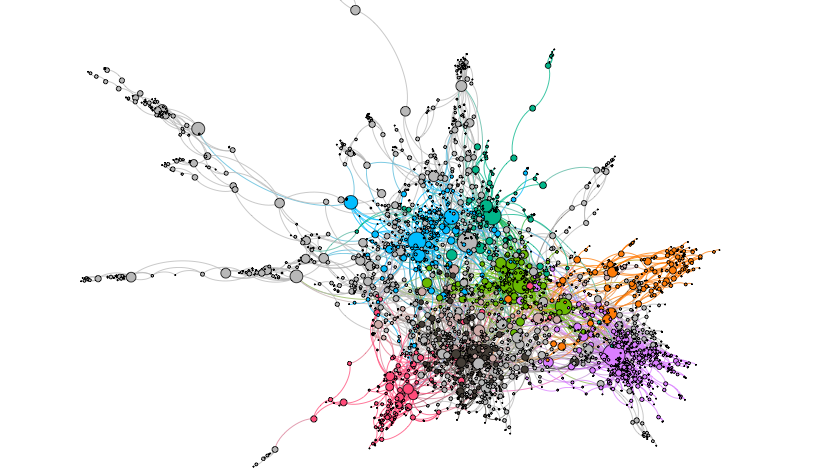


A Synopsis on

**NETWORK GRAPH ANALYSIS**



Under esteemed guidance of

Ms. Ashwairya Saxena

(Machine Learning Project)

By:

Manoj Konda,

NIT Warangal

**INDEX**

|  |  |
| --- | --- |
| **Sl.No.** | **Topic** |
| 1 | Introduction |
| 2 | Objective |
| 3 | Background |
| 4 | Hardware and Software Requirements |
| 7 | Future Scope |
| 8 | Conclusion |
| 9 | Bibliography and References |

**INTRODUCTION**

Network Graph Analysis has real broad applications in the field of networking. Two main areas are involved in the analysis of the application of network graphs, which are graph-based representation and network theory. Complex systems, such as a power grid, the World Wide Web, activity in different regions of the brain, or people within a community, can be understood, studied and visualized based on their connections in a network.

Networks are everywhere.

Networks or Graphs are a set of objects (called nodes) having some relationship with each other (called edges).

We live in a connected world and generate a vast amount of connected data. Social networks, financial transaction systems, biological networks, transportation systems, and a telecommunication nexus are all examples.

**OBJECTIVE**

To Explore and Analyse Network Graph data ,Node Classification, Link prediction and Community detection.

**BACKGROUND**

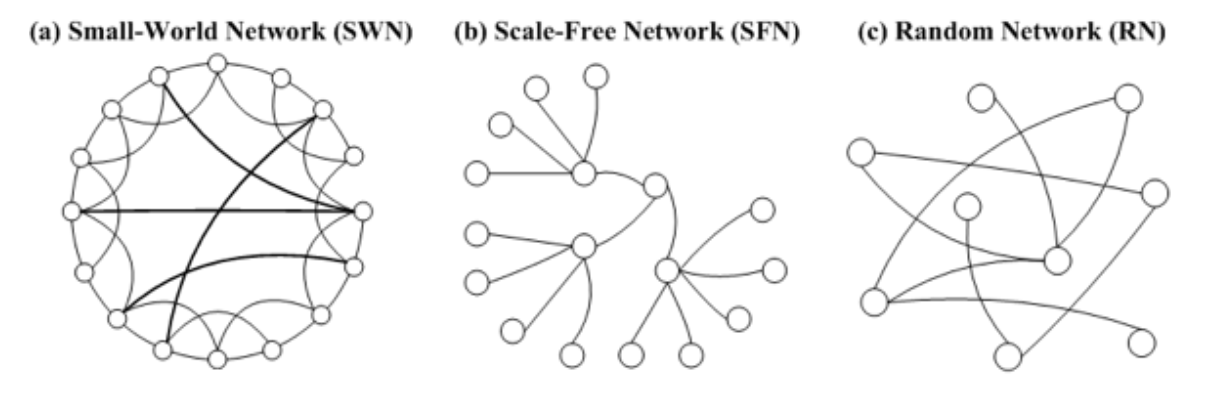
A graph has two components which are nodes and edges. In a graphical representation, these components have natural correspondences with the elements of the problem. In general, nodes in a graph represent features and edges represent interactions between features.

