrality
- Page Rank
- Katz Centrality
- Percolation Centrality

Degree Centrality: Degree centrality makes the assumption that important nodes have many connections. This is the most basic way in which you could define importance or centrality. (For Directed networks either in-degree or out-degree is considered).

$$C_{deg}(v) = \frac{d_v}{|N| - 1}$$

Where d_v is degree of node v and |N| is the number of nodes in the Graph.

Closeness Centrality: Closeness centrality is a measure of the average shortest distance from each vertex to each other vertex. Specifically, it is the inverse of the average shortest distance between the vertex and all other vertices in the network.

$$C_{close}(v) = \frac{|R(v)|}{\sum_{w \in R(v)} d(v,w)}$$

Where R(v) is the set of nodes of Graph G which node v can reach and d(v,w) is the length of path from node w to node v.

Normalization of Closeness centrality:

$$C_{norm\_close} = \frac{|R(v)|}{|N|-1} C_{close}(v)$$

Where |N| is the number of nodes in the graph.

Betweenness Centrality: Betweenness centrality measures the extent to which a vertex lies on paths between other vertices. Vertices with high betweenness may have considerable influence within a network by virtue of their control over information passing between others. They are also the ones whose removal from the network will most disrupt communications between other vertices because they lie on the largest number of paths taken by messages.

$$C_{btw}(v) = \sum_{s,t \in N} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$$

Where $\sigma_{s,t}$ is the number of shortest paths between node s, t and $\sigma_{s,t}(v)$ is the number of shortest paths between node s, t that pass-through node v. s, t pairs chosen from set of nodes N such that we can either include node v or completely exclude. Also, s & t are chosen such that there is at least one path between them.

Normalization of Betweenness centrality: Graphs that have a larger number of nodes will tend to have higher centrality than nodes of graphs that are smaller in terms of the number of nodes. That is because in large graphs, there are more nodes i.e. s and t, to choose from to compute the centrality of the nodes. Therefore, we use normalization in these cases.

For Directed Graphs: $C_{norm\_btw} = \frac{1}{(|N|-1)(|N|-2)} C_{btw}(v)$

For Undirected Graphs: $C_{norm\_btw} = \frac{1}{\frac{1}{2}(|N|-1)(|N|-2)} C_{btw}(v)$

Time Complexity of Betweenness centrality: Computing betweenness of all nodes in a graph can be computationally expensive. Depending on algorithm it can take time complexity of O(|N|^3). This can be reduced by considering subsets of the Nodes set and we can approximate computation using them. We can define subsets by choosing specific sets of source and target nodes to calculate betweenness centrality.

PageRank: Developed by the Google founders to measure the importance of webpages using the hyperlink network structure of the web. PageRank assigns a score of importance to every single node assuming that important nodes are those that have many in-links from important pages. A node's page rank depends upon other nodes page ranks. Page rank can be used for any type of network but it is used mostly for Directed Networks. Page rank is calculated 'k' times (k is any number we choose that is greater than 1)

Steps for calculating page rank of each node:

1. All nodes get a Page rank of 1/n in the first step. (where ‘n’ is the number of nodes in the graph)
2. Perform the Basic Page rank update rule ‘k’ times:
   a. Basic Page rank update rule: Each node gives an equal of share of its current page rank to the nodes it is connected to.
   b. The new PageRank of each node is sum of all PageRank it received from other nodes.

After repeating sufficient k-steps we observe that PageRank of a node converges to a number approximately.

