2014

Web Service Network Analysis

Avani Gade
University of Windsor

Web Service Network Analysis

by

Avani Gade

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor.

Windsor, Ontario, Canada

2014

Web Service Network Analysis

by

Avani Gade

APPROVED BY

May 13, 2014

DECLARATION OF ORIGINALITY

ABSTARCT

A Web Service is a software component that is accessible on the Web. Web Services can collaborate to form composite services, which are called Mashups. Services and Mashups interconnect with each other, forming a complex network. This technological network evolves in a large scale without central control. As a new type of software network, there is an urgent need to study its topological properties. This thesis studies the Web Service Network that consists of 4255 primitive web services and mashups collected from ProgrammableWeb.com.

We study various centralities of the network, including degree, betweenness, closeness and pagerank. We find that it is a scale-free network, whose degree distributions follow a power law. We also identify the top web services according to these centrality measures, and demonstrate correlations between them.

DEDICATION

I dedicate this thesis to my family for their constant love and support. Epecially to my dad and mom for teaching me the importance of hardwork and higher education.

To the rest of my family for their encouragement, and to my sisters for all the trust they have in me, and their belief that I can reach my dreams.

Finally, I would also like to thank Siddarth Reddy Musku, Manoj Gajjarapu,Rithvik and all my friends on and off campus for their constant moral support.

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude to Dr.Jianguo Lu, my supervisor, for his steady encouragement, patience and guidance. He read numerous revisions and helped me to make some sense of the confusion, and for enlightening discussions we led through-out my graduate studies. Without his help, the work presented here could not have been possible.

I also wish to express my appreciation to Dr. Subir Bandyopadhyay, School of Computer Science, Dr. Kevin W Li, Odette School of Business and Dr. Richard Frost, School of Computer Science for being in the committee and sharing their valuable time.

Most of all thanks to the Almighty who continues to make the impossible possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Web Service

The nature of the Web has expanded and changed considerably over the years. The Web that was once a repository of texts and images evolved into a provider of services. It now provides various information such as videos, news, music, shopping, flight bookings, business to business applications and many more[1]. Ten years ago the first generation Web or Web 1.0 was seen as a medium of education and communication resource, analogous to a book, as a means of representing the content or means of communicating[2]. The users browsed, read and obtained information and then were directed through a site from a front-page[3]. Web 1.0 was limited in use, seen as only people with high ability in HTML (Hypertext Mark Language) could post content.

The Web 2.0[4] collectively dubbed as new generation web-based technologies, has provided the growth of applications such as Wikis, Blogs, Social Networks, Podcasts, Mashups, etc. that make it easier for users to publish their own content unlike the unnamed Web 1.0[5][6]. Mashups are the personalized applications created by the user with the help of a programmable interface, i.e., Programmable Web[7]. The importance of these Mashups is that they can combine the content in new and suprising ways thereby creating a new webservice entirely or they can also provide a new visualisation to the already available and existing web services[8]. These Web 2.0 technologies have paved the way for the users to create their own web applications using powerful building blocks provided by third parties by enabling them to go beyond static publishing, for even more profound transformation. According to IanDavis[6] described that the Web 2.0 is still emerging as the Web 1.0 took people to information, whereas the Web 2.0 will take the information to the people[6].

The Programmable Web lists the Web APIs and Mashups by date of introduction, and provides a detail of each, thus simply it is a Mashup ecosystem. According to the W3C, i.e., World Wide Web Consortium glossary defined Web Service[9] as a software system that was designed to support the interoperable machine to machine interaction over a network.

The Web Service is defined as an application accessible to other applications over the web. Nevertheless, the definition of the Web Service to a large extent depends on the underlying concepts and technologies and how it has been interpreted[10]. The UDDI (Universal Description, Discovery and Integration) consortium defines the precise definition of the Web Service as “self-contained, modular business applications that have open Internet oriented, standards-based interfaces”. This definition focuses on putting emphasis on the need for being compliant with internet standards. It also requires the service open, i.e., to have a published interface which essentially invoked across the Internet[10].

The Web Service provides the APIs. APIs are the application programming interfaces that the Mashups build on[6]. A Mashup combines data and services provided by third parties through open APIs as well as the internal sources owned by the users[11]. The Mashups are backed personally by a complex ecosystem of Mashup platforms, interconnected data providers and users[12]. The Mashups is created in a way that hides the details of the source applications to offer a seamless experience for the user by integrating data or functionality from one or more sources to create a new application[7].

The Programmable Web also categorizes the APIs and Mashups through providing taxonomy and tags that users can associate with the entries. It offers information on Mashup tools, as the site is user contributed not all APIs and Mashups are indexed. However, the Programmable Web is the most widely used and recognized Mashup directory, and its content is considered the representative of the state of the Mashup ecosystem[12].

An example Web Service Network is as follows

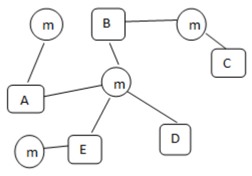


Figure 1.1: An Example Web Service Network

Figure 1.1 depicts an example of a Web Service Network from[7]. The circles represent the Mashups and the rectangles represent the web APIs. If an API is used by a Mashup, there is an edge connecting the API and the Mashup. The line segment connecting them depicts an instance in which the Mashup builds or uses a particular API. In this thesis, we consider the Web Service Network, extracted from Programmable Web, which captures the relationships between Web APIs and Mashups to analyze their topological properties.

1.2 Motivation and Objective

As the Web 2.0 is still evolving, no greater stability is obtained in understanding the growth of the ecosystem. Despite the development of the Web Service Networks and its adamant features, its use is not fully known yet. The key question is how the structure of the Web Service Network looks like and how these Mashups use the APIs. According to [7] the structure of the Web Service Network is a three-tiered network with a layer of Mashups between each API tier. The interesting fact found is that the growth of APIs is slower compared to the growth of Mashups[7].

To understand these Web Service Networks many researchers have worked for permanence and how they evolved. However, mostly their work focused on the software architectures and other dynamic programming languages like Java, Ruby, JDK, Eclipse, etc. confining to private network as they could not be easily integrated with other languages and vice versa due to the interpretations of standards, different standard versions, encodings, etc. Nevertheless, the distributed programming shifted from private to public internet and from using the private and controlled services to increasingly use publicly available Web services[13]. It is critical to understand, and there is an urgent need to study these networks. Despite many adopting the Web Service Network concept and perceiving its importance, it is still challenging and much effort remains before seeing the Mashup applications in a mature stage[14].

For the growth and success of the ecosystem it is critical to find the key players in the network, as they have a faster access to information and resources and to centrally transfer the information to other components. Many researchers have suggested simple rules describing the behavior of individuals in the system, leading to a unique pattern to the entire system [9]. The interaction between the entities became a significant cause of innovation and these studies initiated to use the network analysis to explore the relationship between the entities[15]. Thus, this analysis helps in designing efficient networks. Therefore, the key players in the network play an important role in the development of the network, and in the context of the Web Service Network these central nodes play an important role for the users to create their Mashups by using the influential web APIs for their success and growth.

The main observation in this thesis is to find the pattern of the network to verify if they follow the so-called Power law. According to [16] "when the probability of measuring a particular quantity having some value varies inversely as the power of that value, then the observed quantity is said to follow the power law". The most common property of large networks is the fact that they follow a scale-free Power law distribution. The main result of the occurrence of Power laws is the expansion of the network continuously and the addition of new vertices and these vertices in turn connecting to already well-connected sites[9].

1.3 Problem Definition

The main focus of this thesis is to analyze a new type of software network i.e., web service network and identifying the central players in the network based on the centrality to calculate the topological properties that include Degree, Betweenness, Closeness and PageRank centralities.

The aim is to also calculate the correlation between these centralities to calculate the linear dependency between them and how each of them is influential in the network.

1.4 Thesis Contribution

The main contribution of this thesis includes,

- To find the top Web Services based on the centralities.
- Identify if the Web Service Network follows a Power law function.
- Analyzing the Web Service Network results to see if it is a scale free network.
- Comparison between the earlier work and how it differs.
- Visual and Graphical analysis.
- To analyze the structure and layout of the Web Service Network.
- Different programming techniques with implementation using (Matlab Programming Language) to calculate the centralities and the correlation.
- The aim of this thesis is also to find if the Web Service Network is a bipartite network, which would be of great interest analyzing and drawing conclusions.

1.5 Thesis Organization

This thesis consists of six chapters. A small survey of earlier works has been discussed in Chapter two. Chapter three gives an overview of the Web Service Network, describes the network centrality measurements with the results of the top ten nodes of the Web Service Network followed by a detailed description of PageRank using variations in calculating the rank. Chapter four describes the correlation between the network centralities followed by the top 25 nodes of the Web Service Network and the top highest nodes for all the centralities. Finally, Chapter five draws the conclusion of this thesis with limitations, difficulties and some future work.

Chapter 2

Background Work

Valverde et al.[17] was the first to present a new complex network approach to software engineering based on the advances of complex networks[18, 19, 20] and the first to study the software networks. In this approach, a software network is a class graph that is a directed graph where D = (V, L) that consists of the set V of classes where every class maps to a single node in the class graph and the set of relationships in L.

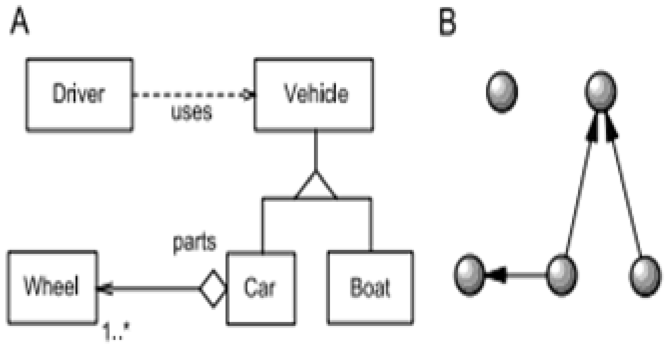


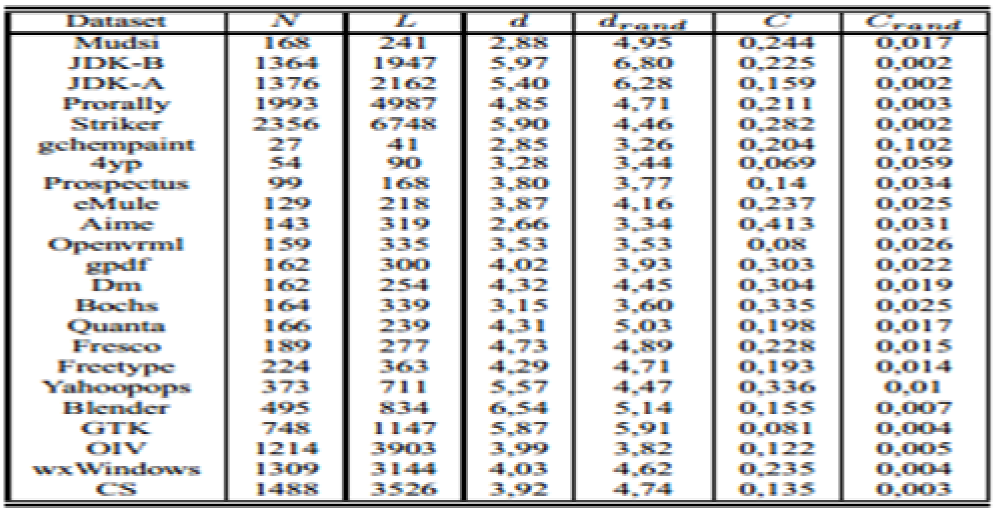
Figure 2.1: A Sample Class Graph
(A) A sample UML class graph (B) Its equivalent class graph

The figure 2.1 has been produced by [17]. In this approach, the nodes represent the software entities, i.e., classes and/or methods and the edges represent the static relations between them namely the inheritance and collaboration between the classes.

They conducted experiments on a large collection of object-oriented software's written in C++ and Java to study the scale free and small world behavior of OO software's in many real systems and random graphs to suggest that they are the universal features of software designs. In addition, they also conducted experiments to verify if the degree distribution follows a Power law.

Based on the experiments the authors found universal network patterns in large OO software's and the topological measurements of the experiments conducted are described in figure 2.2.

GRAPH MEASUREMENTS



Dataset	N	L	d	d_{rand}	C	C_{rand}
Mudsi	168	241	2,88	4,95	0,244	0,017
JDK-B	1364	1947	5,97	6,80	0,225	0,002
JDK-A	1376	2162	5,40	6,28	0,159	0,002
Prorally	1993	4987	4,85	4,71	0,211	0,003
Striker	2356	6748	5,90	4,46	0,282	0,002
gchempaint	27	41	2,85	3,26	0,204	0,102
4yp	54	90	3,28	3,44	0,069	0,059
Prospectus	99	168	3,80	3,77	0,14	0,034
eMule	129	218	3,87	4,16	0,237	0,025
Aime	143	319	2,66	3,34	0,413	0,031
Openvrml	159	335	3,53	3,53	0,08	0,026
gpdf	162	300	4,02	3,93	0,303	0,022
Dm	162	254	4,32	4,45	0,304	0,019
Bochs	164	339	3,15	3,60	0,335	0,025
Quanta	166	239	4,31	5,03	0,198	0,017
Fresco	189	277	4,73	4,89	0,228	0,015
Freetype	224	363	4,29	4,71	0,193	0,014
Yahoopops	373	711	5,57	4,47	0,336	0,01
Blender	495	834	6,54	5,14	0,155	0,007
GTK	748	1147	5,87	5,91	0,081	0,004
OIV	1214	3903	3,99	3,82	0,122	0,005
wxWindows	1309	3144	4,03	4,62	0,235	0,004
CS	1488	3526	3,92	4,74	0,135	0,003

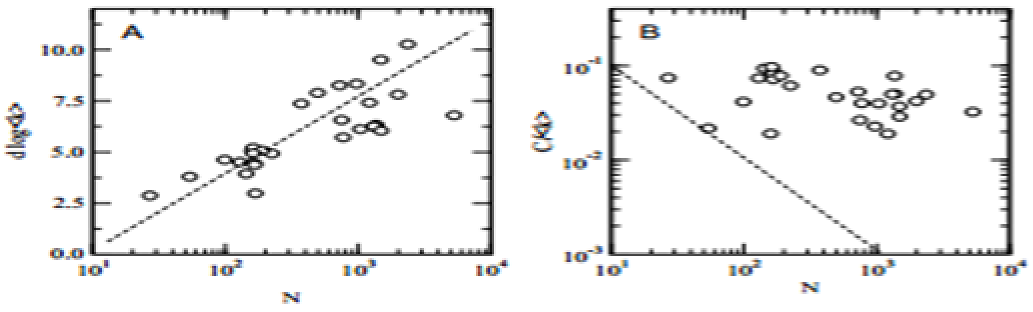
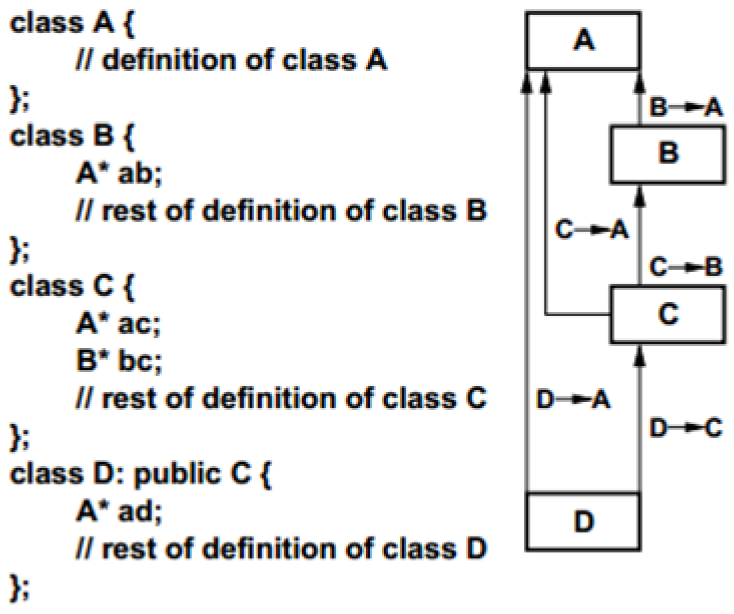


Figure 2.2: Validating the small world from Log-log plot of the class graph

(A) Average path length vs. the Class Graph size (B) Normalized Clustering followed by the random counter parts where N is the number of classes, L is the number of links, d is the average path length, drand is the average path length in random graphs, C is the Clustering Coefficient of the class graph and Crand is the Clustering Coefficient of the random graphs. The figure 2.2 is produced by [17].

Based on the above analysis they distinguished that the class graphs are much more clustered than their random counter parts and the Clustering Coefficient of the Class graphs is well above the random expectation while the average path length is rather small which proves the concept of Small world behavior. They also found the scale free behavior where a very few classes take part in various relations and the majority of classes have only one or two relationships. Even though this approach verifies the behaviors of software networks and it found universal patterns in a large collection of object-oriented softwares, it fails to take the edge direction and the centrality measurement into consideration.

Myers et al.[21] proposed a refracting based model on software evolution to examine the software collaboration graphs contained within several open source software systems that tend to show a scale free and small world behavior similar to [17] but taking the edge directionality into consideration. In his approach, the author examined the collaboration networks associated with six different open-source software systems. Collaboration networks in his research refer to graphs which are decomposed into two subgraphs. Those graphs are the inheritance graph and the aggregation graph.



```
class A {
      // definition of class A
};
class B {
      A* ab;
      // rest of definition of class B
};
class C {
      A* ac;
      B* bc;
      // rest of definition of class C
};
class D: public C {
      A* ad;
      // rest of definition of class D
};
```

Figure 2.3: A simple class collaboration graph representing the relationship among the classes A, B, C, D.

