3rd Year Project Report 2007

Drawing Graphs – Extracting the Main Concepts from Text

Author: Gyanisha Choolun

Supervisor: Dr David Rydeheard

Abstract

Whether it is a book, an article, a poem or any piece of text, understanding its essential meaning can be very lengthy and inefficient. This project is concerned with analysing a text document in order to enable people to identify its key concepts. The application created in the course of this project reveals the concepts present in a document pictorially by means of a graph. The positioning of the concepts on the graph allows the user to instantly identify the concepts present in the text and their similarity to each other. Groups of closely related concepts can then be identified using clustering algorithms. Each of these groups consists of concepts that have a similar meaning and they can be represented by a single main concept. The essential meaning of the text is thus revealed through the identification of these main concepts and a summary can be generated to convey the meaning of the text to the user.

Acknowledgements

I would like to thank my supervisor, Dr. David Rydeheard for his help and support throughout the duration of the project. I would like to thank Mary McGee Wood and Craig Jones for their help during the project. I would also like to thank my family and friends for their constant support and motivation.

Table of Contents

1. Introduction

1.1. Project Overview

It is often the case that when people are looking for particular books, news articles or academic papers on the internet that they have a vast amount of information presented to them. They are often forced to go through the tedious process of reading through all the information carefully in order to identify those that are relevant to what they are looking for. If the main concepts in any document were extracted and presented pictorially to the user, this would enable them to understand what the document is about without having to read it.

This report will describe such an approach to understand the meaning of a document by extracting the concepts that most closely describe the document. It describes how graph drawing techniques can be used together with an interactive graphical user interface and clustering techniques in order to effectively identify the most meaningful concepts in a document.

1.2. Aims and Objectives

The main aim of this project is to build an application that enable users to visualize and identify the main concepts in a document. The main objectives of the project are to:

- Create a graphical user interface to be able to present the desired graph representation of a text document.
- Analyse any file that contains text information.
- Display a meaningful 2-D graph representation of the concepts present in the document.
- Allow the user to interact with the interface in order to choose how many main concepts they want to extract from the document.
- Offer the option of automatically identifying the number of main concepts that should be extracted from the document.
- Design and implement methods to identify which of the concepts present in the document represent the key concepts that most closely describe the text and show this on the 2-D graph.

The secondary objectives of the project are to:

- Label and colour the concepts so that they are clearly visible on the graph.

- Summarise the document using the main concepts identified.

1.3. Report Overview

Chapter 2 – Concept Extraction

This chapter will describe the background information about identifying the main concepts in a document. It also covers the background of the techniques used in the new approach.

Chapter 3 – Requirements Analysis

This chapter will outline the main functions required in the proposed system.

Chapter 4 – Design

This chapter will describe the main components of the system and how they interact with each other. It also describes the design of the graphical user interface and the reasons behind adopting that particular design.

Chapter 5 – Implementation

This chapter will describe the implementation of the most interesting and complex parts of the system in depth.

Chapter 6 – Concept Extraction Evaluation

This chapter will measure the effectiveness of the new approach by testing it with different forms of texts. The outputs from the application will be shown by screen shots.

Chapter 7 – Conclusion

This chapter will provide an overview of what was achieved in this project and cover any improvements, limitations and future work.

2. Concept Extraction

2.1. Chapter Introduction

Concept extraction is the process of extracting conceptual information (the main concepts) from text documents. From this conceptual information, we can then understand the content of the document. There already exists techniques for extracting the main semantic concepts in a document and the most common one is Latent Semantic Analysis (LSA) which is based on statistical computations (mathematical matrix decomposition similar to factor analysis) [1, 2].

This project introduces a new technique for extracting conceptual information from a text document. The technique is mainly based on the use of sophisticated graph drawing techniques and clustering algorithms in order to find the main concepts in any document. The technique consists of six main steps which are summarised in figure 2.1.

1. The first step is syntactic processing and it involves breaking the text down into contexts which can be a group of words or sentences. It also involves pre-processing of these contexts such as omitting the words which we want to ignore before conceptual information is retrieved from the document.

2. This step involves constructing the input file for the graph that is in a format suitable for the graph drawing method.

3. The next step involves visually representing all the concepts present in the text using a graph drawing technique which is based on force-directed methods (spring embedders) [3] and energy minimisation algorithms.

4. This step involves the use of clustering algorithms in order to identify groups of related words or concepts (represented as clusters) in the graph.

5. This step identifies the most important or significant concept from each group of concepts. Each of the clusters identified in step 4 will thus have one main concept representing it.

6. The final step involves ranking the main concepts identified in step 5 in order to construct a text summary of the document.

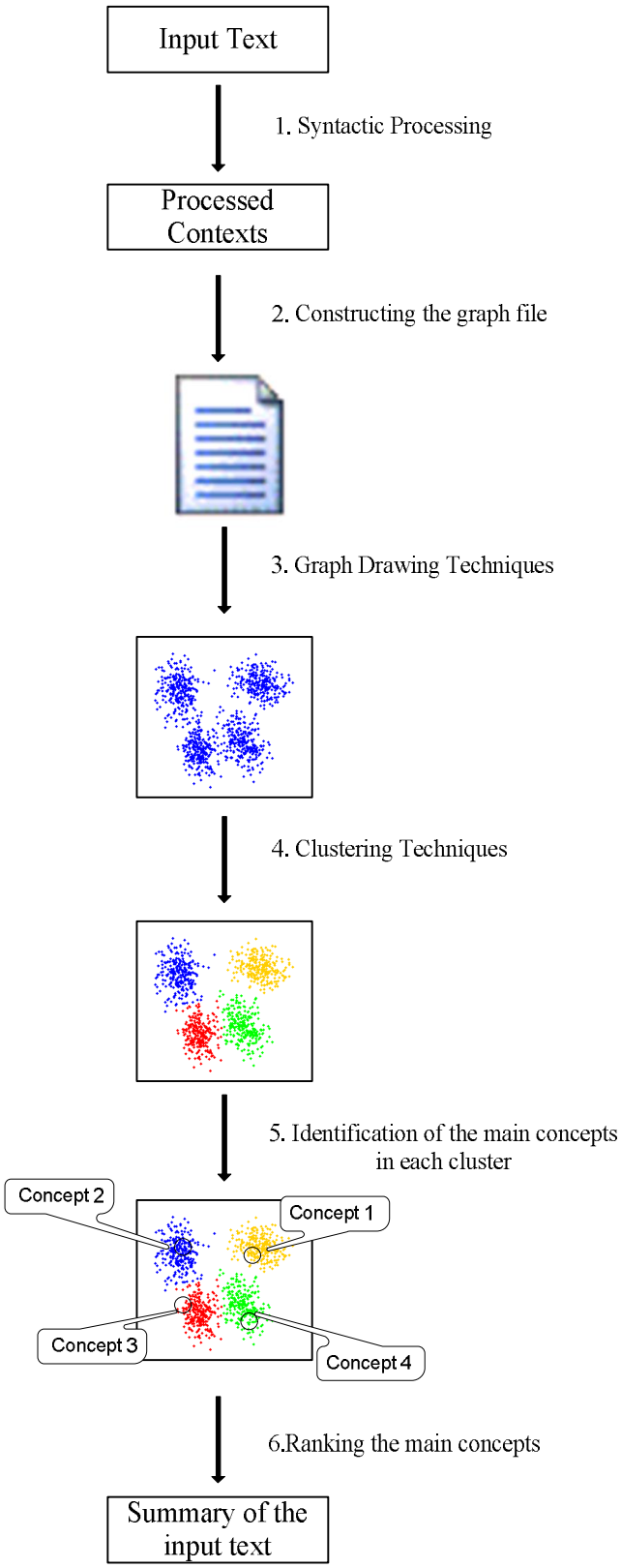

Figure 2.1: Overview of the new technique for extracting conceptual information

In this chapter, the background of the main steps involved in this new technique will be covered and we will also give an overview of an existing technique for extracting conceptual information from documents which is latent semantic analysis.

2.2. Syntactic Processing

Syntactic processing is the first step in our new technique for extracting conceptual information from a document and it involves two main processes. The first process involves breaking the text down into contexts (or propositions). These can be simple sentences or group of words and they simplify the process of concept extraction. The second process involves processing the contexts into a stream of meaningful words that have conceptual meaning in the document.

2.2.1. Constructing the contexts

This is the first step in the syntactic processing process and it involves breaking the text down into contexts (group of words or sentences). It is a known fact that it is not possible for two words to belong to the same grammatical proposition (or context) if they are not closely connected. It is therefore possible to divide the text into these contexts and assume that two words occurring in the same context have a similar meaning and are therefore related. These contexts can be made bigger or smaller by changing the number of words or sentences which represent the context. These contexts will form the basis of concept extraction by allowing us to link closely related words together at a later stage when constructing the input file that will represent the graph.

2.2.2. Processing the contexts

The second step of syntactic processing involves retaining only those words from the contexts which have conceptual meaning in the text. It ensures that the contexts (or the document) are reduced to words that help retrieve the concepts present. There are a number of ways of achieving this and these will be covered in the rest of this section.

Comparison with a list

This is the approach used in this project to filter out words that do not have conceptual meaning in the document. In most documents, it can be noted that words like “if” and “moreover” which are called connectors (words that link together parts of the discourse) do not have any conceptual meaning. Other words that do not have conceptual meaning are adverbs and adverbial phrases (such as “tomorrow” and “probably”). These words are added to a list together with the punctuation marks to be omitted. The contexts are compared with this list and any unwanted words and punctuation marks are filtered out.

Luhn’s upper cut-off

This is another technique to omit words that do not have conceptual meaning in a document.

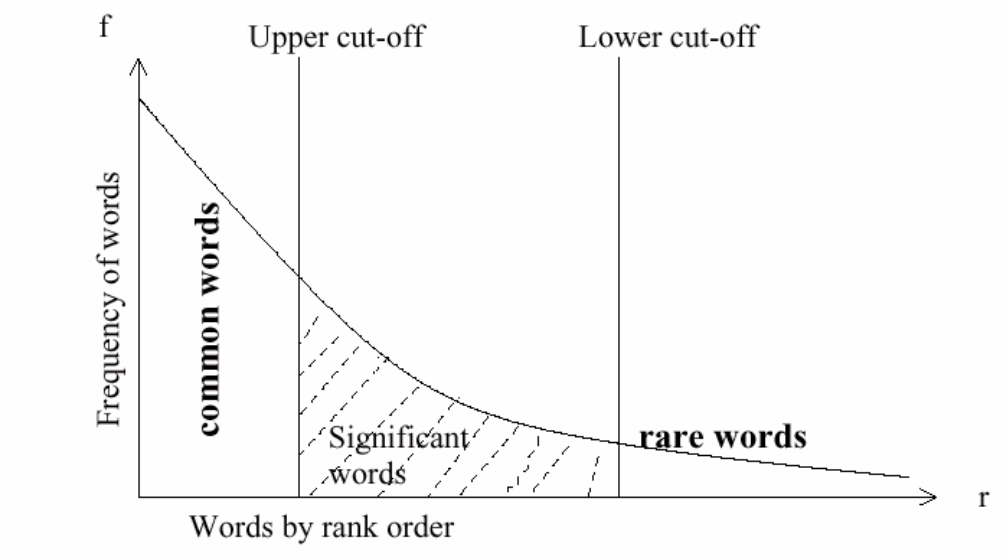

Figure 2.2: Luhn's analysis

Figure 2.2 describes Luhn's analysis [5] which shows how the frequency of a word is related to its significance in a document.

It is noted from the graph that it is mostly high frequency words which do not have significant meaning in a document. Therefore these high frequency words should be omitted when processing a document. These words are again mostly connectors, adverbs and adverbial phrases. The cut-offs have to be established by trial and error and the words below the upper-cut-off are then omitted.

Stemming

The English language consists of many word variants that originate from the same word stem or root and these word variants have a similar meaning. The process of stemming reduces these word variants into a single form or a single root. Stemming consists of suffix removal (such as reducing the word "worker" to "work"), detection of equivalent stems ("absorb" and "absorpt") and plural stemming (such as reducing the word "stories" to "story"). The most common implementation of stemming is the Porter stemming algorithm [6]. One of the problems in the extraction of semantic concepts from a document is that word variants are sometimes treated as different concepts because they do not occur in the same context (group of consecutive words or sentence). Therefore, stemming eliminates this problem by making sure that all word variants are mapped to the same stem and represent a single concept.

2.3. Constructing the Graph File

This is the second step in our implementation and at this stage the contexts have already been reduced sp that they contain only meaningful words. In order to construct the graph file, the contexts need to be converted to a set of relations between the words that have a similar

meaning. In the previous section, it was mentioned that if two words occur in the same proposition (context), then it is most likely that they are closely related. In this step, all the contexts are broken down into relations where each word in a context is related to every other word in the same context. These relations are stored in the graph file. It is also meaningful to keep a count of the frequency of all the words in the text and the number of times two words are related (through contexts) across the whole document as these values will help measure how strongly two words are connected.

2.4. Graph Drawing

Graph drawing is the third step in our new technique for extracting conceptual information. Graph drawing [7] is a form of data visualization and it is the discipline of using models, algorithms and systems in order to derive 2-D or 3-D drawings (or representations) of graphs. A graph consists of a set of vertices and edges that represent the connections between the vertices. There are many ways of drawing a graph by using different layouts and graph drawing tackles the problem of how to draw a graph so that it conveys the most meaning according to the context of the application. Graph drawing is used in applications such as social network analysis, electrical engineering, bioinformatics, project management (PERT diagrams), visualization of hypertext links and UML diagrams among others.

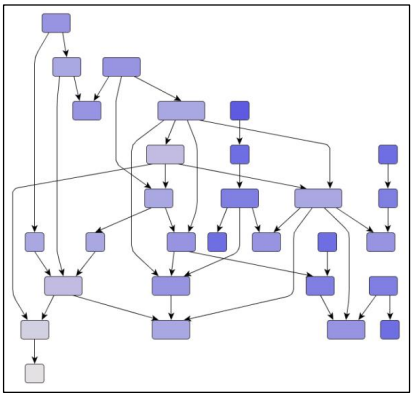

Figure 2.3 (a)

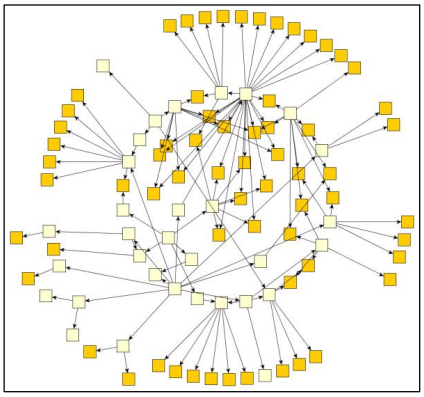

Figure 2.3 (b)

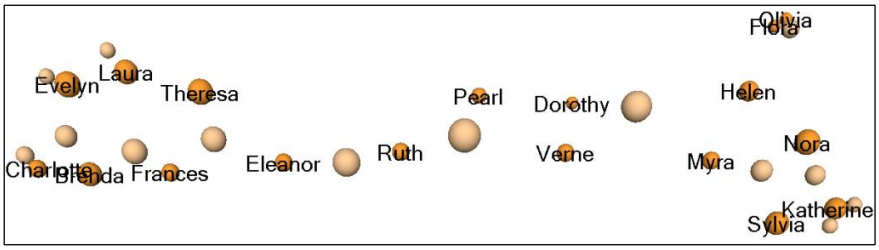

Figure 2.3 (c)

For instance, figure 2.3 (a) [8] shows a hierarchical graph layout which is suited to UML diagramming. Figure 2.3 (b) [8] shows a circular layout which is suited to social network representations and figure 2.3 (c) [9] shows a force-directed layout which is suited to graphs of social networks.

In this project, the force-directed layout is used because this graph drawing technique will draw large graphs in such a way that dense sub graphs are easy to identify. This property is useful because these dense sub graphs will represent semantically related words (main concepts) on the graph.

The two main components of force-directed graph drawing techniques are:

- A system of forces which act on the vertices and edges. This is also called the energy model or force model.

- An algorithm which tries to find a stable minimum energy state drawing (by simulating the evolution of the system over a number of iterations).

The force-directed graph drawing technique being used in this project is called LinLog [10] but there are also other force-directed techniques appropriate in this case namely Simulated Annealing (SA) and Fruchterman-Reingold. Linlog has been chosen because the drawings clearly reveal the structure of the graph and the graph shows clearly separated clusters with interpretable distances between the clusters.

In the rest of this chapter, we will give an overview of the graph drawing technique being used (LinLog) and also give an overview of simulated annealing which is an alternative to the algorithm being used.

2.4.1. Edge-Repulsion LinLog

This is the force-directed graph drawing technique that being implemented in this project. The energy model in this technique is called Edge-Repulsion LinLog model and the system to assign forces is a spring-like force for every pair of nodes (i, j), also called spring embedder [3].