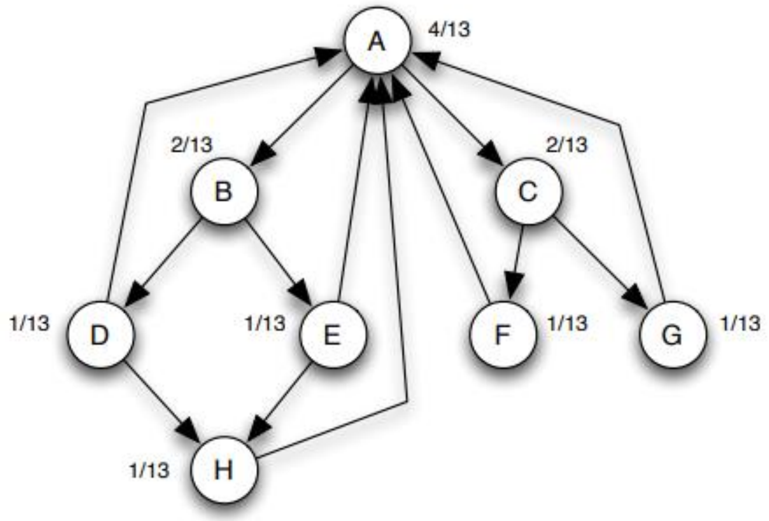


Figure 1: Calculation of Basic PageRank in a network based on edges

Basic PageRank of a node can be interpreted as the probability that a random walker lands on the node after k steps. Random walk of k steps imply that we start at any random node in the network and choose an outgoing edge at random and follow it to the next node until k steps.

Scaled PageRank: It becomes a problem in almost any real network to which PageRank is applied: as long as there are small sets of nodes that can be reached from the rest of the graph, but have no paths back, then PageRank will build up there for example see Figure 2 when a randomly chosen edge reaches node F or G, we see that there is no other way to go except to F or G. So, it builds up the page ranks for F and G and all other nodes nearly equal to 0. To avoid this situation, we use Scaled PageRank.

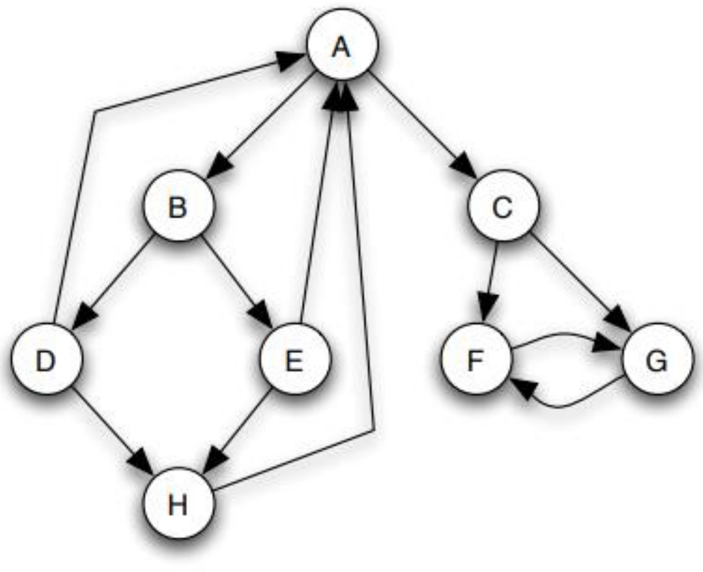


Figure 2: A network with nodes F and G which cannot reach other nodes of network except themselves

We can use this idea here. We pick a scaling factor α that should be strictly between 0 and 1. We then replace the Basic PageRank Update Rule with the following:

Scaled PageRank Update Rule: First apply the Basic PageRank Update Rule. Then scale down all PageRank values by a factor of α.

This means that the total PageRank in the network has shrunk from 1 to s. We divide the residual 1 – α units of PageRank equally over all nodes, giving (1 − α)/n to each.

One can show that repeated application of the Scaled PageRank Update Rule converges to a set of limiting PageRank values as the number of updates k goes to infinity. Scaling factor α is generally chosen between 0.8 and 0.9.

Hubs and Authorities

Given a query to a search engine, instead of searching all the web pages it goes to a subset of web pages with Roots, Hubs. A Root is a set of highly relevant web pages to the query (e.g., pages that contain query string) are the Potential Authorities and the Potential Hubs are all pages that link to a page in Root. This whole Root nodes or any node that link to a node in root i.e., hubs together is considered as Base set. Consider all edges and nodes connecting in Base set.