The figure 2.3 was produced by [21]. The class collaboration in his work is defined to include the interaction of classes both through inheritance, i.e., where one class is defined as a subclass of other and through aggregation i.e., where one class is defined to hold an instance of the other class[21]. According to [21] the nodes represent the classes and the edges represent the directed collaborations between the classes.

The author conducted experiments to examine the WCC (Weakly Connected Components) and SCC (Strongly Connected Components) in his studied collaboration networks associated with six different open-source software systems and to verify the behaviors and features of the networks through his proposed model.

According to [21] based on his experiments concluded that all six systems consist of a single dominant WCC comprising a large fraction of the total nodes in the system and a very small remaining WCC, and conversely only few a nodes belonging to SCC. An interesting part of his work is the symmetry between the in-degree and out-degree which follows a Power law, concluding that the hierarchical nature of software design has an impact on the overall network topology.

This method, however, proves to be efficient in taking certain topological properties into consideration, but fails to explain the topological properties from the view of software engineering and to uncover precise implications of that process for large-scale network.

Ichii et al.[22] proposed a new approach to explore the component graphs, i.e., statistical analysis based on OO software's as single software systems and multiple software systems which have never been studied before. A single software systems composed of the components that are acquired by analyzing the source files which have been retrieved from the distribution packages and multiple software system is composed of the components that are acquired by analyzing the source files which have been retrieved from the distribution packages and/or software repositories. The focus of his approach is to find if the component graphs in-degree and out-degree followed the Power law.

According to [22] his research of a software component graph, referred as component graph, is a graph where the node represents the component and an edge represents the use relation between the components.

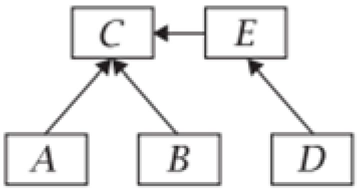


Figure 2.4: An example of Component Graph

The figure 2.4 has been produced from [22]. A component in his work refers to a Java interface or a Java class extracted from the source files and the use relation represents the User Interface, that extends, implements, declares a variable, instantiated, calls, or if an interface refers to a field value of a class or an interface.

The authors conducted experiments to verify if the In-degree and Out-degree followed a Power law[22]-

- (In-degree and Out-degree of a component graph) Based on the experiment 1, the Power law is followed by the in-degree distribution almost ideally and not for the out-degree distribution and the distribution reaches highest in the range of small values while a straight line is identified in the range of larger values.
- (in-degree and out-degree distributions of a component graph for multiple software systems) Based on experiment 2, with the component graphs for multiple software systems: The the Power Law is followed by the in-degree distribution and not the out-degree distribution.
- (In and out-degree distributions of subgraph of a component graph for software systems) based on experiment 3, the subsets whose components are

picked out based on a keyword has similar characteristic with the superset. The Power law is followed by in-degree distribution with similar parameters and the the Power law is partially followed by the out-degree distribution.

- (What aspects of components contribute to the Power Law (or non-Power Law) at the In-degree and/or Out-degree distribution of a component graph) Based on experiment 4, the In-degree relates to the roles of the components, but it has a low correlation with the size, the complexity, and the cohesion. They also found that, the out-degree has a higher correlation with the size and the complexity of the component[22].

Based on their analysis, they concluded that there was an asymmetry between the in-degree and out-degree as the in-degree followed a Power law but, not the out-degree. The authors draw conclusions that the in-degree depends on the role of each component and the out-degree depends on the size and the complexity of each component.

However, this approach failed to explain why some software's follow the Power law and some software's do not. It also failed to consider the topological measurements to find the important components in the graph.

Potanin et al.[23] examined the graphs formed by OO programs written in a variety of languages and proving them to be scale free networks. The author also proposes that his approach helps to optimize language runtime systems, improve the design of future of OO languages and re-examine innovative approaches to software design. The object graph in his work refers to the object instances created by a program and the links between them is the skeleton of the execution of that program.

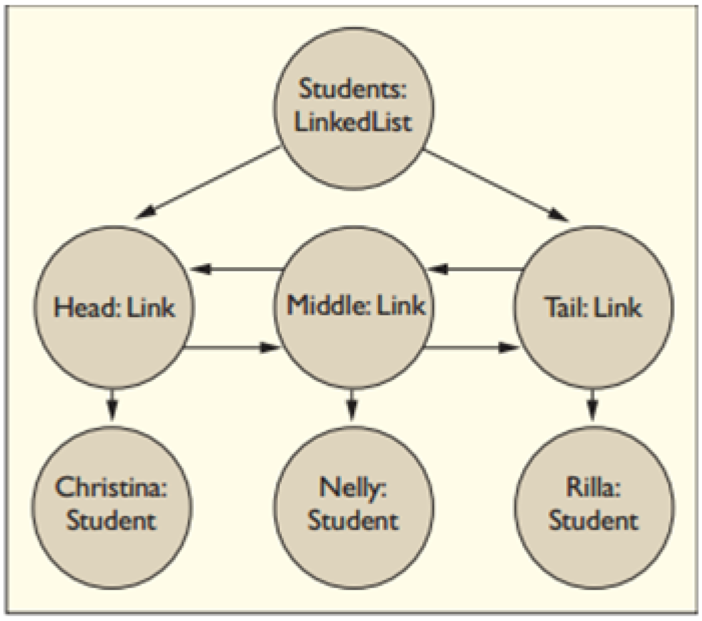


Figure 2.5: A simple object graph of a Linked List

The figure 2.5 has been produced from [23]. In his work, each link object has two references to other link objects, except for the head and tail of the list[23]. The node and edge represent an object, as the graph grows and changes as the program runs after every assignment statement to an object field, may create, change or remove an edge in the graph.

According to [23] conducted experiments to verify if the graphs formed by OO programs written in different languages follow the Power Law and prove to be a scale free network.

The authors claim that they have found symmetry between the in-degree and out-degree and they tend to follow a Power Law. However, they failed to take topological properties into consideration.

The above approaches are all different from each other as [17] failed to take the direction of an edge into consideration[21]. He also failed to explain certain implications in large-scale networks, and this work differs from [22] as in [21] there is symmetry between the In-degree and Out-degree, but in [22] there is no symmetry between the In-degree and Out-degree. This is caused by the difference in their definitions of the component graph particulary the difference between the use-relation as an edge[22][21]. Similarly, there is the difference between the works of [23] and [22] as in [23] component graph analyses a dynamic analysis of the software system in operation, i.e., linked list where as [21] is the static analysis of software system. Despite much work done by many authors they failed to explain why some software follows the Power law in some applications and metrics and some do not.

However, all the proposed models and approaches were well-known for analyzing the software component graphs. These methods were unique in verifying and studying the structures of the network and other studies, mostly related or taken into account these methods depending on the definition of the software component. With regard to the earlier work done by the researchers this thesis differs in a very interesting way, i.e., studying the new form of Web Service Network, analyzing the network centralities and their correlations despite most of the earlier work is related to OO programming languages and how these software' tend to follow the Power law based on only the Degree centrality. As most of the evidence of Power laws in software was studied only at the microscopic level, i.e., at the level of method calls or class references this is an interesting distinctness in this thesis.

Chapter 3

Network Centrality Measurements

3.1 Overview

The major companies like Google, E-Bay, Amazon, etc., have provided interfaces to many of their services at little or no cost at all, to attract individuals or businesses to develop their composite applications with new functionalities. A well-known example of the web API is Google maps. It generates maps for a given location, and its output is integrated with other data and services into Mashups. These Mashups allow the quick creation of custom applications as they often have a short span and are created for only a specific group of users and needs [12]. Each new Mashup then attracts its own set of users additionally increasing the market reach of the API providers[24].

The Web Service Network graph G in this thesis is a directed graph extracted from Programmable Web, G= (V1 ∪ V2, E) with two sets of nodes, V1, representing the web APIs, V2, representing the Mashups, and E being the set of edges or links between the nodes. Mashup nodes are only connected to API nodes. For example, if a Mashup combines with Google maps and Twitter APIs, the graph will contain the edges-(Mashup, Google maps API) and (Mashup, Twitter API). The total number of nodes in the graph is N=| V1 ∪ V2|. In this thesis, the Web Service Network graph consists of 4255 nodes and 7193 edges.

Summary of the Web Service Network

Nodes	4255
Edges	7193
Number of WCC	157
Number of SCC	4255
Diameter Directed	Infinite
Diameter Undirected	14
Average length directed	Infinite
Average length undirected	4.648
Max Indegree(directed)	453
Min Indegree(directed)	0
Max Outdegree(directed)	65
Min Outdegree(directed)	0
Clustering Coefficient	0

Table 3.1: Network Centrality Measures

3.2 Visualization

The ability to visualize and handle a Web service Network is very crucial. Visualization refers to the way of representing the structural information as diagrams of abstract graphs and networks. There are many visualization tools available today. The success depends on how the tool is selected and how the structure is displayed. Taking all these factors into consideration this thesis uses Graphviz open source graph visualization software[25]. Graphviz has various features to represent the layout of the graph the layouts include dot, neato, fdp, sfdp, twopi and circo. The layout can be selected based on the appearance of the structure of the graph. The description of the layouts is as follows[25].

- dot-In Graphviz this is a default layout. It gives the hierarchical or layered structure for directed graphs if the edges have directionality.
- neato-This layout can be used if the graph is not too large and is undirected. It uses a spring model layout and it attempts to minimize global energy function, similar to statistical multidimensional scaling[26].
- fdp-It is similar to note as the “spring model” layout. The major difference however, reduces forces rather than working with energy[27].
- sfdp-This layout may be used for larger and undirected graphs. It is a multiscale version of the fdp and produces layouts in a reasonably short time.
- twopi-It draws graphs using a radial layout[28, 29], where nodes are placed in centric circles depending on their distance from a given root node.
- circo-It draws using a circular layout[30]. This tool is suitable for certain diagrams of multiple cyclic structures or biconnected components like certain telecommunications networks.

The figure 3.1 shows the Web Service Network. The red dots represent the Web API's and the blue dots represent the mashups. The figure was produced using the Graphviz visualization tool with the twopi layout.

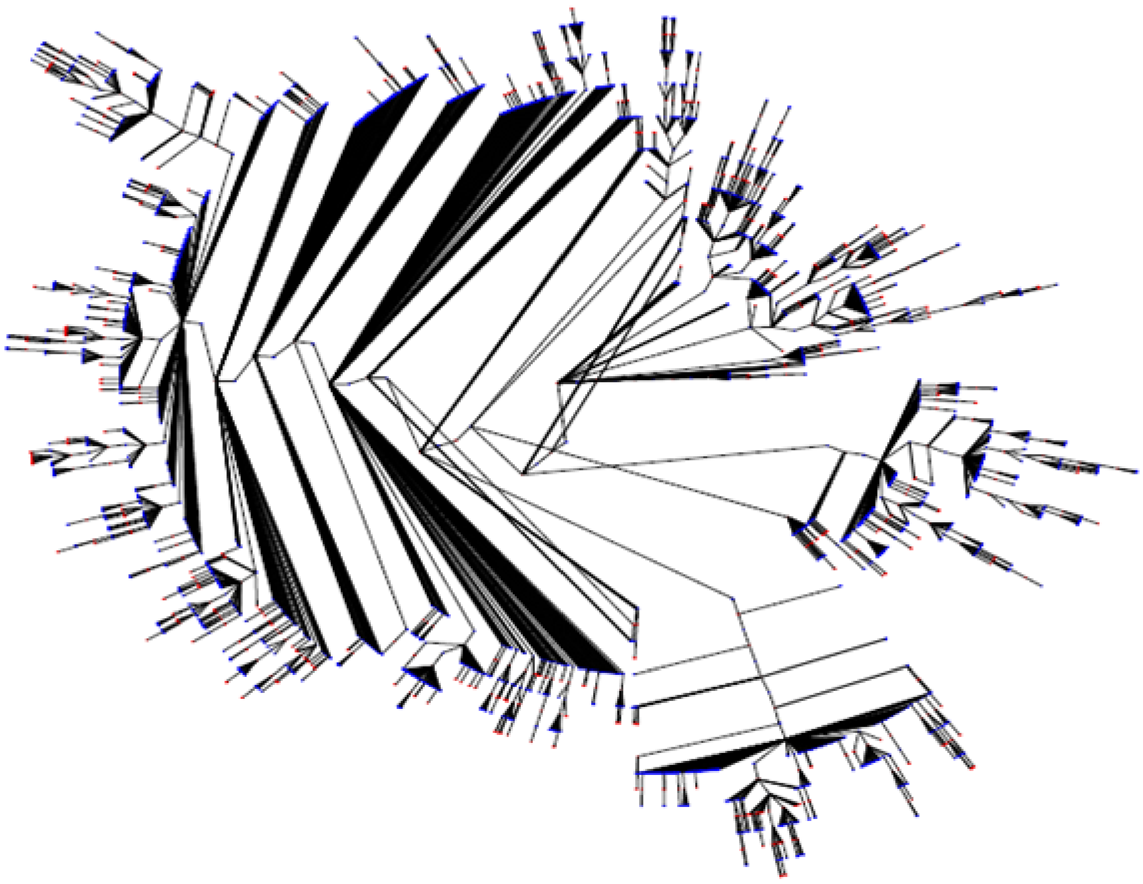


Figure 3.1: The representation of the Web Service Network

Finding the network centralities is an aggregate phenomenon in knowing the frequency of how the API is used in some Mashups. This way helps understand why certain APIs are vastly used and more popular than other APIs[24].

The network centralities or the topological properties contribute an important role in any network. In this thesis, we describe the properties of the network on two levels, global metrics and the actor's or a node individual property. The global metrics describe the characteristics of the network as a whole, such as the diameter, node distance, average length, clustering coefficient, etc., and the actor's individual properties include the analysis of the individual properties of the actors in the network which includes the actor's status in the network.

The node's individual properties are usually expressed in terms of its centrality. The nodes that are centrally located have a more vital and important position in the network and a node has faster access to information and resources. Thus, centrality indicates the extent to which a node's strategic position is defined by its key ties to other nodes in the network [31]. However, if a node is removed, would communication be disrupted is another problem this thesis would be focusing on.

Centrality measures indicate the importance of the Web Services in the Web Service Network. The types of centrality measurements in this research are degree centrality, betweenness, closeness and pagerank. Each of these different centrality measurements will identify the central nodes in the network based on the network position which will be examined by the measurement of each node [32].

3.3 Degree Centrality

The degree of a node is determined by the number of links or ties it has. In-degree is equal to the number of incoming edges of a node. In a network, it is natural for some key nodes to supply services to many other nodes, while most other nodes supply services only to few other nodes if any. Out-degree of a node represents a completely different property, with respect to the in-degree, it is the number of nodes used by a given node. It is merely the number of outgoing edges of a node.

According to [16] the degree distribution is the function P of a network can be calculated by fraction of the number of vertices having degree i to total number of degrees.

$$P(i) = \frac{Number\ of\ vertices\ having\ a\ degree\ i}{Total\ number\ of\ degrees} \tag{3.1}$$

3.3.1 Representation of Matrices

A way to represent a network is by a matrix. A graph can be represented as an adjacency matrix in which each element of this matrix relates to whether there is a link between node i and node j or not. So the expected size of this matrix is square of node numbers. If it assumes that n is the node number so matrix size equals n*n. The adjacency matrix A_{ij} is one if there is a link between i and j otherwise this element equals zero. The best way to represent a matrix is to list the non zero cells in the matrix[33]. A vertex that has no arrow is known as an isolated vertex and it has a degree 0. Edges that have the exact same starting and ending point are called loop edges.

The algorithm used to calculate the in-degree and out-degree is given in algorithm 1.

Algorithm 1 In-degree and Out-degree

```
Input: Matrix
Output: In-degree and Out-degree
for all unique edges do
    Matrix(Source(i, 1), Destination(i, 1))=1
    In-degree=sum (Matrix)
    Out-degree=sum (Matrix')
end for
```

In algorithm 1, i, represents the unique edges, the source and destination represent the Web APIs and Mashups.

A simple example to compute the degree centrality is as follows.

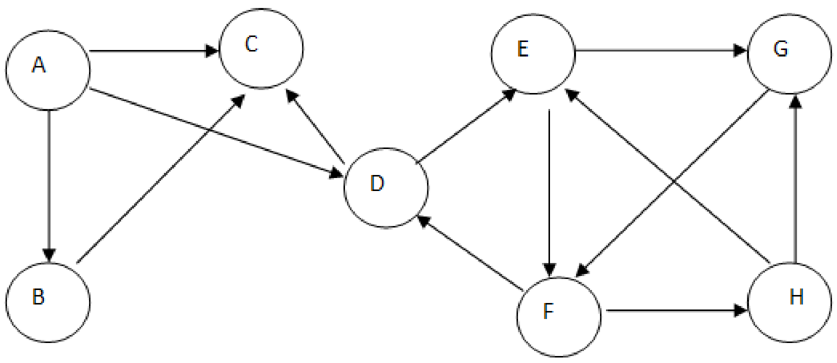


Figure 3.2: Simple example to compute the Degree Centrality

According to [34]the degree centrality of a vertex is defined as,

$$d_i = \sum_j A_{ij} \tag{3.2}$$

Where d_i is the degree centrality of a vertex i, A is the adjacency matrix and A_{ij} =1 if there is a link between node i and j, A_{ij} =0 if there is no link.