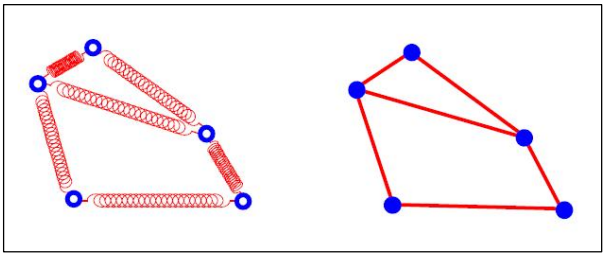

Figure 2.4: System of spring forces [11]

Adjacent vertices (vertices connected by an edge) on the graph are replaced by springs with unit natural length and non-adjacent vertices are replaced by springs with infinite natural length. For every pair of node (i, j), the desired length d_{ij} of each spring is proportional to the distance between the nodes i and j on the graph. The energy-minimisation algorithm used in LinLog is called Barnes and Hut [12].

Edge-repulsion LinLog Energy Model

The energy model in this graph drawing technique differs largely from other force-directed techniques. In most techniques, adjacent graph nodes i.e. nodes that are joined by an edge attract each other and nodes that are not connected repulse each other. In LinLog, the repulsion is based on edges repulsing each other instead of node repulsion.

In order to describe the energy model, a number of assumptions need to be made. The assumptions are:

- A graph G = (V, E) is made up of a set of nodes V and a set of edges E.
- $V^{(2)}$ is the set of all subsets of V which have exactly two elements.
- A drawing p consists of a vector of node positions p_v where v€V.
- Given a node v, its degree deg(v) is the number of nodes that are connected to v by an edge.
- Given a drawing p and two nodes u and v, the distance between u and v is determined by the difference vector pv–pu and is denoted by ||pv-pu||.

The energy $U_{EdgeLinlog(p)}$ of a drawing p is then defined as shown in figure 2.4.

$$U_{EdgeLinLog}(p) = \sum_{\{u,v\}\in E} ||p_u - p_v|| - \sum_{\{u,v\}\in V^{(2)}} \deg(u)\deg(v)\ln||p_u - p_v||$$

A B C D

Figure 2.4: Equation for calculating the energy of a drawing p

The first part of the subtraction in the above equation (A to B) is a calculation of the attraction between adjacent nodes. The second part (C to D) is the calculation for the repulsion between the edges. The repulsion does not mean that entire edges repulse each other. Rather, it means that it is the end nodes of the edges in question that repulse each other. In other words, the repulsion between two nodes is determined by the number of edges that are connected to them (the degree of the node). In this way, each node ends up having an influence on the drawing which is proportional to the number of edges connected to them (the degree). This property of the node can thus be reflected on the graph being drawn by making the size of the node

proportional to its degree. Therefore, in this model, it is noted that symmetry and balance is achieved as both repulsion and attraction are caused by the edges.

Algorithm for Energy Minimization

The energy minimization algorithm used in LinLog is called Barnes and Hut [12]. It is a hierarchical force-calculation algorithm which is based on the idea that the combined effect of a group of particles (distant bodies) can be represented by the effect of the centre of mass of that group. The runtime of the algorithm is of O (n log n). It performs a hierarchical subdivision (decomposition) of the space of particles by using a dynamic divide and conquer approach.

The algorithm consists of three components:

1. Building an octtree for a 3-D space or a quadtree for a 2-D space.

2. The 2-D space decomposition is easier to visualise where each region of the space is divided into 4 sub regions (8 sub regions for 3-D space). If a sub region contains more than one point, it is recursively subdivided and if a sub cube contains no bodies, it is not taken into account as shown in figure 2.5 (a) [13]. A quad tree is then constructed in figure 2.5 (b) [13]. In this process, the algorithm also calculates the centre of mass of each region and associates it with that region.

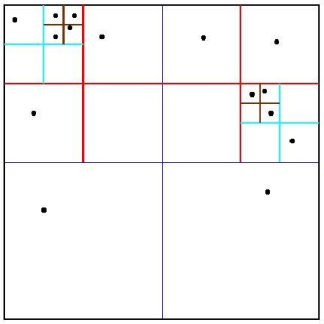

Figure 2.5 (a)

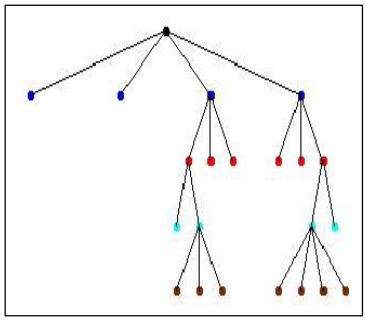

Figure 2.5 (b)

3. Computing the centre of mass of all the cells.

   By traversing the tree bottom up, the centre of mass of each point is calculated from the centres of mass of its children.

4. Calculating the forces on each point.

   The force on each point is a function of the point and the centres of mass of the point and its children. It is calculated by traversing the tree top down, beginning at the root in order for each point to accumulate the total force on the point (by summing up the forces on its children).

2.4.2. Simulated Annealing (SA)

This force-directed technique for graph drawing is an alternative to LinLog and it was introduced by Davidson and Harel [14]. It has not been implemented in this project but it is expected to be a good comparison to the graph drawing output produced by LinLog.

Simulated annealing (SA) is a technique to find a good solution from all the possible solutions to a problem (optimisation problem) by trying random variations of the current solution. This method simulates the annealing process which is the way in which liquids freeze or metals recrystalize into a minimum energy crystalline structure

Each solution s represents a state of a physical system. An energy function E(s) is chosen and it represents the internal energy of the system in that state. E(s) is to be minimized over a number of iterations. The energy function is usually chosen to be the sum of the repulsion among vertices, frame action on the vertices and attraction of edges. The algorithm starts by generating an initial solution which is the initial positions of the nodes on the graph either randomly or according to predefined rules. A control parameter T (also called the temperature) is also initialised. The goal is then to bring the system from the initial state to a state with the minimum possible energy through a series of iterations.

In each iteration, a new solution s' is randomly selected from the set of solutions that are close to the current solution s (the neighbourhood of s). Davidson and Harel chose the neighbourhood of a state s to be the any state that is an exact mirror of the current state but with one vertex moved inside a circle of radius r (where r decreases when T decreases) .The energy of the new solution E(s') is calculated using the energy function chosen at the start. If $E(s') < E(s)$, the change is accepted and the system moves into the new state s'. If $E(s') > E(s)$, the change will be accepted with a probability of $e^{-(E(s')-E(s))/KT}$ following the Boltzmann distribution where K is the Boltzmann constant.

The temperature T is decreased during the search process, thus at the beginning of the search, the probability of accepting moves that result in states of higher energies is high and it gradually decreases, converging to a simple iterative improvement algorithm.

2.4.3. Summary

Fruchterman-Reingold [15] is another force-directed graph drawing technique which is very similar to LinLog. The main difference is that instead of the edges repulsing each other (as is the case in LinLog), in Fruchterman-Reingold it is the nodes that repulse each other.

Most force-directed techniques are similar to Fruchterman-Reingold. These result in graphs where the dense sub graphs are really small and sparse sub graphs are large because the attraction between the nodes in the dense sub graphs is a lot higher than the repulsion between them. The LinLog model restores the balance between attraction and repulsion and this gives a graph with a more uniform density which makes it easier to read.

The figures below show a comparison between a graph drawn using the Fruchterman-Reingold model in figure 2.6 (a) and the Edge-Repulsion Linlog model in figure 2.6 (b) [].

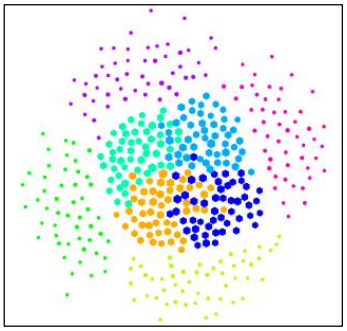

Figure 2.6 (a)
Fruchterman-Reingold

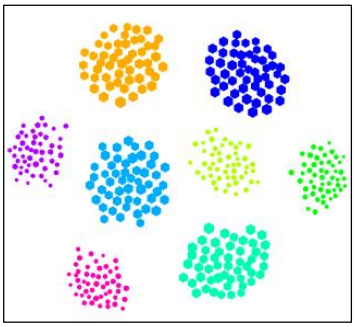

Figure 2.6 (b)
Edge-Repulsion Linlog

2.5. Clustering Techniques

Clustering is the fourth step in the process of extracting conceptual information from a text document. It is the process of determining the intrinsic grouping in a collection of data by finding meaningful structure in the data [17]. More broadly, it can be defined as the process of grouping together objects (or data) where the objects in a particular group are “similar” in some way and “dissimilar” to objects in other groups. These groups are called clusters. For instance, the similarity measure can be distance and therefore two or more objects which are considered to be close together will be part of the same cluster.

Clustering algorithms can be classified into two main categories namely hierarchical and partitional clustering algorithms.

Hierarchical clustering algorithms are subdivided into two main categories which are agglomerative and divisive clustering methods. Hierarchical clustering can be represented by a two-dimensional diagram called a dendogram which represents the grouping or division of data which occurs at each stage of the clustering process. An example of a dendogram is shown in figure 2.7.

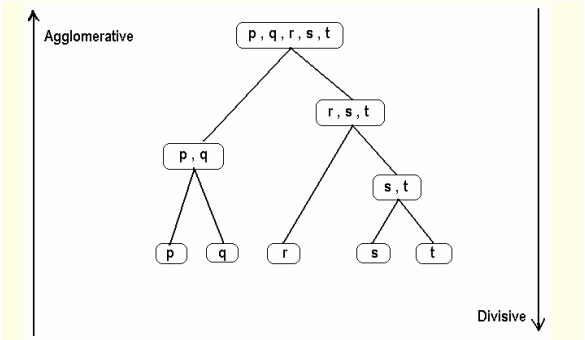

Figure 2.7: Partitional clustering

Partitional clustering algorithms do not produce a structure like the dendogram. Rather, they partition the data into a set of disjoint clusters which is predetermined before the clustering process.

In this project, we have implemented one hierarchical algorithm which is agglomerative hierarchical clustering and K-Means clustering which is a partitional clustering algorithm. An overview of these two algorithms will be covered in the rest of this section.

2.5.1. Agglomerative Hierarchical Clustering

This is a hierarchical clustering algorithm which when given a set of N points, uses an NxN distance (or similarity) matrix derived from the points in order to identify the clusters.

It starts by assigning each point to an individual cluster. Therefore, if there are N points, N clusters will be created. The main steps of the algorithm are:

1. Find the two closest pair of clusters using the distance matrix

2. Merge these two clusters into a single cluster (this step decrements the total number of clusters by one)

3. Repeat steps 2 and 3 until a termination point is reached or until a single cluster with N points is reached

The two most common ways of computing the distance between two clusters in order to find the two closest clusters are single-linkage (also called nearest neighbour) and complete-linkage.

Single-linkage

The distance between two clusters is measured by the distance between the two closest points in these two clusters as shown in figure 2.8.

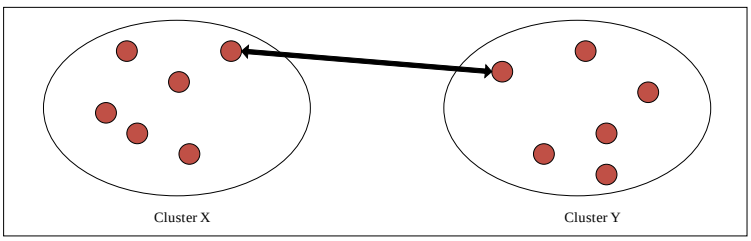

Figure 2.8: Single-linkage

Complete-Linkage

The distance between two clusters is measured by the distance between the two furthest points in these two clusters as shown in figure 2.9.

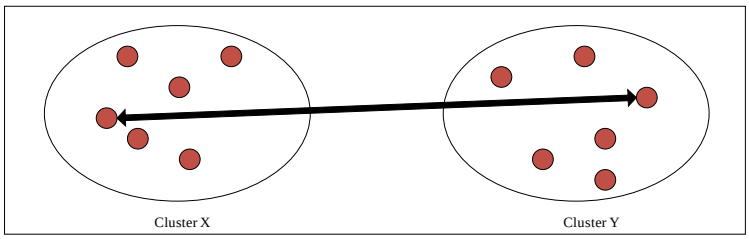

Figure 2.9: Complete-linkage

For example, with the distance matrix in figure 2.10 (a) [19] and using single-linkage, the corresponding hierarchical structure of the points is obtained as shown in figure 2.10 (b) [19] and the algorithm can be stopped at any point.

	BA	FI	MI	NA	RM	TO
BA	0	662	877	255	412	996
FI	662	0	295	468	268	400
MI	877	295	0	754	564	138
NA	255	468	754	0	219	869
RM	412	268	564	219	0	669
TO	996	400	138	869	669	0

Figure 2.10 (a)

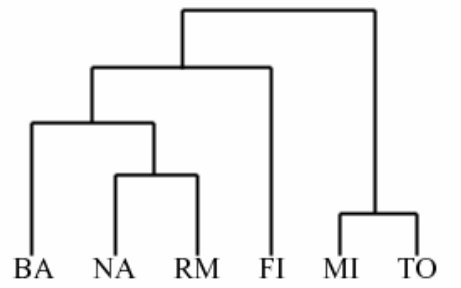

Figure 2.10 (b)

2.5.2. K- Means Clustering

K-means is a partitional clustering algorithm that clusters a set of data or objects into K partitions where K is predefined [20]. It assigns each point to one of the K clusters seeking to minimize the sum of the distances or sum of squared Euclidean distances from each point to the centroid (or mean point) of each cluster. The time complexity of K-Means is O (n) where n is the number of points. It is important to choose a good initial partition for this algorithm to increase the likelihood of the algorithm to find the global minimum value for the sum and reduce the number of iterations.

The following example illustrates how K-Means works assuming K = 5.

1. Given the number of clusters, K centroids are randomly defined and they represent the initial centre locations for each cluster (as shown in figure 2.11 (a)).

2. Each point is then assigned to the nearest centroid and becomes a member of that cluster (figure 2.11 (b)).

3. For each cluster, a new centroid is calculated which is the barycentre of the points in that cluster (figure 2.11 (c)).

4. Steps 2 and 3 are repeated until there is no change in the location of all the centroids.

Figure 2.11 (d) shows the resulting five clusters (coloured with five different colours) after k-means has been run on the dataset [21].

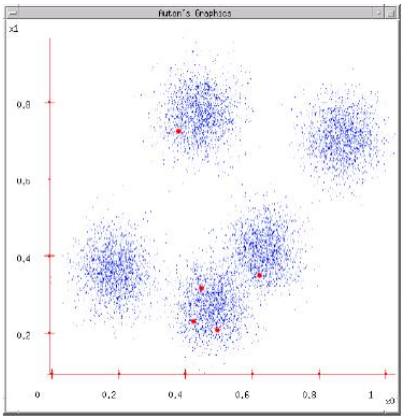

Figure 2.11 (a)

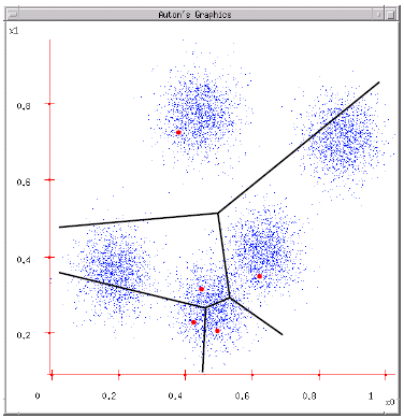

Figure 2.11 (b)

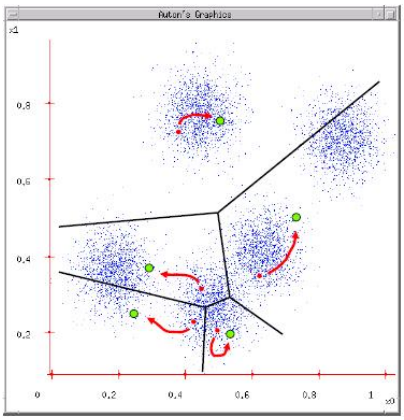

Figure 2.11 (c)

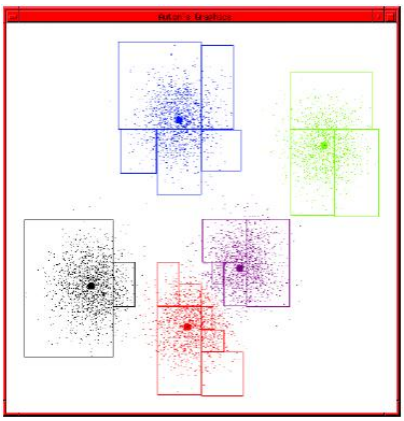

Figure 2.11 (d)

2.6. Identification of Main Concepts

This is the fifth step and it involves determining which concept in each of the clusters identified in step 5 will represent the main concept. In a way, the main concepts summarise each cluster which makes it crucial to identify the correct concept in each cluster. The most logical approach (which has also been implemented in this project) is to identify the word which has the highest number of connections to all the other words in a cluster. There are other alternative ways such as picking the word which occurs with the highest frequency in the text.

