Exploration and application of complex graph properties

by

Zi Yuan Chen

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

The application of mathematics have played a key role in the development of technological advancements and breakthroughs in sciences. Throughout history, mathematics has provided us with an increasing amount of various sub branches such as discrete maths, applied maths, cartesian geometry, algebra, calculus and many more. All of which can be applied to the real world, whether it is for construction, physics or day to day life.

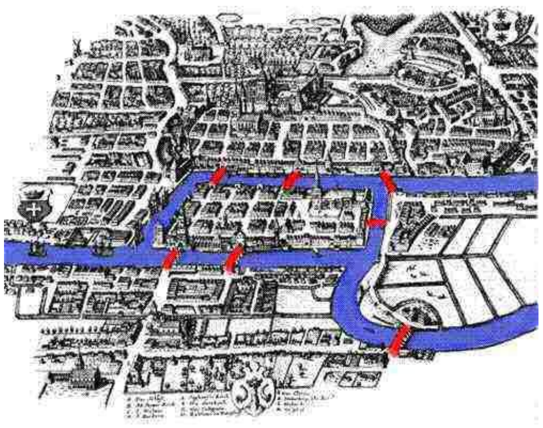
A key branch under discrete mathematics is graph theory where models can be developed that represents relationships between different objects. Graphs has a range of uses both in the mathematical world and the real world. They can be used to visually represent large sets of numerical data so that different properties can be derived from the graph such as clustering of certain areas, the connectivity between vertices or edges and the correlations that they may have. They are also widely known as networks with some examples such as a friendship network, business networks and even a food chain levels. These are the areas that I will explore along with a way to visualise them by using python. Analysing specific properties that may influence the visualisation of the graphs and thus the outcome of the relationships between the vertices.

1.2 History

Initially, graph theory was introduced in 1735 as a form of solution to the seven bridges of Königsberg problem which solved by Leonard Euler[35]. This famous problem involved an island within Königsberg that had a river, Pregel, surrounding the island via a fork, there were seven bridges that crossed this river from the island to other major landmasses of Königsberg, Prussia. The island had four bridges, two north, two south onto the mainland, one bridge to a neighbouring island and the neighbouring island itself had two bridges which totals to the seven bridges stated by the problem, see Figure 1.1a.

Due to the location in which the island was situated, the problem was to determine whether a route exists that manoeuvres through all the bridges exactly once and must return to the starting location. Leonard Euler proceeded to analyse the problem by evaluating the only the key areas, this was the land masses and the bridges. Other information such as the sizes of the island, bridge type or length were irrelevant. Consequently, the problem could be portrayed via dots and lines to give a simplistic view (Figure 1.1b), once developed, this was known as vertices which denoted the key interest and the edges that are incident to the vertices to denote the connections/relations between them.

By removing the irrelevant information, Euler was able to simplify and visualise the problem thus discovering the fact that for there to be a solution, each vertex must have an even number of edges incident to the vertex (even degree) as you require one edge for entering and another for exiting otherwise not all edges will be part of the final path. All vertices in the Königsberg problem have odd degree, thus what's known as an *Eulerian path* (a path that traverses all edges exactly once) doesn't exist for this problem and since a Eulerian path doesn't exist then a *Eulerian circuit* cannot exist either (a Eulerian path that returns to the starting vertex). Therefore Euler's solution proves that there is no solution to the seven bridges of Königsberg and is regarded as the first proof in relation to graphs. This led to the birth of graph theory.


(a) Königsberg and the seven bridges, from MacTutor Archive [38]

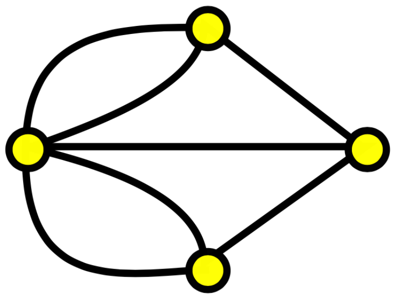

(b) Graph representation to Königsberg problem from online source [41]

Figure 1.1: The original seven bridges of Königsberg and it's graph representation that only focusses on the key details and disregards the irrelevant information. Achieved by the use of vertices and edges.

After Eulerian paths and cycles were introduced, the next famous puzzle in relation to graph theory was invented in 1857 and was known as the Icosian Game[8] by William Rowan Hamilton. The objective of the puzzle was to find a cycle that visits all vertices exactly once and returns to the starting vertex. This type of cycle is later called a *Hamilton cycle* along with the definition of a *Hamilton path* which doesn't have the requirement to return to the starting vertex.

In the mathematical world, Leonard Euler is known for the *Euler's identity* within complex mathematics which states that for a real number x, $e^{ix} = \cos(x) + i\sin(x)$. This has been crucial in many subject areas such as in physics and engineering. Additionally, in 1850, Euler uncovered another formula to be known as *Euler's polyhedra formula* which states that $F + V - E = 2$ where F denotes the number of faces, V as the number of vertices and the number of edges as E of this graph model. As polyhedrons can be depicted as graphs, algebraic topology benefited from Euler's polyhedron formula where more complex surfaces could be studied such as the surface of a torus. Based upon this formula, the *Euler characteristic* was formalised to describe the topological characteristic of various complex surfaces with it's formula as $F + V - E = 2 - 2g$ where g is denotes the number of "holes" the surface has (formally known as the genus).

Furthermore, graph theory has assisted in problems such as the four-colour map problem in the 1850s that states whether all the countries can be coloured with only four colours on a map such that no adjacent countries were coloured with the same colour. In which the solution wasn't found until 1972 by Kenneth Appel and Wolfgang Haken[31] through the assistance of a computer. An example of a four colour problem is colouring the counties in the UK with only four colours and the solution for this is shown in Figure 1.2.

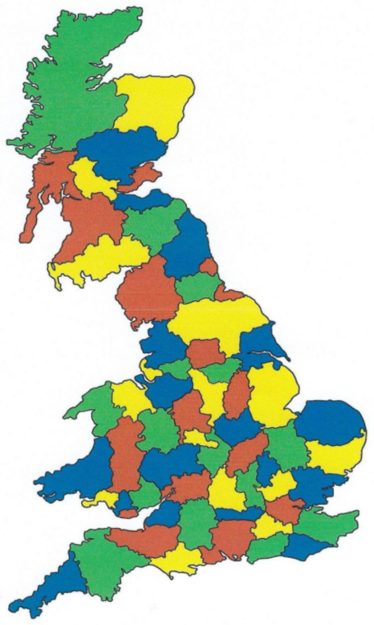


Figure 1.2: The solution for the four colour map problem with regards to the counties in the UK sourced from Robin Wilson [44].

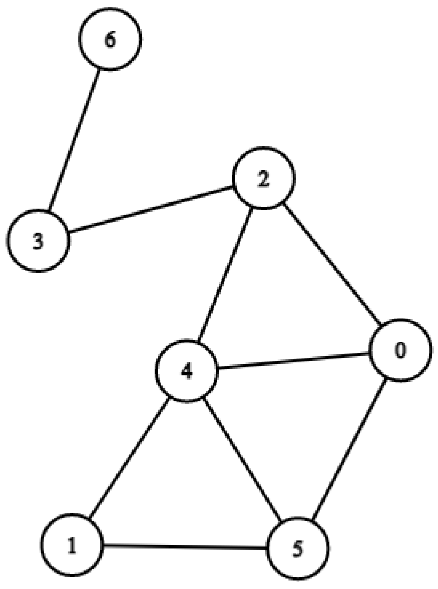
The Chinese postman problem is another such graph theory problem where you must find the shortest path that uses all the edges in the graph at least once. A similar alternate version is called the travelling salesman problem where you find a shortest path that uses all edges exactly once in the graph and must end at the starting vertex. These such problems are used in Linear programming to find optimal solution in routing or pathing between locations. Variants of the CPP (Chinese postman problem) includes undirected Chinese postman problem (UCPP) which contains different restrictions depending on the subset of edges (see the paper [22]) and Chinese Postman Problem with load-dependent costs (CPP-LC[11]) that includes weights upon the edges so that an optimal route can be generated. This can be widely used and applied to problems such as the best route to lower vehicle CO_2 emissions. So these problems can be applied to the modern day scenarios and are still valuable to companies now.

Therefore graph theory studies have been researched extensively since 1735 with many beneficial factors brought into the real world. This is possible due to the versatile nature that data structures in the real world can be represented as a mathematical structure as graphs/networks. Examples of what they can represent ranges from simple relationships between people to the complex structure of the brain by studying it's anatomic structure and assigning vertices according to the sections of the brains and edges as the links between them. The links are typically representations of the neurons in the brain. Further detail of brain mapping into graphs can be read by the paper [19]. Therefore by using graph models, various patterns and correlations can then be derived to generate graph properties that can be studied to develop useful information and possible improvements to the whole graph.

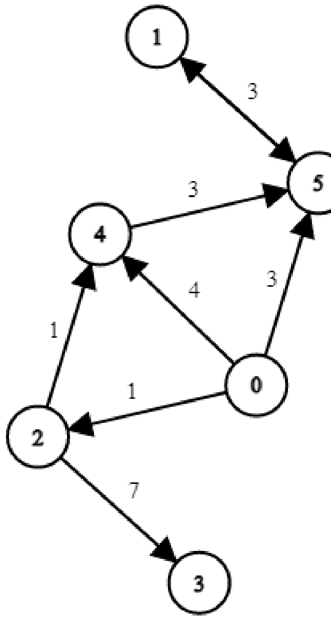
1.3 Basic notation and terminology

Graphs or networks are mathematical constructs that are formed by a collection of vertices and edges. Vertices V represents individual objects such as land masses, companies, houses, people etc. Edges E represents the connections between the vertices such as their relationship, flow of water, supply chain etc. Sets $V(G)$ or V and $E(G)$ or E forms the graph G and can be written as $G = (V, E)$ with E being a subset of $V \times V$ so an edge $e \in E$ can be written as $v_1 v_2$ if e connects v_1 and v_2 where $v_1, v_2 \in V$. There's variation among the graphs as they can be directed or undirected, the edges may carry weights and they may contain self loops. Figure 1.3a and 1.3b shows simple graphs, one of which is undirected and another which is directed. A graph $H = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subset V$ and $E' \subset E$.

Order is the mathematical term that represents the number of the vertices in the vertex set $V(G)$. *Size* is number of vertices in the edge set $E(G)$. The *degree* of a vertex, denoted by $deg(v)$ is the number of edges that are connected (otherwise known as *incident*) to the vertex, discussed previously in Euler's solution to the seven bridges of Königsberg problem. Additionally, $\delta(G)$ and $\Delta(G)$ represents the minimum and maximum degree in G respectively. G is a *regular* graph if $\delta(G) = \Delta(G)$. Vertices v_1 and v_2 are *adjacent* if there exists an edge $e \in E$ that connects them. Vertex v_2 is also known as a *neighbour* to v_1 and is part of the set $N(v_1)$ which denotes the neighbours of v_1.