HITS Algorithm

HITS (hyperlink-induced topic search) is an algorithm that makes use of the link structure of the web in order to discover and rank pages relevant for a particular topic. It assigns a Hub score/ Authority score to the nodes in Base network. It is calculated in k-iterations.

Steps to calculate Hub and Authority scores of each node:

1. Assign each node an Authority and Hub score of 1.
2. Apply the Authority update rule: Each node's authority score is sum of Hub scores of nodes that point to current node.
3. Apply Hub update rule: Each node's Hub score is sum of authority score of nodes that it points to.
4. Normalized Authority score: $Norm\_Auth(j) = \frac{Auth(j)}{\sum_{i \in N} Auth(i)}$ where $Auth(i)$ indicates Authority Score of node i.
5. Normalised Hub score: $Norm\_Hub(j) = \frac{Hub(j)}{\sum_{i \in N} Hub(i)}$ where $Hub(i)$ indicates Hub Score of node i.
6. Repeat k times.

For most networks as k gets larger authority and hub scores converge to a value.

Power Laws

When people measured the distribution of links on the Web, however, they found something very different. In studies over many different Web snapshots, taken at different points in the Web's history, the recurring finding is that the fraction of Web pages that have k in-links is approximately proportional to $1/k^2$. This is so different from normal distribution. The crucial point is that $1/k^2$ decreases much more slowly as k increases, so pages with very large numbers of in-links are much more common than we'd expect with a normal distribution.

A function that decreases as k to some fixed power, such as $1/k^2$ in the present case, is called a power law; when used to measure the fraction of items having value k, it says, qualitatively, that it's possible to see very large values of k.

$$P(k) = Ck^{-\alpha}$$

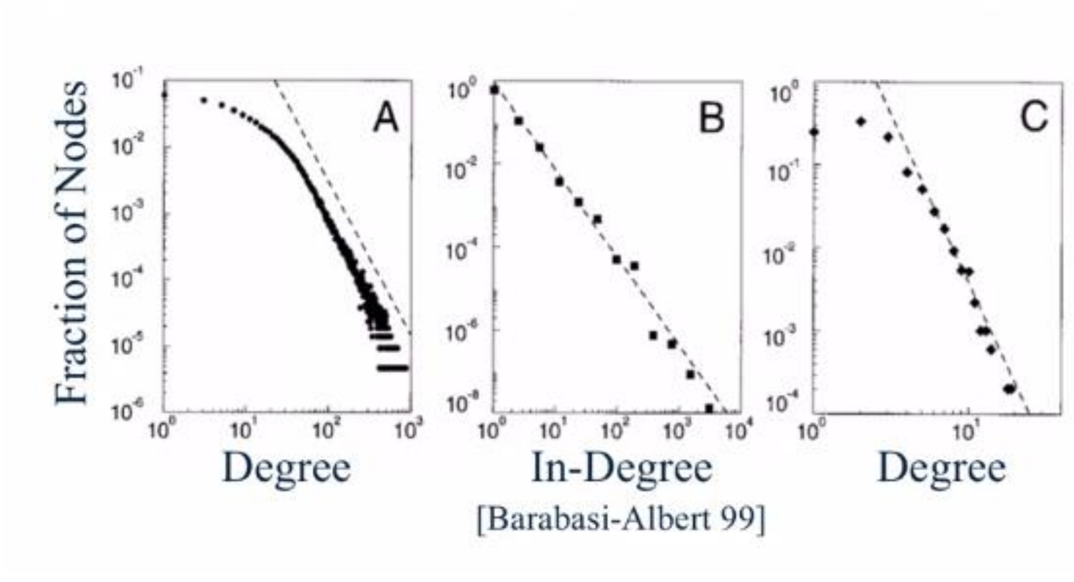
where C and alpha are constants

Degree distribution of a Graph: The Degree distribution of a Graph is the probability distribution of degrees over entire network.

$$D(k) = \frac{V(k)}{|N|}$$

where $V(k)$ is the number of nodes with degree k and |N| is the number of nodes in the network.