Nodes (A,B,C,D,E in the example) are usually representing entities in the network, and can hold self-properties (such as weight, size, position and any other attribute) and network-based properties (such as Degree- number of neighbors or Cluster- a connected component the node belongs to etc.).

Edges represent the connections between the nodes, and might hold properties as well (such as weight representing the strength of the connection, direction in case of asymmetric relation or time if applicable).

These two basic elements can describe multiple phenomena, such as social connections, virtual routing network, physical electricity networks, roads network, biology relations network and many other relationships.

Real-world networks and in particular social networks have a unique structure which often differs them from random mathematical networks:

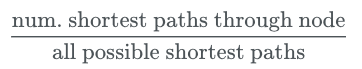


* Small-World Network phenomenon claims that real networks often have very short paths (in terms of number of hops) between any connected network members. This applies for real and virtual social networks (the six handshakes theory) and for physical networks such as airports or electricity or web-traffic routings.
* Scale Free networks with power-law degree distribution have a skewed population with a few highly-connected nodes (such as social-influences) and a lot of loosely-connected nodes.
* Homophily is the tendency of individuals to associate and bond with similar others, which results in similar properties among neighbors.

**Centrality**:

In network analysis, measures of the importance of nodes are referred to as centrality measures. Highly central nodes play a key role of a network, serving as hubs for different network dynamics. However the definition and importance of centrality might differ from case to case, and may refer to different centrality measures:

* Degree : the amount of neighbors of the node
* EigenVector / PageRank : iterative circles of neighbors
* Closeness : the level of closeness to all of the nodes
* Betweenness: the amount of short path going through the node

**Degree** is the simplest and the most common way of finding important nodes. A node’s degree is the sum of its edges. If a node has three lines extending from it to other nodes, its degree is three. Five edges, its degree is five. It’s really that simple. Since each of those edges will always have a node on the other end, you might think of degree as the number of people to which a given person is directly connected. The nodes with the highest degree in a social network are the people who know the most people. These nodes are often referred to as hubs, and calculating degree is the quickest way of identifying hubs

Degree centrality =

**Eigenvector** centrality is a kind of extension of degree—it looks at a combination of a node’s edges and the edges of that node’s neighbors. Eigenvector centrality cares if you are a hub, but it also cares how many hubs you are connected to. It’s calculated as a value from 0 to 1: the closer to one, the greater the centrality. Eigenvector centrality is useful for understanding which nodes can get information to many other nodes quickly.

Eigenvector centrality computes the centrality for a node based on the centrality of its neighbors. The eigenvector centrality for node i is the i-th element of the vector x defined by the equation

Ax=λx

where A is the adjacency matrix of the graph G with eigenvalue λ. By virtue of the Perron–Frobenius theorem, there is a unique solution x, all of whose entries are positive, if λ is the largest eigenvalue of the adjacency matrix A^2

.

**Betweenness centrality** is a bit different from the other two measures in that it doesn’t care about the number of edges any one node or set of nodes has. Betweenness centrality looks at all the shortest paths that pass through a particular node (see above). To do this, it must first calculate every possible shortest path in your network, so keep in mind that betweenness centrality will take longer to calculate than other centrality measures (but it won’t be an issue in a dataset of this size). Betweenness centrality, which is also expressed on a scale of 0 to 1, is fairly good at finding nodes that connect two otherwise disparate parts of a network. If you’re the only thing connecting two clusters, every communication between those clusters has to pass through you. In contrast to a hub, this sort of node is often referred to as a broker. Betweenness centrality is not the only way of finding brokerage (and other methods are more systematic), but it’s a quick way of giving you a sense of which nodes are important not because they have lots of connections themselves but because they stand between groups, giving the network connectivity and cohesion.