(a) An undirected graph with 7 vertices, 9 edges and a average vertex degree of 18/7



(b) A directed graph with 7 vertices and 7 directed edges that contain weights

Figure 1.3: The 2 simple types of graphs within graph theory that are the basic building blocks for more complex structures and theorems.

There are multiple ways to traverse a graph. A *walk* $w = v_1v_2v_3v_4v_5...v_n$ is a sequence of vertices such that $E(w) = (v_1v_2, ..., v_{n-1}v_n)$ where vertices and edges can be repeated. They can be *open* or *closed* depending if the final vertex is equal to the starting vertex. A *trail* is an open walk where no edges are repeated but vertices may be repeated. When all the edges are traversed exactly once, it's known as a *Eulerian trail* (mentioned previously as a Eulerian path) and the graph is called *semi-eulerian* or *traversable.* Similarly for a trail that's closed (returns to the start), then it is known as a *circuit* and if all edges are used then it's called a *Eulerian circuit* (or Eulerian cycle) and the graph is defined to be *Eulerian.* A *path* is a trail but with no vertex repetitions and if the size of the path is equal to the size of the graph then it's a *Hamilton path.* Finally, a *cycle* is a path that ends at the starting vertex and if all vertices are visited exactly once then it's known as a *Hamilton cycle* meaning that the graph is *Hamiltonian.*

When considering network with flows, vertices can be known as nodes and may have capacities that limit the overall flow through the network shown in Figure 1.4. These networks are especially used when considering plumbing, water pipes and even evacuation routes in a building. Within a room there's a capacity that is represented by it's node capacity and the weights of the edges can demonstrate the rate of flow along with it's maximal flow, i.e. when people are evacuating a building, the corridors have a limit to the amount of people that may pass through. Networks can be used to model social network processes through the study of small corporate groups to generate a communications network. This network representation will have a flow of sentiments based on social network theory[46] that is constrained in three ways. Firstly by any existing direct relationships within the group, that will be denoted by vertices. Secondly, the frequency of communication of the relationships defined in the first point. Lastly, the breadth of the existing relationships in the network. Thus, by using graphs and networks, social behaviours with groups or companies can be studied on giving ways to more phycological information through numerical data.

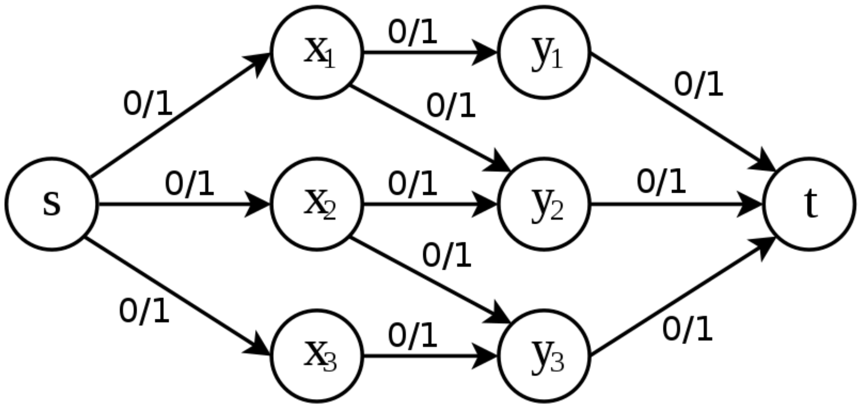
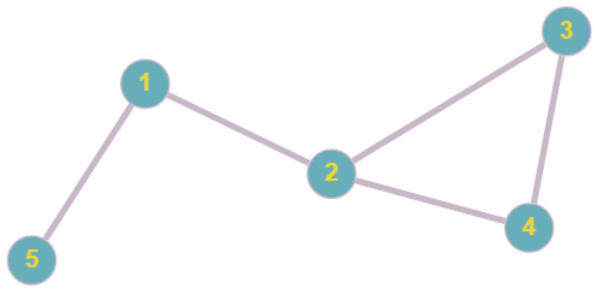


Figure 1.4: A simple network flow with a source node and a sink node, the current flow is 0 and the edge capacities are shown. The image is sourced from Wikipedia [29].

Additionally they can be represented by the use of matrices to enable the use of matrix calculations on the data sets. They're known as *adjacency matrices*[27] and within this matrix holds the number of edges incident to each vertex and it's connection based on the location of this value as the rows and columns represents the vertices. A *weighted adjacency matrix* will instead hold the weights of each edge in the matrix. An $n \times n$ adjacency matrix $A = (a_{ij})$ for $i, j = 1, ..., n$ is defined by $a_{ij} = 1$ if there exists an edge from vertex i to vertex j. A matrix is always symmetric when considering an undirected simple graph as an edge will contribute to both sides of the matrix. Examples of an adjacency matrix along with it's graph representation is shown in Figure 1.5.



$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Figure 1.5: A simple graph along with it's adjacency matrix.

The main focus for future chapters will be on weighted directed graphs and the data that can be perceived by them. Chapter 2 will describe and outline specific graph properties that will be applied to datasets. This means that by analysing the graphs, numerical values can be generated to represent certain factors of the graph. These factors can then be applied to the graph to rearrange it's shape so that further correlation can be identified between all the vertices and edges.

Chapter 2

Graph Properties

There exists many different graph properties that can be applied according to specific graph types. Hence we will focus on properties that will be applied to connected weighted graphs to provide a collection of views targeted for the same layout structure and to enable a wider selection of properties. Hence, unless stated otherwise, the properties described in this chapter will be of the form of connected weighted graphs.

2.1 Trophic Coherence

Firstly the graph property of *trophic coherence*[25], this property determines the stability of the graph and provides trophic levels to the graph and it's individual vertices. Trophic levels are taken from ecology and applied to graphs to generate a height based format for the vertices of a graph. Thus, for a graph $G = (V, E)$ (can be directed), we have V which is the set of all vertices in G and E which is the set of all the edges within G. The graph can be represented with an adjacency matrix A where a_{ij} denotes the elements within the matrix. The standard trophic level definition on vertex level uses the in degree and the out degree of the vertex v_i given by Equation 2.1.

$$k_i^{\text{in}} = \sum_j a_{ij}, \qquad k_i^{\text{out}} = \sum_j a_{ji} \tag{2.1}$$

So, the standard trophic levels for vertices $v_i \in V$ is formulated as:

$$t_i = 1 + \frac{1}{k_i^{\text{in}}} \sum_j a_{ij} t_j \tag{2.2}$$

By ecological convention, $t_i = 1$, if the vertex v_i is *basal*. A vertex is known to be basal if it has no edged directed to it, i.e. $k_i^{\text{in}} = 0$. The trophic level equation can also simply be written in matrix form by 2.3 with $\mathbf{z}$ be defined as $z_i = \max(k_i^{\text{in}}, 1)$ and $\Delta = \text{diag}(\mathbf{z}) - A$.

$$\Delta \mathbf{t} = \mathbf{z} \tag{2.3}$$

All vertices will receive a trophic level if and only if the laplacian Δ is invertible as the row sum of elements in Δ for a vertex that isn't basal is zero. If there are no basal vertices in the graph then Δ will be equal to zero and thus be singular (i.e. no inverse). This is discussed by Samuel Johnson [24] in the investigation of stability and such dynamical features within graphs. So for the standard trophic levels equation to work for the entire graph, there must exist at least one basal vertex meaning that this is a limitation to the definition of trophic levels.

2.1.1 Inclusion of weights and equation improvement

In a weighted graph, edges carry a weight between a vertex u to a vertex v. Weighted matrix W are used as a representation of the entire graph along with incorporating the direction of each edge and if there exists self loops. The elements of the weighted matrix are w_{uv}. If the graph isn't weighted then the edge is valued as 1 if the edge exists and 0 otherwise, i.e. the adjacency matrix of G. The total weight (also known as the strength) for each vertex is defined by the weights into a vertex u and the weights out of vertex u, which is shown by s_v in Equation 2.4. This is essentially the same as the in and out degrees of Equation 2.1 mentioned previously but instead of ones and zeros, the weight values are taken into account. The imbalance of the vertex u is defined by the weights in of a vertex minus the weights out of the vertex shown by i_v in Equation 2.5.

$$s_v = \sum(w_{uv}) + \sum(w_{vu}) \tag{2.4}$$

$$i_v = \sum(w_{uv}) - \sum(w_{vu}) \tag{2.5}$$

Vectors $\mathbf{s}$ and $\mathbf{i}$ holds all the values of the strength and imbalance for all vertices respectively. We let $\mathbf{h}$ be a vector, then the graph Laplacian operator in matrix form is defined by Equation 2.6.

$$\Delta = \text{diag}(u) - W - W^T \tag{2.6}$$

Therefore to get the trophic levels for each vertex with consideration for the additional weights, we solve the system of equations for vector $\mathbf{h}$ as shown by Equation 2.7.

$$\Delta \mathbf{h} = \mathbf{v} \tag{2.7}$$

The values within this vector h corresponds to the trophic levels for the relative vertices. The values are used to illustrate the various trophic levels within a graph to give a hierarchical format visualisation. Trophic levels aren't unique solutions due to the fact that an arbitrary constant can be added to each component of the graph if there are multiple components to generate new levels that would be correct. The benefit that this is that Equation 2.7 can use an arbitrary constant c for a vertex in $\mathbf{h}$. If there are multiple components to the graph then a vertex in each component. A unique solution can be found this way in which the trophic level values can be shifted so that a better graphical display can be generated. For example, have the lowest level be zero.

Trophic level values can also be used to equate the overall *trophic incoherence* of the graph as a whole rather than just looking at the graph on a vertex level. By using the trophic levels from $\mathbf{h}$, the equation for the trophic incoherence is defined by R. S. MacKay, S. Johnson, B. Sansom[24] as:

$$F_0 = \frac{\sum_{uv} w_{uv}(h_v - h_u - 1)^2}{\sum_{uv} w_{uv}} \tag{2.8}$$

The possible shifting of the trophic levels doesn't affect the trophic incoherence hence it is independent. The incoherence is strictly ranged from zero to one. If $F_0 = 0$ then the graph is *maximally coherent* as it would mean that all levels in the graph has a difference of exactly one which means that the graph is perfectly separated into levels. Whereas if $F_0 = 1$ then the graph is *maximally incoherent* and levels are harder to decipher. As F_0 measure the incoherence, by taking $1 - F_0$, this would instead measure the coherence of the graph as they're each others converses.

In conclusion, trophic levels can be applied to weighted directed graph and any subcategories to achieve a hierarchical view of the graph. This eases the visualisation of many datasets and is used to decipher valuable information that may be of use. Through the combination of other graph properties which are described in this chapter, various combinations of these properties will yield different visualisations.

2.2 Clustering Coefficient

The *transitivity* [37] of a graph, also known as *clustering*, is a property of a graph that measured the density of triangles within the graph, where three vertices are connected together. Used to quantify the graph's connectivity strength as it determines the fraction of triangles over the possible triangles that could be formed within the graph. Another perspective is that the coefficient quantifies the probability of a vertex a having an edge to vertex c if ab, $bc \in E(G)$. Thus, the *clustering coefficient* determines how complete the graph is with a value of 1 meaning it is complete. There are two popular introductions of clustering coefficient, the *local clustering* and the *global clustering*. The global clustering coefficient essentially measures the completeness of the graph by measuring the number of exisitng triangles divided by the number of possible triangles in the graph. The local clustering coefficient measures the clustering coefficient for each vertex rather than the whole graph, the measurement is taken by the number of triangles that has a connection to the vertex over the number of triples centred on this vertex. In other words, the local value demonstrates how close the neighbours of this vertex is to being a complete graph (a *clique*).