2.7. Ranking of Main Concepts

The last step in our implementation involves generating a text summary of the document. In order to achieve this, the main concepts (identified in step 6) need to be ranked according to their importance in the document. This will enable us to position the more important concepts first in the summary and the least important ones at the end. One method is to place the main concepts that have the highest connectivity (through edges) to the other concepts in the text first in the summary. The other method is to summarise the main concepts in descending order of the size of the cluster they represent.

2.8. Latent Semantic Analysis (LSA)

LSA is an existing technique for extracting conceptual information [22]. It has not been implemented in this project but its output will be compared with the output from our new technique in chapter 6. LSA uses no humanly constructed dictionaries, knowledge bases or grammars which makes it very similar to our technique. The rest of this chapter gives an overview of LSA.

The first step in LSA involves representing the document or set of documents in terms of a term-document matrix or term-context matrix where each row represents a term (word) and each column represents a context (which can be a group of words, sentences, paragraphs or documents). Each cell entry in the matrix represents the frequency with which each term (rows) occurs in each context (columns). The entries in the matrix are then weighted according to the importance of the term in that particular context and also according to how much it conceptual importance it has to the document or set of documents. It is assumed that the resulting matrix is a term-context matrix M with t rows (terms) and N columns (contexts).

LSA then decomposes this matrix M into the product of three other matrices K, S and D' by a process called Single Value Decomposition (SVD) as shown in figure 2.12. SVD is a mathematical matrix decomposition technique which is similar to factor analysis [2].

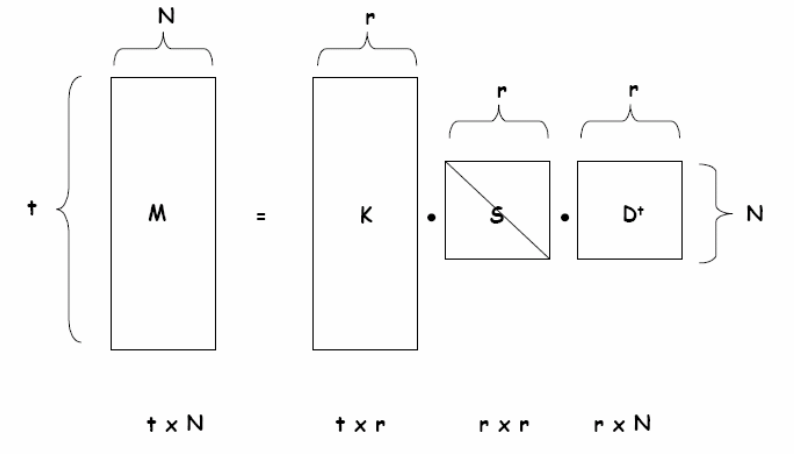

Figure 2.12: Single Value Decomposition

K is an orthonormal[1] matrix whose row values correspond to the row values of M and D' is an orthonormal matrix whose column values correspond to the column values of M. The matrix S is an r x r diagonal matrix with non-zero entries (singular values) only along the central diagonal such that when the three matrices are multiplied, the original matrix M is obtained.

The next step is to reduce S so that the s most important concepts can be obtained from the document or set of documents. The value s will determine the amount of dimensionality reduction and it should be chosen carefully to fit the intrinsic structure of the data in order to get the right number of concepts. This is done by choosing the s highest singular values in S and omitting the other values to get the resulting matrix S_s. The corresponding singular vectors in the matrices K and D' are deleted to get two reduced matrices K_s and D_s'. The resulting matrices are shown in figure 2.13.

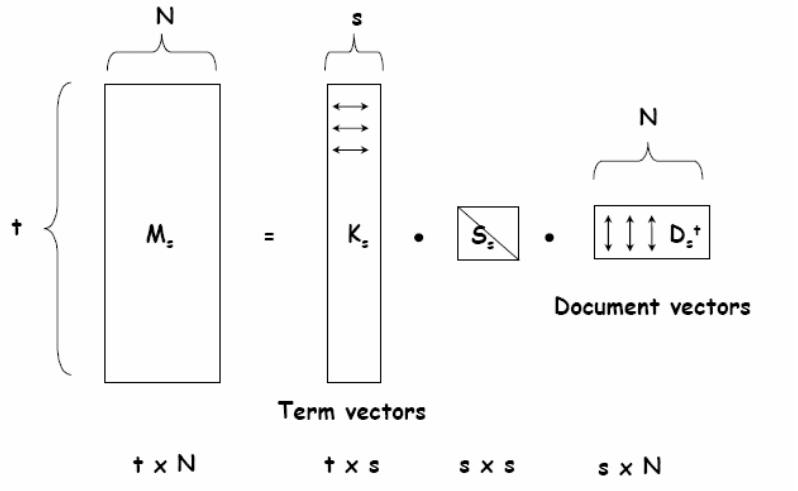

Figure 2.13: Choosing s singular values

[1] An orthonormal matrix is an m x n matrix A such that when the rows and columns of the matrix is swapped to yield an n x m matrix A', the resulting matrix A' is the inverse of the original matrix A.

This reduced dimensionality generates a vector which is a translation of the term-context vector into a concept space. From the resulting matrix M_s, we can measure how similar documents i and j are by comparing the column vectors d_i and d_j in matrix DsS_s by using the cosine similarity measure. In the same way, we can measure how similar terms k and p are by comparing the row vectors in the matrix K_sS_s by the same cosine similarity measure. These cosine similarity measures thus enable us to cluster the documents or terms into groups which will then represent concepts.

LSA has been proven to yield similarity between terms that closely match what humans would describe as similar which is why it is one of the most popular techniques for retrieving conceptual information from a document or set of documents.

Application using LSA

The following is an example of an application which uses LSA [23] to find the main concepts from the text provided. It compares the input text with semantic spaces which are thousands of texts from different areas such as encyclopedias and articles in order to retrieve the main concepts in the text provided. The text shown in figure 1.2 is fed into the application. The semantic space chosen is general reading which contains the text hundreds of articles about general subjects.

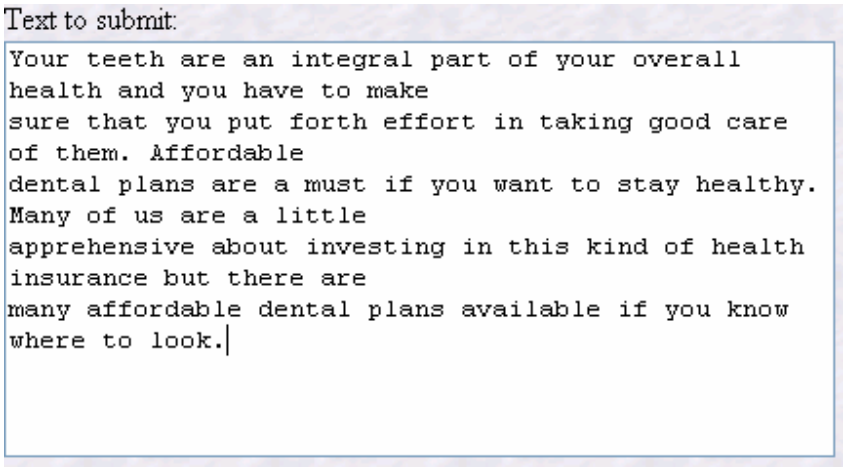

Figure 2.14

The number of concepts to be retrieved is chosen to be five. The application returns the five main concepts it has determined to be the main concepts in the document as shown in figure 2.15. It is noted that the concepts returned match the text provided very closely. The LSA similarity is the similarity matrix values of the terms in the document.

LSA Similarity	Term
0.73	health
0.69	checkups
0.64	wellness
0.63	dental
0.60	hygienist

Figure 2.15

3. Requirements Analysis

3.1. Chapter Introduction

This chapter will give an overview of the functional and non-functional requirements that will be used in the design and implementation of the system. Chapter 2 forms the basis for requirements analysis where it was explained that the main objective of this project is to model a text document as a graph and provide techniques to extract the main concepts of the document from the graph.

In this project, it was not attempted to fully define all the requirements before starting implementation. Rather, the requirements analysis was done in an evolutionary and iterative fashion combined with early implementation, testing and feedback [23]. This iterative and evolutionary development approach has been chosen over the waterfall lifecycle approach because studies have shown that the latter is associated with higher rates of project failure. This is because it tries to fully define all the requirements before starting any development and research has shown that the requirements change significantly during a project. However, iterative and evolutionary development expects changes and readily adapts to the changes to produce a more refined and productive system.

The rest of this section will describe the final system requirements (functional and non-functional requirements).

3.2. Functional Requirements

Functional requirements are concerned with the core functionalities of the system. The main functional components of the system have already been identified in chapter 2 (section 2.1). They are listed below:

1. Syntactic processing
2. Constructing the graph file
3. Graph drawing
4. Clustering
5. Identification of the main concepts
6. Ranking of the main concepts (Summarisation)

In more detail, the actual functions provided by the system are as follows:

- To load a text file stored on the computer and turn it into a set of vertices (nodes) and edges suitable for graph drawing.

- To calculate a minimal energy graph drawing of the words (represented by nodes) in the document using a force-directed graph drawing technique.

- To display the graph with the words represented by nodes. The size of the nodes on the graph should reflect the connectivity of the word in the document (its degree).

- To cluster the graph where the clusters will represent concepts using a number of clustering algorithms according to the number of clusters specified by the user. The system should also be able to determine the optimal number of clusters for a graph with each of the clustering algorithms.

- To display the clusters of words on the graph.

- To identify and display the most important concept in each cluster using one or more methods.

- To provide a textual summary of the input text indicating the main concepts in the text. The summary should rank the main concepts in order of their importance in the text.

The system requires a lot of user interaction and therefore, the following functions need to be satisfied by the Graphical User Interface (GUI).

- Before plotting the graph, the user should be able to choose a different context (group of words or sentences that the text will be broken down into) from the GUI. Choosing a different context will yield a different graph.

- The user should be able to select a different clustering algorithm to use to cluster the graph.

- The user should be able to specify the number of clusters when using each of the clustering algorithms.

- The GUI should provide the ability to view the text being processed.

- The GUI should provide more than one algorithm which can be used when summarising the text.

3.3. Non-Functional Requirements

Usability

The system should be simple and easy to use. It should be clear to the user what the commands available are at each stage of the process of extracting the main concepts from the text. Moreover, the graphs should be displayed in an aesthetic way and the user should immediately understand the concepts present in the document from it.

Performance

The individual components of the system vary considerably in their performance. The main consumers of the processing power of the system are the graph drawing component and the clustering component but out of these two components the graph drawing component is the most resource-consuming component.

If N is the number of iterations, E denoted the number of edges and V denotes the number of vertices (nodes)

- K-means clustering algorithm has a time complexity of O (V)
- Agglomerative hierarchical clustering algorithm has a time complexity of O (V^2)
- And the graph drawing component has a time complexity of O (E + VlogV) per iteration (of the energy minimisation algorithm)

This is why it takes a very long time to draw large graphs (where there is a high number of nodes and edges). This is because the number of iterations needed to find a minimum-energy drawing of the graph increases when the number of nodes increases. On a graph of average size (about 100 nodes), the running time of the graph drawing component is approximately 6 seconds and on a larger graph (about 450 nodes), the running time is about 49 seconds. The system should therefore be able to draw small and average sized graphs reasonably fast.

Flexibility

The system should be flexible and reusable. This will ensure that new functionality can be easily added in the future. The new functionality may be a different graph drawing algorithm or more clustering algorithms. This has been achieved by the use of object-oriented design patterns as much as possible when designing the software components of the system [24].

4. Software Design

4.1. Chapter Introduction

This chapter is concerned with the design of the system and it is based on the requirements discussed in chapter 3. It will give an overview of the structure of the system and describe its main components. It will also cover the design of the user interface as well as the choice of programming language to be used in the implementation.

4.2. System Architecture

Figure 3.2 shows the main components of the system and the way in which these components interact with each other.

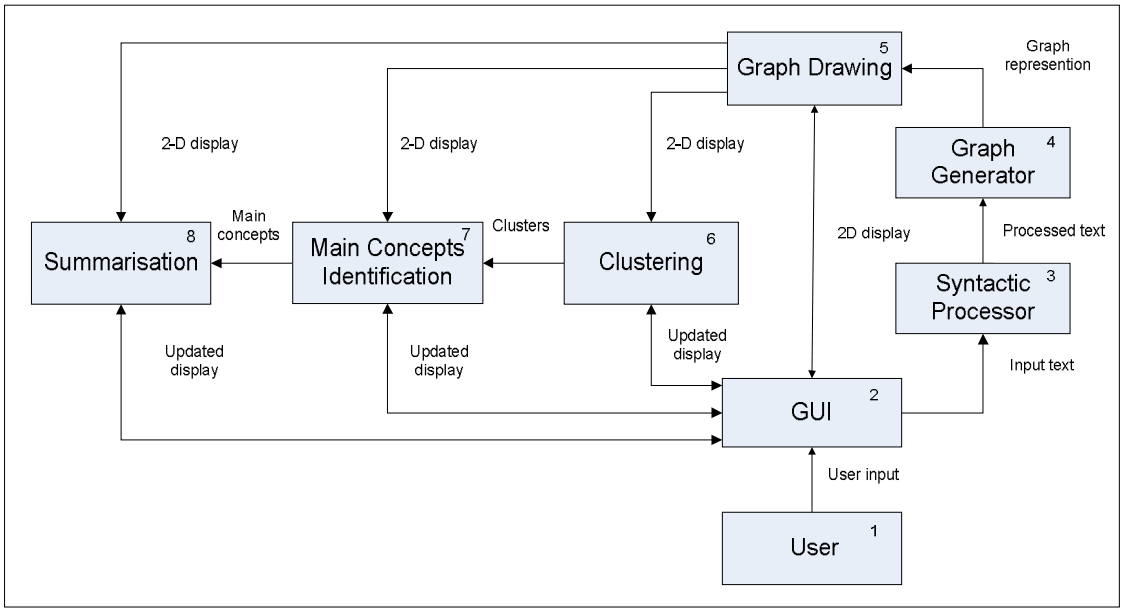

Figure 3.2: System architecture

1. The process starts with the user inputting the text using the options from the GUI (shown by the bottom box numbered 1 in the above diagram).

2. This component is responsible for the system interacting with the user. The user can choose commands from the GUI and the GUI will initiate the appropriate actions in the required component. The GUI also displays the 2-D graph drawing every time the graph is updated. When the user has selected a text file and selects the command to generate a graph, the GUI sends the chosen file to the syntactic processor which is the next component in the system.

3. The syntactic processor receives the input text from the GUI. It has a number of classes designed to first take out the words in the text that occur only once, break the remaining text down into contexts (depending on the context chosen in the GUI) and then removing the unwanted words from the contexts. The output from this component is the contexts which are sent to the graph generator component. This component also stores the first three paragraphs of the original text in a vector of String objects in order to rank the main concepts at a later stage.

4. The graph generator component accepts the contexts and generates a vector of Edge objects (these objects have a source node name, a target node name and an edge weight which defines how many times the edge occurs in the graph). This vector of Edges is written to a text file which will form the input to the graph drawing component. Each line of this file consists of a source node, a target node and the weight of that edge separated by a space character.

5. The key component of the system is the graph layout component. This is the central engine of the system and most of the other components depend on it in order to perform their functions. The graph layout component will read the text file generated by the graph generator component. Each line of this file will represent an edge which should be present on the graph. The graph drawing component takes this file as a parameter and uses it to generate a graph of the nodes and edges. The two main classes in graph layout are LinLogLayout and MinimiserBarnesHut. LinLogLayout first stores the nodes and edges as a nested hash map. Each of the source nodes from the text file is mapped to a map of the target nodes it is connected to (its edges). The nested hash map maps each of the target nodes to a value representing the strength of the edge (the number of times the source node is connected to the target node by an edge i.e. the weight).

   Map (Source Node → Map (Target Node → Strength of edge)).

   LinLogLayout then calculates random coordinate positions for all the nodes in the graph. The MinimiserBarnesHut class then takes the initial hash map and the initial coordinates and computes a drawing of the graph nodes with minimum energy over a number of iterations. The output from MinimiserBarnesHut is consists of the final positions (coordinates) of the nodes and the size of each node which is the number of edges it has (its degree). These values are used to update the GUI and display the initial 2-D drawing of the graph.

6. The clustering component consists of two main packages and takes its data from the graph drawing component. The first package contains classes that are used to cluster the graph using K-Means. The second package contains classes that are used to cluster the graph using the agglomerative clustering algorithm. The main problem encountered in clustering data in general is deciding on the number of clusters. The same problem is encountered here when clustering the graph with K-Means and the agglomerative hierarchical algorithms when the user has chosen not to specify the number of clusters. If the right number of clusters was known for each graph, then clustering the graph would be easy. In order to determine the optimal number of

clusters for a graph, appropriate methods will be implemented in both the K-Means and the agglomerative hierarchical packages. These methods will be based on some intrinsic measure of the points in the graph and how this measure changes with the number of clusters. A detailed explanation of these methods will be covered in chapter 5. The output from this component will be a vector of Cluster objects, each containing a set of points which is used to update the GUI. Each cluster of points will be represented by a unique colour on the graph.