Degree distribution in Real Networks



Consider the graphs above, A shows the degree distribution of network of 2,25,000 actors connected when they appear in a movie together, B shows the degree distribution of network of 3,25,000 documents on WWW connected by URLs**,** C shows the degree distribution of network of 4,941 power generators connected by transmission lines.

We can notice that all these Degree Distribution graphs look like a straight line when on log-log scale which gives us the equation of Power Law,

$$P(k) = Ck^{-\alpha}$$

with α being 2.3, 2.1, 4 for A, B, C respectively.

Many real networks tend to have the Power law distribution. Networks with Power law distribution have small number nodes with very large degree and large number of nodes with small degree. This is sort of like Rich get richer.

Preferential Attachment Model

Growing network models based on growth and preferential attachment are known to generate networks with power-law in-degree distributions. This model generates these networks by a process of "preferential attachment", in which new network members prefer to make a connection to the more popular existing members.

The model starts with two nodes connected by an edge. At each step, a new node is added. A new node picks an existing node to connect to randomly, but with some bias. More specifically, a node's chance of being selected is directly proportional to the number of connections it already has, or its "degree." This is the mechanism which is called "preferential attachment."

Steps:

1. Start with two nodes connected by an edge
2. At each time add a new node with an edge connecting to an existing node.
3. Choose the node to connect at random with probability proportional to each node's degree.
4. The probability of connecting to a node u of degree k_u is $\frac{k_u}{\sum_j k_j}$

Small world networks

Real networks exhibit High average Clustering Co-efficient and low Shortest path distance. Preferential attachment model fails to produce High average Clustering Co-efficient but shows low Shortest path distance. We need a model that exhibits both High average Clustering Co-efficient and low Shortest path distance. Small world models are designed for this purpose.

Steps:

1. Start with a ring of n nodes where each node is connected to k nearest neighbours.
2. Fix a parameter p in [0,1]
3. Consider each edge (u,v) in the network, with a probability of p we select a node w at random and rewire it to (u,w)

For small values of p, small world models have High average Clustering Co-efficient and low Shortest path distance, matching what we observe in real world.

Link Prediction in a Network

Problem- Given a network, predict edges that will be formed in future. There are five basic measures to do this. They are listed as follows.

1. Number of common neighbours
2. Jaccard co-efficient
3. Resource Allocation index
4. Academic-Adar index
5. Preferential attachment
6. Number of common neighbours

1. Number of common neighbours: The number of common neighbours of nodes X and Y is

$Neigh(X,Y) = |N(X)| \cap \ |N(Y)|$ where, $|N(X)|$ is the number of neighbours of node X.

Community common neighbours: Some methods use community structure of networks to predict future links. It assumes that pairs of nodes which belong to same communities and have many neighbours in common are more likely to form a new link.

$$CNeigh(X,Y) = |N(X)| \cap \ |N(Y)| + \sum_{u \in N(X) \cap N(Y)} f(u)$$

where, f(u) is 1 if u is in same community as X and Y and 0 otherwise.

2. Jaccard co-efficient: It is the same as the common neighbours except it is normalized.

$$J(X,Y) = \frac{|N(X)| \cap \ |N(Y)|}{|N(X)| \cup \ |N(Y)|}$$

3. Resource Allocation index: It indicates the fraction of "resource" that a node can send to another through their common neighbours.

$$res\_alloc(X,Y) = \sum_{u \in N(X) \cap N(Y)} \frac{1}{|N(u)|}$$

Community resource allocation: Similar to Resource allocation index but only considers nodes in same community

$$Cres\_alloc(X,Y) = \sum_{u \in N(X) \cap N(Y)} \frac{f(u)}{|N(u)|}$$

4. Academic-Adar index: Similar to Resource allocation index

$$acad\_adar(X,Y) = \sum_{u \in N(X) \cap N(Y)} \frac{1}{\log(|N(u)|)}$$

5. Preferential attachment: Similar to the Preferential attachment model. The nodes with more connections tend to form new connections.

$$pref\_attach(X,Y) = |N(X)||N(Y)|$$

Project Work

We have performed three tasks under this project:

1. Analyse the algorithm by which a set of randomly generated graphs are constructed i.e., Preferential Attachment/Small World
2. Predict whether or an employee in a company will receive a management position salary.
3. Predict the future possible connections between nodes in a network.