2.2.1 Clustering for simple graphs

To determine the values of the clustering coefficient for simple connected graphs that are unweighted and undirected, the global clustering coefficient is defined by equation 2.9 where $\sum T$ denotes the number of triangles (closed triplets) and $\sum \tau$ denotes the number of connected triplets in the graph.

$$C = \frac{3(\text{Number of total triangles})}{\text{Number of total connected triples}} = \frac{\sum T}{\sum \tau} \tag{2.9}$$

An alternative equation which was demonstrated by M.E.J. Newman [30] through the studies of complex networks in terms of social networks where the clustering coefficient determines the likelihood that a friend of your friend is also your friend. So, the alternative equation is written in the form of equation 2.10 where $\sum P_2$ denotes the number of paths with length two within the graph.

$$C = \frac{6(\text{Number of total triangles})}{\text{Number of paths with length 2}} = \frac{\sum T}{\sum P_2} \tag{2.10}$$

By considering the vertices of the graph, the local clustering coefficient can be defined to give such a value to each vertex $v \in V(G)$ and is given by the equation 2.11 where i is the index of the vertex. This definition is from [30] and proposed by Watts and Strogats [42] where they analysed small world networks in relation to various real world systems by the use of clustering coefficients and random graphs to formulate certain similarities. Note that if the degree of a vertex is one then the coefficient can be determined as 0, otherwise the equation will lead to 0/0.

$$C_i = \frac{\text{Number of triangles connected to } i}{\text{Number of triples centred on vertex } i} \tag{2.11}$$

Another representation of the global clustering coefficient is to take the averages of all the local coefficients[36]. When the vertices have a degree of 0 or 1 then $C_i = 0$ so clustering coefficient is defined by equation 2.12.

$$C = \frac{1}{n} \sum_i C_i \tag{2.12}$$

Later the clustering coefficients will be used as assistance to model graphs generated off of a dataset. However to provide more accurate values, then weights and directions would need to be taken into account. Thus, the definitions of clustering coefficients must be developed further.

2.2.2 Clustering for weighted graphs

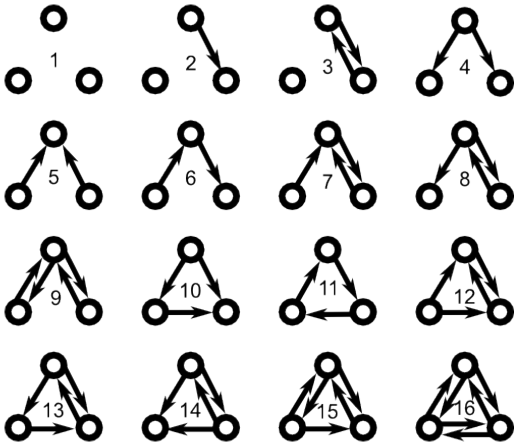
Now by considering graphs as before but weighted, the equations undergo changes. For the instance of weighted graphs, there are multiple different definitions of clustering coefficients, each with slight variation in values depending on the type of graph. This section will summarise a couple of the different definitions for weighted and directed graphs and further detail can be analysed from Tanguy and Anna Levina on weighted directed clustering [15]. In this paper, four different definitions are reviewed which are the Barrat definition, Onnela's definition, Zhang & Horvath and their own continuous definition for weighted graphs. Zhang & Horvath[47] have used their definition of weighted clustering coefficients to analyse gene co-expression networks to review their functionality. Additionally, by soft or hard thresholding, it enables them to determine relationships between the clustering coefficient and gene networks within biology.

A simple idea to associate the clustering coefficient with regards to the edge weights is to define a value w that represents the value of the triplet. w can be the summation of the triplet, the mean of the triplet or another suitable method depending on the purpose. Then equation 2.13 calculates the weighted clustering coefficient[33] where T denote the triangles in the graph and τ, the triples.

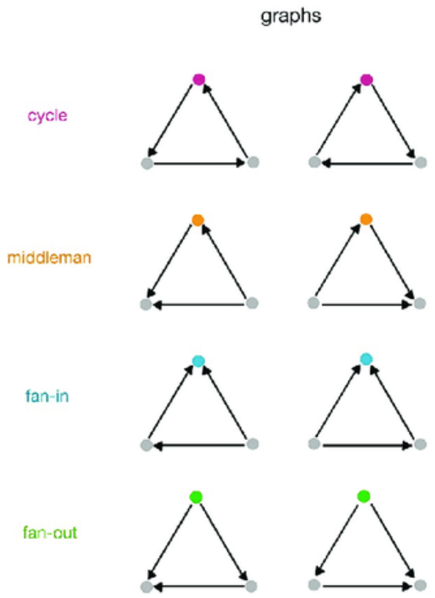
$$C = \frac{\text{Total of closed } w}{\text{Total of } w} = \frac{\sum_T w}{\sum_\tau w} \tag{2.13}$$

Weights can be added trivially through this way, in which it won't affect the equations formalised beforehand. So now considering the addition of directions as well as weights which causes further complexities in the values of clustering coefficients due to the various number of different motifs used to describe the nature of the triangles. For instance, there are sixteen possible motifs for directed graphs of three vertices shown in Figure2.1a. However if we consider only connected triangles, they can be organised into four types of motif groups known as a cycles, Middleman, Fan-in and Fan-out as demonstrated in Figure 2.1b which are used in the study of higher order motifs and synaptic integration by Bojanek, Zhu and Maclean[4]. An interesting result used in this paper is the isomorphisms between the middleman, fan-in and fan-out motifs.

Consideration of the edge's directions yields better accuarancy in the coefficient values. One of the versions mentioned in the paper [15] was Fagios where he introduces the clustering coefficient to binary directed networks which are equivalent to simple directed connected graphs. Firstly the equation for the directed version without the consideration of weights is defined by the ratio of all directed triangles centred on a vertex $i \in V(G)$ and the number of all possible triangles that could be formed with vertex i, These are called t_i and T_i respectively. Before this equation, prior properties of the graph are necessary so that the equation can be easily formulated. Thus, consider a graph $G = (V, E)$ with its matrix representation as the adjacency matrix A along with V_1 as the column vector, dimension n of the



(a) All sixteen motifs of possible connections between three nodes and their directions shown on the edges. Ranges from zero edges to the maximum of six edges. Image sourced from Math Insight[21].



(b) The four named categories of motifs in which all directed connected triangles fall under. Each pair of motifs is listed as the corresponding types which are cycles, middleman, fan-in and fan-out. Image displaced from [32].

graph, of only 1s. A_i is the i-th row of the adjacency matrix. The in-degrees and out-degrees of a graph are the total number of edges going in or out of a vertex $i \in V(G)$ respectively, the total degree is the sum of the in and out degrees shown by equations 2.14.

$$\begin{aligned} \text{in}(d_i) &= \sum_{i \neq j} a_{ji} = (A^T)_i V_1 \\ \text{out}(d_i) &= \sum_{i \neq j} a_{ij} = (A^T)_i V_1 \\ \text{tot}(d_i) &= \text{in}(d_i) + \text{out}(d_i) \sum_{i \neq j} a_{ji} + \sum_{i \neq j} a_{ij} = (A^T + A)_i V_1 \end{aligned} \tag{2.14}$$

When the edge is directed both ways, this degree is the summation of the products of all the edges of vertex v that are bidirectional. Formally can be shown as equation 2.15 with A_{ii} as the i-th element of the diagonal for the matrix product of A.

$$\text{bi}(d_i) = \sum_{i \neq j} a_{ij} a_{ji} = A^2_{ii}$$

This equation is demonstrated by Fagio [14] that measures the clustering coefficient for each vertex with directed edges with the consideration of the eight triangles that this vertex could form shown previously in figure 2.1b.

So this equation can be demonstrated with vertex i and pairs of neighbours j and k that essentially shows that the equation 2.15 calculates the triangles formed by v over the possible triangles with the deduction of $2\text{bi}(d_i)$ as otherwise if vertex i and j had edges directed to each other, this causes a count of two addtional triangles.

$$\begin{aligned} C_i &= \frac{\frac{1}{2}\sum_j \sum_k (a_{ij}+a_{ji})(a_{ik}+a_{ki})(a_{jk}+a_{kj})}{\text{tot}(d_i)(\text{tot}(d_i)-1)-2\text{bi}(d_i)} \\ &= \frac{(A+A^T)^3_{ii}}{2(\text{tot}(d_i)(\text{tot}(d_i)-1)-2\text{bi}(d_i))} = \frac{t_i}{T_i} \end{aligned} \tag{2.15}$$

Weights of the edges can be implemented into Fagio's equation of clustering coefficients by simple using a weighted adjacency matrix W instead of the adjacency matrix A for graph G. Mentioned before, the weights of the triangles can be considered differently however in Fagio's generalisation of the clustering coefficient equation, the mean weights of the triangles are utilised. This is shown by taking a cube root all elements of the weighted matrix W which can be denoted as $W^{[1/3]}$. Therefore by subbing $W^{[\frac{1}{3}]}$ in the place of A in equation 2.15, the weighted directed version can be achieved, formally as equation 2.16.

$$C_i = \frac{(W^{[\frac{1}{3}]}+(W^{[\frac{1}{3}]})^T)^3_{ii}}{2(\text{tot}(d_i)(\text{tot}(d_i)-1)-2\text{bi}(d_i))} = \frac{t_i}{T_i} \tag{2.16}$$

However with this generalisation of the formula, it considers any triangle formed by a vertex i as equal values. This means that the directions in the triangles are meaningless however due to the nature of directed graphs, their directions are what gives the graph it's flow of information and different directions will lead to different interpretations of the graph. Thus an improvement to properly include the directions of edges is to consider each motif separately or in Fagio's case, treat them by considering the 4 types of categories the motifs can fall under mentioned before in Figure 2.1b.