7. The component responsible for the identification of the main concepts in each cluster takes its input data from the clustering component and from the graph drawing component. For each cluster, the main concept is the node which has the maximum number of edges connecting it to another member of the same cluster. The 2-D drawing is updated on the GUI and these main concepts are indicated on the graph by means of arrows.

8. The last component in the system is the summarisation component. This component uses the main concepts, their degree (number of edges) and their position in the text to generate a text summary of the document. Two methods will be implemented to generate the summary. Both methods will take the position of the main concept in the text into account. The only difference is that one method places the main concept of the biggest cluster first and the other method places the main concept with the highest degree in the text first. The main concepts will be highlighted in the summary when displayed in the GUI.

4.3. Graphical User Interface Design

As this system requires a lot of user interaction, a lot of thought has been put into the design of the GUI using Schneiderman's eight golden rules [25] as guidelines.

The main objectives of the design of the user interface are for it to be:

- Easy to use

  The user interface has been designed so that it is easy for the user to choose the main functions they want the system to perform (just by clicking on the appropriate button from the toolbar for example) and change the parameter of the different functions (changing the number of clusters by moving a slider bar). This gives the user the feeling of being in control.

- Intuitive

  The GUI has been designed so that the appropriate buttons or options are enabled or disabled when the system has just performed a particular function. This makes the user aware that the system has finished processing their command.

- Able to fulfil the requirements in the requirements analysis

The GUI will have all the required options to fulfil the user requirements. The user can either select commands from the menu bar or from the toolbar. The menu bar and the toolbar contain the same options. In addition to these commands, the user can select the parameters to use when clustering using the side bar (settings for k-means and agglomerative hierarchical).

Another feature of the design of the GUI is concerned with displaying the main part of the system which is the graph in an attractive way such as making use of colours when drawing the nodes and using clear fonts when labelling the nodes. The graph should also occupy the major part of the interface as it is the main focus of the application.

Menu Bar

- File menu: Load File (browse a text file on the computer), Context (choose from sentence or word and the number of words and sentences to use), Exit (exits the application)
- Graph menu: Plot Graph (plots a graph), Reset (displays original graph if clustering has been done)
- View menu: View Text File (opens a new window showing the text being processed), View Concepts (indicates the main concepts on the graph)
- Cluster menu: K-Means (clusters the graph with k-means), Agg Hier (clusters the graph with agglomerative hierarchical)
- Summary menu: Cluster Summary and Text Summary (choose method to summarise text), Summarise Text (generates a text summary)
- Help menu: About (information about the author of the application and version number)

Toolbar

The toolbar contains buttons for all the functions present in the toolbar except the choice of the method used to summarise the text (the Summary menu), context choice (the File menu) and information about the application (the Help menu). It provides a quicker way for users to select commands without having to scroll through the menu items.

Graph Display

The system is very pictorial being based on graph drawing and the quality of the graph crucial to the good use of the system. The appropriate use of colours improves the quality of the graph and makes it easier to understand. The nodes on the graph will be represented by coloured oval shapes. The size of each node will vary depending on its degree. In the initial graph, all the nodes will be coloured dark blue and labelled with the word in black coloured fonts. After clustering has occurred, a unique colour will be used to colour each cluster (all the nodes in a cluster being the same colour). This will allow the user to instantly identify which words

belong to which cluster and get an overall picture of how concepts are distributed in the text. During the identification of the main concepts, it should be clear to the user which of concepts in each cluster represents the main concept on the graph. This is achieved by using an arrow pointing to the main concepts in order to draw the user's attention to them.

Figure 3.4 shows the final design of the GUI.

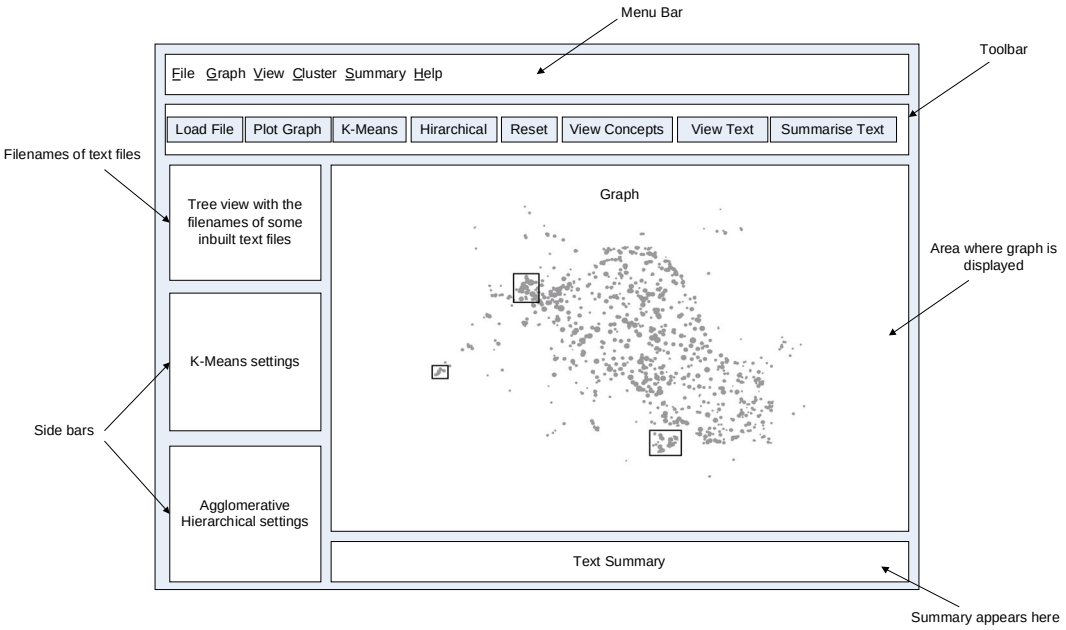

Figure 3.4: GUI design

4.4. Programming Language Choice

The programming language which will be used to implement the system is Java. The main reasons for using Java are:

- The system has been designed using object-oriented design principles and the class structure will closely model the system architecture and its components. An object-oriented language is therefore appropriate in this case and Java is an OO language.

- Java is platform-independent which means that the system can be run on a number of operating systems including Windows and Linux.

- Java provides numerous components for the creation of graphical user interfaces (AWT and Swing).

- This application could be extended for use in combination with web searches and web applications. Also, implementing the system in Java makes it straightforward to interact with, for example, XML web services running on web servers.

Graphical User Interface

Java provides two components for creating user interfaces:

- AWT (Abstract Windowing Toolkit)
- Swing

AWT is the traditional approach to GUI design and utilises the operating systems resident graphical libraries. This makes the code dependent in its implementing operating system. However, Swing which is an updated version of AWT is platform independent and utilises raw graphics commands for drawing lines and boxes which can be adopted to use the "look and feel" theme that matches the look of the current operating system. Swing was the component used in the implementation as it contains the appropriate GUI components required for the system [26].

5. Implementation

5.1. Development Environment

Choosing an Integrated Development Environment (IDE) is an important decision to make when coding an application. A good IDE makes programming less tedious and also lessens the development time.

Eclipse

The Eclipse environment was chosen to develop the system because it has the following features:

- Written in Java for Java application development.
- Class creation is simple using wizards.
- Debug mode allows inspection of variables at runtime.
- When creating a private variable, options become available to create *getter* and *setter* methods for that variable.
- The text editor automatically formats code to a given format.
- Possible fixes to common problems are displayed in a quick fix menu displayed, besides compiler errors.
- Compiles as you type, displaying errors instantly.
- Creates Java documentation tags as you type.
- JAR file creation is just a right click operation.
- It is freely available (open source).
- Allows javadoc comments to be viewed.
- It allows version control.

The only drawbacks of Eclipse are that there is no user interface design functionality and that it is memory intensive especially in the debug mode. The benefit of using Eclipse was mainly familiarity and therefore the lack of a user interface designer did not adversely affect the project.

The next section will provide an in-depth description of how the most interesting and challenging parts of the system were implemented. Then, screen shots of the GUI will be shown and it will be explained how it fulfils all the requirements of the system.

5.2. Generating the Graph File

After the user has selected a file from the GUI and the input text has been broken down into contexts (group of sentences or words) during syntactic processing of the text, these contexts are used to create the file that will be used to generate the graph. This file needs to be a list of nodes and edges. The nodes will be the words in the contexts and an edge will be present between every two words that are present in the same context. A count of the number of times two words are related in the same context should also be recorded because this value will be used to compute how close two nodes should be on the graph (the weight of the edge).

Before processing the contexts, we need a way to represent a node and an edge. Each node in the graph is going to be connected to at least one other node. This is why only an edge representation is needed. A fairly standard representation of an edge has been implemented here by the Edge class and the nodes are represented by String attributes of the Edge class. The declaration of the Edge class is shown below:

```
public class Edge {
      private String source;        // the start node of the edge
      private String target;        // the end node of the edge
      private int weight;           // the weight of the edge
```

The contexts are stored in a vector of String objects. Each of these contexts is then broken down into a vector of Edge objects by linking every two words in the same context and keeping a count of the number of times these two words are linked by the same context (weight). This is done by the GraphFile class and the main part of the code which produces this vector of Edges is shown below.

```
int size = words.size();
for(int j=0; j<size; j++) {
      for(int k=j+1; k<size; k++) {
            String node1 = words.elementAt(j);
            String node2 = words.elementAt(k);
            if(!node1.equals(node2)) {
                  Edge existingEdge = getEdge(result, node1, node2);
                  if(existingEdge == null) {
                        result.add(new Edge(node1, node2));
                  }
                  else {
                        existingEdge.incrementWeight();
                  }
            }
      }
}
```

This vector of Edge objects is then written to a text file called “graph.txt”. This is resulting file that will be used for the graph drawing and its format is shown in figure 5.1.

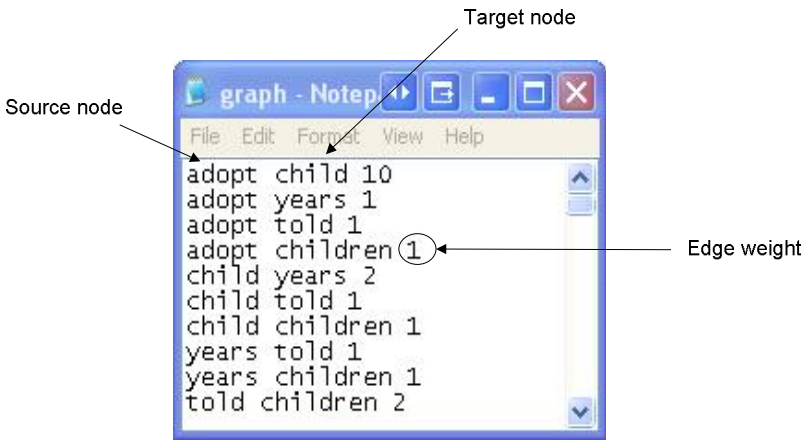

Figure 5.1: Graph file

5.3. Drawing the Graph

The graph drawing technique is responsible for computing a drawing of the nodes with minimum energy [12]. It consists of two main classes, LinLogLayout and MinimizerBarnesHut. LinLogLayout reads the file “graph.txt” and converts it into a nested hash map called *graph*. The hash map maps each source node to a second map which maps each of the source node’s target nodes to the edge weights by the method readGraph(textFile) which takes the filename as a parameter.

```
Map<String,Map<String,Float>> graph = readGraph(textFile);
```

This hash map forms the basis of the graph drawing technique. A private vector of Edge objects is constructed from this hash map in order to count the number of edges of each node. Using the hash map, the next step involves assigning unique id’s to the nodes in order to help identify the nodes resulting in a hash map mapping each node (represented by a string) to an integer(number from 0 to number of nodes-1) . This hash map is called *nodeToId*.

The class then uses the *graph* hash map to compute random initial positions of the nodes which represent an initial drawing of the graph. The result from this is a two dimensional array of the type float [i] [j] called *positions* where i is the id of the node and j is 0, 1 or 2 used to store the x, y and z coordinates. In this case, only 2D graphs are drawn, therefore, only the x and y coordinates are set and the z coordinate is always set to zero.

The next step is to minimise the energy of the initial graph. This is done by the MinimizerBarnesHut class. LinLogLayout creates an instance of MinimizerBarnesHut as shown below.

```
MinimizerBarnesHut minimizer = new MinimizerBarnesHut(
      makeAttrIndexes(graph, nodeToId), makeAttrWeights(graph,
      nodeToId), makeRepuWeights(graph, nodeToId),1.0f, 0.0f, 0.01f,
      positions);
```

All these parameters are needed by the MinimizerBarnesHut class and they are calculated by methods in LinLogLayout. The constructor of MinimizerBarnesHut is shown by the code below.

```
public MinimizerBarnesHut(
      final int[][] attrIndexes, final float[][] attrWeights, final
      float[] repuWeights, final float attrExponent, final float
      repuExponent, final float gravFactor, final float[][] pos) {
            nodeNr = attrWeights.length;
            this.attrWeights = attrWeights;
            this.attrIndexes = attrIndexes;
            this.repuWeights = repuWeights;
            this.attrExponent = attrExponent;
            this.gravFactor = gravFactor;
            this.repuExponent = repuExponent;
            this.pos = pos;
      }
```

The attribute attrIndexes is adjacency list of the edges of the graph using their unique id. The attribute attrWeights is an adjacency list of the edge weights. The attribute repuWeights is the repulsion weights for each of the nodes. The repulsion weight of a node is a measure of the number of edges going out of it (its degree) and it is obtained by adding the weights of all the edges connected to that node. The attribute *attExponent* is an exponent used for the distance when calculating the attraction energy and it is given a value of 1.0f. The attribute *repuExponent* is an exponent of the distance used when calculating the repulsion energy and it is given a value of 0.0f. The attribute *gravFactor* is used for calculating the gravitational energy which attracts all the nodes to the barycentre of all the nodes and its value is 0.01f. This is done in order to make sure that nodes are not placed too far apart on the graph. The attribute *pos* is the 2-Darray of the initial positions of the nodes.

Having set these private attributes in the constructor, the method minimiseEnergy is then called on the instance of MinimizerBarnesHut and this method accepts an integer as its paramater which is the number of iterations to perform in order to compute a drawing of minimum energy. A typical value for the number of iterations is 100.

The minimiseEnergy works on the 2-D array *pos*, updating the position values of the nodes in each iteration. The initial energy of the graph can be computed from the initial positions of the nodes for analysis. The main steps of the energy minimisation process is summarised below:

1. The barycentre of all the nodes is calculated.

2. All the nodes are added to an octtree (represented by the OctTree class) using their positions.

3. The *attrExponent* and the *repuExponent* are adjusted depending on the number of iterations already performed. The values are higher if a large number of iterations are still left in order to converge to a minimum energy state.

4. The energy of each node is then calculated using the octTree. The energy of a node i is the sum of the repulsion energy (between that node and the other nodes in the octtree which is mainly a function of the *repuWeights[i]*, distance to the other nodes and *repuExponent*), the attraction energy (a function of the *attrWeights[i]* , distance to the other nodes and *attrExponent*) and the gravitational energy (related to the distance of i from the barycentre and the *gravFactor*) of the node.

5. The resultant direction of movement of the node which is equal to the direction of the total force acting on the node is calculated (by adding the directions of each of the forces acting on the node).

6. The distance by which the node will be moved is calculated using the direction of movement and a number of values which are multiples of the number 32 (random number chosen).

7. The old position of the node is stored, the new position of the node is computed and the energy of the node is recalculated each time a new distance value is calculated.

8. If the new energy of the node is less than the initial energy of the node, the final *pos* values of the node are updated with the new position and the node is moved in the octtree.

9. Steps 1 to 8 are repeated for every iteration (in this case 100).

The main part of the code for the steps described above is shown below.

```
// move each node
energySum = 0.0f;
for (int i = 0; i < nodeNr; i++) {
      final float oldEnergy = getEnergy(i, octTree);
      // compute direction of the move of the node
      getDirection(i, octTree, bestDir);
      // line search: compute length of the move
      oldPos[0] = pos[i][0]; oldPos[1] = pos[i][1]; oldPos[2] =
pos[i][2];
      float bestEnergy = oldEnergy;
      int bestMultiple = 0;
      bestDir[0] /= 32; bestDir[1] /= 32; bestDir[2] /= 32;
      for (int multiple = 32;multiple >= 1 && (bestMultiple==0 ||
      bestMultiple/2==multiple); multiple /= 2) {
            pos[i][0] = oldPos[0] + bestDir[0] * multiple;
            pos[i][1] = oldPos[1] + bestDir[1] * multiple;
            pos[i][2] = oldPos[2] + bestDir[2] * multiple;
            float curEnergy = getEnergy(i, octTree);
            if (curEnergy < bestEnergy) {
                  bestEnergy = curEnergy;
                  bestMultiple = multiple;
            }
      }
```

```
        for (int multiple = 64;  multiple <= 128 && bestMultiple ==
        multiple/2;  multiple *= 2) {
              pos[i][0] = oldPos[0] + bestDir[0] * multiple;
              pos[i][1] = oldPos[1] + bestDir[1] * multiple;
              pos[i][2] = oldPos[2] + bestDir[2] * multiple;
              float curEnergy = getEnergy(i, octTree);
              if (curEnergy < bestEnergy) {
                    bestEnergy = curEnergy;
                    bestMultiple = multiple;
              }
        }
        pos[i][0] = oldPos[0] + bestDir[0] * bestMultiple;
        pos[i][1] = oldPos[1] + bestDir[1] * bestMultiple;
        pos[i][2] = oldPos[2] + bestDir[2] * bestMultiple;
        if (bestMultiple > 0) {
              octTree.moveNode(oldPos, pos[i], repuWeights[i]);
        }
        energySum += bestEnergy;
}
```

The new positions of the nodes is converted into a final hash map *nodeToPosition* in LinLogLayout which is a map from each node to its position (Map <String, float []>).