The sum of the rows of the matrix represent the in-degree of the figure 3.2 and the sum of column represents the out-degree. The degree of the node from the adjacency matrix is the sum of the row i and column j for each node.

$$\begin{vmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{vmatrix}$$

From figure 3.2 we can see node A has no incoming but has three outgoing edges. The in-degree of node A is 0 and out-degree is 3. The degree of node A is given by the summation of the in-degree and out-degree, i.e., 0+3=3. Similarly the degree of all the nodes can be computed. As the web API-Mashup Web Service Network is large, to calculate the topological properties this thesis uses MATLAB programming language to analyze the Web Service Network.

For analysis purpose, we first need to represent the Web Service Network in a format that allows the mathematical computation. Web Service Network in a text file format is converted to a matrix using Matlab in a way that each row and column represent the source, i.e., Web API and the destination, i.e., Mashup of the Web Service Network. And the value of row i and column j is 1 if there is a link between the source and destination or 0 otherwise. Having the matrix of the network enables us to calculate the in-degree and the out-degree by summing the corresponding column or row respectively. Once the in-degree and out-degree are calculated it is easy to find the maximum and minimum in-degree and out-degrees.

We can see from figure 3.3 that the in-degree distribution seems to be crowded and out-degree distributions are however less crowded, this is because the variation of out-degree among the Web API's was limited.

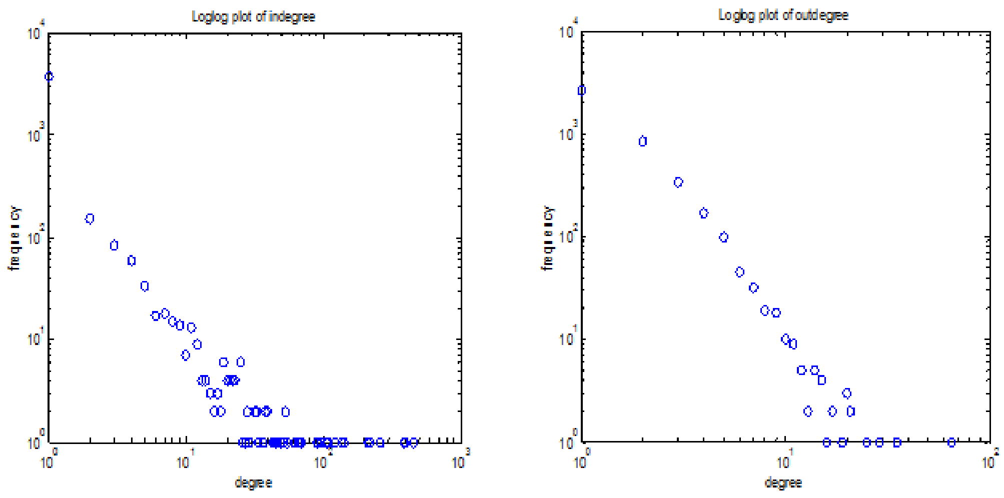


Figure 3.3: In-degree and Out-degree distribution for the Web Service Network

Figure 3.3 shows a log-log plot, the distributions of in-degree and out-degree with the base 10. The x-axis represents the degree and y-axis shows the frequency of certain degree.

Figure 3.4 gives the overall degree distribution for the Web Service Network.

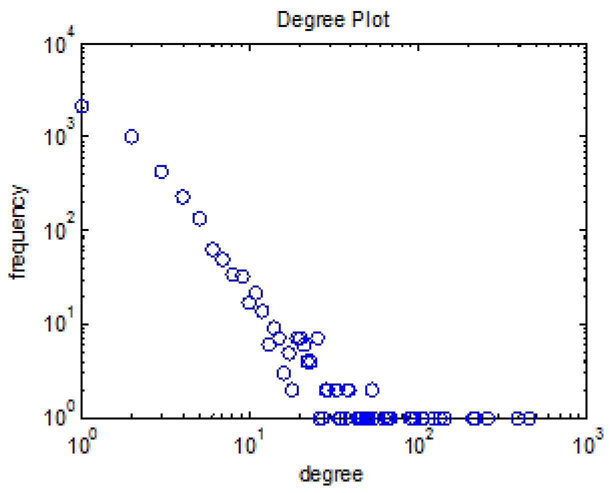


Figure 3.4: Degree distribution for the Web Service Network

Figure 3.3 shows that both in-degree and out-degree distributions follow the Power law. The best way to analyze if the network exhibits the Power law is by ranking the vertices according to their degree and plotting them on a logarithmic axis to ascertain whether it has a straight line. Answers to what a Power law is, and why do they occur is followed in the next section.

3.3.2 Power law

The Power law has attracted attention over the years for its mathematical and sometimes physical consequences[35]. Power law implies that small occurrences are extremely common, since large instances are extremely rare. This regularity or 'law' is sometimes also referred to as Zipf's law and sometimes Pareto law or the 80-20 rule i.e.; roughly 80% of the effects come from 20% of the causes [35]. According to many sources and evidences Power law distributions occur in an extremely diverse range of phenomena. They include the population of cities, sizes of earthquakes [36], moon craters [37], solar flares [38], computer files [39], the frequency of occurrence of personal names in most cultures [40], people's annual incomes [41], the number of citations received by papers [42], the constancy of the use of words in any human language [43] and many more [16].

A network is called scale free if its degree distribution follows a Power law function. This means a high number of nodes have a small degree and only a few nodes have a high degree. This would mean that many APIs are used by few Mashups and a small number of APIs are used by many Mashups. This is the case with our Web Service Network from figure 3.3, so we can conclude that the Web Service Network follows a Power law and corresponds to a scale free network. Identifying Power laws in man-made systems or either in nature can be tricky and challenging [16].

There are many ways of generating the Power laws. A "log-log" plot provides a quick and a standard way to identify if the data exhibits an approximate Power law as it plots a straight line on log axes. But how can the Power law function be calculated is the crucial question. The best way to check if it's approximately a Power law $\frac{1}{i^\alpha}$ is, for some i, estimating the exponent α.

The notion of Power law probability distribution from equation 3.1 according to [16]is expressed in the form,

$$P(i) = \frac{C}{i^\alpha} \tag{3.3}$$

P(i) is the probability that i occurs, C and α are constant, α is usually expressed as the exponent. The equation 3.3 can be then expressed as,

$$P(i) = Ci^{-\alpha} \tag{3.4}$$

The equation 3.4 can be expressed further by taking logarithms on both sides as,

$$logP(i) = -\alpha\, logi + logC \tag{3.5}$$

If we have a Power law relationship and we plot log P (i) as a function of log i then the equation above plots a straight line. Here, – α is the slope and log C is the y intercept. Such a log log plot is a quick way to verify if the data exhibits a Power law relationship. From the straight line of the plot one can read the exponent from the slope.

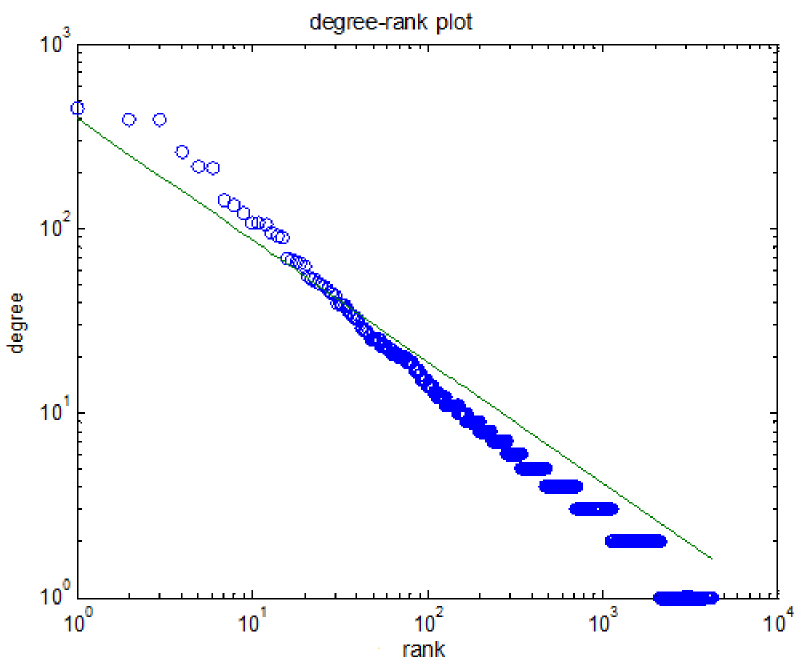


Figure 3.5: Power law fit to the degree distribution

Figure 3.5 represents the Power law fit for degree distribution with an exponent α being 1.66. The x-axis represents the rank, it is the decreasing order of the degree of the vertices and the y-axis represents the degree of the vertices. We can also try to fit the data of in-degree and out-degree to verify the Power law. Let us look at the plots regarding the mentioned. In the in-degree plot in figure 3.3, the tail of the in-degree seems to have a lot of noise we can try to smoothen it by plotting it on a logarithmic axis and a power law fit.

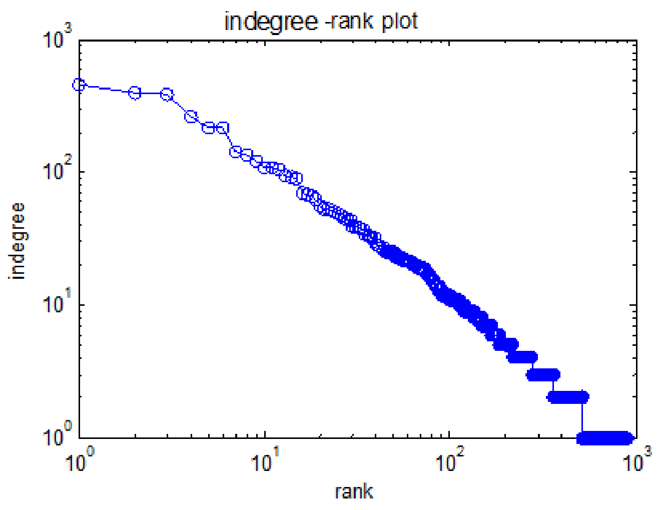


Figure 3.6: Rank plot of in-degree

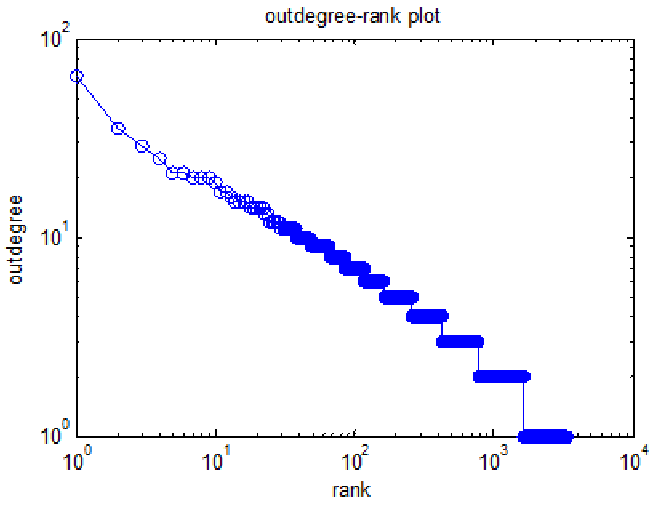
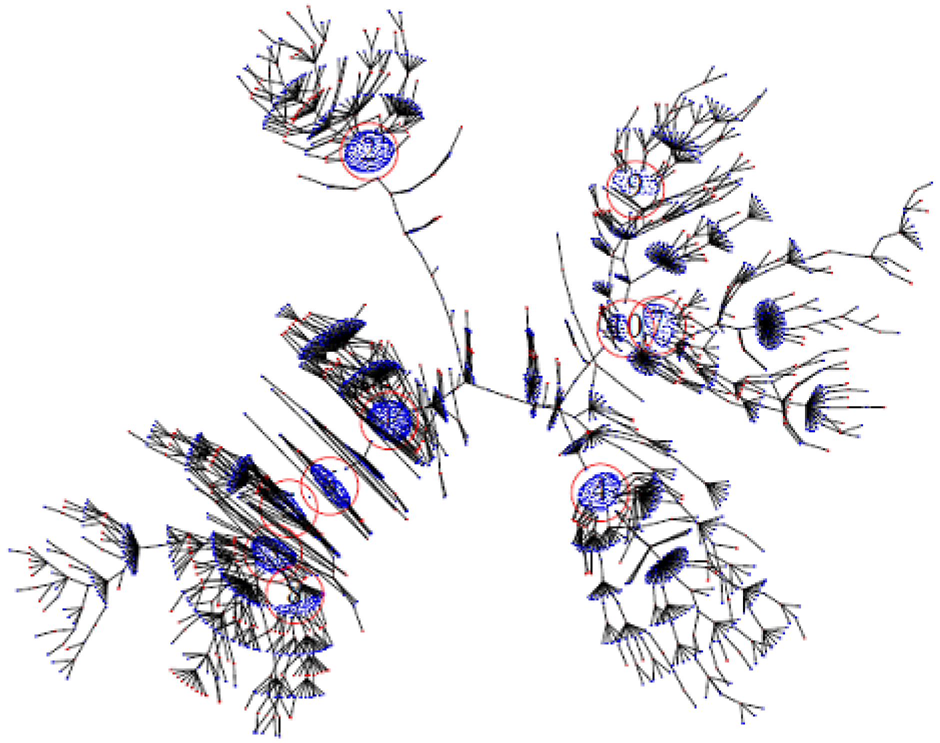


Figure 3.7: Rank plot of out-degree

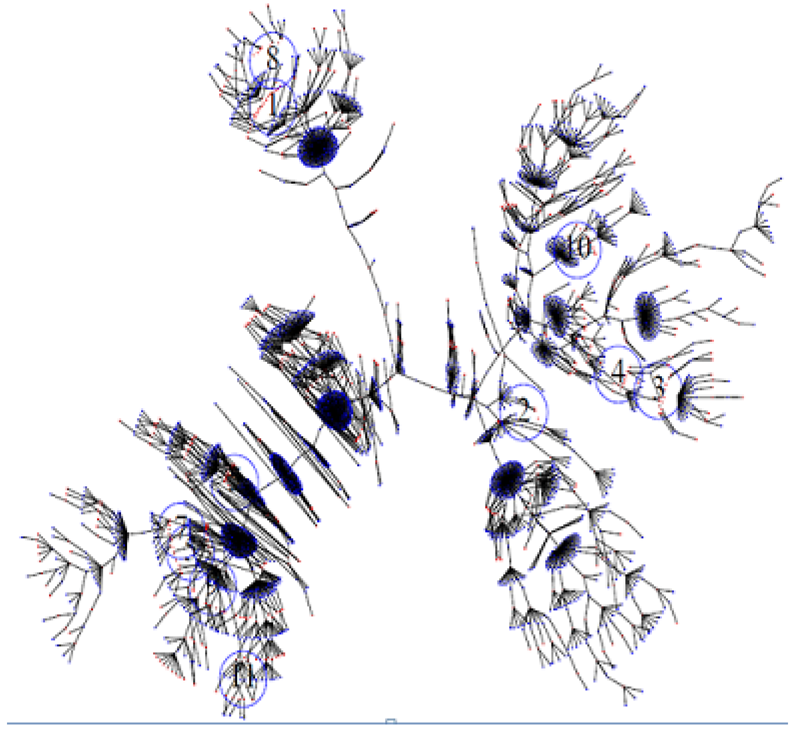
The tail of the in-degree and out-degree cumulative distribution is shown in figure 3.6 and 3.7. Note how the measure on the lower right corresponds to, and whose services are used by almost every other node, and this also corresponds to a straight line. The power law fit for indegree has an expoenet of 1.66 and for out-degree is 2. If the slope of the network is high, the number of nodes having a high degree is smaller than the number of nodes having a low degree. Therefore, we can conclude that both the in-degree and out-degree follow a Power law for the Web Service Network based on the shape of the plot. The Web Service Network is also a scale free network. In the context of the Mashup ecosystem, this would imply that a large number of APIs are used by few Mashups and a small number of APIs are used by many Mashups.

The top ten nodes with the highest in-degree for the Web Service Network are shown in figure 3.8 with their value of in-degree and figure 3.9 represents some of the nodes with the highest out-degree. We can observe that the nodes with the high out-degree are not similar to that of the in-degree.



Node Name	Indegree value
1. Twitter	453
2. YouTube	393
3. Flickr	388
4. Amazon	262
5. Facebook	218
6. Twilio	213
7. EBay	143
8. LastFM	135
9. Google	122
10. del_icio_us	108

Figure 3.8: Top ten nodes with highest In-degree



Node Name	Outdegree value
1. Tagbulb	65
2. Headup	35
3. Pixelpipe	29
4. Gawk_com	25
5. G4ng	21
6. Connector Local	21
7. MapsShownTo Me	20
8. Pageflakes	20
9. What'sPublic	20
10. TheMusicFeed	19

Figure 3.9: Top ten nodes with highest Out-degree

3.4 Betweenness Centrality

The position of a Web API in a Web Service Network is characterized by the shortest path based centrality, which brings forth more information than the degree centrality . The fraction of shortest paths across a node is called Betweenness Centrality [44]. The shortest paths between each pair of nodes, and the number of times that each node appeared in the shortest paths can be computed easily. Betweenness centrality provides more topological information than rest, which is necessary in the analysis of Web Service Network. APIs with high betweenness centrality can be seen as “bridges” between clusters of APIs. The betweenness centrality is considered as an important measure for determining the control of a node on the communication between other nodes in a network[44].