These were cycles, middleman, fan-in and fan-out and by measuring each specific category then we can get more accurate coefficients for the relative pattern. The definition of number of all possible triangles was defined in equations 2.15 and 2.16 as T_i. Similarly with the directed triangles that are actually formed by a vertex i by t_i. These can be both decomposed into the 4 types of motifs which can then be used to create the clustering coefficient for each specific motif. Note that the sum of the clustering coefficient for specific motifs will be the general clustering coefficients for all triangles. The equations are decomposed as follows:

$$\begin{aligned} T_i &= \text{tot}(d_i)(\text{tot}(d_i) - 1) - 2\text{bi}(d_i) \\ &= \text{in}(d_i)\text{out}(d_i) - \text{bi}(d_i) + \text{in}(d_i)\text{out}(d_i) - \text{bi}(d_i) \\ &+ \text{in}(d_i)(\text{in}(d_i) - 1) + \text{out}(d_i)(\text{out}(d_i) - 1) \\ &= \text{cyc}(T_1) + \text{mid}(T_2) + \text{fan-in}(T_3) + \text{fan-out}(T_4) \end{aligned} \tag{2.17}$$

$$\begin{aligned} t_i &= (W^{[\frac{1}{3}]} + W^T)_{ii} \\ &= (W^{[\frac{1}{3}]})^3_{ii} + (W^{[\frac{1}{3}]} W^{[\frac{1}{3}]^T} W^{[\frac{1}{3}]})_{ii} + (W^{[\frac{1}{3}]^T} W^{[\frac{1}{3}]^2})_{ii} + (W^{[\frac{1}{3}]^2} W^{[\frac{1}{3}]^T})_{ii} \\ &= \text{cyc}(t_1) + \text{mid}(t_2) + \text{fan-in}(t_3) + \text{fan-out}(t_4) \end{aligned} \tag{2.18}$$

So both equations can be split according to their different motifs through logical and algebraic reasoning. For T_i, the maximum number of directed triangles for cycles, middleman, fan-ins and fan-outs that can be formed equates to the total number of triangles for a vertex i. For example, when calculating cycles, the maximum number of directed cycles that can be formed with vertex i is it's degree in multiplied by it's degree out, hence $\text{in}(d_i)\text{out}(d_i)$. However if a neighbour of i has an edge to and from the same vertex then this would account to an additional triangle counted hence the subtraction of the bidirectional edges by $-\text{bi}(d_i)$. Notice that middleman and cycles only differ according to the direction of the pairs of neighbours connected to vertex i that forms the triangle hence the reason why $\text{cyc}(T_1) = \text{mid}(T_2)$. Then for the calculations of fan-in motif, it's the multiplications of the in degrees of i and in degrees - 1 (as an edge is already considered) which gives the maximum fan-in motif styled triangles. Similarly for the fan-out motif.

Then for the actual triangles formed, they can be broken down by algebra shown by equation 2.19 which equates to the weighted matrix component seen in the original equation. Hence by algebraic manipulation, t_i can be separated into the 4 motifs shown before.

$$\text{cyc}(t_i) = \frac{1}{2} \sum_j \sum_h w_{ij} w_{jh} w_{hi} + w_{ih} w_{hj} w_{ji} \tag{2.19}$$

$$\begin{aligned} &= \frac{1}{2}((W^{[\frac{1}{3}]})_{(i)} W^{[\frac{1}{3}]} (W^{[\frac{1}{3}]})^{(i)} + (W^{[\frac{1}{3}]})^T_{(i)} (W^{[\frac{1}{3}]})^T (W^{[\frac{1}{3}]}) (W^{[\frac{1}{3}]})^{(i)} \\ &= (W^{[\frac{1}{3}]})_{(i)} W^{[\frac{1}{3}]} (W^{[\frac{1}{3}]})^{(i)} = (W^{[\frac{1}{3}]})^3_{ii} \end{aligned} \tag{2.20}$$

Finally, the clustering can be calculated according to the specific motifs of cycles, middleman, fan-in and fan out. Which can be formally defined as:

$$x(C_i) = \frac{x(t_i)}{x(T_i)} \tag{2.21}$$

where $x = \{\text{cyc, mid, fan-in, fan-out}\}$.

By demonstrating one specific formulation of the clustering coefficient applied to weighted directed graphs, we notice that depending on variations of properties in relation to the triangles, different values may occur for the same graph. Such properties include the various different motifs of the triangles and the consideration of the edge weights. This is what gave way to the multiple different versions as each has their benefit depending on what the graph represents which are thoroughly examined in the paper [15], all versions can also be analysed against each other to determine which has the best performance depending on the ideal requirements of the task[10].

2.3 Centrality

Positioning of the graph is a vital area in displaying and extracting important information whcih can then be assigned to the vertices and edges. The benefit of this is that it may be used to identfy key areas within a graph or network such as the the links between companies and which one has the most influence. This can also be applied to brain networs, the spreading patterns of disease and is useful to characterise sepcific areas to give a new interpretation of the graph. One of the major centrality value is the betweenness of vertices in a graph which will be describved in further detail in the next section.

2.3.1 Betweeness centrality

Betweenness centrality measures the centrality of a graph by using the shortest paths of pairs of vertices. The introduction of this idea was through the view of a comunications network. Where a point(or vertex) of the communicaitons network is deemed to be central if it lies on the shortest path between another pair of points in the network. Alex Bavelas[2] formulalised this idea of centrality where he suggests that a person in a group is in the central position if that person lies on the shortest path between other connecting pairs. With the immplication that this person then holds the power or responsibility for the others due to information exchange that must go through that point. A simplistic way of viewing the betweenness value for a vertex is that the larger the value, the greater number of shortest path connections it has to other vertices. In other words, if the value is large for a vertex, then the travel time from this vertex to other vertices is shorter.

The betweenness centrality will be discussed based on Freeman's interpretation of betweenness centrality meaure[16]. For a simple unconnected and undirected graph $G = (V, E)$, consider all the unordered pairs of vertices $v_i, v_j \in V$ with $i \neq j$. This pair must either be disconnecter or has at least one path connecting them with it's path length based on the number of edges contained within. The path or paths connecting v_i to v_j with the shortest length is known to be the *geodesics*. If the path is larger than one edge (the vertices were adjacent) then vertices in this geodesic are considered to be central. Depending if the vertex is the only central vertex between the pair of vertices determines whether it has complete or partial control of their link. The inutuion of control in betweenness was expressed by Shimbel[39] where work sites are considered as the vertices. Meaning that the vertices with control will be the connecting sites between other sites. Which in terms means that the connecting sites holds responsibility to the other sites and must relay information and resources to them. Figure 2.2 expresses the idea of central vertices and the geodesics between them. Vertices v_1 and v_3 has two geodesics meaning that they share power. Also vertices v_3 and v_6 only has one geodesics through v_4 and v_0 so either v_4 or v_0 can have complete control.

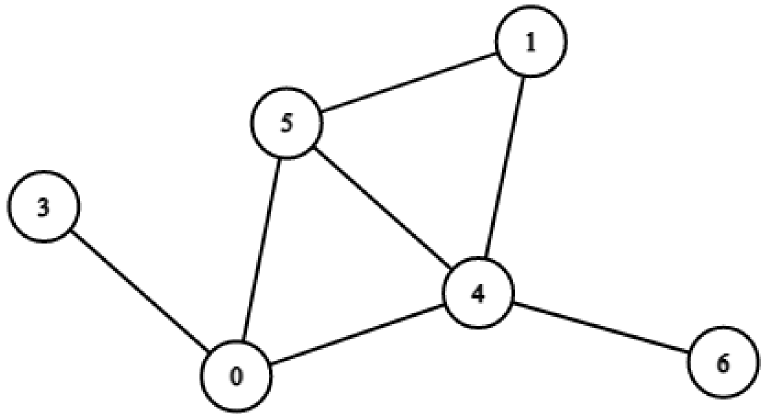


Figure 2.2: A simple graph of six vetices and seven edges. Vertices v_1 and v_3 have two geodesics meaning that the central vertices are v_4 and v_5 who share partial control but v_0 has full control and is most central between v_1 and v_3. Note that v_3 and v_6 cannot be central as they only have one edge each and they only have one geodesics beacuse if v_5 or v_1 is included then it is no longer the shortest path between v_3 and v_6.

To use the geodescides for a pair of vertices, if there are more than one geodescis then they are considered to have equal propability in deciding which one to be used. This is just simply given by $\frac{1}{g_{ij}}$ where g_{ij} are the number of geodesics between vertices v_i and v_j. The formulisation of partial betweenness $b_{ij}(v_k)$ is defined using the idea of geodesics, so for a vertex v_k in G and a vertex pair v_i and v_j, the partial betweenness for v_k can be calculated based on the vertex pair and is given by the equation

$$b_{ij}(v_k) = \frac{1}{g_{ij}}(g_{ij}(v_k)) = \frac{g_{ij}(v_k)}{g_{ij}}(i \neq j \neq k) \tag{2.22}$$

where $g_{ij}(v_k)$ is the amount of geodesics connecting vertices v_i and v_j that inlcude

v_k in it's path. This essentially translate to the probability that the geodesic chosen for v_i and v_j contains v_k. Additionaly, notice that $b_{ij}(v_k) = 0$ if there doesn't exist a path between the vertex pair, v_i and v_j. This can be extended to calculate the centrality of each vertex. So, for a graph of size n, the centrality value for a vertex $v_k \in V$ can be defined by

$$C_B(v_k) = \sum_{i}^{n} \sum_{j}^{n} b_{ij}(v_k) \tag{2.23}$$

where $i < j$. So this defines the betweenness centrality for v_k and it's value increases depending on the amount of geodesics that v_k is a part of. The maximum value[17] was proved by Freeman through the use of a star with the central point as v_k as all vertices are rachable through this central one. Furthermore, this means there are $n(n-1)/2$ paths between all the unordered pairs of the star graph S. With $n-1$ of these connected to the central vertex v_k. So the betweenness centrality for S is

$$C_B(v_k) = \frac{n(n-1)}{2} - (n-1) = \frac{n^2 - 3n + 2}{2} \tag{2.24}$$

And if any new edge is added that doesnt increase the branches of the star, then a new geodesic would form without v_k meaning that the value will fall. Therefore Equation 2.26 expresses the betweenness centrality of any vertex in a graph G by it's represention though a ratio with the maximal value.

$$C'_B(v_k) = \frac{2C_B(v_k)}{n^2 - 3n + 2} \tag{2.25}$$

2.3.2 Generlisation to directed graphs

The key idea of betweenness centrality is to evaluate the graph and produce values according to the shortest paths of all the possible pairs of the graph. The higher the betweenness value, the more the vertex is likely to lie on the shortest paths of any two vertices. As this concept investigates the vertices and shortest paths, weighted edges wouldn't benefit this graph evaluation as weights could cause a pair of vertices to be seen as further apart than they actually are within the graph. In the experimentations of social networks[18], the centrality values are used on undirected graphs however an idea to generlise the betweenness to weighted graphs is to take the weights as indication of the distance of the vertices. Meaning that the geodesics of any pair will be defined on the smallest total value of paths between them rather than the shortest path length as shown on Figure 2.3.