5.4. Clustering the Graph

There are two algorithms used for the clustering of the graph generated. They are K-Means and agglomerative hierarchical clustering algorithms. The clustering algorithms use the *nodeToPosition* hash map from the LinLogLayout class in order to cluster the nodes.

5.4.1. K-Means Implementation

Given the number of clusters K, clustering a graph using the K-Means algorithm [20] is straightforward. The main part of the K-Means algorithm is implemented in the KMeans class. The problem with K-Means is finding the optimal number of clusters for a particular graph drawing to ensure that concept extraction is accurate. The number of clusters selected by the user might not result in an accurate clustering of the graph and therefore, the concepts extracted might not describe the text properly.

The method implemented to determine the optimal number of clusters for a particular graph is based on intrinsic measures. Assuming that a graph has been clustered for a given K, the quality of the clustering is be measured by E(K) where E(K) is the sum of the squared distances from each point to its centroid. E(K) can be referred to as an energy measure or a similarity measure. In order to determine the optimal number of clusters for a particular graph drawing, a graph of E(K) against the number of clusters has been plotted using different texts to see how E(K) varies depending on K. These graphs have been obtained by running K-Means on the text starting with K=1 and calculating E(K). K-Means is then run with K incremented by 1 and the new value for E(K) is calculated. E(K) deceases when K increases. Figure 5.2 shows the results obtained using four different texts.

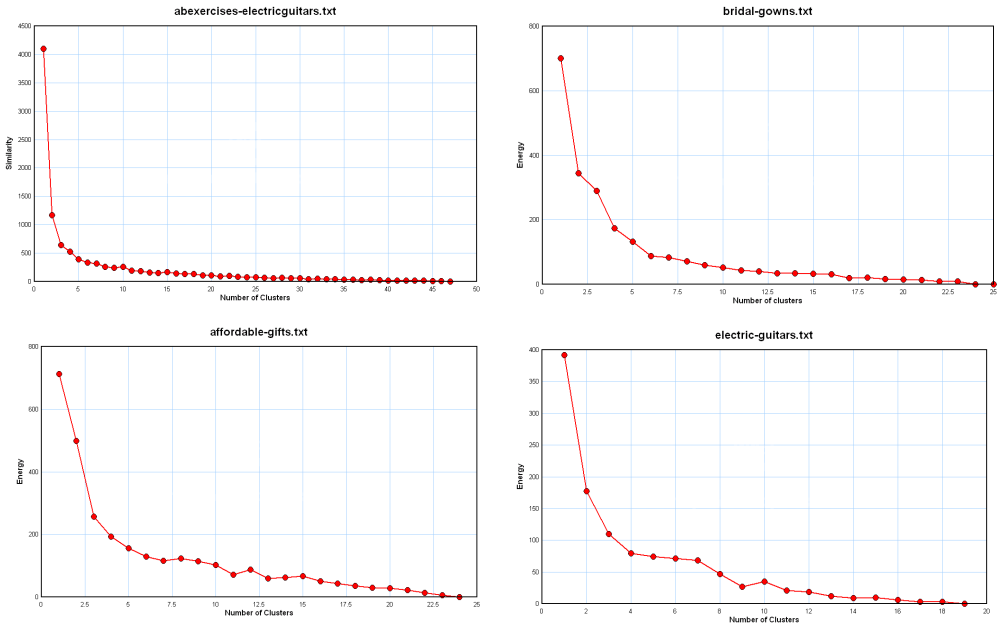

Figure 5.2: Graphs of energy (or similarity) against number of clusters

It can be noticed from the graphs that there is a point after which the change in E(K) is not significant. After analysing the values for E(K), it was found that the cut-off point (point where E(K) stops changing significantly occurs when (E(K)- E(K+1))/E(K) < 0.4.

This technique for determining the optimal number of clusters is implemented in the KMeansAnalyser class using the hash map *nodeToPosition* from LinLogLayout. The main part of the code is shown below:

```
public int getOptimalK1(double target) {
      int start = 1;          //start at number of clusters=1
      double oldMeasure = distanceMeasure1(start);
      int i = start+1;
      boolean doNext = true;
      //for all k's starting from 2 to maxClusters
      while(i <(maxClusters+1) && doNext) {
      //calculate the distance measure for this run of K-Means
      double newMeasure = distanceMeasure1(i);
      //then measure the difference with the previous measure
      double pre = oldMeasure - newMeasure;
      //if this is greater than or equal to the target, then do another
      k-means with number of clusters+1
      if(pre/oldMeasure < target) {
            doNext = false;
            return i-1;
      }
      i++;
      oldMeasure = newMeasure;
      }
      return i-1;
}
```

K-Means is then run on the optimal number of clusters returned by the above method to get the clustered graph.

5.4.2. Agglomerative Hierarchical Implementation

The agglomerative hierarchical clustering algorithm is more complex than the K-Means algorithm. The first step in its implementation involved constructing the distance matrix (or similarity matrix). The distance matrix is represented by the SimMatrix class and it contains the distances between every two nodes on the graph. The nodes are converted into TermAccessor objects so that they can have attributes like an id as well as their string representation. A SimVector class is created and it represents each row of the matrix. The main part of the code for constructing the distance matrix is shown below:

```
public void genMatrix(Map<String, Integer> nodeToId, Map<String,
float[]> nodeToPosition) {

      simVecs = new SimVector[nodeToId.size()];

      termAccessors = new Vector<TermAccessor>();
      int count = 0;

            //for each node's unique number
            for (String term1 : nodeToId.keySet()) {
                  //iterate through all the nodes
                  SimVector simVector = new SimVector(term1);
                  TermAccessor ta = new TermAccessor(term1);
                  termAccessors.add(ta);
                  float[] position1 = nodeToPosition.get(term1);
                  for (String term2 : nodeToPosition.keySet()) {
                        //calculate the distance between the two nodes
                        float[] position2 = nodeToPosition.get(term2);
                        Double value = getEuclideanDistance(position1,
position2);
                        //put it in the vector
                        simVector.add(term2, value.toString());
                  }
                  simVecs[count] = simVector;
                  count++;
            }

            simMatrix = new SimMatrix("Matrix", simVecs);
      }
```

The next step involves using the matrix to cluster the nodes. Assuming that there are n nodes in the graph, at the start of the clustering process, each node is assigned to its own cluster resulting in n initial clusters. A series of iterations then begin whereby the two closest clusters are merged together into a single cluster resulting in n-1 clusters. The single-linkage measure (section 2.5.1) is used here to calculate the distance between two clusters. This means that the distance between two clusters is the distance between two of their closest points. This process continues until a stopping point is reached (such as the number of clusters specified by the user or target value). The main part of the algorithm is shown below:

```
private void step(){
            //identify clusters with greatest similarity
            TermCluster[] clusters = findMostSimilarClusters();

            //keep the larger cluster
            TermCluster combined;
            TermCluster old;
            if(clusters[0].countTerms() > clusters[1].countTerms()){
                  combined = clusters[0];
                  old = clusters[1];
            }else{
                  combined = clusters[1];
                  old = clusters[0];
            }
            //keep a history of the steps
            stepHistory.add(old);
      }
```

The same problem identified with K-Means occurs here where the optimal number of clusters is not known for any particular graph. The same technique used with K-Means is used here to determine the right number of clusters. The only difference is the measure E(K) is calculated using a different formula. In this case, E(K) is called the "average within cluster similarity" and it is given by the sum of the "within cluster similarity" of all the clusters divided by the total number of terms (words). The "within cluster similarity" of each cluster is given by the sum of the distances between every two points in that cluster divided by the number of terms in the cluster.

The graphs below show the results obtained when the average within cluster similarity is plotted against the number of clusters.

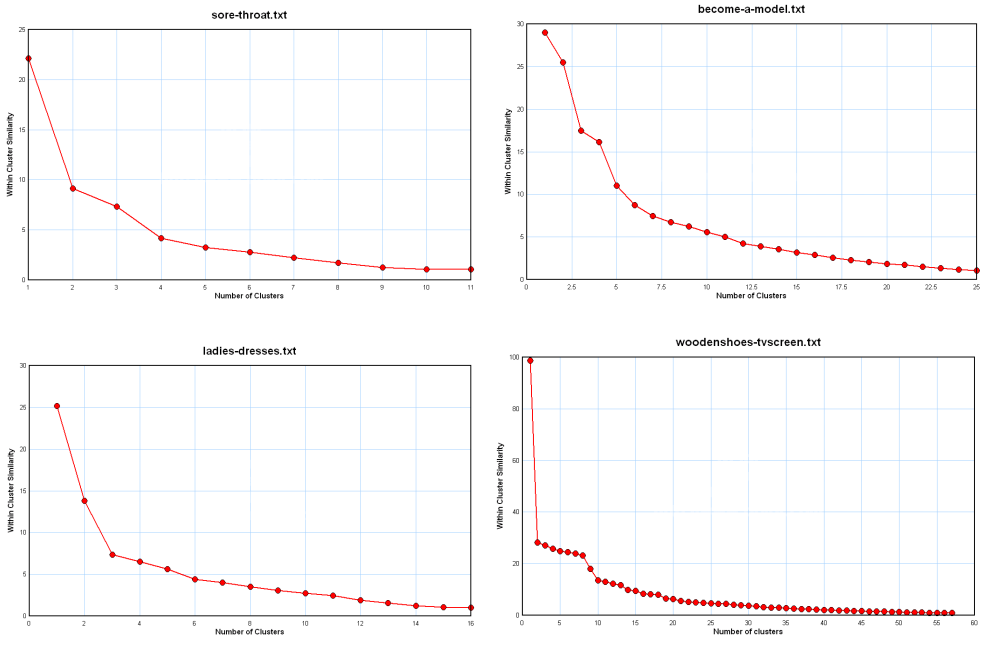

Figure 5.3: Graphs of average within cluster similarity against number of clusters

After analysing the graphs, a reasonable stopping point is where (E(K) – E(K-1) / E(K)) < -0.4 and this technique is used to cluster the graph when the user has chosen to automatically cluster the graph with the optimal number of clusters.

5.5. Graphical User Interface

The implementation of the GUI was done in the ClusterMain class. This class acts as an interface to the underlying classes that perform graph drawing and clustering in order to identify the main concepts in a document. Figure 5.4 shows the final GUI which focuses on the main point of interest (the graph) and provides drop down menus, sliders and buttons for easy interaction with the application.

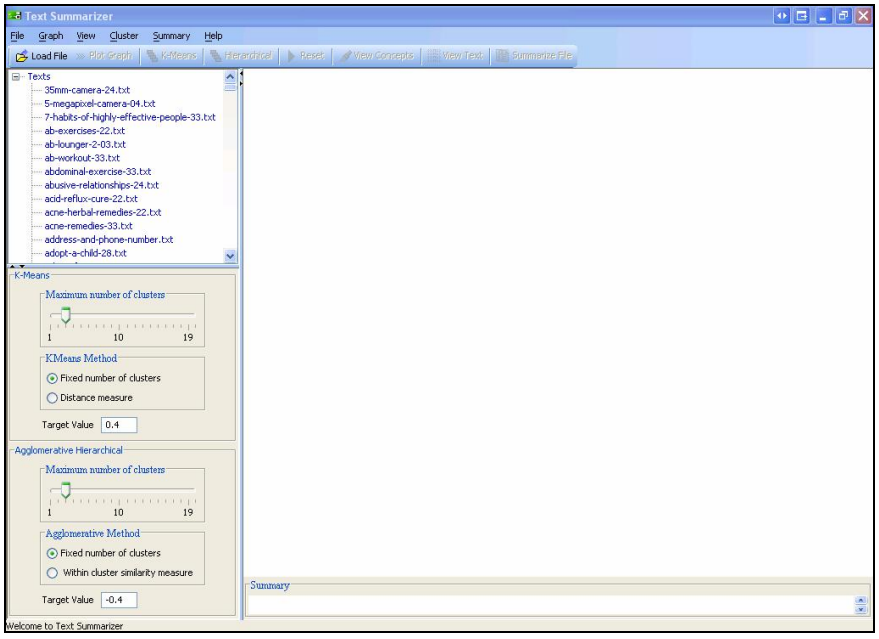

Figure 5.4: Main window

File Selection

When the user clicks on the Load File button (toolbar) or menu item, they are prompted to select a file though a file dialog selection window (Figure 5.5). The user can also select a file from the filenames provided by the tree structure at the top left.

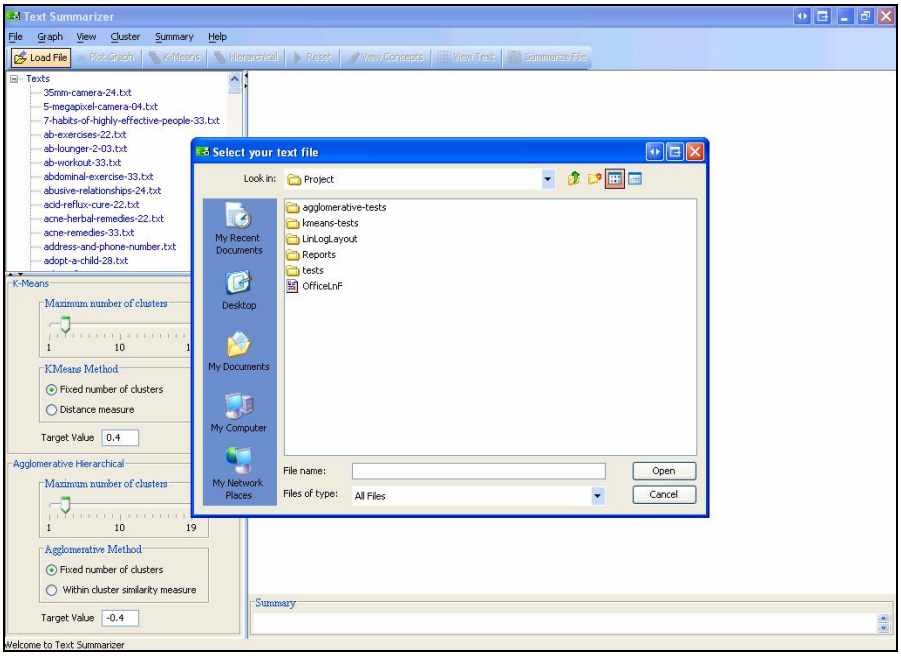

Figure 5.5: File selection

Graph Drawing

The graph is drawn when the user clicks on the Plot Graph button or menu item. The nodes are drawn according to their coordinates and their diameter (Figure 5.6). The appropriate buttons get activated when the graph has been drawn showing the next commands that are available to the user.

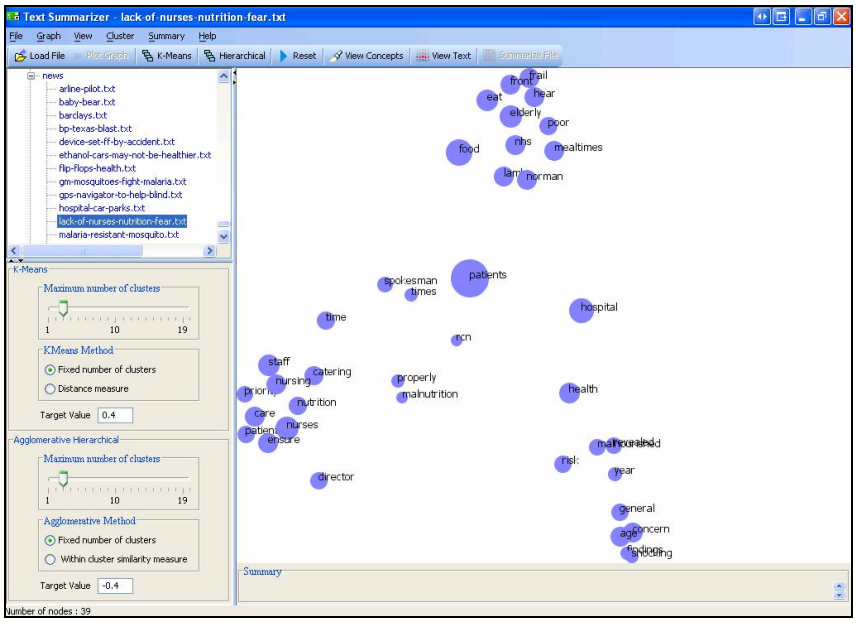

Figure 5.6: Initial Graph

Clustering

The clustering options can be chosen from the side bar (Figure 5.7). Once these options are set, the k-means or agg-hier buttons or menu items are selected to cluster the graph.