For the graph G (V, E), let i and j be two nodes that are in V. The betweenness centrality for the node $v \in V$ according to [44] is defined as

$$C_B(v) = \sum_{i \neq v \neq j \in V} \frac{\sigma_{ij}(v)}{\sigma_{ij}} \tag{3.6}$$

where σ_{ij} is the number of shortest paths connecting i to j and $\sigma_{ij}(v)$ is the number of shortest paths connecting i to j passing through the node v.

To calculate the centralities such as betweenness, closeness, diameter and average path length the Web Service Network is considered, as undirected. We use the Dijakstra’s shortest path algorithm in Matlab, for the Web Service Network, to calculate the shortest path between all pairs of nodes.

Algorithm 2 Dijkstra’s Algorithm to calculate the shortest path between each pair of nodes

```
Input: The Web Service Network graph represented as a matrix
Queue = Keeps the list of nodes to be visited
Neighbor= Keeps the list of all the neighbors for each node
Output: Matrix Path and the Matrix Distance
for $i$=1:Number of Nodes do
  distance(i,i)=0
  Put i in Q=Queue
  while Q is not empty do
    Pick a node from Q
    for all neighbors of node i do
      if the neighbor has not been visited before then
        distance(i,neighbor(Q))=distance(i,Q)+1
        path(i,neighborqueue)=Q;
        add to Q=neighbor(Q);
      end if
    end for
  end while
end for
```

The Matrix Path stores the path between the pairs of nodes, and Matrix distance stores the length of the shortest path between each pair of nodes; and if there is no path between a pair of nodes it is equal to -1. The Algorithm 2 returns the distance and the path by calculating the shortest paths between each pair of nodes.

Let us consider a simple example to calculate the betweenness centrality.

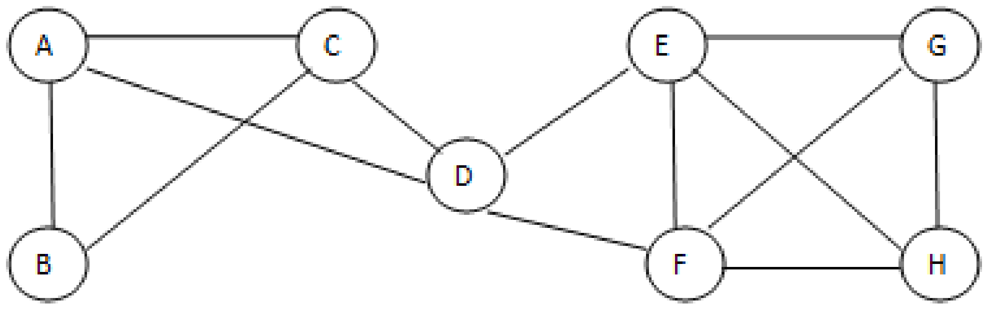


Figure 3.10: A simple example for betweenness centrality

For networks like figure 3.10 it is easy to calculate the betweenness centrality of the nodes. Vertices that occur on many shortest paths between other vertices have higher betweenness than those that do not. For example, calculating the betweenness centrality for node D of figure 3.10. The matrix for figure 3.10 is as follows.

Node	A	B	C	D	E	F	G	H
A	0	1	1	1	0	0	0	0
B	1	0	1	0	0	0	0	0
C	1	1	0	1	0	0	0	0
D	1	0	1	0	1	1	0	0
E	0	0	0	1	0	1	1	1
F	0	0	0	1	1	0	1	1
G	0	0	0	0	1	1	0	1
H	0	0	0	0	1	1	1	0

Table 3.2: Matrix for Betweenness Centrality example

For calculating the betweenness centrality let us list the number of shortest paths first. The number of shortest paths from A to E from figure 3.10 is 1. Similarly the number of shortest paths from A to F from figure 3.10 is 1 and the same follows for all other nodes. Then, we list the number of shortest paths that path through node D. For example, the number of shortest paths from A to E that pass through D is 1 and the number of shortest paths from B to E passing through D is 1 and so on. We then calculate the betweenness centrality for node D as shown in table 3.3.

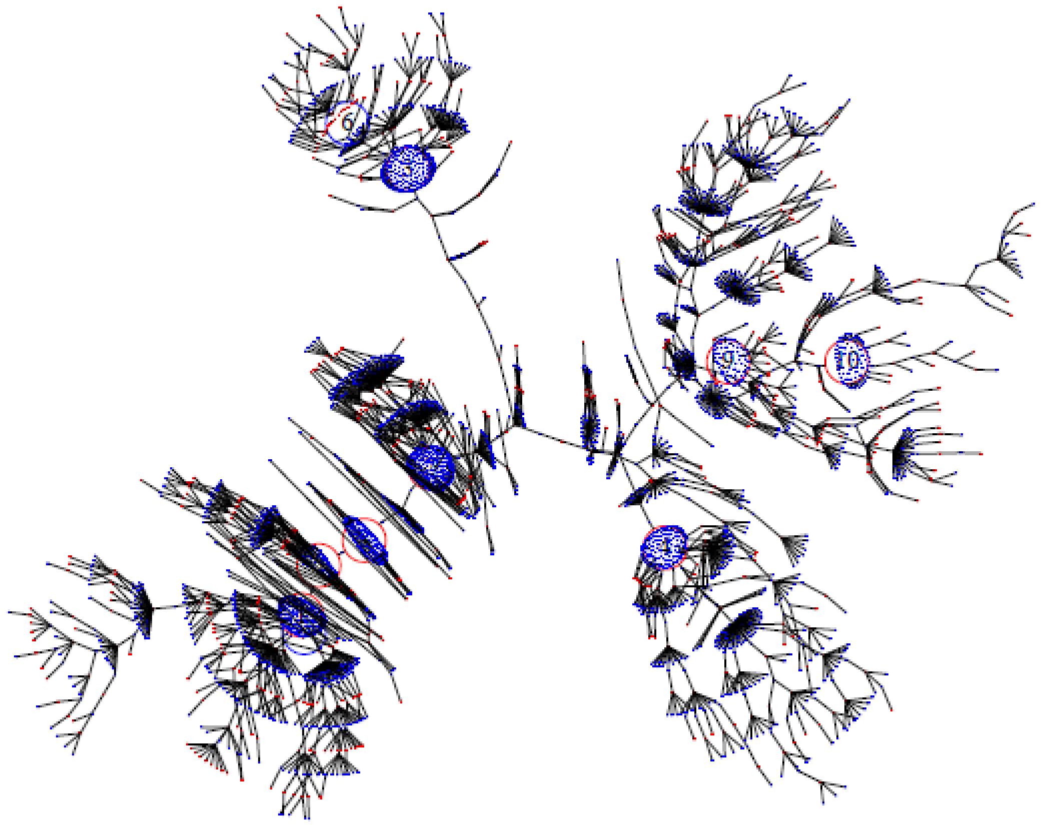
Nodes	σ_{ij}	$\sigma_{ij}(D)$	$\frac{\sigma_{ij}(D)}{\sigma_{ij}}$
(A, B)	1	0	0
(A, C)	1	0	0
(A, E)	1	1	1
(A, F)	1	1	1
(A, G)	1	1	1
(A, H)	1	1	1
(B, C)	1	0	0
(B, E)	1	1	1
(B, F)	1	1	1
(B, G)	1	1	1
(B, H)	1	1	1
(C, E)	1	1	1
(C, F)	1	1	1
(C, G)	1	1	1
(C, H)	1	1	1
(E, F)	1	0	0
(E, G)	1	0	0
(E, H)	1	0	0
(F, G)	1	0	0
(F, H)	1	0	0
(G, H)	1	0	0

Table 3.3: Calculating Betweenness Centrality

From Table 3.3 the betweenness centrality for node D

$$C_B(D) = 12$$

The betweenness centrality is interesting because they control the flow in a network. From figure 3.10 node D is important as it controls the overall flow of the network. Once node D is disconnected the communication between all the other nodes may be affected. Similarly the betweenness between all the other nodes can be calculated. As the Web Service Network is large we use the Dijakstra's shortest path algorithm given in algorithm 2.



Nodes	Betweenness value	Degree
1. Flickr	3039233	388
2. Twitter	2786558	453
3. YouTube	2122121	393
4. Amazon	1699774	262
5. Facebook	1465646	218
6. Tagbulb	1307982	65
7. Audi411	1022316	6
8. Twilio	992611	213
9. EBay	818512	143
10. MicrosoftBingMaps	794205	108

Figure 3.11: Top ten nodes with highest Betweenness

In figure 3.11 the nodes with high betweenness centrality can be seen as "bridges" between clusters of other nodes. An interesting observation is that nodes with high betweenness do not necessarily have a high degree.

Let us consider a simple example to verify the above case.

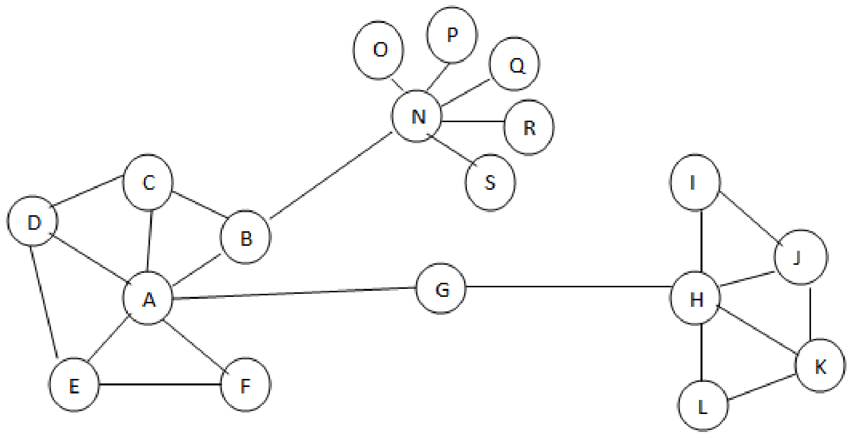
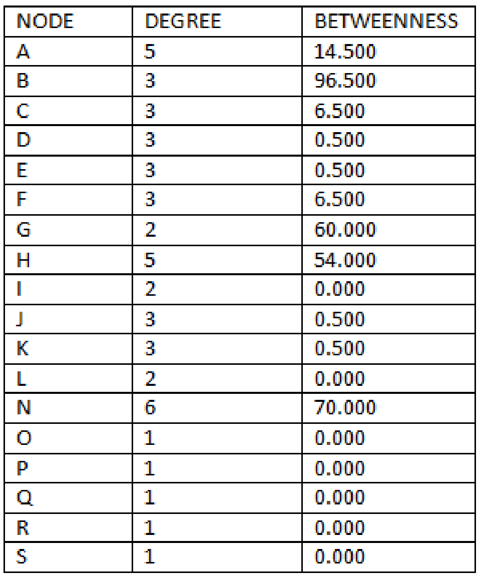


Figure 3.12: A larger example for betweenness centrality



NODE	DEGREE	BETWEENNESS
A	5	14.500
B	3	96.500
C	3	6.500
D	3	0.500
E	3	0.500
F	3	6.500
G	2	60.000
H	5	54.000
I	2	0.000
J	3	0.500
K	3	0.500
L	2	0.000
N	6	70.000
O	1	0.000
P	1	0.000
Q	1	0.000
R	1	0.000
S	1	0.000

Figure 3.13: Table for the network

In the example figure 3.12 the node B has a very low degree of 3 but has the highest betweenness of 96.500. Similarly node G has a very low degree of 2 but has one of the highest centralities this is because these nodes lie in the center connecting the central nodes in different areas. In particular, from figure 3.11 Audi411 has a relatively low degree of only 6, usually a low degree implies the node is unimportant. However, their high betweenness plays a critical role in connecting the web services in different domains. A close inspection of the service Audi411 reveals that it is used in six services that are in separate areas that are crucial for the network flow.

3.5 Closeness Centrality

Between all pairs of nodes in connected graphs there is a natural distance metric, this is defined by the length of their shortest paths[45]. The farness of a node can be identified as the sum of its distances to all the other nodes, and its closeness is defined as the inverse of the farness[46][45]. If the closeness centrality of a node is 0, then its farness must be infinite in which case it is infinitely far from some node or they may be infinitely many nodes in the graph. Thus, the more dominant a node is, the lower the distance to all the other nodes. The distance between two vertices is equal to the number of edges in the shortest path connecting them. Therefore, subsequently closeness can be regarded as how fast it takes to spread the information from a node to all other nodes[47]. According to [48] closeness centrality for node v is defined as

$$C(v) = \frac{1}{\sum_{i \in V} \quad d(v,i)} \tag{3.7}$$

where d(v,i) is the distance of the length of the shortest path between v and i and V is the set of all vertices.

As mentioned earlier, to calculate the closeness of each node the Web Service Network is considered undirected, and using algorithm 2 discussed earlier, the length of shortest paths for all pair of nodes are calculated and summed over the distance of each node to all other nodes.

An example network to calculate the Closeness centrality measure is as follows.

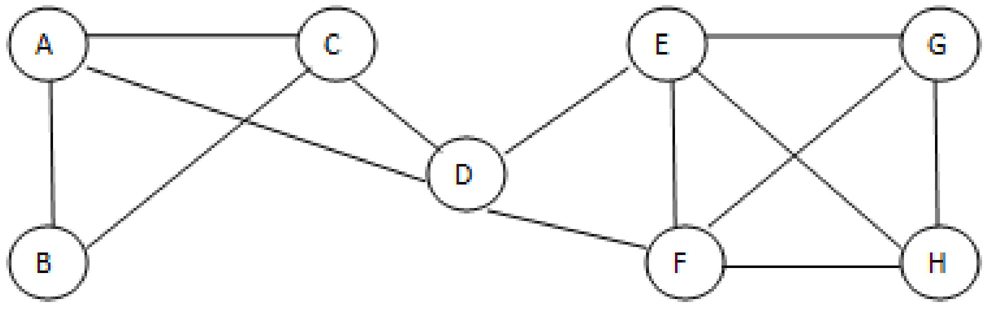


Figure 3.14: A simple example for Closeness centrality

The distance between the nodes for figure 3.14 can be given by a matrix.

Node	A	B	C	D	E	F	G	H
A	0	1	1	1	2	2	3	3
B	1	0	1	2	3	3	4	4
C	1	1	0	1	2	2	3	3
D	1	2	1	0	1	1	2	2
E	2	3	2	1	0	1	1	1
F	2	3	2	1	1	0	1	1
G	3	4	3	2	1	1	0	1
H	3	4	3	2	1	1	1	0

Table 3.4: Calculating the Closeness Centrality

The closeness for the node B and D from table 3.4 can be calculated as

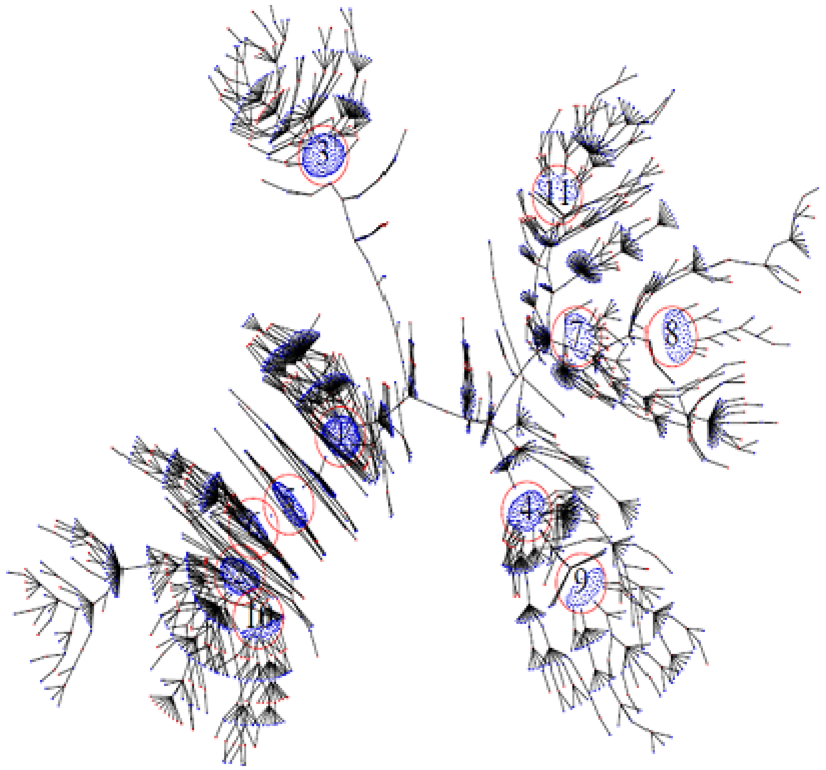
$$C_C(B) = \frac{1}{sum\ of\ all\ its\ distances} = \frac{1}{1+1+2+3+3+4+4} = \frac{1}{18}$$

$$C_C(D) = \frac{1}{sum\ of\ all\ its\ distances} = \frac{1}{1+2+1+1+1+2+2} = \frac{1}{10}$$

Nodes with high closeness centrality are noteworthy as they propagate information with more efficiency in a network in comparison with the rest of the nodes. Therefore, if we are looking to send information quickly we would start sending to the node with the highest centrality.

In the case of figure 3.14 we would first send the information to the node D. Since D can reach the rest of the network with the smallest effort. As D has a closeness of just 1/10 to reach all other nodes in the network were as B has a closeness of 1/18 to the rest of the network whereas the other nodes in the network have 13, 15 or 18.