Part I – Analyse Algorithm

For this part we analysed randomly generated graphs and determine which algorithm created them

A list containing 5 NetworkX graphs are loaded. Each of these graphs were generated by one of three possible algorithms:

- Preferential Attachment (PA)
- Small World with low probability of rewiring (SW_L)
- Small World with high probability of rewiring (SW_H)

Code snippet for algorithm identification:

```
def arry(graphs):
    arr=[]
    for G in graphs:
         arr.append([nx.average_clustering(G),
,nx.average_shortest_path_length(G),len(degree_distribution(G))])
    return arr

def graph_identification():
a=[]
     arr=arry(P1_Graphs)
      for i in range(5):
          if arr[i][2]>10:
              a.append('PA')
          elif arr[i][0]<0.1:
              a.append('SW_H')
          else:
              a.append('SW_L')
      return a
 graph_identification()
```

Here, basically we analysed the average shortest path, degree distribution and average clustering co-efficient. As we have previously established that Preferential Attachment has very lengthy degree distribution. Also, Small World with low probability of rewiring exhibit high average clustering co-efficient and high average shortest path distance and High probability shows low average clustering co-efficient and low average shortest path distance.

Part II – Salary Prediction

For this part we worked with a company's email network where each node corresponds to a person at the company, and each edge indicates that at least one email has been sent between two people. The network contains the node attributes Department and ManagementSalary. Department indicates the department in the company which the person belongs to, and ManagementSalary indicates whether that person is receiving a management position salary.
To accomplish this, we trained a sklearn classifier on nodes that have ManagementSalary data, and predicted a probability of the node receiving a management salary for nodes where ManagementSalary is missing.

```python
from sklearn import svm
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import MinMaxScaler

G = nx.read_gpickle('email_prediction.txt')

def salary_predictions():
    def ismanagement(node):
        return node[1]['ManagementSalary']

    df = pd.DataFrame(index= G.nodes())
    df['Clustering'] = pd.Series(nx.clustering(G))
    df['Degree'] = pd.Series(nx.degree(G))
    df['Degree Centrality'] = pd.Series(nx.degree_centrality(G))
    df['Closeness'] = pd.Series(nx.closeness_centrality(G))
    df['Betweenness'] = pd.Series(nx.betweenness_centrality(G))
    df['PageRank'] = pd.Series(nx.pagerank(G))
    df['Management'] = pd.Series([ismanagement(node) for node in G.nodes(data=True)])

    df_train = df[~pd.isnull(df['Management'])]
    df_test = df[pd.isnull(df['Management'])]

    features = ['Clustering', 'Degree', 'Degree Centrality', 'Closeness', 'Betweenness', 'PageRank']
    X_train = df_train[features]
    Y_train = df_train['Management']
    X_test = df_test[features]

    scaler = MinMaxScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    clf = MLPClassifier(hidden_layer_sizes = [10, 5], alpha = 5,
                        random_state = 0, solver='lbfgs', verbose=0)
    clf.fit(X_train_scaled, Y_train)

    test_proba = clf.predict_proba(X_test_scaled)[:,1]

    return pd.Series(test_proba, X_test.index)
salary_predictions()
```

Part III – New Connections Prediction

In this part, we predicted the future connections between employees of the network. The future connections information has been loaded into the variable `future_connections`. The index is a tuple indicating a pair of nodes that currently do not have a connection, and the `Future Connection` column indicates if an edge between those two nodes will exist in the future, where a value of 1.0 indicates a future connection.