Betweenness centrality of a node v is the sum of the fraction of all-pairs shortest paths that pass through v

cB(v)=∑s,t∈Vσ(s,t|v)σ(s,t)

where V is the set of nodes, σ(s,t) is the number of shortest (s,t)-paths, and σ(s,t|v) is the number of those paths passing through some node v other than s,t. If s=t, σ(s,t)=1, and if v∈s,t, σ(s,t|v)=0^2.

**Libraries Used**:

* numpy
* matplotlib
* networkx
* warnings
* pandas
* Subprocess - check\_output
* Networkx.algorithms - community
* Networkx.algorithms - node\_classification
* **Numpy**: NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed
* **Matplotlib:** Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack
* **Pandas**: Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.
* **Networkx**: NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

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, analyze network structure, build network models, design new network algorithms, draw networks, and much more.

Applications: It helps to visualize Network data i.e., graph data , Node classification, Link Prediction(predicting relation between nodes), Community detection of a node etc.,

**Algorithms Used**:

* **Node Classification** - there is an inbuilt module in networkx i.e., node\_classification.

This module provides the functions for node classification problems.

The functions in this module are not imported into the top level `networkx` namespace.

You can access these functions by importing the `networkx.algorithms.node\_classification` modules, then accessing the functions as attributes of `node\_classification`

* **Link prediction** - There are various algorithms for Link prediction

1. Triadic closure
2. Jaccard Coefficient
3. Resource Allocation Index
4. Adamic Adar Index
5. Preferential Attachment
6. Community Common Neighbor
7. Community Resource Allocation

In these we use Jaccard Coefficient, Resource Allocation Index, Adamic Adar Index, Preferential Attachment,

**Jaccard Coefficient** :

It is calculated by the number of common neighbors normalized by total number of neighbors. It is used to measure the similarity between two finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets.

Jaccard Coefficient(X, Y) = 

**Resource Allocation Index** :

Among a number of similarity-based methods to predict missing links in a complex network, Research Allocation Index performs well with lower time complexity. It is defined as a fraction of a resource that a node can send to another through their common neighbors.

Research Allocation Index(X, Y) = 

**Adamic Adar Index** :

This measure was introduced in 2003 to predict missing links in a Network, according to the amount of shared links between two nodes. It is calculated as follows:

Adamic Adar Index(X, Y) = 

**Preferential Attachment** :

Preferential attachment means that the more connected a node is, the more likely it is to receive new links. Nodes with higher degree get more neighbors.

Preferential Attachment(X, Y) =|N(X)|.|N(Y)|

* **Community detection:**

1. Kernighan–Lin bipartition algorithm- Partition a graph into two blocks using the Kernighan–Lin algorithm.
2. K-Clique- Find k-clique communities in graph using the percolation method.
3. Modularity-based communities-
4. Find communities in a graph using the greedy modularity maximization.
5. Find communities in graphs using Clauset-Newman-Moore greedy modularity maximization.
6. Lukes Algorithm for exact optimal weighted tree partitioning.
7. Label propagation community detection algorithms. etc.,

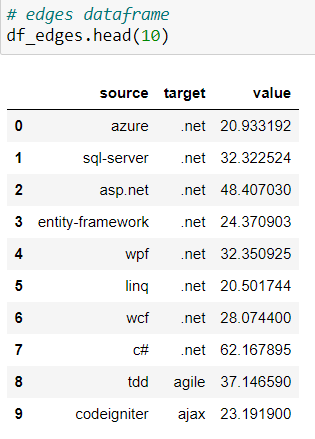
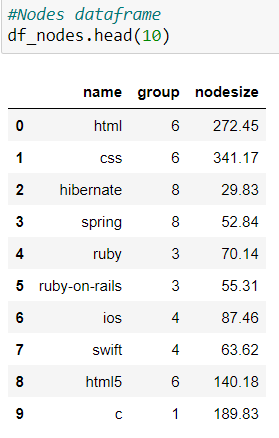
I used greedy\_modularity\_communities(G) and label\_propagation\_communities(G)