Similar to Betweenness Centrality this thesis uses Matlab programming language to calculate the Closeness centrality for the Web Service Network. Based on different goals of interacting, different nodes might take the role of being most important in the network. All of the above centralities are crucial in order to identify the critical nodes in the network.



Node name	Closeness value
1. Twitter	0.029
2. Flickr	0.023
3. YouTube	0.022
4. Amazon	0.017
5. Twilio	0.016
6.Facebook	0.011
7. EBay	0.009
8. Microsoft Bing Maps	0.009
9. 411Sync	0.008
10.LastFM	0.008

Figure 3.15: Top ten nodes with highest Closeness

To calculate the closeness centrality we consider the nodes that have the smallest closeness unlike other as betweenness, as these nodes are close to every other node through which the flow of information can be spread to all the nodes in a small amount of time. The list of the top nodes for all these centralities is followed in Chapter 4.

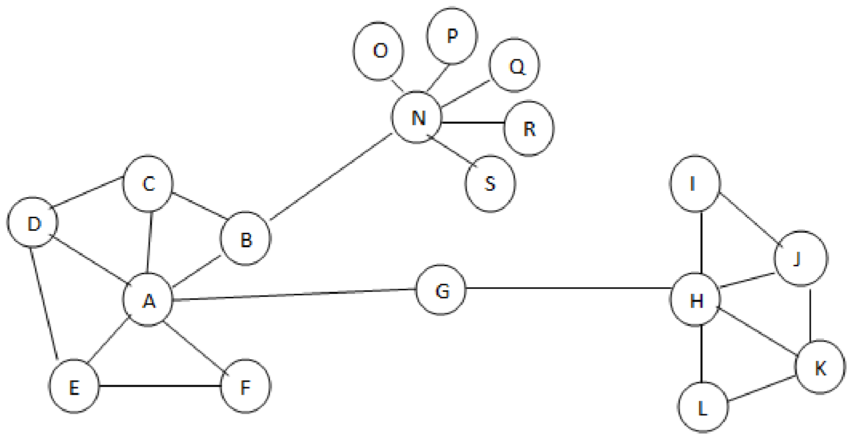
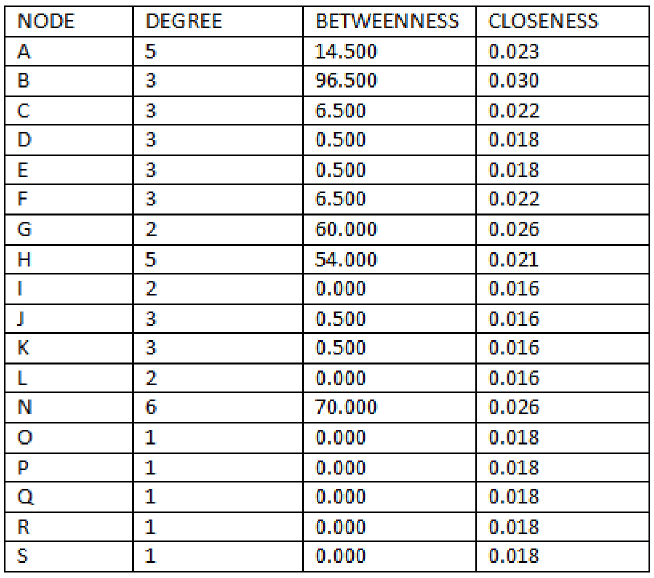


Figure 3.16: A larger example for closeness centrality



NODE	DEGREE	BETWEENNESS	CLOSENESS
A	5	14.500	0.023
B	3	96.500	0.030
C	3	6.500	0.022
D	3	0.500	0.018
E	3	0.500	0.018
F	3	6.500	0.022
G	2	60.000	0.026
H	5	54.000	0.021
I	2	0.000	0.016
J	3	0.500	0.016
K	3	0.500	0.016
L	2	0.000	0.016
N	6	70.000	0.026
O	1	0.000	0.018
P	1	0.000	0.018
Q	1	0.000	0.018
R	1	0.000	0.018
S	1	0.000	0.018

Figure 3.17: Table for the network

A node with high closeness may or may not have a high degree and high betweenness. In the example figure 3.16 the node C has a low degree of 3, a low betweenness of 6.500 but has a high closeness of 0.022 as this node is close to every other node through which the flow of information can be spread to all other nodes in a small amount of time.

3.6 The Average Path Length and Diameter

The average path length and diameter is based on the shortest path. The average path length is defined as the average number of steps along the shortest path for all the possible pairs of nodes in the network. The concept of average path length should not be confused with diameter. The shortest paths should be extracted among the available nodes and the longest shortest path is counted as the diameter. For a complete network, where every node is connected to every other node, the diameter of the network is 1.

Let us consider a small example to understand the concept of diameter is as follows.

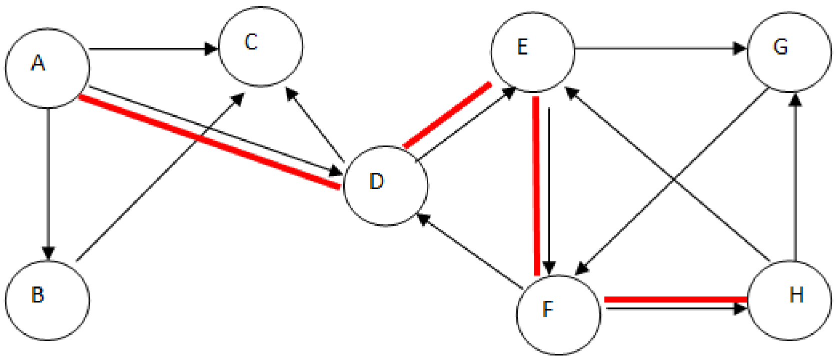
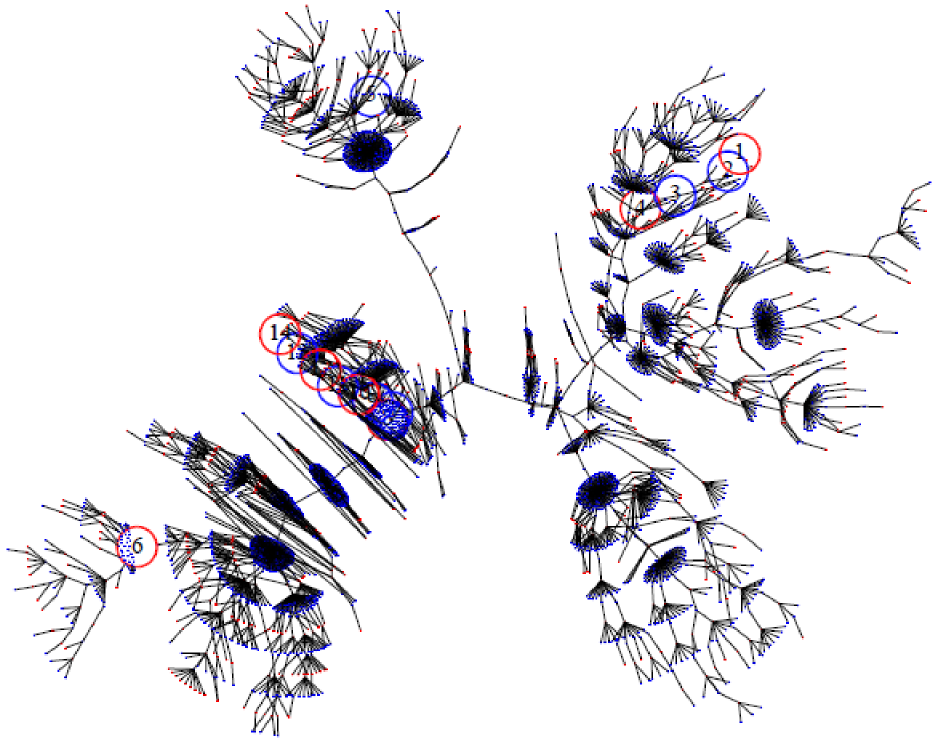


Figure 3.18: An example network to calculate the diameter

For the figure 3.18 the longest shortest path is from A-H which is 4. For the directed graph, the average is taken only over the distances of those nodes that are reachable from each other and in case of those nodes that are not reachable from each other their distance is infinite. Thus, as node H cannot reach node A the longest distance is infinite. Hence the diameter of this network is infinite. If we consider the figure 3.18 as undirected every node can reach every other node.

In this thesis, the diameter and average length are calculated for Web Service Network for both directed and undirected . The greatest length of any of the shortest paths in this thesis for a directed network is 2, and as every node cannot reach every other node the diameter is infinite and for the undirected network is 14. The average path length for directed network is infinite and for undirected network is 4.648.



Nodes	Node name
1	PubNub
2	InDashSMSDashboard&auto-responderforNexmo
3	NexmoSMSSpreader
4	GoogleSpreadsheets
5	GoogleSpreadsheetsEditor
6	GoogleHomepage
7	What'sonTwitter
8	Twitter
9	Snappr
10	ClockworkSMS
11	MediaburstSMSthroughShopify
12	Shopify
13	ShopTalk
14	Cloudvox

Figure 3.19: Diameter for the Web Api-Mashup Network-Undirected

The diameter of figure 3.19 is a unique chain as length of the longest shortest path for the undirected network is 14. The nodes which appeared in the diameter are visualized in the table in figure 3.19.

3.7 Strongly and Weakly connected parts(SCC and WCC)

A network is strongly connected if for any pair of nodes there is a path linking them. For finding the strongly connected parts of the network, it is important that we first find all the nodes that are reachable from each other. This approach enables us to find all small parts of strongly connected nodes if there is any. It is important to find strongly connected and weakly connected components to find the connectivity between the nodes and also to identify the core nodes that constitute the entire network.

Let us consider a simple example to calculate SCC.

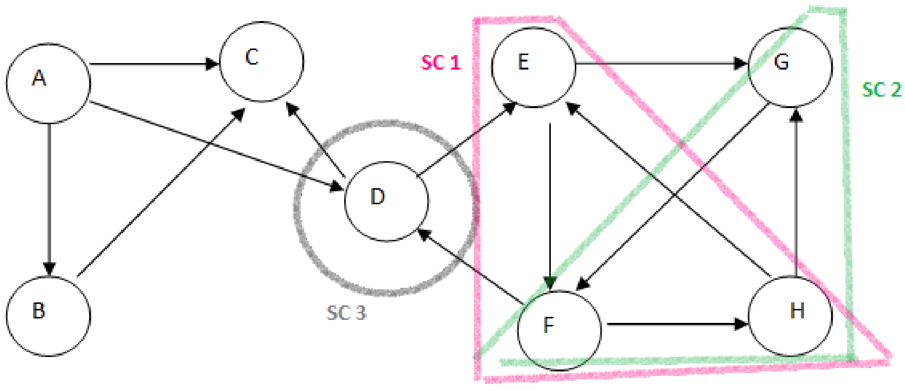


Figure 3.20: Simple example network to calculate the SCC

Figure 3.20 can be partitioned into a unique set of strong components of SC1:(E, F, H), SC2:(G, F, H), SC3:(D). Node D is a standalone strongly connected componen. However, every other node cannot reach node A and node C cannot reach other nodes they are weakly connected components. Thus, the graph is a weakly connected graph.

The information about all the nodes that are reachable from all the other nodes can be found by finding the shortest path between the nodes. In the shortest path algorithm 2, if the value of distance (i,j) is equal or greater than zero it means that the node j is reachable from the node i. We can then create a matrix which has a row for each node and in the columns it saves the index of all the nodes that are reachable form the node corresponded to the index of the row. Using this matrix, we can check for all the nodes that are reachable from node i to see whether node i is also reachable from those nodes. If the node i is also reachable from any of its connected nodes, that connected node is stored as a strongly connected node with node i in a strongly connected matrix. After evaluating the mentioned condition for all nodes, in the strongly connected matrix we will have a list of nodes that are strongly connected to each nodes.

This approach enables us to find all small parts of strongly connected nodes if there is any. The largest strongly connected part of the graph can be verified by finding the longest strongly connected part. For finding the weakly connected parts of the graph the same algorithm can be employed to the undirected graph to find the weakly connected part of the nodes if there is any. The largest strongly connected part of the graph can be checked by finding the longest strongly connected part.

For the Web Service Network, G=(V, E) in this thesis the number of components i.e., the maximal number of sets of nodes in which every node can reach every other node by some path no matter how long is 157. The component 1 is a large component and has 3858 nodes and it represents 91% of the network. The number of standalone strongly connected components is 4255 and the number of weakly connected components is 157. Thus, the Web Service Network is Weakly Connected.

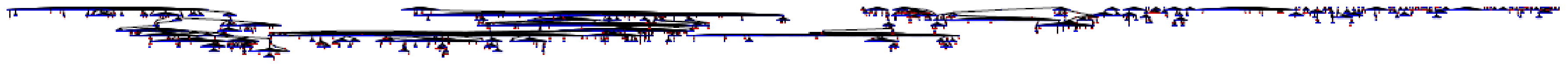


Figure 3.21: Strongly connected component for the Web Api-Mashup Network

3.8 Clustering Coefficient

The clustering coefficient was first introduced by Watts and Strogatz [9] in the context of social network analysis. In most real world networks, evidence suggests that tightly knit groups tend are created by the nodes by a relatively high density[9]. This kind of a likelihood, tends to be greater compared to the average probability of a connection randomly established between two nodes [9, 49].

The clustering coefficient measures the extent to which the network nodes tend to form groups with internal connections but few times connections leading out of the group. It is a property of a node in a network. The measure is 1 if every neighbor connected to a vertex, is also connected to every other vertex within the neighborhood and is 0 if none of the neighbors is connected. Clearly stating, clustering coefficient defined as the probability that two randomly selected neighbors are connected to each other.

According to [9] clustering coeffcient is defined as

$$CC(v) = \frac{Actual\ number\ of\ edges\ between\ neighbors\ of\ v}{Maximum\ possible\ number\ of\ edges\ between\ neighbors\ of\ v} \tag{3.8}$$

The actual number of edges between neighbors of v can be expressed as k(k-1)/2.

Let us consider a simple example to calculate the clustering coefficient is as follows

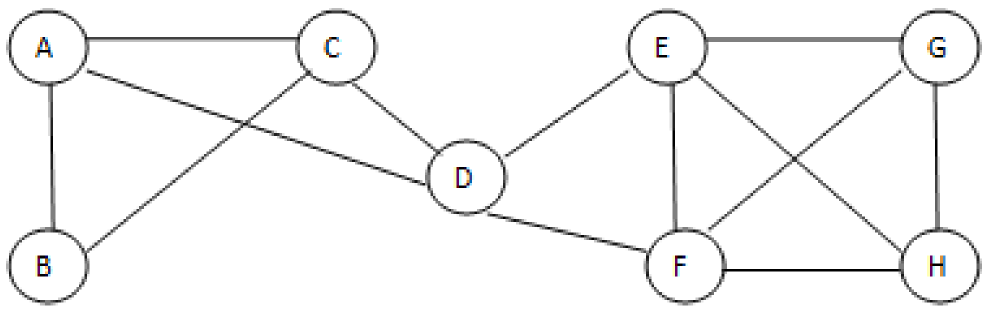


Figure 3.22: An example network to calculate clustering coefficient

Let figure 3.22 is a graph G. Let us consider vertex D, whose clustering coefficient has to be found. The actual number of edges between neighbors of D is 2 they are (a:c,e:f). The maximum possible number of edges between neighbors of D ie., k is 4= k(k-1)/2=4(3)/2=6 . Thus clustering coeffcient of D can be expressed as

$$CC(D) = \frac{2}{6} = 1/3 = 0.333 \tag{3.9}$$

A tree structure has no clusters by definition and the clustering coefficient of a tree is 0 [18]. For the Web Service Network the clustering coefficient is zero.

3.9 Ranking Web APIs

PageRank is considered an interesting metric to determine the relative importance of Web Services within a Web Service Network. They can be ranked in terms of 'importance' or 'usefulness' within a network so that the discovery of the solutions can be done in an effective way.

The PageRank is measured by the number of Mashups that use a particular API. Their rank again is indicated by the rank of services which use them. The algorithm for computing PageRank is provided later. For ranking the web API's based on the Directed popularity method, each web API score is computed based on the number of its incoming-links, but all incoming-links are not equal in terms of importance, for example an incoming-link from a node that has itself a high number of incoming-links from other node has more value than an incoming-link from a node that does not have any incoming-link from the other nodes, therefore a recursive formula is required to calculate the score of each API. The Web Service APIs with higher rank have higher popularity.
According to [33] PageRank is defined as

$$Mv = \lambda * v \tag{3.10}$$

M is the matrix representation of the network, v is the eigenvector of M and λ is the principal eigenvalue.

The PageRank algorithm is provided; the iteratios for the algorithm equals 20, and for each node in Matrix M if a node j has a link to i then M_{i_j} is $1/Outdegree(j)$ if that element equals 1 in M otherwise 0.

Algorithm 3 PageRank

```
Input:Web Service Network N represented as the list of edges, Node Number(the total number of nodes)
Output: $v$ is the rank vector containing the values
M= Matrix representation of Web Service Network N
$v$=[1..1]/Node Number
for i=1:20 do
  $v$=M*$v$
end for
```

In the above algorithm the vector v is the principle eigenvector of the matrix M. Initially we assign this rank vector v values to be 1 and divide the vector with the total number of nodes N to get the rank vector v which is a [1/N] matrix. This will be the first approximation. The rank can be computed by applying M to the initial rank vector matrix [1/N] repeatedly, where [1/N] is an N*1 matrix in which all the entries are (1/N). Let v_0 be the intermediate rank, v_1 be the initial rank,

v_2 be $v_1 * M$, v_3 be $v_2 * M$. The rank is converged to a fixed point as number of iterations increase. This is called the Power method. The reason for using Power method is to compute the Pagerank vector, concerns the number of iterations it requires. The converged rank i.e.,v contains PageRank values of all the nodes. The Power method is still the predominant method in finding PageRank. For the Web Service Network after a number of iterations (20) is able to calculate the rank of each node. Let us consider a simple example to calculate PageRank.