To accomplish this, we trained a sklearn classifier on those edges in `future_connections`, that have `Future Connection` data, and predict a probability of the edge being a future connection for those edges in `future_connections` where `Future Connection` is missing

```python
future_connections = pd.read_csv('Future_Connections.csv',index=0)
def new_connections_predictions():
    for node in G.nodes():
        G.node[node]['community'] = G.node[node]['Department']

    pa = list(nx.preferential_attachment(G))
    df = pd.DataFrame(index=[(x[0], x[1]) for x in pa])
    df['preferential_attachment'] = [x[2] for x in pa]

    cns = list(nx.cn_soundarajan_hopcroft(G))
    df['cns'] = [x[2] for x in cns]

    df['resource_allocation'] = [x[2] for x in list(nx.resource_allocation_index(G))]

    df['jaccard_coefficient'] = [x[2] for x in list(nx.jaccard_coefficient(G))]

    df = future_connections.join(df,how='outer')

    df_train = df[~pd.isnull(df['Future Connection'])]
    df_test = df[pd.isnull(df['Future Connection'])]

    features = ['cns', 'preferential_attachment', 'resource_allocation', 'jaccard_coefficient']
    X_train = df_train[features]
    Y_train = df_train['Future Connection']
    X_test = df_test[features]
    scaler = MinMaxScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    clf = MLPClassifier(hidden_layer_sizes = [10, 5], alpha = 5,random_state = 0, solver='lbfgs', verbose=0)
    clf.fit(X_train_scaled, Y_train)

    test_proba = clf.predict_proba(X_test_scaled)[:, 1]
    predictions = pd.Series(test_proba,X_test.index)
    null_values = future_connections[pd.isnull(df['Future Connection'])]
    null_values['prob'] = [predictions[x] for x in target.index]
    return null_values['prob']
```

Summary

- Represented and manipulated networked data using the NetworkX library
- Analyse the connectivity of a network
- Measure the importance or Centrality of a node in a network with measures like Degree, Closeness, betweenness
- Discussed the algorithms by which networks are constructed: Preferential Attachment, Small World Model.
- Discussed the link prediction problem and measures: Resource Allocation, Jaccard Co-efficient, Number of neighbours, Preferential Attachment
- Predict the evolution of networks over time

Conclusion

In this report we provided an overview of the methods and techniques for analysing, measuring and visualizing evolving social network analysis. In the past, static techniques were adapted to dynamic networks with relative success, but nowadays, with the advent of social media, scale and velocity of most of those static techniques reveal weaknesses that only can be addressed by methods and techniques designed for dealing with evolving data. Future link prediction, Graph Algorithm generation, Network manipulation, strategies and methods of analysis were discussed.

In recent years this area of research will continue to have significant development in the future, several problems are still unsolved and many of them can be significantly improved. The areas of applicability of evolving networks and social network analysis are also broader, with many of the abovementioned techniques moving from well-succeeded areas like world wide web, communication, telecommunication and mobile networks, to newer areas like social network recommendations, news and blog analysis.

Future Trends

Future trends of social network analysis will continue to be driven by future trends and characteristics of the network data, such as the size of data, which is incredibly getting large, and changes in space and time. On one side, there is the urge for scalable and efficient social network analysis methods, and on the other side, there is the need for methods to study the dynamics and evolution of social networks, able to deal with future velocity and timescale dimensions of the network data. Therefore, it is expected that the study of evolving networks will continue to be a significant strand of research in the context of social network analysis in the near future.

References

[1] Easley D., Kleinberg J., ‘Networks, Crowds, and Markets (2010): Reasoning About a Highly Connected World’, 1st Edition, Cambridge University Press, July 2010.

[2] Watts, D., Strogatz, S. ‘Collective dynamics of ‘small-world’ networks’. Nature Volume 393, pp 440–442 (1998).

[3] Hovland I., ‘Making a difference: M&E of policy research’, Overseas Development Institute, 111 Westminster Bridge Road, London, UK, 2007

[4] Kim J., Hastak M. ‘Social network analysis: Characteristics of online social networks after a disaster’, International Journal of Information Management, Volume 38, Issue 1, February 2018, pp 86-88.

[5] Hagberg A., Schult D., Swart P., ‘NetworkX documentation’ Last Updated on Dec 09, 2020.

[6] Can U., Alatas, B. ‘A new direction in social network analysis: Online social network analysis problems and applications’, Physica A: Statistical Mechanics and Its Applications, Volume 535, 1 December 2019.