Hence, we discuss the generalisation in this section only to directed graphs and if the graph has weights, then they are not implented to ensure betweenness values will not be influenced. The geodesic proportions of paths from v_i and v_j was defined earlier in Equation 2.22. Consequently, this eqation can be utilised to define pair-dependency of vertices v_i to v_k where the vertext v_i has to depend on vertex v_k in

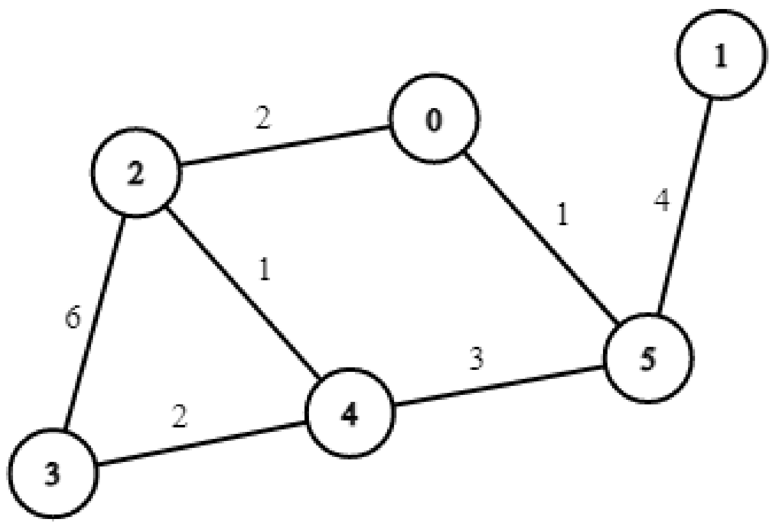


Figure 2.3: a

order to get to other vertices such as v_j on it's geodesics. In other words, v_k acts like a gatekeeper to v_i. Therefore, for a graph with n vertices, the pair dependency is defined to be

$$d^*_{ik} = \sum_{j=1}^{n} b_{ij}(v_k) \tag{2.26}$$

where $i \neq j \neq k$. Matrices can be used to store the pair-depency values to provide an ease of use and is a better representation for all the values. The results can be arranged into the matrix D defined as $D = (d^*_{ik})$. The elements of the matrix measures how much the vertex corresponding to the row number depends on vertex corresponding to the column number to be able to connect to other vertices in the graph. Additionally the betweenness centrality can also be calculated based on this matrix D through the summation of the columns in which the sum will give the betweenness centrality for the column number that represents the vertex. Otherwise shown as

$$\sum_{i=1}^{n} d^*_{ik} = 2C_B(v_k) \tag{2.27}$$

Which means that the betweenness centrality of v_k is double of the pair dependency column sum[43]. This is because for an undirected graph, the upper and lower diagonal matrix are equal due to the symmetry of graph G. The genereralisation for directed graph can then be shown to be

$$C_B(v_k) = \sum_{i=1}^{n} d^*_{ik} \tag{2.28}$$

2.3.3 Other centrality values

Centrality in itself is a larger area within graph theory and other than the betweenness values, there are other similar atrributes such as the closeness centrality, eigenvalue centrality, Katz centrality[26] and the Hyperlink-Induced Topic Search (HITS) centrality. All of which calculate values for the vertices or edges given their positions within the graph by various diffent methods. Interestingly the closeness centrality can be seen as the duality of betweenness centrality as they can be obtained from row and column summatiuons of the depency relation defined in the paper by Brandes, Borgatti and Freeman[5]. Betweenness studies the vertices that acts as bridghes between other geodesics whereas the closeness centraliuty measures the average distance of the shortest paths between any pairs of vertices. A vertex with high closeness value means that the distance to any other vertex is short on average. Closeness centrality[6] can simply be calculated as the inverse of total shortest distance from a vertex i to all of vertices, demonstrated by equation 2.29.

$$C_C(v_i) = \frac{1}{\sum_j^n d(v_i, v} \tag{2.29}$$

where $i \neq j$. The calculation is for simply connected graphs however can be easily expanded to directed and weighted graphs by modying the calculation of the mesaure of distances for the graph in question. I.e. Take the total weights of the path length rather than the path length for weightedness and to consider only correct paths(travelling along an edge in the permiteed direction) when investigating directed graphs.

Therefore by taking these properties, the graph can be rearranged in accordance to their values. This is accomplished by having their property values as their position vector and is done so similarly in the paper by Juan, Alvarez, Villasante and Ruiz-Frau[12] where they relabelled graphs with their normalised centrality values. This ranges from the betweenness centrality to eignevector centrality. Thus a similar idea can be applied along with the other properties explained and studied in this chapter.

2.4 Webpages

One of the largest graphs in the modern world is the network of web-pages, especially Google's search engine. To help navigate through the vast quantity of pages, Brin and Page[7] developed an algorithm known as the Page Rank algorithm. The Page rank gives a quantified meaning to the importance of the webpages and their links to other web-pages. Additionally the Page Rank is known as a centrality value which was discussed briefly in the last section along with the Hyperlink-Induced Topic Search (HITS) which was also created with a similar goal to Page Rank. Both these values are based upon digraphs (directed graphs) and was generated to help with the navigation of the world wide web and provide user's with the highest quality webpages that were also most relvant to their search criteria.

2.4.1 Hyperlink-Induced Topic Search

HITS calculate the ranks of *authorities* and *hubs* in relation to their in-links and out-links[28], i.e. the edges pointing in and the edges going out of the vertices in the graph/network. Authorities and Hubs are assigned to webpages(vertices) depending on their number of in-links and out-links. For the HITS algorithm, the webpages who has lots of in-links pointing to it are denoted as authorities and the webpages who has lots of out-links pointing to other webpages are denoted as the hubs. These can be identified through the use of the HITS algorithm as the calculations are an iterative process where it enforeces the authorities and hubs by bringing the authorities to the surface of the graph. Thus isolating them from other the other webpages. This is achieved by generating the hub and authority values by mutual reinforcement. Based on vertices $i \in V$ from a graph of webpages $G = (V, E)$, the hub value h_i and authority value a_i are first set to a value of 1. Then by Kleinberg's HITS algorithm, they are updates iteratively through the formulas:

$$a_i^(k) = \sum_{j:j->i\&j\neq i} h_j^{(k-1)}, \qquad h_i^(k) = \sum_{j:i\to j\&j\neq i} a_j^{(k)} \tag{2.30}$$

where j are denoted as the links from and to the webpages. The k depicts the k^{th} iteration of the algorithm, so the authority values depend on the previous iteration of hub values and the hubs are calculated based on the current authority values. Which gives the mutual reinforcement of both values. As the graph G can be represented by an adjacency matrix $A = [a_{ij}]$, the formulas can be expressed through matrices[9] and vectors instead as:

$$\mathbf{a}^{(k)} = \mathbf{A}^T\mathbf{h}^{(k-1)}, \qquad \mathbf{h}^{(k)} = \mathbf{A}^T\mathbf{a}^{(k)} \tag{2.31}$$

where the hub and authority values are adapted into vectors $\mathbf{a}^{(k)}$ and $\mathbf{h}^{(k)}$. The vectors are shown as

$$\mathbf{a}^{(k)} = \begin{bmatrix} a_1^{(k)} \\ a_2^{(k)} \\ \vdots \\ a_n^{(k)} \end{bmatrix}, \qquad \mathbf{h}^{(k)} = \begin{bmatrix} h_1^{(k)} \\ h_2^{(k)} \\ \vdots \\ h_n^{(k)} \end{bmatrix} \tag{2.32}$$

Thus accomplishing the goals of generating the values which depict the hubs and authoritiies more clearly within the graph. Although this algorithm is currently defined only for directed weightless graphs so we extend the alrithm to include a weighted edge version.

Normalisation and weights

As the number of iterations increases, the values generated may increase. This means that at some point, the values will become too large to be used for any calculations so they can be normalised such that this doesnt occur. After k iterations, the normalised formulas are demostrated by Equations (2.34) and the general outline for the HITs algorithm is demonstarted by Agosti and Pretto[1] as the following:

```
$\mathbf{a}^{(0)} := \mathbf{u}$ , $\mathbf{h}^{(0)} := \mathbf{u}$;
**for** $k := 1$ to $K$ **do**
        $\mathbf{a}^{(k)} = \mathbf{A}^T\mathbf{h}^{(k-1)}$;
        $\mathbf{h}^{(k)} = \mathbf{A}^T\mathbf{a}^{(k)}$;
        normalise $\mathbf{a}^{(k)}$ such that $\left\|\mathbf{a}^{(k)}\right\| = 1$;
        normalise $\mathbf{h}^{(k)}$ such that $\left\|\mathbf{h}^{(k)}\right\| = 1$;
**end for**
$\mathbf{a} := \mathbf{a}^{(K)}$ , $\mathbf{h} := \mathbf{h}^{(K)}$;
```

where K denotes the maximum number of iterations and $\mathbf{u}$ be the vector for the first iteration of hub and authority values, also known as the base case. The base case for $\mathbf{u}$ will just be the vector of ones known as $\mathbf{1}$ or $\mathbf{e}$ in linear algebra. So then the normalised values after k iterations is given as follows:

$$\mathbf{a}^{(k)} = (\mathbf{A}^T\mathbf{A})^{k-1}\mathbf{A}^T\mathbf{u}, \qquad \mathbf{h}^{(k)} = (\mathbf{A}\mathbf{A}^T)^k\mathbf{u} \tag{2.33}$$

To incorporate weights of the edges into the algorithm, the weighted matrix W for the graph G can be used in place of the adjaceny matrix. Simply by replacing the A in the formulas, the weighted version can be generated as the formulas:

$$\mathbf{a}^{(k)} = (\mathbf{W}^T\mathbf{W})^{k-1}\mathbf{W}^T\mathbf{u}, \qquad \mathbf{h}^{(k)} = (\mathbf{W}\mathbf{W}^T)^k\mathbf{u} \tag{2.34}$$

2.4.2 Page Rank

A widley known alorithm that contributes to the internet of navigation within Google is the Page Rank. The Page Ranks are needed because many different search query's can be entered onto the search engine to produce lots of results that contain the same or similar words to the search enquiry so a method to help organise or prioritise is necessary. The Page Rank is used to rank the web-pages according to the amount of backlinks a page may have and the number citations that reference a particular page. An example of a webpages and links as a graph is shown in Figure 2.4.

For the graph $G = (V, E)$ the webpages can be seen as vertices$v \in V$ with their links as the edges directed edges between pages, the same way to how they were

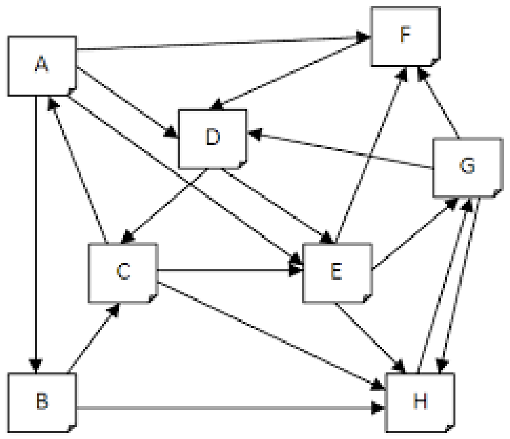


Figure 2.4: A graph representation of webpages and their in and out links. Algoithms such as Page Rank and HITS uses this graph format to help demonstrate their calculations. This image was sourced from the paper by Devi, Gupta and Dixit[13] whom describes and compares Page Rank algorithm to the HITS algorithm.

represented in the HITS formulas. We can let F_v be the set of webpages that v points to, i.e. the forward links. Similarly the backwards links as B_v which is the set of webpages that points to v. The simplified version of Page Rank[34] can then be defined with $N_v = |F_v|$ as the following:

$$PR(v) = c \sum_{w \in B_v} \frac{PR(w)}{N_w} \tag{2.35}$$

where c is a variable used for normalisation pruposes so can be modified accordingly. So Page Rank is calculated by the even distribution of the Page Rank for webpage v among webpages that v points to. The values that links to other webpages from v are then used to calculate their Page Rank. Hence giving an iterative approach to Page Rank as the algorithm travels along the links of the webpages. However if a webpage doesnt have outlinks and only inlinks from other webpages then their Page Rank is never distibuted to others causing it's valley to accumulate. This situation is known as a *rank sink*.