Common thing to ask about a network dataset is what the subgroups or communities are within the larger social structure. Is your network one big, happy family where everyone knows everyone else? Or is it a collection of smaller subgroups that are only connected by one or two intermediaries? The field of community detection in networks is designed to answer these questions. There are many ways of calculating communities, cliques, and clusters in your network, but the most popular method currently is modularity. Modularity is a measure of relative density in your network: a community (called a module or modularity class) has high density relative to other nodes within its module but low density with those outside. Modularity gives you an overall score of how fractious your network is, and that score can be used to partition the network and return the individual communities.13

Very dense networks are often more difficult to split into sensible partitions. Luckily, as you discovered earlier, this network is not all that dense. There aren’t nearly as many actual connections as possible connections, and there are several altogether disconnected components. It's worthwhile partitioning this sparse network with modularity and seeing if the result makes historical and analytical sense.

Community detection and partitioning in NetworkX requires a little more setup than some of the other metrics.

**Dataset**: My datasets “stack\_network\_links.csv” and stack\_network\_nodes.csv” are the info regarding different technologies(like html, css, c#, react ect.,) are considered as nodes whereas links are source and target.



**HARDWARE AND SOFTWARE REQUIREMENTS**

|  |  |
| --- | --- |
| **HARDWARE TOOLS** | **MINIMUM REQUIREMENTS** |
| Processor | i5 or above |
| Hard disk | 10GB |
| RAM | 8GB |
| Monitor | 17’’ Coloured |
| Mouse | Optical |
| Keyboard | 122 Keys |

|  |  |
| --- | --- |
| **SOFTWARE TOOLS** | **MINIMUM REQUIREMENTS** |
| Platform | Windows/Linux/MacOS |
| Operating System | Windows/Linux/MacOS |
| Technology | Machine learning – Python |
| Scripting language | Python |
| IDE | Pycharm & Jupyter notebook |

**FUTURE SCOPE**

Now-a-days trending apps like twitter, facebook, maps, netflix, and many others are networks and these functionalities can be improved by network graph analysis. Friend suggestions, Nearest Person to you, People you may know these all are derived using Network analysis.

And also fields like marketing, stocks, blockchain, Telecom are like networks these can be improvised more using Network analysis. These marketing and other strategies can be done using link prediction , communities, marketing and advertising will be easier

.

**CONCLUSION**

We conclude that using Network Graph Analysis, We can predicts the future node, relation between nodes, divide nodes into communities, and also analysing Network data give you more information in short time.

By using social network analysis techniques to identify information about people, their role in the network, and their relationships can be useful to help sort, aggregate, and filter information in social media.

People can share content directly by sharing links on their social networking pages, posting them on social-sharing websites like reddit, or more passively sharing with social readers that show friends everything a person has looked at. These methods highlight information that a person’s friends have found interesting, and that filter is often very useful for identifying good content. Further user input, like votes up or down on content, can further help sort and filter information shared in this way.

Recommender systems move up a level, aggregating ratings or behavior and using that to personalize suggestions for items that a person might like. Collaborative filtering systems use similarity estimates to show items that people similar to the user like. Social recommender systems replace or enhance the similarity measures with social features, like trust relationships, to recommend items. This leverages social information in several ways, using ratings that users supply and their social connections to highlight interesting items.

Leveraging social information is already effective for finding information, and as the amount of information people encounter online grows, it will be important to develop new methods that incorporate social information and techniques for filtering, sorting, and aggregating content.

**REFERENCES**

* Zhou, D., Bousquet, O., Lal, T. N., Weston, J., & Schölkopf, B. (2004). Learning with local and global consistency. Advances in neural information processing systems, 16(16), 321-328.
* networkx(<http://networkx.lanl.gov/>)
* Link detection Algorithms networkx (<https://www.geeksforgeeks.org/link-prediction-predict-edges-in-a-network-using-networkx/>)
* Node Detection Algorithms (https://networkx.org/documentation/stable/reference/algorithms/node\_classification.html)
* Community Detection Algorithms networkx (<https://networkx.org/documentation/stable/reference/algorithms/community.html#module-networkx.algorithms.community.kernighan_lin>)
* Dataset(<https://github.com/>)