A simple example to calculate the PageRank is as follows.

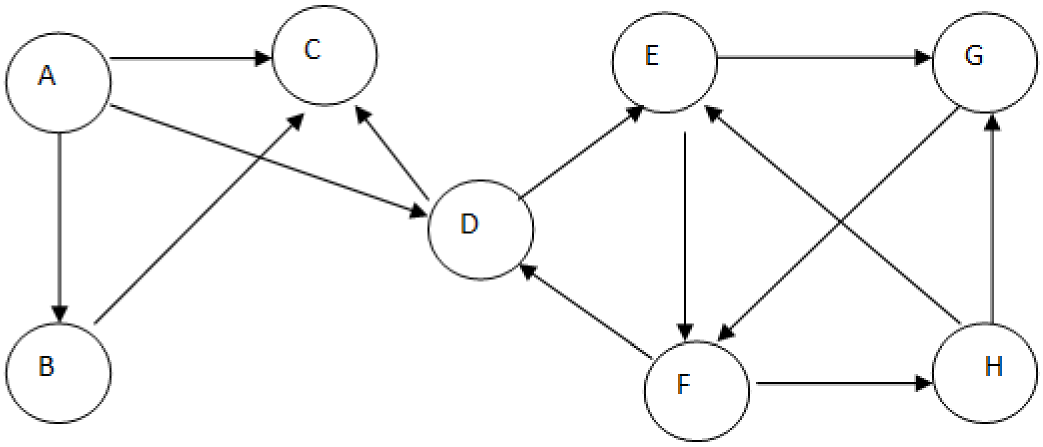


Figure 3.23: An example of a network to calculate PageRank

Assume a random surfer starting at node A in figure 3.23. In order to explain what happens to the random surfers at the next step it is better to describe the matrix, if they are 'n' nodes, matrix M consists of 'n' rows and columns. If a node 'j' has 'k' arcs out and if one of them is to node 'i' then 'i' is the row and 'j' is the column for the element M(ij) then the value is 1/k else M(ij)=0[33].

The Matrix Representation for figure 3.23 is as follows.

M=

$$\begin{vmatrix}
0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & 0 \\
0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0
\end{vmatrix}$$

M is the Matrix. The order of the surfer for the above matrix is a natural one A, B, C, D and E. Matrix M has one row and one column for each node. Thus, the initial column represents that the surfer A has a probability of 1/3 for being at the nodes B, C and D based on the matrix. The surfer at node B has one link C and it has a probability 1 as it is certain that surfer would be at only that link next. Similarly this is followed in all other nodes[33].

A column vector can be used to best describe a random surfer's probability. The jth component is the probability that the random surfer is at node j. This function is the PageRank[33]. Assume a random surfer starting at any of the 'N' nodes with a balanced probability. For each component, an initial vector v_o would have a probability of $1/N$[33]. The distribution of the surfer after one step would be

$$M * v_o \tag{3.11}$$

and for the next step would be

$$M(Mv_o) = M^2 v_o \tag{3.12}$$

and so on if M is the matrix. In order to know the surfers distribution after

'i' steps it can be found by multiplying the transition matrix M with the initial vector v_o for a total of 'i' times[33].

A random surfers probability distribtion at node i after each step according to [33] is defined as

$$\sum_j M(ij) * v_j \tag{3.13}$$

where M(ij) is the probability that the surfer would make a move to node i in the later step being at node j and this value would be 0 as there is no link from node j to i and v_j represents that the surfer was at node j at the former[33]. Given the example of this type of comportment usually relates to ancient theory called as Markov process. A surfer reaches a limiting distribution v that satisfies

$$v = M * v \tag{3.14}$$

And meet the conditions:

- If it is capable of reaching from any node to every other node, i.e., if a graph is strongly connected
- If it has no dead ends, i.e., the nodes that have no arcs out

  As we see the example 1 satisfies both of these above stated conditions. The distribution of M does not change by multiplying another time, rather it reaches a limit, in an alternative way the v is called as the eigenvector of M[33]. For some principal eigenvalue λ if the vector v satisfies according to [33] is defined as

$$Mv = \lambda * v \tag{3.15}$$

then it is said to be an eigenvector for matrix M. As v has the highest eigenvalue associated to all the other eigenvalue's it is termed as principle eigenvector and as the columns and rows of M each sum to 1 it is called a stochastic matrix. The principal eigenvalue identified with the eigenvector is 1 as M is stochastic and v is called the rank vector[33].

The position of the surfer after a long time can be known with this principle eigenvector. This intern reminds us of the intuition of PageRank i.e.: the more important a node is the more likely that a surfer will be at that node. Thus, we can start by multiplying v_o the initial vector with the Matrix M for number of rounds to see how the value differes [33]. To overcome the error limits of double precision the Web intern has 50-75 iterations for sufficient coverage [33].

Example 2: Let us see what will happen by applying the above process to matrix M. As they are five nodes, the initial vector v_o has eight components, each 1/8. The array of approximations to the limit that we get at each step by multiplying each step of M

$$\begin{vmatrix}1/8\\1/8\\1/8\\1/8\\1/8\\1/8\\1/8\\1/8\end{vmatrix} \begin{vmatrix}1/8\\1/8\\0\\1/8\\1/8\\1/16\\1/16\\1/8\end{vmatrix} \begin{vmatrix}1/2\\0\\0\\1/16\\1/16\\1/16\\1/32\\3/32\end{vmatrix} \begin{vmatrix}1/48\\0\\0\\1/32\\3/64\\3/64\\1/62\\3/64\end{vmatrix} \begin{vmatrix}1/48\\0\\0\\1/16\\7/64\\3/32\\3/32\\1/8\end{vmatrix} \text{ and goes on } \qquad \begin{vmatrix}1/28\\0\\0\\3/28\\3/14\\3/14\\3/14\\3/14\end{vmatrix}$$

After a number of iterations the probability of A is 0.0357, the probability of B is 0, the probability of C is 0 etc. The difference of the probabilities of the nodes may not be great but in the real world situations with billions of nodes they would be greatly varying.

A similar approach of PageRanking can be applied and used for the Web Service Network. The PageRank representation for the Web Service Network is as follows.

In figure 3.24, x axis represents the number of iterations and y axis represents the ranks. PageRank of the nodes calculation in 20 iterations is shown. We can see from figure that the PageRank of most of the nodes is '0'. From this it is clear that most of the nodes could not be observed after a few iterations and they tend to be zero.

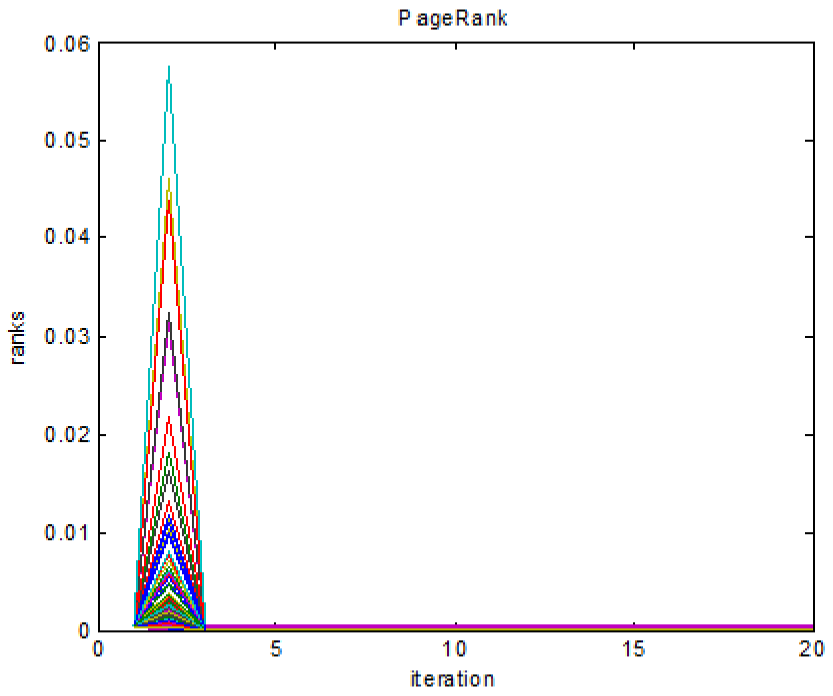


Figure 3.24: PageRank

3.10 PageRank using Damping Factor

If there is a self loop for a node, the random surfer entering that node can never leave. Thus, all the pagerank calculated is trapped at that node. To resolve such issues the concept of teleportation came into existence. In many studies, scientists have confirmed this jump factor or the damping factor to be β and having a value of 0.8. For example consider a random surfer. The probability of a random surfer being introduced is precisely equal to that the random surfer will decide not to follow a link from their current page[33] if it has no dead ends (which will be discussed in later sections). At this point, it is clear to visualize that a random surfer decides if it has to follow the link from a current node or teleport to a random node. It is important to use the jump factor for traversing to the next nodes with the mentioned probability. Nevertheless, if a node has dead ends there is a possibility that the surfer can go nowhere.

It is simply the probability of traversing to the next nodes. Thus the probability v=M*v is replaced by

$$v = (\beta * M + (1 - \beta) * R) * v \tag{3.16}$$

Where at each step we could either follow the link with the probability of β or either jump to some other nodes with a probability of $1 - \beta$. Here "R" is defined as the Rank and dividing by 1/N is the value for each element of R.

Algorithm 4 PageRank using Jump Factor

```
Input:Matrix M represented as the list of edges
Output: $v$ is the rank vector containing the values
for i=1:20 do
    $\beta$=0.8
    $v = (\beta * M + (1 - \beta) * R) * v$
end for
```

The algorithm 4 calculates the PageRank using a jump factor 0.8. The result of applying this change is shown in figure 3.25.

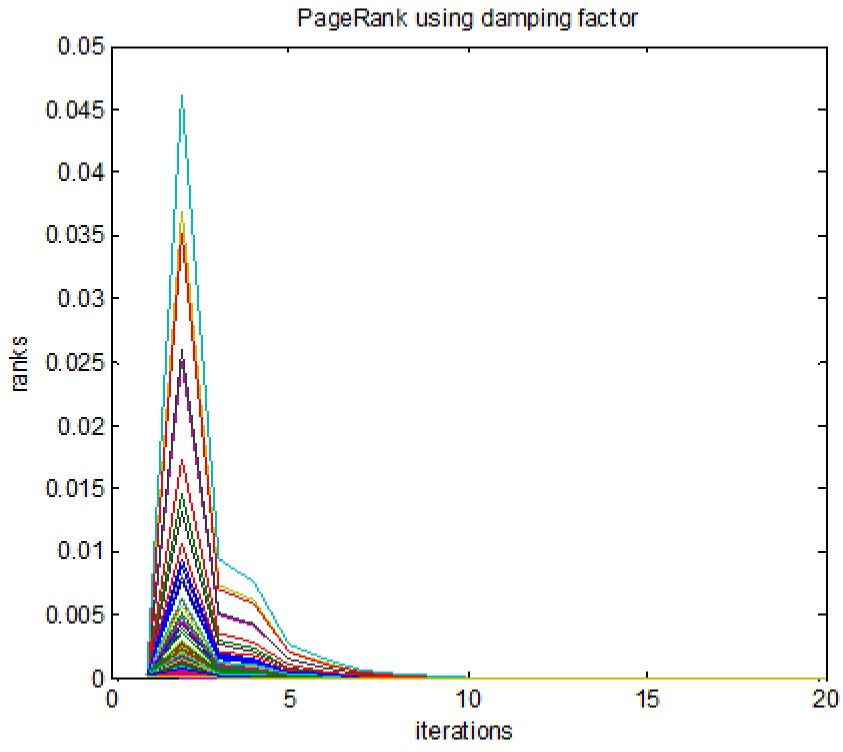


Figure 3.25: PageRank using Jump Factor 0.8

We can observe a major difference compared to figure 3.25 that after removing the dead ends the PageRank converges after a few number of iterations. The figure 3.24 of PageRank differs from figure 3.25 of PageRank as it postpones the iteration of all observable probability to go to zero. However, sometimes this problem cannot be solved due to dead ends. For confronting this problem we could replace all the zero elements in the adjacency matrix with [1/N] or even removing that nodes from the matrix to solve the dead ends result.

3.11 PageRank removing the Dead ends

Generally, a node that has no links out is called as a dead end. By allowing the dead ends or such nodes the transition matrix can no longer be stochastic as some columns may sum to 0 instead of 1. The matrix whose columns may sum to solely 1 is called as substochastic. All the components or some of the vectors may be 0 for a stochastic matrix by computing M_{iv} for increasing powers. This implies that most of the important information for the network 'drains out' and no relative importance of the nodes is determined.

The other major problem is a spider trap. The set of nodes with no dead ends or any arcs out is called a spider trap[33]. It is because of these spider traps the pagerank calculation of a node can be trapped and the surfer cannot move any further.

If there are dead ends in the Web Service Network we replace the corresponding column in the matrix M with 0 and to remove the dead ends we have initially replaced them with 1/N.

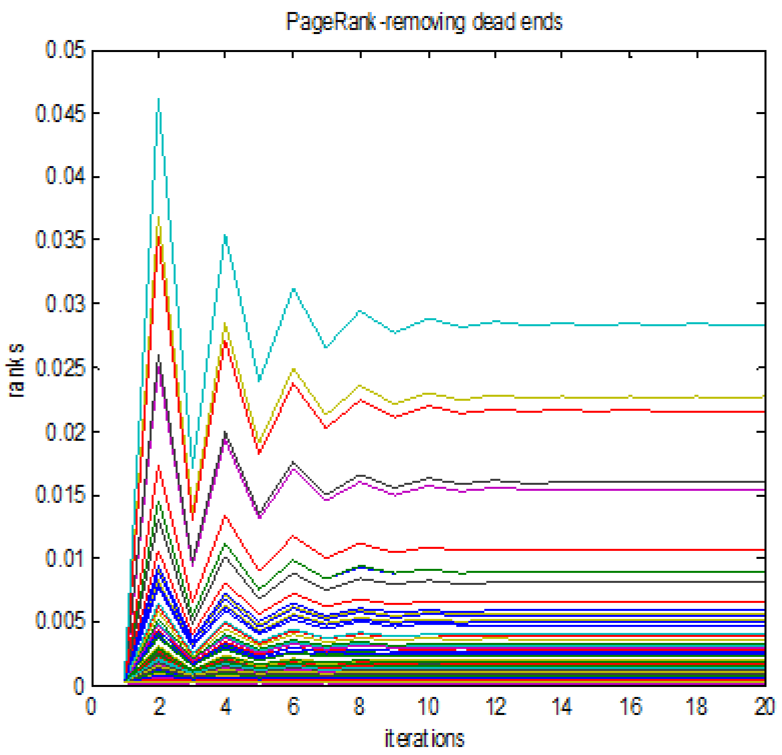
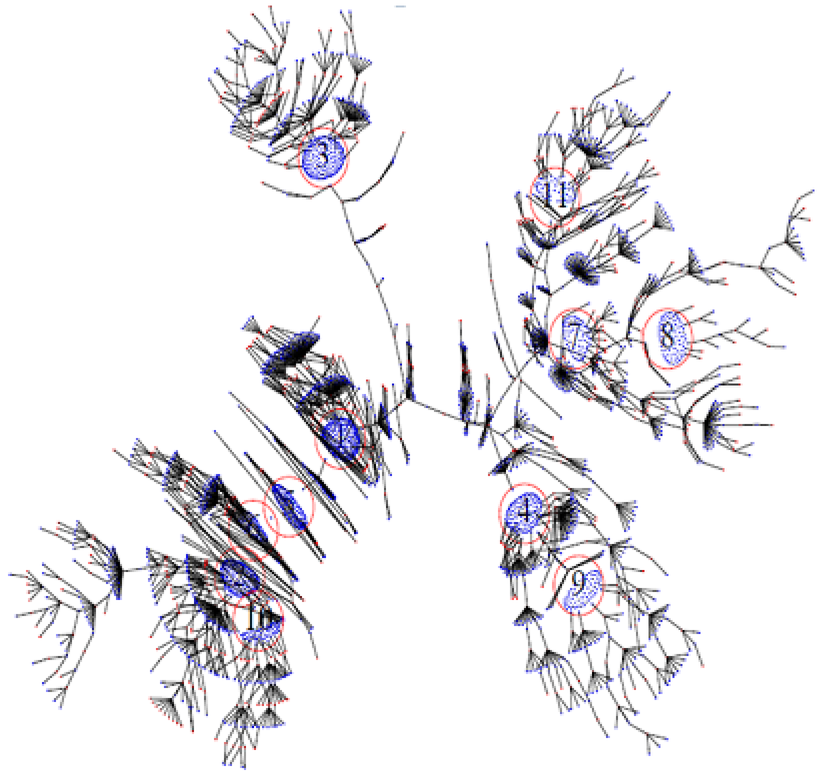


Figure 3.26: PageRank after removing the dead ends

From figure 3.26 we can observe that after using the damping factor and removing the dead ends the values converge very quickly after a very few number of iterations.

The Web Service APIs with higher rank and popularity attract own set of users to create their Mashups for the development of personal and business applications. Thus, PageRanking method is a simple and elegant approach to searching and ranking web service APIs significantly enhancing the discovery process for outputting the services with the highest rank.