A remedy to having a rank sink is to use a a damping factor to illustrate the probablilty that the user follows the links on the webpage. This takes into the consideration that users could skip pages or go directly to another webpage that wasn't linked throught the url. Hence $(1 - d)$ is considered as the distribution of the Page Rank from webpages that wasn't directly linked to it, i.e. no direct edges between them. Thus for the page v with $b_i \in B_v$, the Page Rank[7] is defined through Equation 2.36.

$$PR(v) = (1 - d) + d(PR(b_1)/F(b_1) + \ldots + PR(b_n)/F(b_n)) \tag{2.36}$$

where F(v) was retrieved from the set F_v for webpage v. Page Rank is applied to each page and it is repeated on further pages until the equation converges. The damping factor adds randomness into the network of webpages so that the rank sink doesnt occur and ensures the convergence isn't reached too quickly. Usually the damping factor is taken to be 0.85.

By using summation, the Page Rank formula can be simply reduced to

$$PR(v) = (1-d) + d \sum_{w \in B_v} \frac{PR(w)}{N_w} \tag{2.37}$$

Personlisation

The algorithm for Page Rank can be modified depending on the use and aims. Such modifications include adjusting the vertex and edge values, modifying the damping factor or to introduce a new variable into the algorithm. An example of a personlisation is the Weighted Page Rank[45] where larger values are assigned to more popular/important webpages rather than the even distribution that occured beforehand. Webpages that are outlinked will instead recieve a value proportional to the pages popularity which are based off of their number of in links and out links. These popularity values of the in links and outlinks are represented as $W^{in}_{v,w}$ and $W^{out}_{v,w}$ accordingly. So $W^{in}_{v,w}$ is calculated by the inlinks of webpage v shown to be I_v and the inlinks of all the webpages that the webpage w references, call this set of pages $R(w)$, as I_p where $p \in R(w)$. This can then be formulased into the equation

$$W^{in}_{v,w} = \frac{I_v}{\sum_{p \in R(w)}} I_p \tag{2.38}$$

Similarly, the same can be done for the outlinks

$$W^{out}_{v,w} = \frac{O_v}{\sum_{p \in R(w)}} O_p \tag{2.39}$$

Where O_v, O_p is defined the same as the inlinks previously but with the use of the outlinks as replacement. Therefore, the Page Rank formula can be modified to include the webpages importance giving the formula

$$PR(v) = (1-d) + d \sum_{w \in B_v} PR(w) W^{in}_{v,w} W^{out}_{v,w} \tag{2.40}$$

So, this formula is focussed more upon the webpages that are visited more frequently by users and ensures they end up with a higher Page Rank.

However, for the purpose of general directed weighted graphs, the edge weights are lost in the formula as the algorithm updates the Page Rank of every vertex in each iteration meaning that the weights can be disregarded and replaced with the Page Ranks instead. To ensure this doesn't happen, the weights of all the edges must be included in the ranks calculation. This is accomplished by summing up the weight values of the in and out edges and incorporating it into the formula as achieved by Equation 2.41.

$$PR(v) = (1 - d) + d \sum_{w \in B_v} \frac{PR(w) w_{(w \to v)}}{N_w} \tag{2.41}$$

where $N_w = \sum_y A_{w,y} w_(w -> y)$ is redefined as the sum of weights of the out linked edges in relation to vertex w.

Whilst HITs and Page Ranks are designed for the search engine, the network of web-pages essentially is a large directed graph that can contain weights hence why the Page Rank can be used on any derivatives of a weighted directed graph. Thus, there are additonal graphical property that can be used to help analyse the structure and linkage of a graph.

Chapter 3

Application of Graph Properties

As various graph properties have been discussed and defined, we now apply these properties onto connected graphs to demonstrate their use. As well as this, they are used to outline and modify the graph such that an alternative layout may be given, revealing correlations between the vertices or edges within the graph. Through the use of Python, I have coded a program to display a graph either generated from a weighted matrix, a adjacency matrix or a graph data set that's pre-existing. To help accomplish this, I have used Tiago's Graph Tool library for python which contains useful documentation and functions to achieve the graph generations as well as other mathematical libraries for complex arithmetic. The general idea is to compare various graph properties by modifying their positions according to the values of their graph properties. For simplicity and the goal of being comparable, we choose the y-axis of the graph to be based upon the trophic coherence and the x-axis to vary between the different properties discussed in the last chapter.

3.1 Early Experimentations

By programming and using tools from the various libraries within python, I was able to generate graphs and define each property value based upon the generated graphs. Then the positioning of each vertex can be modified in relation to the values calculated.

Initially, out of the many pre-existing graphical datasets from Tiago's library, I experimented on a smaller dataset that demonstrates the relationships between karate clubs in a city so that I can test and generate a visualisation of this dataset. This dataset involved 34 karate clubs where the initial graph can be shown by Figure 3.1 with the adjacency matrix in Appendix A.1.

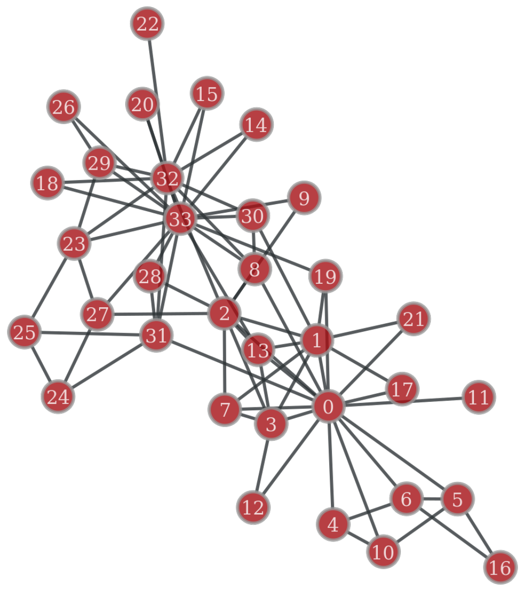
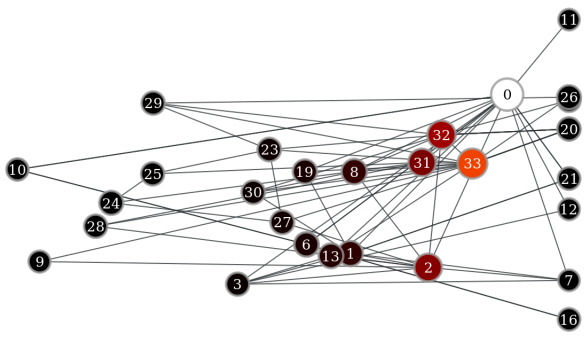


Figure 3.1: The initial graph based on the karate club dataset pre-existing within the library that was generated through the python program. Contains 34 clubs and their connections to one another with no important meaning with the positions of the vertices.

Current positioning of the graph are determined based on the idea that the vertex do not overlap and the connections are all easily visible. So the dataset's positioning has no real benefits other than having good visibility. Any correlations or vital information can not be derived from the initial graph as the positions do not represent anything. The only information that can be derived easily are the certain outliers who only have one edge, vertices 22 and 11. The vertices that have more edges can also be seen such as vertices 32, 33 and 0 under closer examination. But these vertices are difficult to distinguish at a first glance. Thus, we now include various different graph properties discussed in the last chapter to ensure that more can be derived. This can be seen in Figures 3.2 and 3.3 by using the trophic coherence values for each vertex as their y-value and another property value for their x-value.

For karate clubs, as the graph is not directed, the trophic levels do not represent a clear hierarchical format since there is no distinction between "upstream" or "downstream" of information with the edges of the graph. Consequently, the vertices in the karate graph do not have information that passes in one direction. So for vertices such as 1 or 16, they could be recognised as the start or end of the network hence the reason they have the largest and smallest trophic levels. Even if the flow of information is bidirectional, trophic levels are still useful in implementing some structure into the karate dataset. Further datasets involving languages that will be explored will be directional so that trophic levels can be used optimally.


(a) Graph generated by karate dataset, the y-axis represents the trophic coherence values and the x-axis represents the betweenness centrality values for all the vertices. Additionally added colour changes between the x-axis to give a clearer visualisation of the separations.

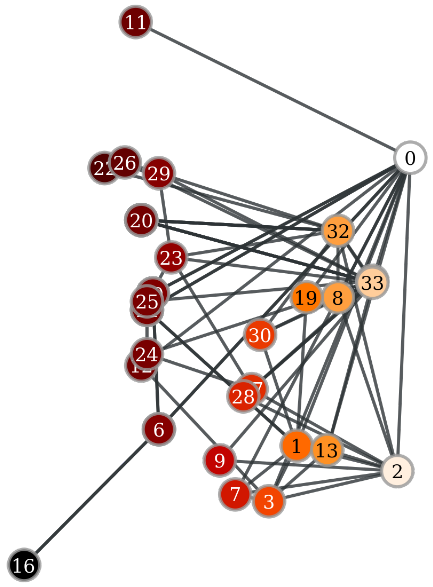

(b) Graph generated similarly to the betweenness graph for the karate dataset but the vertices are plotted with local clustering coefficient as the x-axis and trophic levels as the y-axis.

Figure 3.2: Centrality values as the x-axis

Figure 3.2a shows the karate graph with the x-axis representing the betweenness value. The betweenness value are scaled by a factor or 10 to give a clearer visualisation with larger betweenness value further on the x-axis. Which also means the vertex is more frequently involved among short paths of the connections. In this case, the connections of the clubs. We can see that vertex 0 is the furthest right vertex so is involved in the most short paths between all the vertices of the karate graph. On the other hand, vertices such as 11, 7, 16 and more are clubs who are on the outskirts with no proper connections to other clubs. Can be seen on the left side on the figure. Therefore, in relation to this dataset, the clubs with larger betweenness are the clubs who are more centralised in a city and have more meaningful links to others.

We compare this to another centrality value, the closeness values who also has been scaled by a factor of 10. Figure 3.2b represents this in place of the betweenness value. Through comparison of both, vertices are positioned similarly to betweenness. This is because betweenness are values when considering all vertices within the graph whereas closeness considers all neighbours of a specific vertex. In other words, betweenness measured the control a vertex has over the flow of information through the entire graph, whereas closeness measures the control over the flow of information with vertices in close proximity (i.e. neighbours). Therefore the vertices on the right of both the betweenness and closeness graph would be the most important/largest clubs. Additionally, clubs such as 19 who have a larger closeness compared to betweenness means that it is important to the clubs in close proximity of itself, in other words, the club is the most important/largest within its local area.