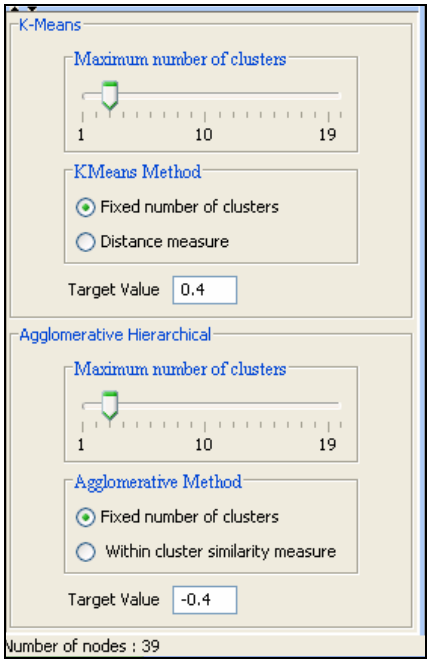

Figure 5.7: Options for clustering

Figure 5.8 shows an example of a clustered graph after k-means is run with the settings in figure 5.7.

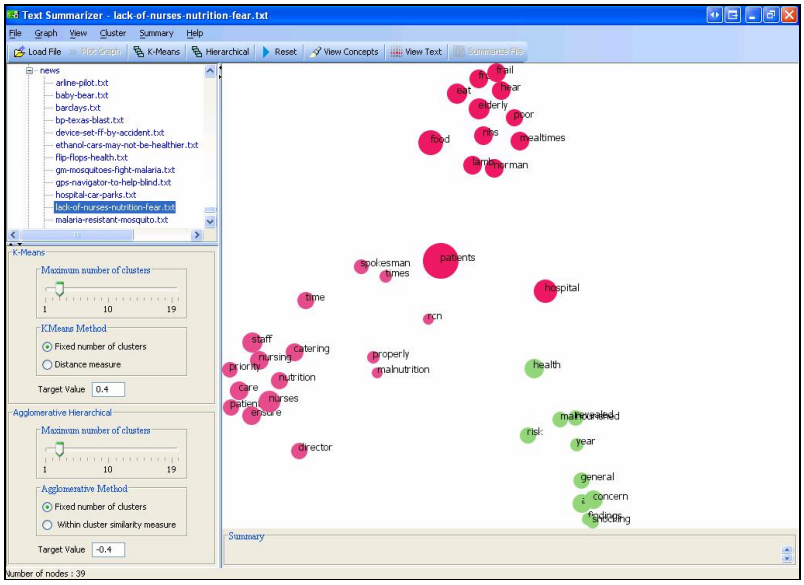

Figure 5.8: Clusters

Main Concepts & Summary

The main concepts are identified when the View Concepts option is selected. The main concepts are indicated with the help of arrows. The summary is displayed at the bottom (when the summary button or menu item is selected) of the graph and the main concepts are highlights in green (figure 5.9).

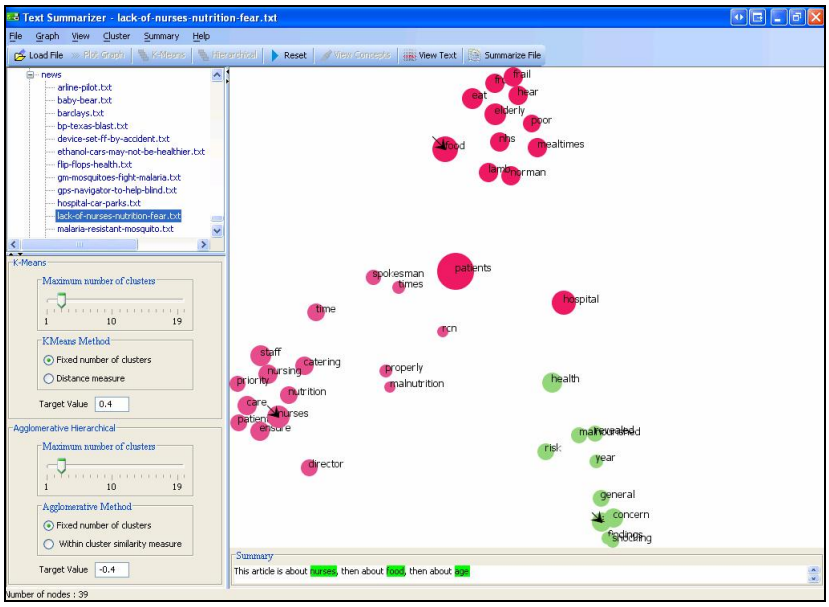

Figure 5.9: Indicating the main concepts on the graph and in the summary

6. Concept Extraction Evaluation

6.1. Chapter Introduction

In this chapter, we will first evaluate the tool in order to see how effectively it extracts the main concepts in different text documents. This evaluation will also cover the effectiveness of the tool when presented with different sizes or lengths of text. We will then proceed to discuss the different ways in which the tool can be improved. Finally, we will discuss how good other techniques are at extracting the main concepts from documents.

Different forms of text present their concepts in rather different ways and it is not clear whether one single technique for concept extraction will suffice. To assess the methods of this project, we have tested it on the following forms of text

- Webpage articles
- Newspaper articles
- Poems
- Novels

It can be noted that in newspaper and webpage articles, words tend to be repeated a lot and that each paragraph is usually about a single concept. This is different for other types of texts such as poems where there tends to be overlapping in the way concepts are spread across the whole document. For instance, poetry tends not to repeat words a lot such that a paragraph in a poem can be about several concepts.

The strength of the tool implemented in this project is that it is very flexible and it can be tuned in various ways in order to identify concepts when dealing with different forms of texts. The tuning allows the user to optimise the process of concept extraction for specific forms of texts. Tuning can take place the following ways:

- Pre-processing of the text document. This can be done in two ways:
    1. Varying the size of the context. The contexts are the small sections in which the input text is broken down into before processing. They can be made bigger or smaller (by changing the number of words or sentences to treat as the context). For example, a bigger context (several sentences long) is needed in novels because several paragraphs may be about a single concept.

2. Changing the words that we choose to ignore in the process of concept extraction. For instance, we might want to ignore the word “truly” in a newspaper article but not in a poem.

- Graph drawing part of the tool.

    1. A different graph drawing technique (such as simulated annealing) can be used in order to give a different drawing of the concepts in the document.

    2. Some parameters of the graph drawing technique can be changed. For example, in the case of LinLog, the number of iterations to find the minimum energy state can be changed which might result in a different graph with higher or lower energy.

- Clustering of the graph.

    1. A different clustering algorithm can be used (k-means or hierarchical).

    2. The user can specify the number of clusters they want or the tool can compute the optimal number of clusters for the graph (calculations involving the similarity of the terms in the clusters) and automatically cluster it.

- Changing the way the main concept is chosen from each cluster. For example, instead of having the node with the highest number of connections to the other nodes in the same cluster as the main concept, we could choose the biggest node in the cluster (the term in that cluster which has the highest degree in the text compared to the other terms in that same cluster).

- Summary construction.

    We can also tune the way in which the text summary is generated after the main concepts have been picked out from each cluster. This can be done by putting the highest degree main concepts (highest number of edges to all the other concepts in the whole text) first or by putting the main concept occurring in the biggest clusters first. The other important point to note when summarising the document is that in some texts for example newspaper articles, the most important concepts tend to be at the beginning of the document. This is taken into account when the text is summarised.

The rest of this chapter will outline the effectiveness of the tool in extracting the main concepts from the four different forms of text mentioned above. When evaluating the tool with each form of text, the effect of the some forms of tuning will also be covered. The tuning categories covered will be:

- Changing the size of the context (Pre-processing of the text).

- Using the different clustering algorithms implemented (Clustering).

- Fixing the number of clusters and using the algorithms for determining the optimal number of clusters (Clustering).

- Generating a text summary of the text

6.2. Webpage Articles

The content of webpage articles tends to be very structured in the sense that most paragraphs usually focus on a particular concept. Therefore, a reasonable context size in this case is one sentence long. Figure 6.1 shows the full text of a webpage article which is about "Becoming a model". The text is about 425 words long.

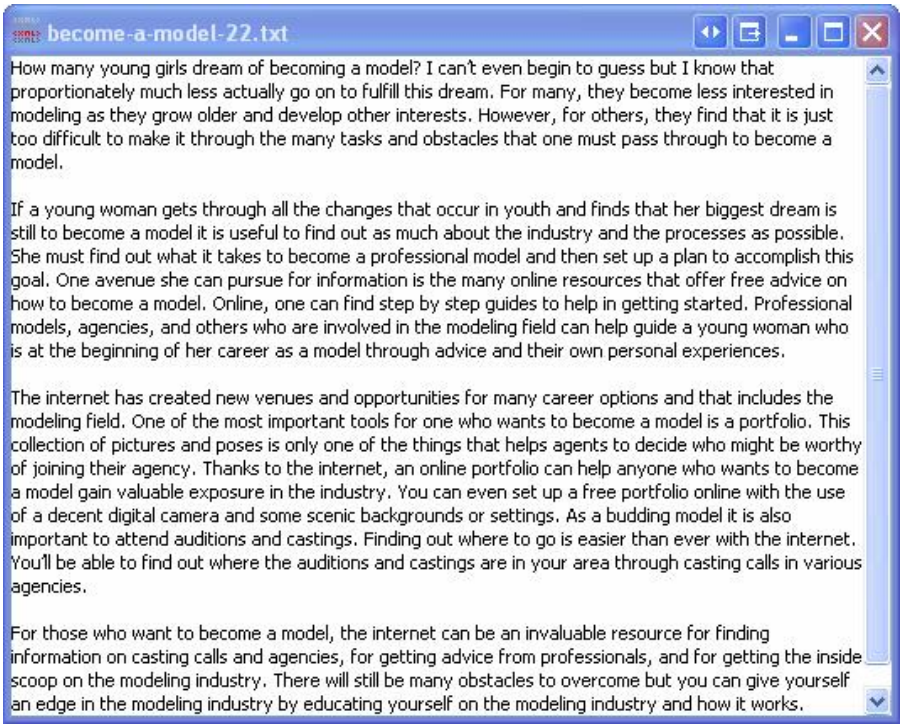

Figure 6.1: Text content of the webpage article

The text of this article is very structured as expected. It can be noted that the first paragraph is introducing the fact that many young girls dream of becoming models. The second paragraph is about where young women can get advice about what it takes to become a model such as online resources. The third paragraph is mainly pointing out how the internet helps budding models by allowing them to set up online portfolios and find out where to attend auditions. Finally, the last paragraph is about how the internet is useful in finding information about casting calls and getting advice from professionals as well as an overview of the modelling industry.

After pre-processing the text above, the tool then generates a graph of the concepts in the text as shown in figure 6.2.

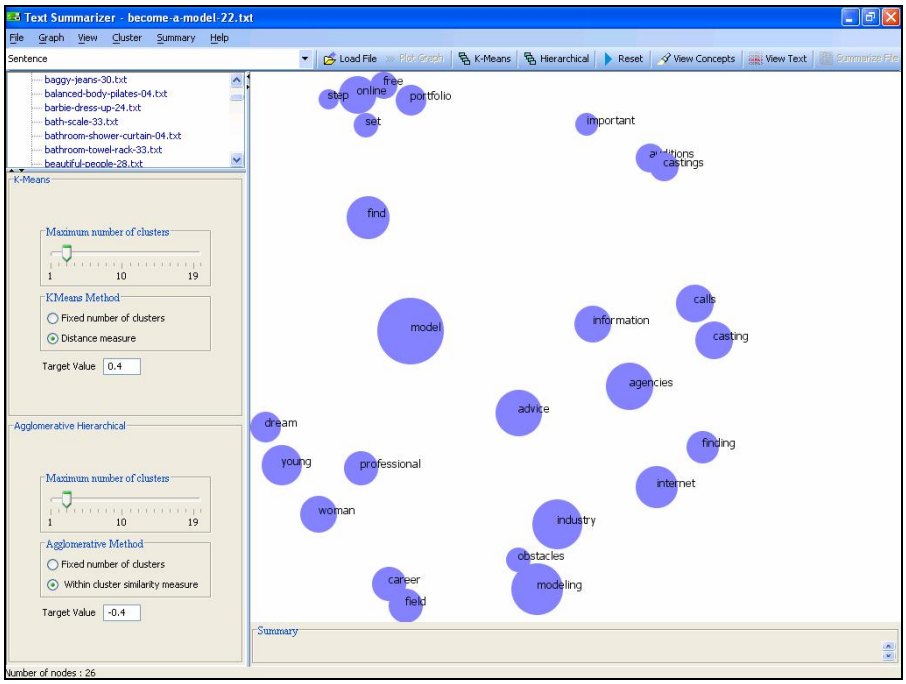

Figure 6.2: Graph showing the concepts in the article

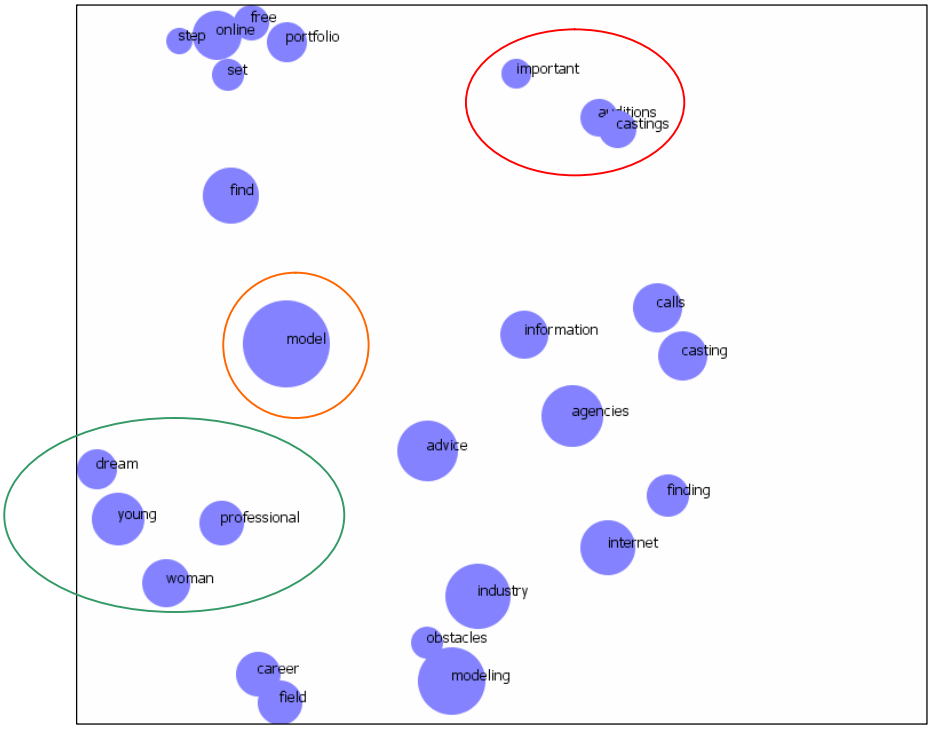

Figure 6.3: Distinct clusters

It can be spotted straightaway from the graph that the text document is mainly about the concept “model” due to the size of the node on the graph shown by the orange circle in figure 6.3. Clearly separated clusters can also be seen. The cluster circled in green consists of concepts like “young” and “woman” whereas the cluster circled in red consists of a clearly separated set of concepts which are “auditions” and “castings”. From the graph, it can already be noted what this article is about from the way the concepts have been drawn. The graph drawing algorithm can be seen to be very effective in this case with its ability to draw a diagram where clearly separated concepts can be visually spotted. The next step is to now cluster the graph with the two clustering algorithms implemented.

Agglomerative Hierarchical Clustering

First of all, the agglomerative hierarchical algorithm will be used to cluster the graph. The formula implemented for determining the optimal number of clusters for this algorithm is based on a graph plot of the number of clusters and the intra-cluster similarity of the clusters. To make sure that the formula works, a plot of the intra-cluster similarity against the number of clusters is plotted before clustering the above graph.