Node Name	PageRank value
1. Twitter	0.028
2. Flickr	0.022
3. YouTube	0.021
4. Amazon	0.015
5. Twilio	0.015
6. Facebook	0.010
7. EBay	0.009
8. MicrosoftBingMaps	0.008
9. 411Sync	0.008
10. LastFM	0.006

Figure 3.27: Top ten nodes with highest PageRank

Figure 3.27 depicts some of the nodes with the highest PageRank. The number of web services that offer an API to access their functionality has risen rapidly over the past years. Thus choosing an appropriate web service API to achieve a specific goal may be time consuming and scary for the mashup developers. PageRank is considered as an interesting metric to determine the relative importance of web services within a Web Service Network. The ranking of these Web Services serves as a heuristic guide to the success of the ranking of these web services. They can be ranked in terms of 'importance' or 'usefulness' within a network so that the discovery of the solutions can be done in an effective way.

It is not necessarily that nodes with high PageRank also have high degree, betweenness or closeness. Let us consider a simple example to verify the above scenario.

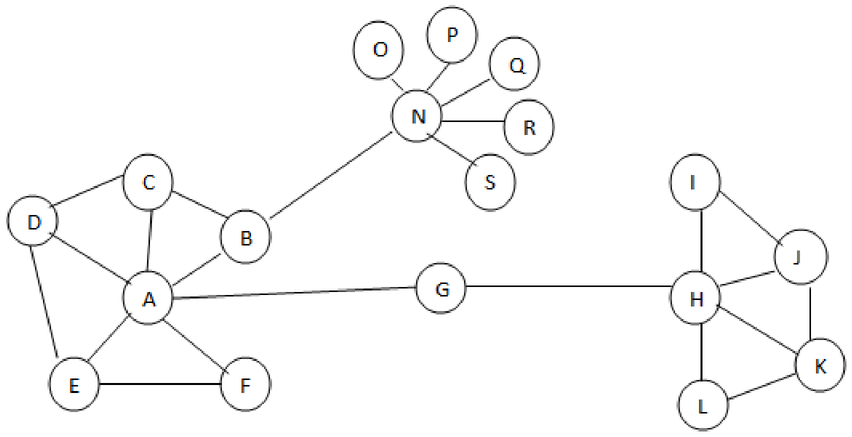
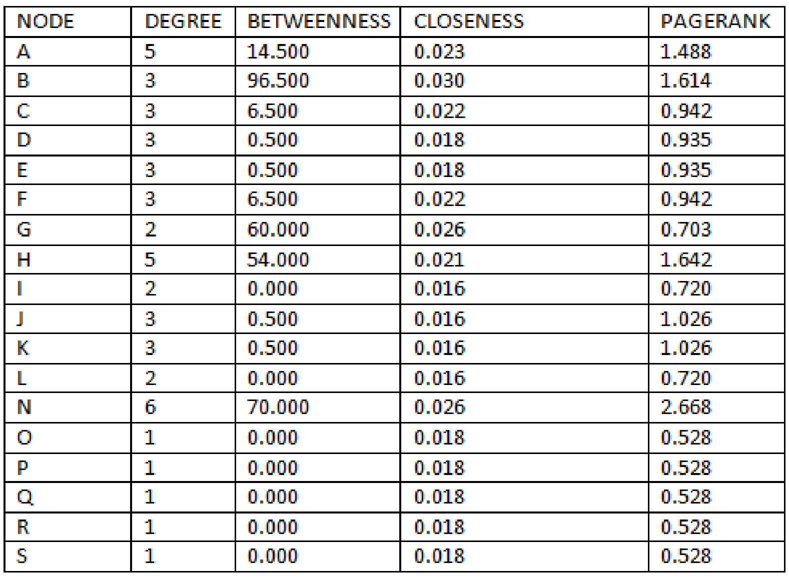


Figure 3.28: A larger example for PageRank centrality



NODE	DEGREE	BETWEENNESS	CLOSENESS	PAGERANK
A	5	14.500	0.023	1.488
B	3	96.500	0.030	1.614
C	3	6.500	0.022	0.942
D	3	0.500	0.018	0.935
E	3	0.500	0.018	0.935
F	3	6.500	0.022	0.942
G	2	60.000	0.026	0.703
H	5	54.000	0.021	1.642
I	2	0.000	0.016	0.720
J	3	0.500	0.016	1.026
K	3	0.500	0.016	1.026
L	2	0.000	0.016	0.720
N	6	70.000	0.026	2.668
O	1	0.000	0.018	0.528
P	1	0.000	0.018	0.528
Q	1	0.000	0.018	0.528
R	1	0.000	0.018	0.528
S	1	0.000	0.018	0.528

Figure 3.29: Table for the network

From table 3.29 we can see that node F has a low degree, low betweenness, low closeness but has one of the highest pagerank. Observing these kind of results is really interesting to see how these nodes affect the overall network.

After PageRank was implemented a proposal called "Hubs and Authorities" was introduced. Similar to Pagerank this method also involved iterative computation of vector matrix multiplication until a fixed point. However, there are huge distinctness between these ideas, and none can be a substitute of the other. The PageRank algorithm preprocesses the steps before handling any serach queries whereas the hubs-and-authorities algorithm, was intially proposed alone for processing a search query, and to rank and produce only the responses relevant to that particular query. The Hubs and authorities may be sometimes referred to as HITS-Hyperlink Induced Topic Search[33]

However, PageRank provides a standard definition that " if important pages link to a page it is considered as important", "where as HITS may be defined iteratively as a definition of two concepts:"if good authorities are linked to page it is a good hub , and if a page is linkedby good hubs its a good authority"[33]. Thus in the context of Web Service Network a service is important when it is used by many services and is used by important web services. Though they are many other ranking tools available today, PageRank still stands efficient to test large web data or large web service networks for precise results.

Chapter 4

Correlation Between the Centralities

4.1 Pearson's Correlation coefficient

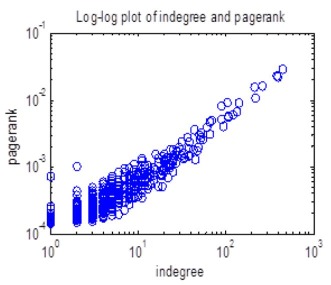
In order to gain access to the relation between different centralities, in this thesis we use the Pearson's correlation coefficient. Pearson's correlation coefficient is a measure of linear dependency between two variables. The dependence relationship between two variables can be described by using a linear function[50][51]. Pearson's rank correlation coefficients are calculated for each pair of metrics for the Web Service Network, and the correlation between the centralities is listed in table 4.1 and are plotted in figure 4.1. The correlation between different metrics in the scatter logarithmic plot for the Web Service Network is given below.

The Pearson correlation is +1 in the case of a perfect increasing (positive) linear relationship (correlation), -1 in the case of a perfect decreasing (negative) linear relationship and some value between -1 and 1 in all other cases, indicating the degree of linear dependence between the variables[51]. As it approaches zero, there is less of a relationship which is closer to uncorrelated. The closer the coefficient is to either -1 or 1, the stronger the correlation between the variables[51].

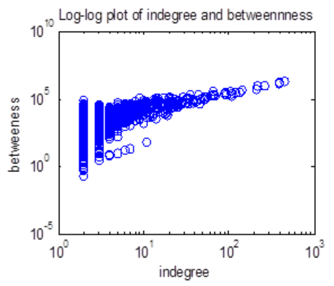
Centralities	In-degree	PageRank	Betweenness	Closeness	Out-degree
In-degree	*	0.9880	0.9721	0.9760	-0.0874
PageRank		*	0.9104	0.9876	-0.0804
Betweenness			*	0.9507	0.1554
Closeness				*	-0.0040
Out-degree					*

Table 4.1: Correlation between the Centralities

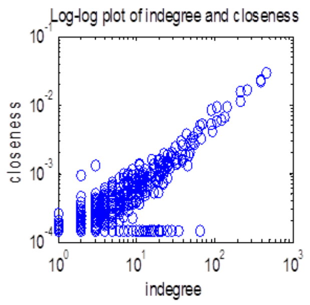
The logarithmic plots of the correlation table 4.1 is shown in figure 4.1



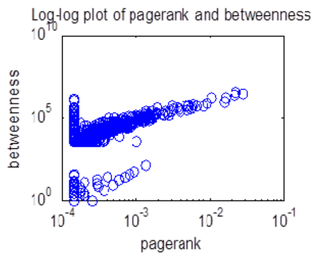
(a) In-degree vs PageRank Correlation



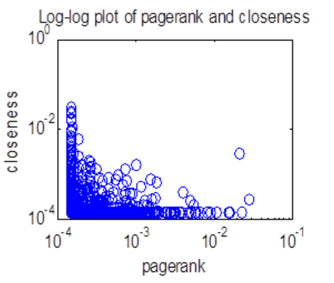
(b) In-degree vs Betweenness Correlation



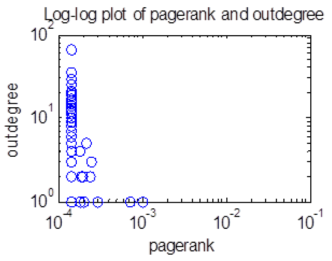
(c) In-degree vs CLoseness Correlation



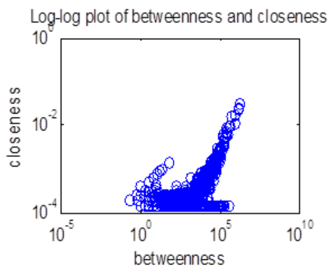
(d) PageRank vs Betweenness Correlation



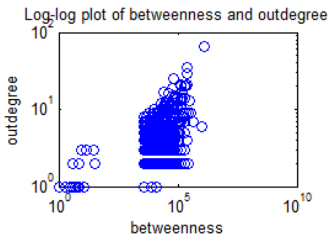
(e) PageRank vs Closeness Correlation



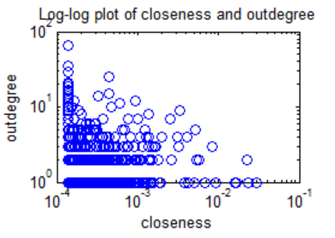
(f) PageRank vs Out-degree Correlation



(g) Betweenness vs CLoseness Correlation



(h) Betweenness vs Out-degree Correlation



(i) Closeness vs Out-degree Correlation

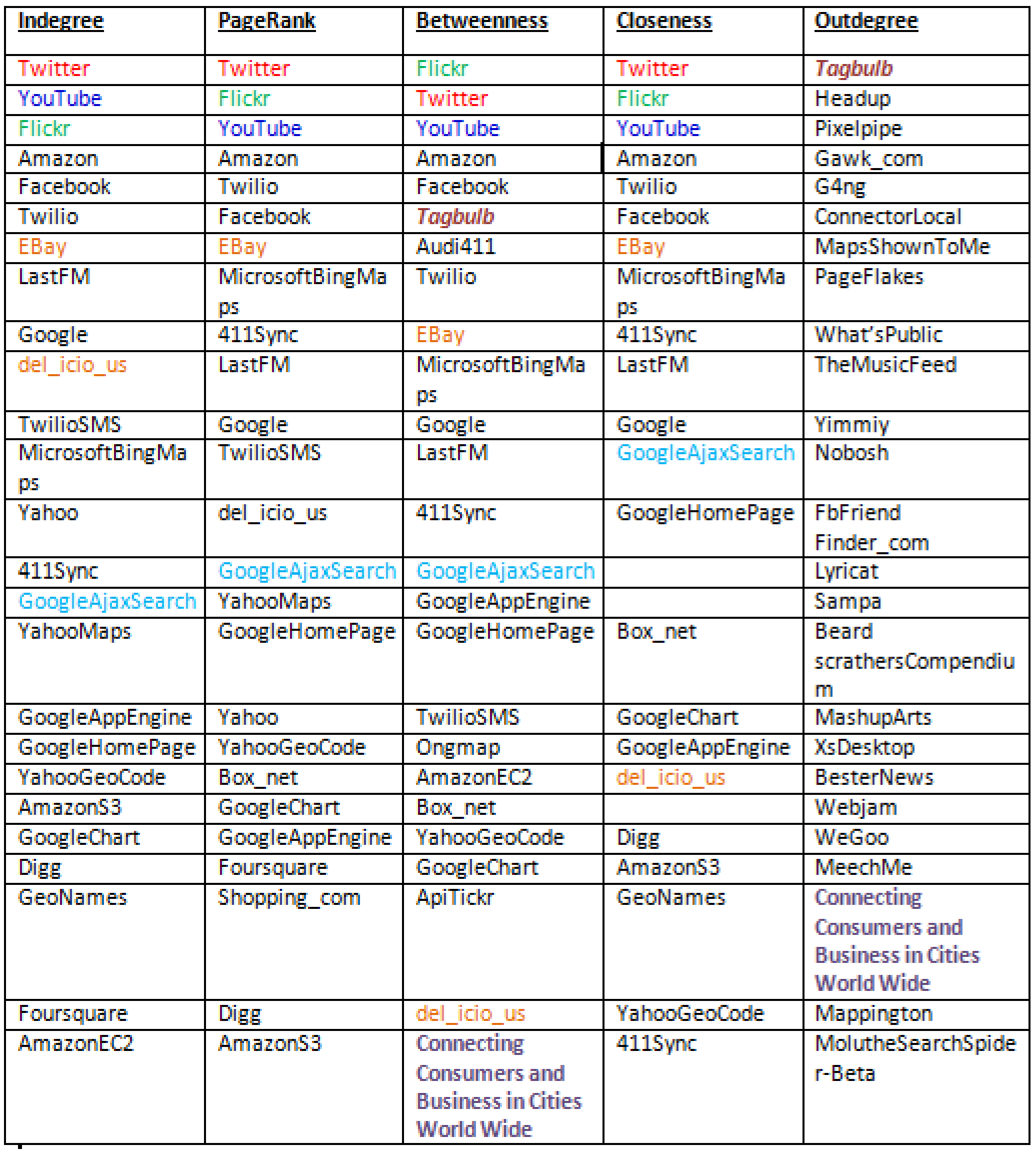
Figure 4.1: Logarithmic plot of Correlation between centralities for the Web Service Network
a. In-degree and PageRank b. In-degree and Betweenness c. In-degree and Closeness d. PageRank and Betweenness e. PageRank and Closeness f. PageRank and Out-degree g. Betweenness and Closeness h. Betweeness and Out-degree i. Closeness and Out-degree

As mentioned earlier in table 4.1, there is a positive correlation between in-degree, pagerank and betweenness. Similarly, there is a positive correlation between betweenness and out-degree. From figure 4.1 we can clearly note there is no correlation between other centralities to out-degree and hence this is negative in Table 4.1. The plot d in 4.1 is really interesting as betweenness and pagerank have a high correlation of 0.9104 but in the plot we see huge cluster in the bottom separated from the main cluster this is because the betweenness for the nodes drops exceptionally from high to really low at a point creating a cluster separated from the main.

4.2 Top Nodes

The figure 4.2 represents the top 25 vertices which have the highest measures of the explained centralities for the Web Service Network. Figure 14 shows the position of the top highest nodes for centralities.

It can be observed that in the studied Web Service Network there are some vertices that appeared in two or more highest centralities, which shows the relative importance of those nodes in the network. For example, the node Twitter has the highest in-degree, highest pagerank and also is the center of the network, which reveals the fact that most of the Mashups use the Twitter API.



Indegree	PageRank	Betweenness	Closeness	Outdegree
Twitter	Twitter	Flickr	Twitter	*Tagbulb*
YouTube	Flickr	Twitter	Flickr	Headup
Flickr	YouTube	YouTube	YouTube	Pixelpipe
Amazon	Amazon	Amazon	Amazon	Gawk_com
Facebook	Twilio	Facebook	Twilio	G4ng
Twilio	Facebook	*Tagbulb*	Facebook	ConnectorLocal
EBay	EBay	Audi411	EBay	MapsShownToMe
LastFM	MicrosoftBingMaps	Twilio	MicrosoftBingMaps	PageFlakes
Google	411Sync	EBay	411Sync	What'sPublic
del_icio_us	LastFM	MicrosoftBingMaps	LastFM	TheMusicFeed
TwilioSMS	Google	Google	Google	Yimmiy
MicrosoftBingMaps	TwilioSMS	LastFM	GoogleAjaxSearch	Nobosh
Yahoo	del_icio_us	411Sync	GoogleHomePage	FbFriend Finder_com
411Sync	GoogleAjaxSearch	GoogleAjaxSearch		Lyricat
GoogleAjaxSearch	YahooMaps	GoogleAppEngine		Sampa
YahooMaps	GoogleHomePage	GoogleHomePage	Box_net	Beard scrathersCompendium
GoogleAppEngine	Yahoo	TwilioSMS	GoogleChart	MashupArts
GoogleHomePage	YahooGeoCode	Ongmap	GoogleAppEngine	XsDesktop
YahooGeoCode	Box_net	AmazonEC2	del_icio_us	BesterNews
AmazonS3	GoogleChart	Box_net		Webjam
GoogleChart	GoogleAppEngine	YahooGeoCode	Digg	WeGoo
Digg	Foursquare	GoogleChart	AmazonS3	MeechMe
GeoNames	Shopping_com	ApiTickr	GeoNames	Connecting Consumers and Business in Cities World Wide
Foursquare	Digg	del_icio_us	YahooGeoCode	Mappington
AmazonEC2	AmazonS3	Connecting Consumers and Business in Cities World Wide	411Sync	MolutheSearchSpider-Beta

Figure 4.2: Top 25 nodes in the Web Service Network (directed)

The nodes that have a correlation in the centralities are highlighted in colors in this table. From this we can conclude that all these nodes play an important role in the Web Service Network based on their centralities, and removing of a central node may change the correlation between these centralities. All these node fonts are shown in different colors in order to show the similarity between them.