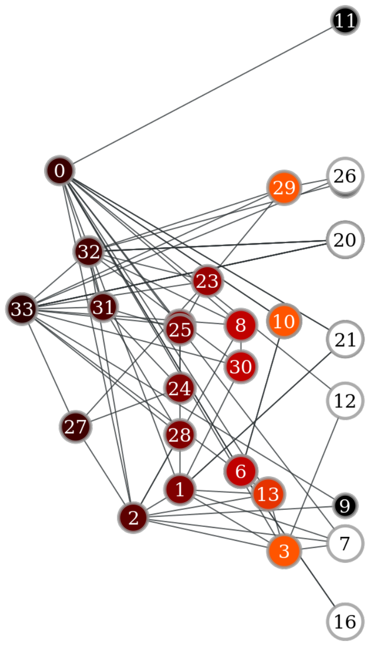


Figure 3.3: Graph generated similarly to the betweenness graph for the karate dataset but the vertices are plotted with local clustering coefficient as the x-axis and trophic levels as the y-axis.

Figure 3.3 is shown with local clustering coefficient as the x-axis instead of betweenness. After manipulation the vertices positions, notice that the clubs with less connections to the major clubs (vertices of high degree) are visually seen to the left in both graphs as these vertices would have a low clustering coefficient as well as a low betweenness. On Figure 3.3, the vertices with the best connections to major clubs are seen further to the right which we see are the karate clubs 26, 20 and 12 etc. However vertices with high betweenness seen before such as vertex 33 instead have a smaller local clustering as their club has connections to smaller clubs, decreasing the overall value of it's own connections. So this means that clubs on the right have quicker access/communication with better clubs and are the closest to them.

All the values for each club can be shown in table format, Table 3.1, where apart from the initial column, each column is one of the different graph property value.

Table 3.1: Table containing all the values calculated for each vertex of the graph.. The order is in the club number depicted in the first column.

Club	**Trophic Levels**	**Betweenness**	**Closeness**	**Local Clustering**
0	0	0.437645803	0.568965517	0.15
1	2.12094015	0.053873557	0.485294118	0.333333333
2	2.30804964	0.152263709	0.559322034	0.244444444
3	2.53039414	0.011961881	0.464788732	0.666666667
4	1	0.000631313	0.379310345	0.666666667
5	2	0.029987374	0.38372093	0.5
6	2	0.029987374	0.38372093	0.5
7	2.47969196	0	0.44	1
8	1.02851103	0.056737013	0.515625	0.5
9	2.22893206	0.000847763	0.428571429	0
10	1	0.000631313	0.379310345	0.666666667
11	-1	0	0.366666667	0
12	1.53039414	0	0.370786517	1
13	2.15210654	0.04159151	0.507692308	0.6
14	0.45900156	0	0.370786517	1
15	0.45900156	0	0.370786517	1
16	3	0	0.284482759	1
17	1.12094015	0	0.375	1
18	0.45900156	0	0.370786517	1
19	1.02788172	0.02936057	0.492537313	0.333333333
20	0.45900156	0	0.370786517	1
21	1.12094015	0	0.375	1
22	0.07623827	0	0.34375	0
23	0.73415591	0.018244949	0.392857143	0.4
24	1.44596889	0.002209596	0.375	0.333333333
25	1.05781988	0.003840488	0.375	0.333333333
26	0.03492233	0	0.358695652	1
27	1.70452843	0.02170214	0.452054795	0.166666667
28	1.75702472	0.001794733	0.445945946	0.333333333
29	0.1140399	0.003869048	0.38372093	0.666666667
30	1.30422637	0.014727633	0.458333333	0.5
31	0.90660502	0.140348425	0.540983607	0.2
32	0.53811913	0.182157888	0.515625	0.181818182
33	0.92088243	0.275055616	0.540983607	0.116666667

This is the early experimentations of the positioning of vertices within the graph so instead of using simple datasets, I will generate the datasets based upon various different languages as well as their sentence structure. To understand this further, words in languages must be given a rank which can be demonstrated through Zipf's Law discussed in the next section.

3.2 Zipf's Law

Zipf's law analyses the natural languages and the frequency of words that appear in them. Alternatively, Zipf's Law[20] is generally seen as the frequencys of specific events are inversely proportional to their rank that is determined through this law. The law was proposed by George Kinbgsley Zipf when researching the various frequencies of words within the English language. The law states that the r^{th} most frequent word in the language has a frequency of $f(r)$ that has a relation with the inverse of r where r is the *frequency rank* for the word and $f(r)$ as the frquency of the word in the corpus examined (The *corpus* means the collection of written text).

$$f(r) \propto \frac{1}{r^{\alpha}} \tag{3.1}$$

This is the scale for $\alpha \approx 1$ and means that the most frequent word in the examined text which is $r = 1$ has it's frequency of appearance to 1, the next most frequent word which is $r = 2$ has a frequency appearane of $\frac{1}{2^{\alpha}}$ and so on. This Zipf's law can be drawn on a graph to show a relation and when $log(f)$ is drawn against $log(r)$, the graph generates a curve that closely resembles a straight line with a slope of -1. This is known as Zipf's curve and later in the 1960s, this was reinforced by the law being correct for smaller corpora[40]. However the curve varies depending on the corpora as expected and the higher ranking words deviated more from the straight line. Therefore, Mandelbrot derived a generalisation for Zipf's law to adjust to the frequency distributions within the different languages. Mandelbrot proposed to adjust the rank by a constant β, demonstrated by

$$f(r) \propto \frac{1}{(r+\beta)^{\alpha}} \tag{3.2}$$

Generalisation of Zipf's law can then be applied to various different corpus of languages so that a frequency distribution can be viewed for the corpus. An example of this can be seen in Figure 3.4.

So words in a corpus has a systematic relationship between their rank in their occurence table and their frequency that they appear in their corpus. Meaning that their are words within languages that are used more commonly such as "the", "or", "of" that account for most of the word occurences and other words such as "xylophone" and "accordion". A larger corpus is studied by Bentz, Kiela, Hill and Buttery [3] where they study zipfs law for Old English and Modern English. They study the frequency and ranks of each word and then compare them between the old and new English. In doing so, for Old English, the words "and" is ranked first with a frequency of 1731 whereas in Modern English, "the" is ranked first with a frequency of 1775 and "and" is second instead with a frequency of 1024. By looking at more words and the comparisons between them, Old English has a larger number of distinct types whilst Modern English had less but this meant that they had higher frequencies within their first 100 words. In conclusion Zipf's Law is a useful tool as

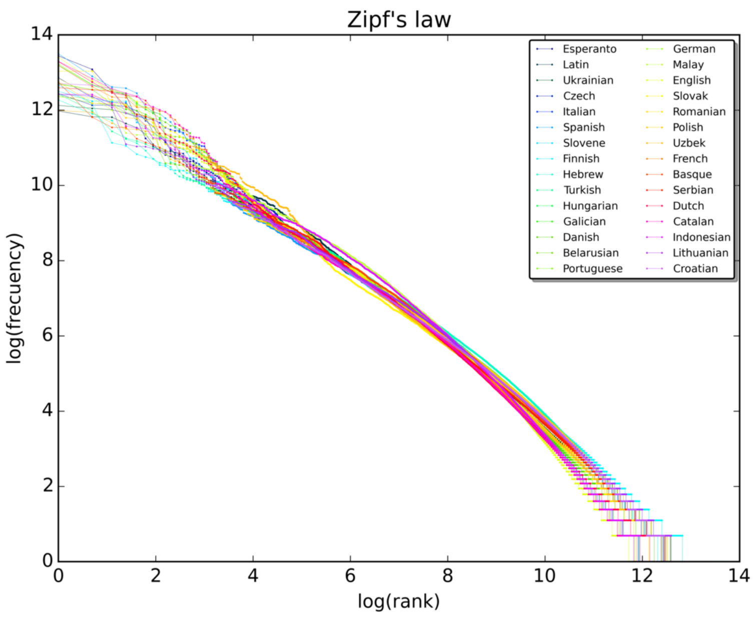


Figure 3.4: The plot of Zipf's law containing 30 different language corpus generated from the first 10 million words in each language from Wikipedias. The image was sourced from Wikipedia[23].

languages tend to follow Zipf's curve in terms of their frequencies and rank meaning that by following an existing corpus of language, the data can be extrapolated and used in other corpus's to determine similarities between the known and the unknown texts. As well as to use it to determine the types of words that are deemed to be most common in a language. We will use Zipf's Law when comparing texts of different languages in further experiments.

a	7	0	0.28174	0	**kingdom**	1	0.027993	0.01017	0
acquaintances	1	0.135998	0.018304	0	**left**	1	0.727384	0.009838	0
also	1	0.303191	0.007227	0	**lived**	1	-0.19525	0.01559	0
and	9	0.253523	0.304283	0.057143	**lives**	1	0.618904	0.022175	0
as	2	0.242183	0.022354	0.333333	**might**	1	0.417113	0.008503	0
bathing	1	-0.18498	0.047444	0	**none**	1	1.364857	0	0
be	3	0.449879	0.095843	0	**not**	2	-0.49384	0.064906	0
beautiful	1	0.326277	0.023314	0	**of**	4	0.801328	0.188965	0
before	1	0.14996	0.025352	0	**on**	1	0.428216	0.012338	0
bid	1	0.28077	0.011625	0	**once**	1	0.26579	0.00544	0
bore	1	0.113036	0.022239	0	**one**	1	0.714174	0.0224	0
bring	1	0	0.059647	0	**only**	2	-0.02083	0.062283	0
but	3	0.018472	0.03931	0	**ordained**	1	0.054413	0.016009	0
by	1	0	0.058069	0	**other**	1	0.592159	0.029245	0
came	1	-0.19525	0.01559	0	**out**	2	1.004888	0.074518	0
child	2	1.293955	0	0	**past**	1	-0.38382	0.021713	0
contain	1	-0.29869	0.034392	0	**plates**	1	0.069116	0.032401	0
could	1	-0.16199	0.06438	0	**provided**	1	0.001655	0.030823	0
daughter	2	-0.19599	0.043738	0	**queen**	3	0.226072	0.093925	0.1
day	1	0.731605	0.030297	0	**relations**	1	0.109517	0.019819	0
did	1	0.043998	0.022204	0	**said**	2	0.908794	0.050292	0
each	1	0.522436	0.028719	0	**shall**	1	0.450446	0.030297	0
eat	1	0.539867	0.021348	0	**shalt**	1	0	0.059121	0
every	1	0.661882	0.029771	0	**so**	2	0.216075	0.049709	0
favourable	1	0.353118	0.035284	0	**squatted**	1	0.268522	0.011812	0
feast	1	2	0	0	**that**	4	0.436479	0.106106	0.03714
for	2	0.091603	0.078154	0	**the**	9	0.587911	0.279782	0.044118
foretold	1	0.351014	0.015958	0	**their**	1	0.710116	0.021649	0
friends	1	0.18152	0.020345	0	**them**	3	0.228276	0.153123	0.1
frog	2	0.485953	0.079587	0	**there**	3	-0.39051	0.079992	0
from	1	0.627021	0.021874	0	**they**	2	0.384346	0.045086	0
fulfilled	1	0.29992	0.024826	0	**thirteen**	1	0.404049	0.030297	0
golden	1	0.046629	0.031875	0	**thou**	1	0	0.058595	0
gone	1	0	0.057543	0	**thy**	1	0.45158	0.029245	0
great	1	1	0.009888	0	**times**	1	-0.37713	0.021187	0
ground	1	0.420717	0.018217	1	**to**	6	0.452713	0.227705	0.018182
had	4	0.364857	0.16175	0.035714	**twelve**	1	0.024142	0.031349	0
happened	2	0.095101	0.032509	0	**was**	1	0.020545	0.046918	0
has	1	0	0.057017	0	**water**	1	0.420717	0.018217	1
he	4	0.108827	0.112105	0.047619	**we**	1	0.400668	0.019524	0
her	1	0.452146	0.028719	0	**were**	1	0.006771	0.029771	0
himself	1	-0.10354	0.034918	0	**when**	1	0.512195	0	1
his	2	0.037515	0.039246	0	**who**	1	0.567433	0.005729	0
in	2	-0.37045	0.039218	0	**wise**	1	0.537434	0.02267	0
into	1	0.195961	0.011039	0	**wish**	1	0.451013	0.029771	0
it	3	0.382995	0.062455	0	**women**	1	0.486956	0.023196	0
joy	1	0.172563	0.023776	0	**world**	1	1.587911	0	0
kind	1	0.351701	0.042138	0	**would**	1	0.527692	0.022701	0
king	2	0.169862	0.176086	0.166667	**year**	1	0	0.056491	0