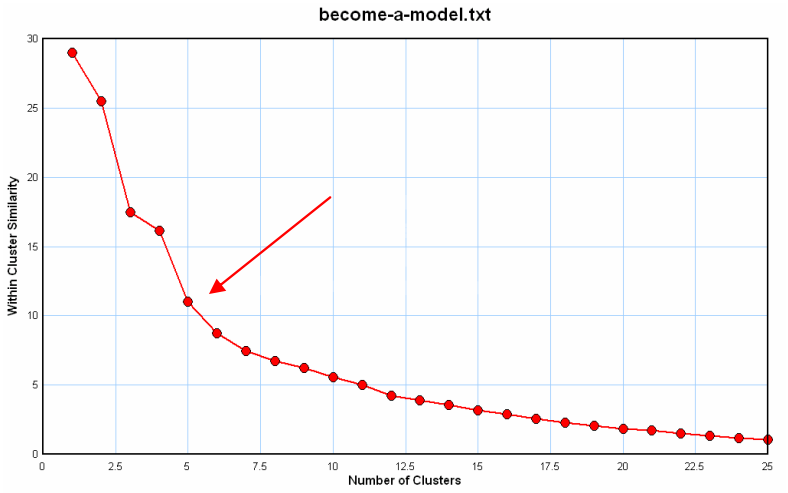

Figure 6.4: Graph of intra-cluster similarity against number of clusters

It is clearly seen from the gradient of the graph in figure 6.4 that there is not much change in the “within cluster similarity” after clustering to more than five clusters. Therefore, according to figure 6.4, the optimal number of clusters for agglomerative hierarchical clustering is five clusters. According to this graph, the tool automatically calculates this optimal value and clusters the graph accordingly and the clustered graph in figure 6.5.

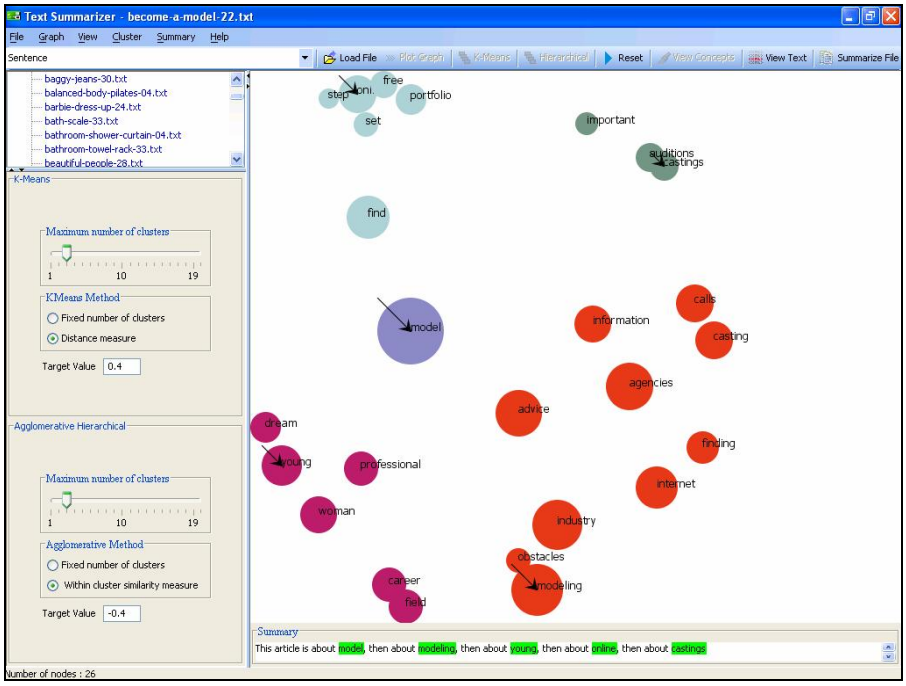

Figure 6.5: Clustered graph

It can be seen from the graph above that the algorithm has clustered the graph into five distinct clusters. The top left cluster contains concepts like “portfolio”. The top right cluster contains concepts like “auditions” and “castings”. The middle cluster contains one concept which is “model”. The bottom left cluster contains concepts like “young” and “woman”. Finally the bottom right cluster contains concepts like “agencies” and “internet”.

The next step involves determining the main concepts in each of these clusters. The tool reveals these five main concepts by means of arrows on the above graph. It can be seen that the five main concepts that the tool has picked out are “online”, “modeling”, “young”, “castings” and “model”. These five concepts indicate the accuracy of the tool in extracting the main concepts in the webpage article. A textual summary of the article can then be obtained. The tool generates the summary from the five main concepts it picked out.

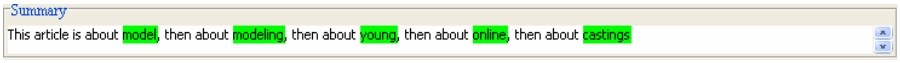

Figure 6.6 (a): Text summary of the article

In this case, the summary generated from the main concepts is “This article is about model, then about modeling, then about young, then about online, then about castings. The summary points out the importance of the main concepts of the text document by putting the most important concepts first and the least important ones last. In this webpage article example, the chosen ranking method for the concepts is based on the frequency of the word in the text.

Overall, it is noted that the main concepts extracted by the tool are very relevant to the text and summarising the text effectively. It can also be deduced that the optimal number of clusters

determined by the tool is relatively good because it resulted in effective clustering by the algorithm and effective main concept extraction.

When the same text is fed into the LSA application described in chapter 2 (section 2.8) and the number of main concepts is chosen to be five, the output below is obtained (figure 6.6 (b)).

LSA Similarity	Term
0.42	careers
0.41	career
0.38	opportunities
0.36	become
0.34	successful

Figure 6.6 (b)

From the concepts extracted above, it is noted that our tool has been more effective in extracting the main concepts from the document.

The user can also choose the number of clusters (main concepts) to generate from the graph instead of the tool automatically doing it. Assuming the user wants to summarise the document into less concepts, three for example, the graph is clustered a second time. A graph with three distinct clusters is obtained. The same process is repeated again whereby the main concepts from these three clusters are obtained and the graph is summarised. The main concepts retrieved are “modeling”, “online” and “castings”.

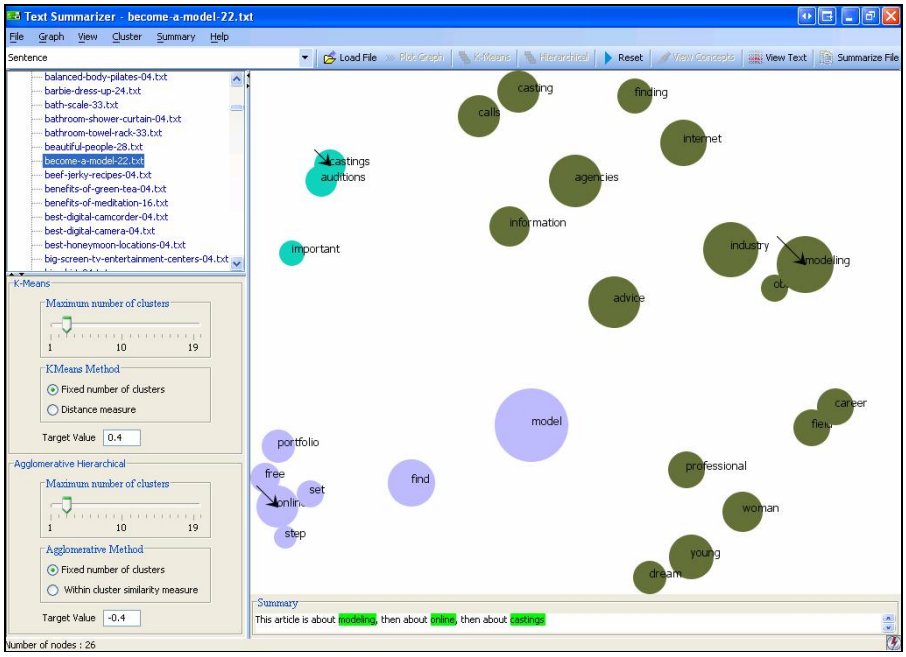

Figure 6.7: Graph showing three clusters and their main concepts

The summary obtained from the text is "This article is about modeling, then about online, then about castings. The tool has again managed to extract the main concepts correctly when the user has specified their preferred number of concepts to be extracted.

K-Means Clustering

The graph below is plotted in order to find the optimal number of clusters for the article using K-Means as the clustering algorithm. It is noted from figure 6.8 that after clustering to three clusters, there is no significant change in the distance measure. Therefore, the tool automatically determines the optimal number of clusters to be three for this article.

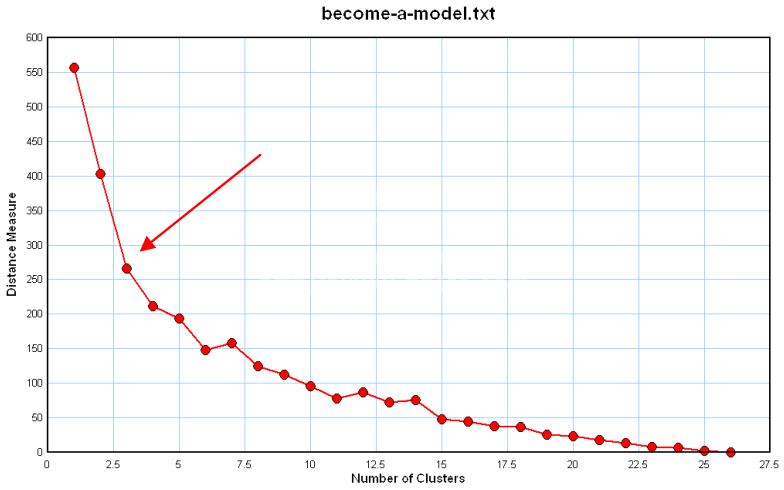

Figure 6.8: Graph of distance measure against number of clusters

The resulting clustered graph in figure 6.9 below shows clusters of the concepts in the article. The cluster coloured yellow contains concepts like "model", "online" and "portfolio". The purple coloured cluster contains concepts like "auditions", "casting" and "agencies". Finally, the green coloured cluster contains concepts like "professional", "woman" and "modeling".

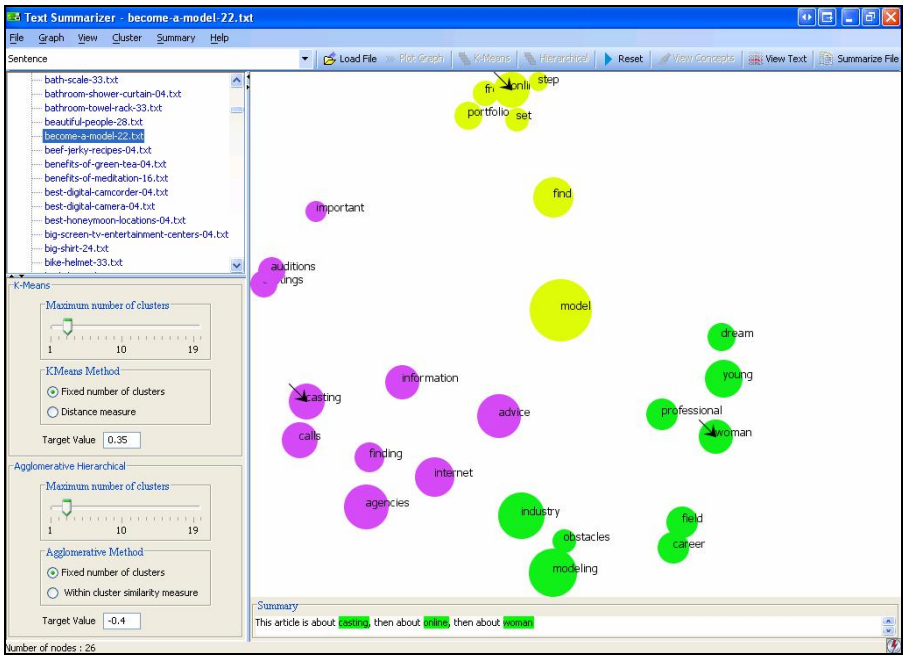

Figure 6.9: K-Means clustering with K=3

The main concepts which have been picked out from these three clusters are “casting”, “woman” and “online” indicated by the arrows in figure 6.9 above. These three concepts are in fact part of the main concepts in the document which shows that this algorithm is effective. However, it can be noted that the concept of “modeling” or “model” have not been picked out and these are the main concepts in the article. The textual summary obtained from the main concepts is “This article is about casting, then about online, then about woman”.

If the number of clusters is now chosen by the user (say five in this case) and the original graph is clustered again using k-means, the graph in figure 6.10 is obtained.

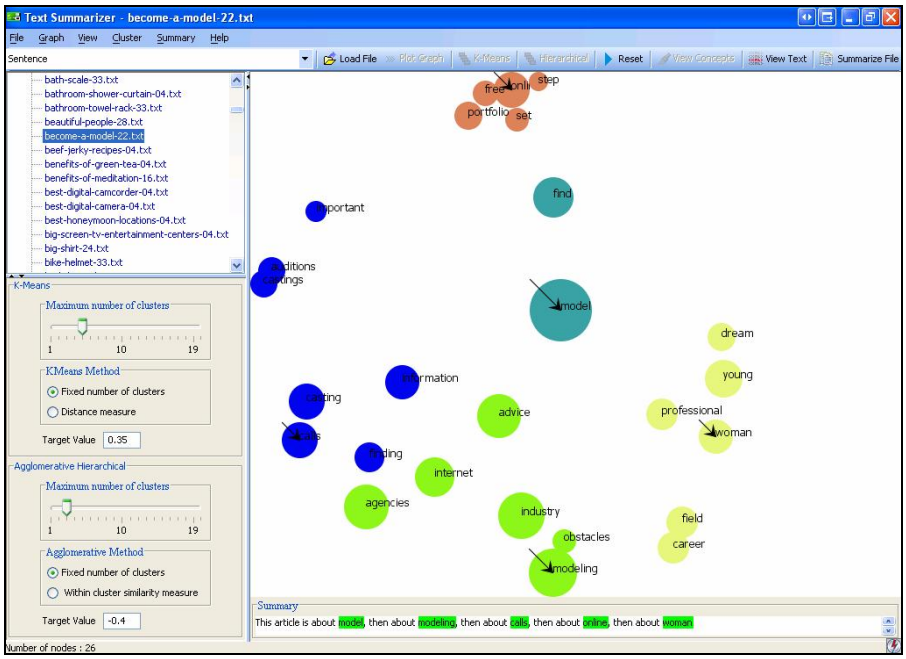

Figure 6.10: K-Means clustering with K=5

It is noted the five clusters obtained this time are most clearly separated than the clusters obtained with three clusters. The main concepts picked out from the five clusters are “online”, “model”, “calls”, “modelling” and “woman”. The summary generated by the tool is “This article is about model, then about modeling, then about calls, then about online, then about woman”. The main concepts picked out from the text are in fact very relevant to the article which proves the effectiveness of k-means as a clustering algorithm.

By clustering the graph with three clusters first, then with five clusters, it is noticed that the number of clusters chosen determines how effective the process of extracting the main concepts is. The main concepts obtained when clustering with five clusters is more relevant to the article than when clustering with three clusters.

It can also be noted that the optimal clustering for the agglomerative hierarchical algorithm gave a different number of clusters (five clusters) from the optimal clustering for the k-means algorithm (3 clusters). In this case, the agglomerative hierarchical algorithm gave better results. However, when clustered with the same number of clusters (five clusters), it is noted that the results from the two clustering algorithms are very similar.

6.3. Newspaper Articles

Newspaper articles differ a lot in structure from other forms of text. The first few paragraphs are usually the most important ones where most of the essential information is. The following paragraphs usually contain important background information and elaborate upon main ideas of the article. The information in the last few paragraphs is usually non-essential information. It is also noted that each paragraph in a newspaper article is usually about one or two particular ideas or concepts. This is why the context is chosen to be one sentence long. In order to evaluate our tool, the article in figure 6.11 (full text in Appendix A) has been chosen which is about a space shield that scientists want to develop in order to protect astronauts from radiation. The text is about 800 words long.

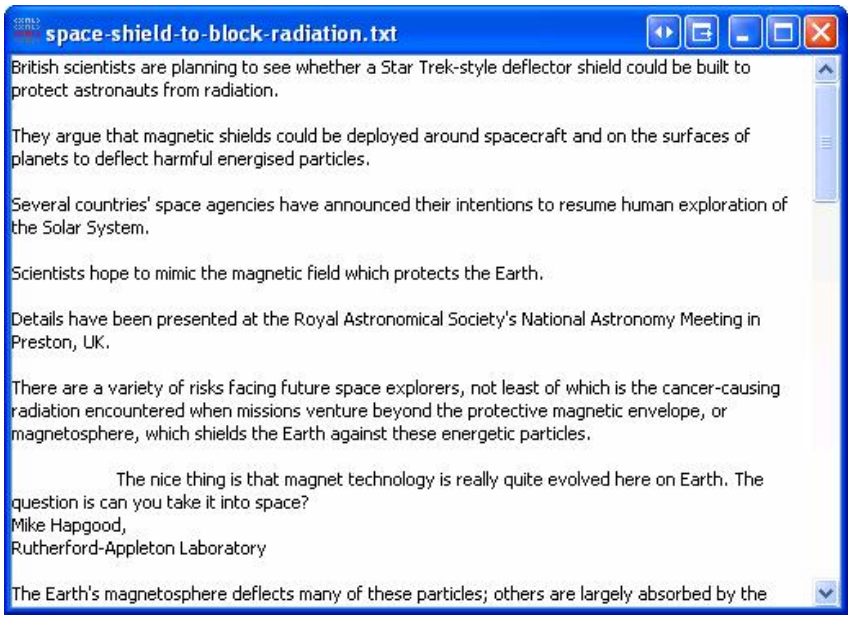

Figure 6.11: Part of a news article

The first few paragraphs of the text are about the most important information of the article. It introduces the fact that British scientists are planning to develop a magnetic shield to protect astronauts from radiation (harmful energised particles). The following paragraphs are about the advantages and disadvantages of these shields. The last few paragraphs present some background information about the shield such as the fact that the idea came from technology in nuclear fusion reactors and other related topics like how a similar technique can be used in fusion reactors.