We can see many interesting results from figure 4.2. In the table of 4.1, top 25 nodes of the Web Service Network which have a high in-degree, pagerank, betweenness, closeness and out-degree are given, and it is obvious that there is a high correlation between pagerank and in-degree of 0.9880. Therefore it can also be observed that there are some nodes that appear in one or more highest centralities, thus the correlation between these centralities is positive showing the most important nodes in the Web Service Network.

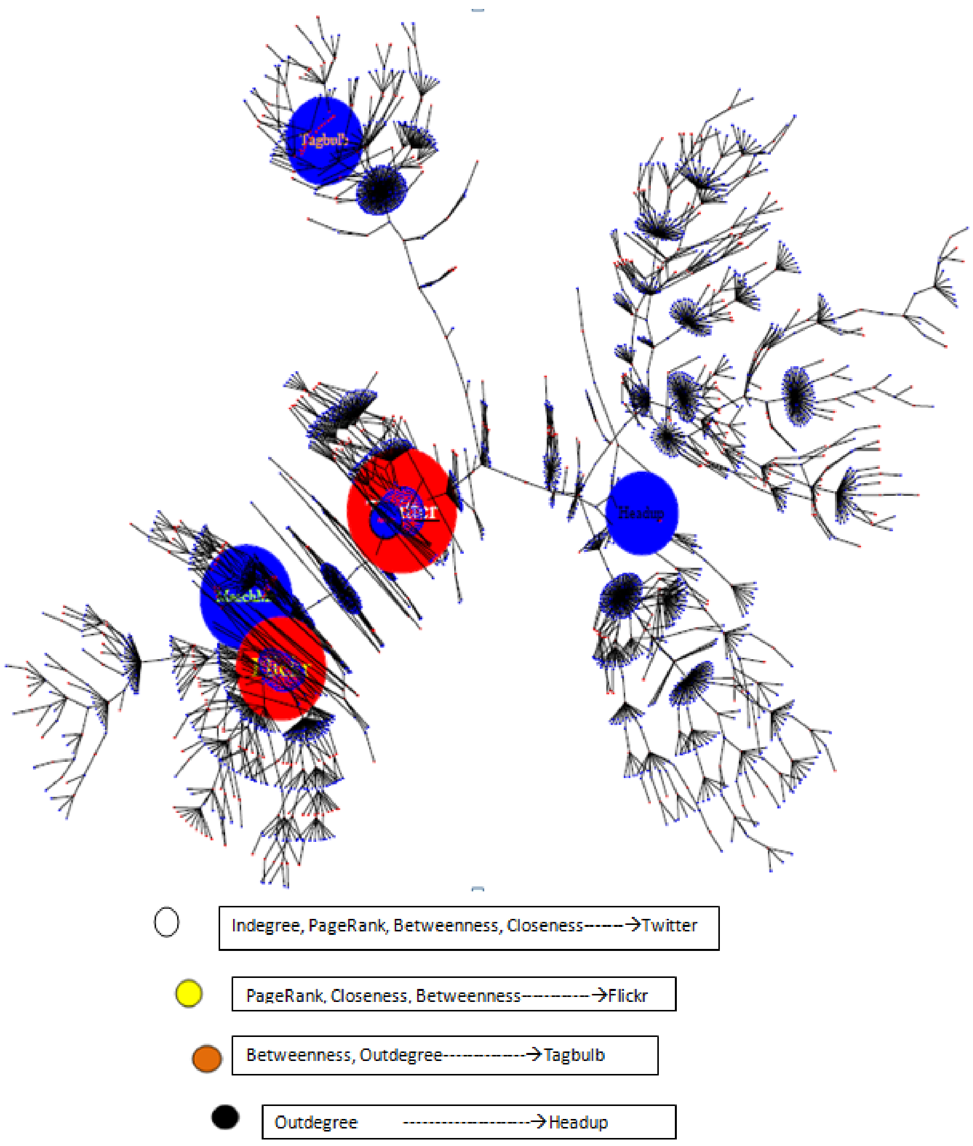

Tagbulb
Twitter
Headup
Flickr
Indegree, PageRank, Betweenness, Closeness-----→Twitter
PageRank, Closeness, Betweenness---------→Flickr
Betweenness, Outdegree------------→Tagbulb
Outdegree -----------------→Headup

Figure 4.3: Top highest score for In-degree, PageRank, Betweenness and Out-degree

The figure 4.3 highlights the top nodes for all the centralities. It is important to note that nodes with high degree sometimes may or may not have the highest pagerank.

4.3 Comparison to Previous Work and Observations

Power law distributions have been found in many areas including software metrics, network and social networks including information and computer science. Many researchers started to analyze the field of software in the perspective of studying and finding scale-free and small world behavior[52]. A phenomenon called "the small world structure" has been identified in numerous networks found in nature and society[53]. These type of networks show special properties, which allows them to spread and find information efficiently and effectively. These may include a high clustering coefficient and a small diameter[53]. Various authors have also found significant Power laws in software systems.

Potanin et al.[23] studied the object graphs which were formed by runtime object oriented programs written in various languages - Java ArgoUML, Java Forte, Java Jinsight, Java Satin GCCand SmallTalk programs etc, and they found significant Power Laws and these turned out to be scale free networks without exception. The average slope value of fitted line for incoming and outgoing links is close to 2 and 3.5 respectively, proving that larger programs use more objects and more levels of abstraction.

Valverde et al. and Ferrer Cancho et al.[17, 52] started the development of scaling in software architecture graphs belonging to open source software systems like Java, C/C++. They found Power laws in the graph vertexes input and output edge distribution. They also found the Power law for the two largest components of degree distribution with a gradient between 2.5-2.65. They observed the small world phenomena between any two nodes in a graph with an average distance of 6.39 and 6.91 and many other authors found Power Laws in links between C/C++ source code files[54, 55, 56].

Myers et al.[21] studied some open source systems like Linux operating system,MySQL relational database, Abiword word processing program, XMMS multimedia system with their class collaboration network etc. All the collaboration networks studied in his research exhibit scale free behaviour (Power Law) and heavy tailed degree distributions. They found the gradient for the in/out degree distribution of all systems in between 1.9-3.1. For most of the systems, they found the value to be around 2.5.

4.4 Summary of Power law distribution in software

Summary					
Paper	Dataset	Graph Model	Study Focus	Power law-slope	Other Metrics
Valverde et al.2002	JDk1.2	Software architectecture graph	In/Out degree distribution	2.5-2.65	N/A
Myers 2003	Open source systems	Class diagram	In/Out degree	1.9-3.1	N/A
Potanin et.al.2005	Java systems	Object graphs	In/Out degree distribtion	1.22-3.50	N/A
Louridas et al.2008	Java system	Module Graph	In/Out degree distributions	1.22-3.50	N/A

Finding the network properties in software has become a popular topic among computer scientists in recent years. The software is built of interacting components and subsystems at different levels of granularity (classes, interfaces, functions, libraries, etc) and also various kinds of interactions among these subsystems can be utilized to define graphs to form a description of the system.

In this thesis we studied the Web Service Network with 4255 nodes and 7193 edges. The Web Service Network is very interesting and unique topic compared to all the previous studies. We measured five different centralities to find the most popular Web APIs. We found the Power Law-distribution in this network denoting that a large number of APIs are used by a few Mashups, and a small number of APIs are used by many Mashups making them popular in the network. The best fit of the Power law for the Web Service Network α= 1.66. We also found a correlation between these centralities in order to calculate the linear dependency between these centralities showing the important nodes in the network.

- With the so called definition of the small world, the studied Web Service Network cannot be considered small world as the longest shortest path is 2 the diameter of the network is infinite as every node cannot reach every other node and their clustering coefficient is 0 i.e. very small. Thus it cannot be considered as a small world.
- Following, as the Degree Distribution follows a Power Law it is a "scale free network".
- It is not a bipartite graph as all the vertices cannot be divided into disjoint sets and every edge does not connect all the vertices.
- The key observation of this thesis is the structure of the network. As the clustering coefficient is zero and by definition a tree has no clusters which bring us to a conclusion that the Web Service Network is a tree structure.

Chapter 5

Conclusion and Future Work

In this thesis, we analyzed a new type of software network i.e., web service network extracted from the largest online web API-Mashup repository i.e., Programmable web. For this network, we analyzed its Power Law distribution. We also got the best fit of the Power Law for the logarithmic degree distribution, and we also concluded it is a scale free network.

We studied the In-degree and Out-degree distributions including other centralities like betweenness, closeness, pagerank, clustering coefficient, diameter, etc to identify all the top Web Services in the network for the growth and success of people creating their own Mashups by using the influential web APIs. We also found the Pearson correlation to measure the linear dependency between these centralities. We observe that all these centralities have a positive correlation, exhibiting a linear dependence relationship between them.

We also analyzed the network to detect if it exhibits a small world behavior, but it may not be considered one due to relatively small diameter and small clustering coefficient. It is also not a bipartite graph. An interesting aspect of this network is the tree structure and how the Mashups use the Web APIs.

The research presented in this thesis could be taken further in three directions. Firstly it could be useful to repeat this analysis on an evolving network to identify the top web services of the network. Furthermore, we hope to give a detailed comparison between the software, complex networks and the Web Service Networks.

Secondly, despite many adopting the Web Service Network concept and perceiving its usefulness, it still remains as a challenge and much work remains before seeing the Mashup applications in a fullygrown stage.

Thirdly, researchers interested in this topic can study networks online and can broaden the metric analysis. Then following the concept of small world, it could be interesting to find why these networks have a very small diameter and clustering coefficient. It would be just an encouraging sign to observe how the Mashup ecosystem will develop a substantial level of novelty and surprise.

Bibliography

[1] Sheila A McIlraith, Tran Cao Son, and Honglei Zeng. Semantic web services. *Intelligent Systems, IEEE*, 16(2):46–53, 2001.

[2] Graham Cormode and Balachander Krishnamurthy. Key differences between web 1.0 and web 2.0. *First Monday*, 13(6), 2008.

[3] Raven McCrory Wallace. A framework for understanding teaching with the internet. *American Educational Research Journal*, 41(2):447–488, 2004.

[4] San Murugesan. Understanding web 2.0. *IT professional*, 9(4):34–41, 2007.

[5] Jacob Benesty, Jingdong Chen, and Yiteng Huang. On the importance of the pearson correlation coefficient in noise reduction. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(4):757–765, 2008.

[6] Paul Miller. Web 2.0: Building the new library. *Ariadne*, 45(30):10, 2005.

[7] Jin Yu, Boualem Benatallah, Fabio Casati, and Florian Daniel. Understanding mashup development. *Internet Computing, IEEE*, 12(5): 44–52, 2008.

[8] Arto Salminen and Tommi Mikkonen. Mashups—software ecosystems for the web era. In *International Workshop on Software Ecosystems, CEUR*, volume 879, pages 18–32, 2012.

[9] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[10] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. Web services. In *Web Services*, Data-Centric Systems and Applications, pages 123–149. 2004.

[11] Michael Weiss, Solange Sari, and Nadia Noori. Niche formation in the mashup ecosystem. *Technology Innovation Management Review*, (May 2013: Technology Evolution), 2013.

[12] Michael Weiss and GR Gangadharan. Modeling the mashup ecosystem: Structure and growth. *R&d Management*, 40(1):40–49, 2010.

[13] E Michael Maximilien, Hernan Wilkinson, Nirmit Desai, and Stefan Tai. A domain-specific language for web apis and services mashups. In *Service-oriented computing–ICSOC 2007*, pages 13–26. Springer, 2007.

[14] Djamal Benslimane, Schahram Dustdar, and Amit Sheth. Services mashups: The new generation of web applications. *Internet Computing, IEEE*, 12(5):13–15, 2008.

[15] Palie Smart, John Bessant, and Abhishek Gupta. Towards technological rules for designing innovation networks: a dynamic capabilities view. *International Journal of Operations & Production Management*, 27(10): 1069–1092, 2007.

[16] Mark EJ Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005.

[17] Sergi Valverde and Ricard V Solé. Hierarchical small worlds in software architecture. *arXiv preprint cond-mat/0307278*, 2003.

[18] Sergei N Dorogovtsev and José FF Mendes. *Evolution of networks: From biological nets to the Internet and WWW.* Oxford University Press, 2003.

[19] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4):175–308, 2006.

[20] Stefan Bornholdt, Heinz Georg Schuster, and John Wiley. *Handbook of graphs and networks.* Wiley Online Library, 2003.

[21] Christopher R Myers. Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs. *Physical Review E*, 68(4):046116, 2003.

[22] Makoto Ichii, Makoto Matsushita, and Katsuro Inoue. An exploration of power-law in use-relation of java software systems. In *Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on*, pages 422–431. IEEE, 2008.

[23] Alex Potanin, James Noble, Marcus Frean, and Robert Biddle. Scale-free geometry in oo programs. *Communications of the ACM*, 48(5): 99–103, 2005.

[24] Shuli Yu and C Jason Woodard. Innovation in the programmable web: Characterizing the mashup ecosystem. In *Service-Oriented Computing–ICSOC 2008 Workshops*, pages 136–147. Springer, 2009.

[25] Emden R. Gansner and Stephen C. North. An open graph visualization system and its applications to software engineering. *SOFTWARE - PRACTICE AND EXPERIENCE*, 30(11):1203–1233, 2000.

[26] Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989.

[27] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11): 1129–1164, 1991.

[28] Ulrich Fößmeier and Michael Kaufmann. Drawing high degree graphs with low bend numbers. In *Graph Drawing*, pages 254–266. Springer, 1996.

[29] Michael Kaufmann and Roland Wiese. Maintaining the mental map for circular drawings. In *Graph Drawing*, pages 12–22. Springer, 2002.

[30] Achilleas Papakostas, Janet M Six, and Ioannis G Tollis. Experimental and theoretical results in interactive orthogonal graph drawing. In *Graph Drawing*, pages 371–386. Springer, 1997.

[31] Devi R Gnyawali and Ravindranath Madhavan. Cooperative networks and competitive dynamics: A structural embeddedness perspective. *Academy of Management review*, 26(3):431–445, 2001.

[32] Monique Bardawil. Key player identification in the mashup ecosystem. *Open Source Business Resource*, 2011.

[33] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[34] Linton C Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1979.

[35] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

[36] Beno Gutenberg and Charles F Richter. Frequency of earthquakes in california. *Bulletin of the Seismological Society of America*, 34(4):185–188, 1944.

[37] BA Ivanov, G Neukum, WF Bottke, and WK Hartmann. The comparison of size-frequency distributions of impact craters and asteroids and the planetary cratering rate. *Asteroids III*, 1:89–101, 2002.

[38] Edward T Lu and Russell J Hamilton. Avalanches and the distribution of solar flares. In *Bulletin of the American Astronomical Society*, volume 23, page 1044, 1991.

[39] Mark E Crovella and Azer Bestavros. Self-similarity in world wide web traffic: evidence and possible causes. *Networking, IEEE/ACM Transactions on*, 5(6):835–846, 1997.

[40] Damian H Zanette and Susanna C Manrubia. Vertical transmission of culture and the distribution of family names. *Physica A: Statistical Mechanics and its Applications*, 295(1):1–8, 2001.

[41] Vilfredo Pareto. *Cours d'economie politique*. Librairie Droz, 1964.

[42] D Price. Statistical studies of networks of scientific papers. In *Statistical Association Methods for Mechanized Documentation: Symposium Proceedings*, volume 269, page 187. US Government Printing Office, 1965.

[43] George Kingsley Zipf. Human behavior and the principle of least effort. 1949.

[44] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[45] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4): 581–603, 1966.

[46] Alex Bavelas. Communication patterns in task-oriented groups. *Journal of the acoustical society of America*, 1950.

[47] Mark EJ Newman. A measure of betweenness centrality based on random walks. *Social networks*, 27(1):39–54, 2005.

[48] Ulrik Brandes. A faster algorithm for betweenness centrality*. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[49] Paul W Holland and Samuel Leinhardt. Transitivity in structural models of small groups. *Comparative Group Studies*, 1971.

[50] Karl Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58(347-352):240–242, 1895.

[51] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[52] Sergi Valverde, R Ferrer Cancho, and Richard V Sole. Scale-free networks from optimal design. *EPL (Europhysics Letters)*, 60(4):512, 2002.

[53] Christoph Schmitz. Self-organization of a small world by topic. In *LWA*, pages 303–310. Citeseer, 2004.

[54] AA Gorshenev and Yu M Pis'mak. Punctuated equilibrium in software evolution. *Physical Review E*, 70(6):067103, 2004.

[55] Alessandro PS De Moura, Ying-Cheng Lai, and Adilson E Motter. Signatures of small-world and scale-free properties in large computer programs. *Physical review E*, 68(1):017102, 2003.

[56] Sergi Valverde and Ricard V Solé. Logarithmic growth dynamics in software networks. *EPL (Europhysics Letters)*, 72(5):858, 2005.

Vita Auctoris

NAME:	Avani Gade
PLACE OF BIRTH:	Hyderabad, India
YEAR OF BIRTH:	1990
EDUCATION:	St.Anns High School, Secunderabad, India, 2005
	University of JNTU,B.tech(2007-2011)
	University of Windsor, M.Sc., Windsor, ON, 2014