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

$$\begin{pmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{pmatrix}$$

```
0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0
0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 1 1
0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1
0 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 0 0 1 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 1 1
1 0 0 1 0 1 0 1 1 0 0 0 0 0 1 1 1 0 1
1 0 0 1 1 1 0 0 1 0 0 1 1 1 1 1 1 1 0
```

Chapter 4

Analysing Languages

Through the earlier experimentations with the graph generations based on specific properties, improvements have been made. This includes normalisation of the values into the range of 0-10 rather than using a scale factor. Which ensures that all the graphs will be similarly shaped and have the same axis size for an easier visual comparison. Also I've included page rank as well as the other graph properties used before. Finally the datasets I will use will be generated through my program by an input of text. Afterwards will be converted into graphs where the words represent the vertices and the edges are directed to the next word in the order of the text. A factor that affected the dataset was the punctuation so to achieve congruent data, the punctuation are stripped unless they are sentence enders such as full stops, question marks, exclamation marks etc.

Therefore, the graph we look upon are directed graphs that can have cycles based upon different languages.

4.1 Text Corpus

The best way in comparing the results to other languages is to have a dataset that is based on the same text extract. Thus, the text extract that is chosen should be simple and well know, in my case, I have chosen to use the popular story in all languages, "Sleeping Beauty". To ensure that the same version is used, the Grimm Brothers version is utilised where the original was in German. So using translations of this story, graphs are generated graphs based on each language translation. The languages being English, German, French, Spanish, Polish, Japanese, Chinese, Russian and Dutch which total to nine different languages. Additional instead of using the entirety of the story, the first two paragraphs are used so that the graphs are not overwhelmingly dense which tells the story as follows:

> In times past there lived a king and queen, who said to each other every day of their lives, "Would that we had a child!" and yet they had none. ... There were thirteen of them in his kingdom, but as he had only

provided twelve golden plates for them to eat from, one of them had to be left out.

In conclusion the original nine dataset created in this way can be used for graph property calculations and firstly we study the English language.

4.2 English

Chapter 5

Conclusion

Bibliography

[1] Maristella Agosti and Luca Pretto. A theoretical study of a generalized version of kleinberg's hits algorithm. *Information Retrieval*, 8(2):219–243, 2005.

[2] Alex Bavelas. A mathematical model for group structures. *Human organization*, 7(3):16–30, 1948.

[3] Christian Bentz, Douwe Kiela, Feli Hill, and Paula Buttery. Zipf's law and the grammar of languages: A quantitative study of old and modern english parallel texts. *Corpus Linguistics and Linguistic Theory*, 10(2):175–211, 2014.

[4] Kyle Bojanek, Yuqing Zhu, and Jason Maclean. Cyclic transitions between higher order motifs underlie sustained asynchronous spiking in sparse recurrent networks. *PLoS computational biology*, 16:e1007409, 09 2020.

[5] Ulrik Brandes, Stephen P Borgatti, and Linton C Freeman. Maintaining the duality of closeness and betweenness centrality. *Social networks*, 44:153–159, 2016.

[6] Ulrik Brandes and Christian Pich. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos*, 17(07):2303–2318, 2007.

[7] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[8] Stephen C. Carlson. graph theory. Dec 2022.

[9] Alexander Chatzigeorgiou, Nikolaos Tsantalis, and George Stephanides. Application of graph theory to oo software engineering. In *Proceedings of the 2006 international workshop on Workshop on interdisciplinary software engineering research*, pages 29–36, 2006.

[10] G.P. Clemente and R. Grassi. Directed clustering in weighted networks: A new perspective. *Chaos, Solitons & Fractals*, 107:26–38, 2018.

[11] Ángel Corberán. The chinese postman problem with load-dependent costs. *Transportation Science*, 52(2):370–386, March 2018.

[12] Silvia de Juan, Andres Ospina-Alvarez, Sebastián Villasante, and Ana Ruiz-Frau. A graph theory approach to assess nature's contribution to people at a global scale. *Scientific Reports*, 11(1):9118, 2021.

[13] Pooja Devi, Ashlesha Gupta, and Ashutosh Dixit. Comparative study of hits and pagerank link based ranking algorithms. *International Journal of Advanced Research in Computer and Communication Engineering*, 3(2):5749–5754, 2014.

[14] Giorgio Fagiolo. Clustering in complex directed networks. *Physical Review E*, 76(2), aug 2007.

[15] Tanguy Fardet and Anna Levina. Weighted directed clustering: Interpretations and requirements for heterogeneous, inferred, and measured networks. *Phys. Rev. Res.*, 3:043124, Nov 2021.

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

[17] Linton C Freeman et al. Centrality in social networks: Conceptual clarification. *Social network: critical concepts in sociology. Londres: Routledge*, 1:238–263, 2002.

[18] Linton C Freeman, Douglas Roeder, and Robert R Mulholland. Centrality in social networks: Ii. experimental results. *Social networks*, 2(2):119–141, 1979.

[19] Michael Hart, R.J.F. Ypma, Rafael Romero-García, Stephen Price, and John Suckling. Graph theory analysis of complex brain networks: New concepts in brain mapping applied to neurosurgery. *J. Neurosurg.*, 124:1665–1678, 06 2016.

[20] William L. Hosch. Zipf's law. 2009.

[21] Math Insight. The 16 possible network motifs involving three nodes. [Online; accessed 8 February, 2023].

[22] Stefan Irnich. Undirected postman problems with zigzagging option: A cutting-plane approach. *Computers & Operations Research*, 35(12):3998–4010, December 2008.

[23] Sergio Jimenez. Zipf 30wiki en labels, 2015.

[24] Samuel Johnson. Digraphs are different: Why directionality matters in complex systems. *Journal of Physics: Complexity*, 1(1):015003, 2020.

[25] Samuel Johnson, Virginia Domínguez-García, Luca Donetti, and Miguel A Munoz. Trophic coherence determines food-web stability. *Proceedings of the National Academy of Sciences*, 111(50):17923–17928, 2014.

[26] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[27] U. Knauer. *Algebraic graph theory : morphisms, monoids, and matrices / by Ulrich Knauer.* De Gruyter studies in mathematics ; 41. 2011.

[28] Amy N Langville and Carl D Meyer. A survey of eigenvector methods for web information retrieval. *SIAM review*, 47(1):135–161, 2005.

[29] Chin Ho Lee. Maximum bipartite matching to max flow, 2009. [Online; accessed 5 February, 2023].

[30] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, jan 2003.

[31] Koji Ohnishi. Towards a brief proof of the four-color theorem without using a computer: theorems to be used for proving the four-color theorem. *Artificial Life and Robotics*, 14(4):551–556, Dec 2009.

[32] Scientific Figure on ResearchGate. Cyclic transitions between higher order motifs underlie sustained asynchronous spiking in sparse recurrent networks, 2020. [Online; accessed 12 February, 2023].

[33] Tore Opsahl and Pietro Panzarasa. Clustering in weighted networks. *Social networks*, 31(2):155–163, 2009.

[34] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[35] James Powell and Matthew Hopkins. *A Librarian's Guide to Graphs, Data and the Semantic Web.* Chandos Information Professional Series. Chandos Publishing, 2015.

[36] Liudmila Ostroumova Prokhorenkova and Egor Samosvat. Global clustering coefficient in scale-free networks, 2014.

[37] Thomas Schank and Dorothea Wagner. Approximating clustering coefficient and transitivity. *Journal of Graph Algorithms and Applications*, 9(2):265–275, 2005.

[38] School of Mathematics and Statistics, University of St Andrews, Scotland. Königsberg bridges, 2000. [Online; accessed 2 February, 2023].

[39] Alfonso Shimbel. Structural parameters of communication networks. *The bulletin of mathematical biophysics*, 15:501–507, 1953.

[40] Elvira I Sicilia-Garcia, Ji Ming, Francis J Smith, et al. Extension of zipf's law to words and phrases. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[41] Katie Steckles. The bridges of königsberg, 2018. [Online; accessed 2 February, 2023].

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

[43] Douglas R White and Stephen P Borgatti. Betweenness centrality measures for directed graphs. *Social networks*, 16(4):335–346, 1994.

[44] Robin Wilson. Four colours suffice, 2017. [Online; accessed 3 February, 2023].

[45] Wenpu Xing and Ali Ghorbani. Weighted pagerank algorithm. pages 305–314, 2004.

[46] Wayne W. Zachary. Modeling social network processes using constrained flow representations. *Social Networks*, 6(3):259–292, 1984.

[47] Bin Zhang and Steve Horvath. A general framework for weighted gene co-expression network analysis. *Statistical Applications in Genetics and Molecular Biology*, 4(1), 2005.

Appendix A

Karate Club

A.1 Karate Club Adjacency Matrix

This is the adjacnency matrix generated for the karate club dataset.

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

Appendix B

Langauages

B.1 Text Corpus

In times past there lived a king and queen, who said to each other every day of their lives, "Would that we had a child!" and yet they had none. But it happened once that when the queen was bathing, there came a frog out of the water, and he squatted on the ground, and said to her: "Thy wish shall be fulfilled; before a year has gone by, thou shalt bring a daughter into the world."

And as the frog foretold, so it happened; and the queen bore a daughter so beautiful that the king could not contain himself for joy, and he ordained a great feast. Not only did he bid to it his relations, friends, and acquaintances, but also the wise women, that they might be kind and favourable to the child. There were thirteen of them in his kingdom, but as he had only provided twelve golden plates for them to eat from, one of them had to be left out.

B.1.1 foo title

Some text