After pre-processing, the following graph shown in figure 6.12 is generated by the tool.

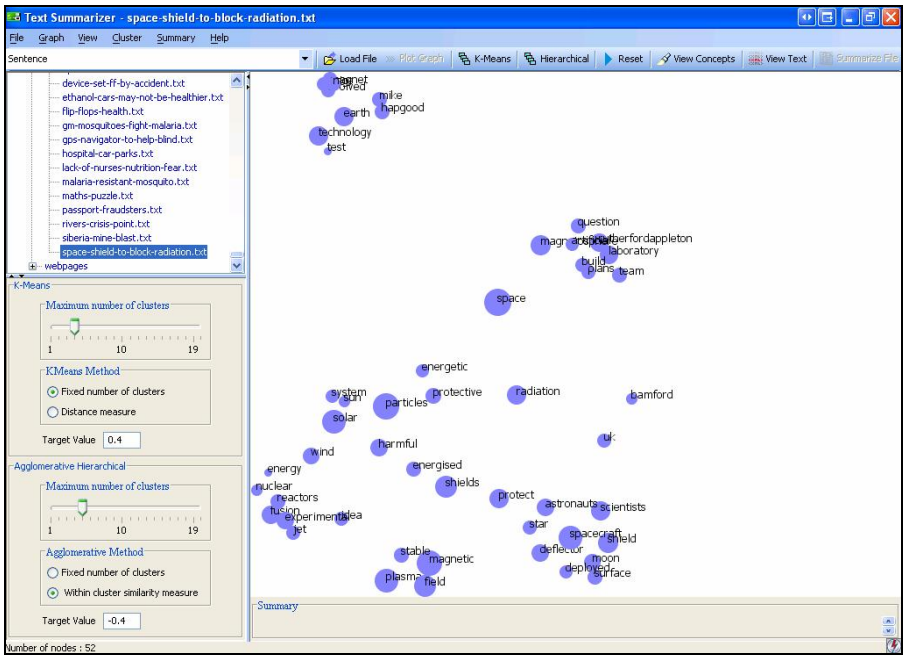

Figure 6.12: Graph shown the concepts

From the graph in figure 6.12 and 6.13, it can be seen that the concept “space” is the main concept in the document because of the size of its node.

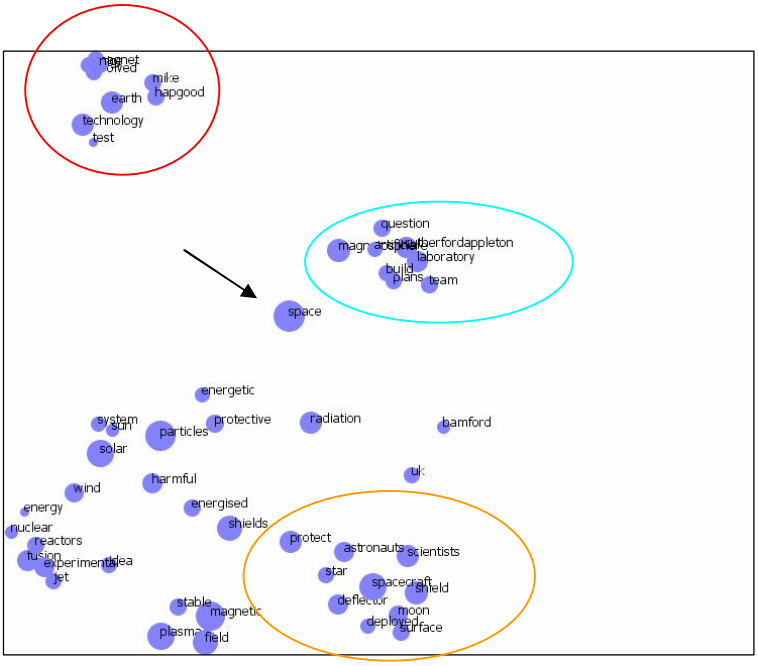

Figure 6.13: Showing some of the clearly separated clusters of concepts in the graph

The next step is to apply the clustering algorithms to the graph generated above.

Agglomerative Hierarchical clustering

Using the method that automatically determines the optimal number of clusters for the agglomerative clustering algorithm, the clustered graph in figure 6.14 is obtained.

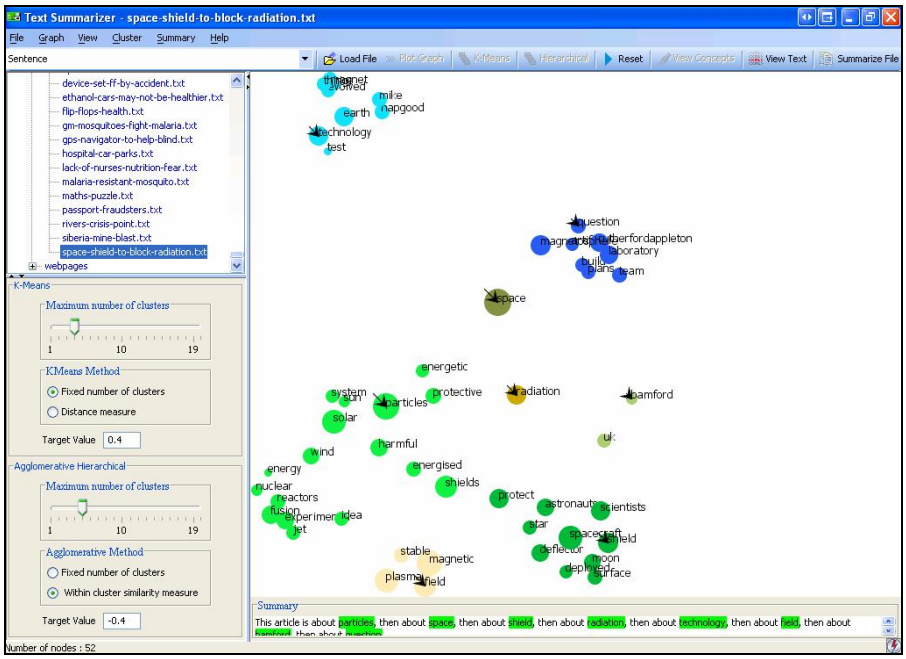

Figure 6.14

The clustering algorithm has clustered the graph into eight distinct clusters and the summary generated is "This article is about particles, then about space, then about shield, then about radiation, then about technology, then about field, then about bamford, then about question".

It is noted that the first six concepts in the summary describe the news article accurately. The two concepts "bamford" which is the name of a place in this case and "question" are not very relevant to the article. Although the last two main concepts the tool has picked out do not describe the article very well, they have been positioned last in the summary which shows how effective the tool is. The other feature of the summary generator is that it prioritises concepts that occur at the top of the document and this is clearly brought out in this example. The concept "technology" actually occurs more then the concept "radiation" in the article if the size of the nodes are compared but the reason "radiation" is placed before "technology" in the summary is because the former occurs in the first three paragraphs of the document as well as the following paragraphs whereas the latter only occurs after the first three paragraphs.

K-Means Clustering

When the K-Means algorithm automatically clusters the graph, only two clusters are obtained and the summary generated is "This article is about magnetic, then about space" as shown in figure 6.15. This summary does not really identify the main concepts of the article.

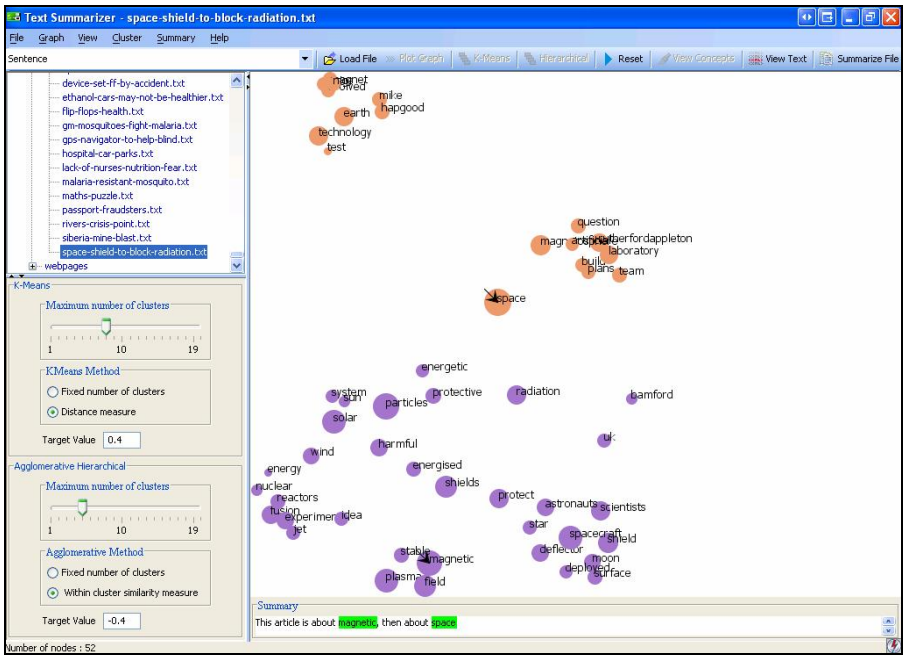

Figure 6.15: Graph showing two clusters

However, if we want to compare the output if the user had specified the number of clusters to be eight (to match the optimal number of clusters obtained for agglomerative clustering), the graph in figure 6.16 is obtained.

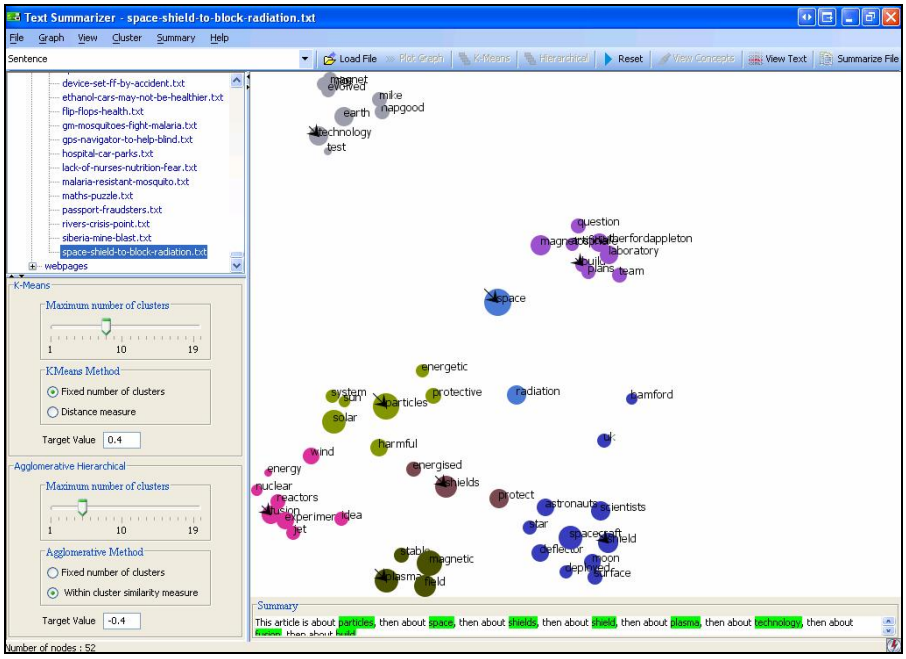

Figure 6.16: Graph showing eight clusters

This time the summary obtained is “This article is about particles, then about space, then about shields, then about shield, then about plasma, then about technology, then about fusion, then about build” where the first four concepts describe the article well but not the other four. It can be noted that the tool failed to identify that “shield” and “shields” are in fact the same concept and stemming would have fixed this problem. In this example, it can be noted that the agglomerative hierarchical algorithm gave a better result than the K-Means algorithm although both algorithms gave reasonably good results. This shows the effectiveness of our tool with respect to newspaper articles.

6.4. Poems

Poems are usually not very long and they tend not to repeat words a lot. The poem chosen is called “Fancy” by the poet John Keats. Part of the text of the poem is shown in figure 6.17 [27]. It is describing things that are fancy. From the text, it can be deduced that the context should be smaller than one sentence.

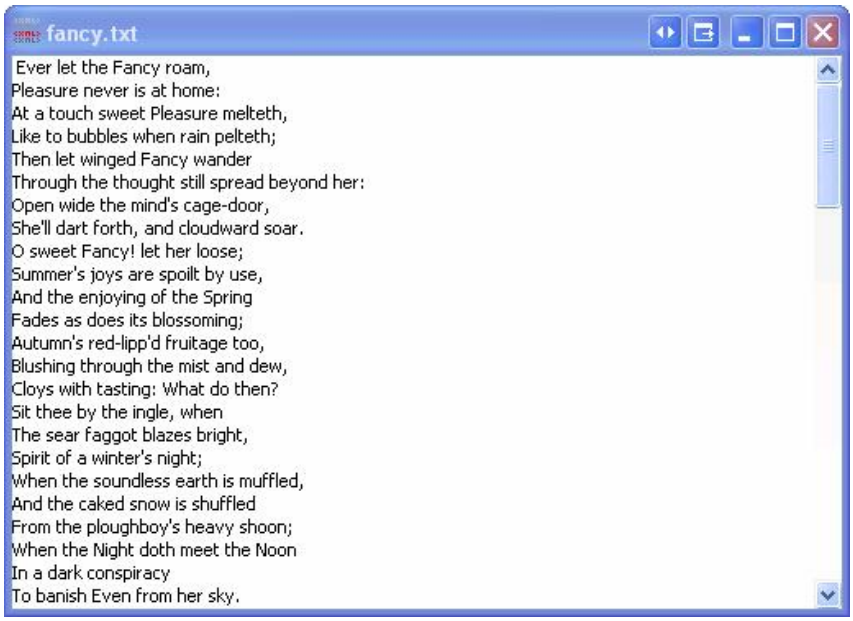

Figure 6.17: “Fancy” poem

The context is kept one sentence long for comparison. A graph is plotted and the optimal agglomerative hierarchical algorithm is used to cluster the graph. The output is shown in figure 6.18. The summary obtained is “This article is about hear, then about shalt, then about meet, then about bubbles, then about fancy”. Only the last concept in the summary (“fancy”) is relevant to the poem in this case.

When the context is made smaller (five words long), the graph obtained is almost unreadable because some of the concepts are very close to each other (figure 6.19). The optimal number of clusters obtained is eight and the summary is “This article is about doth, then about winged, then about thee, then about bubbles, then about home, then about earth, then about shalt, then

about fancy". The tool has again not been very effective in extracting the main concepts and has put the main relevant concept in the poem as the least important concept in the summary. Hence, it can be noted that the tool does not extract the main concepts from poems very well.

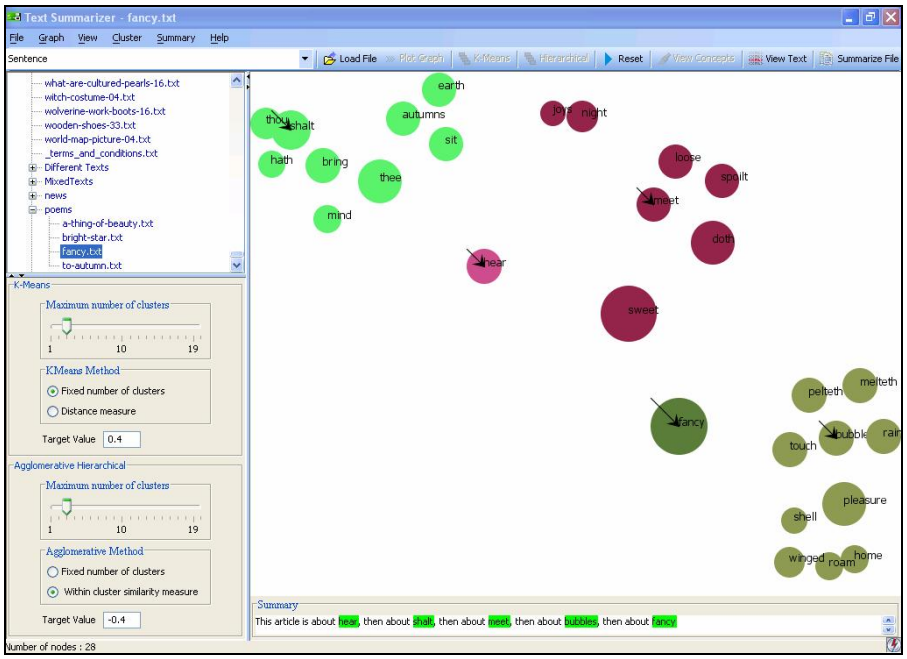

Figure 6.18: Clustered graph when the context is one sentence long

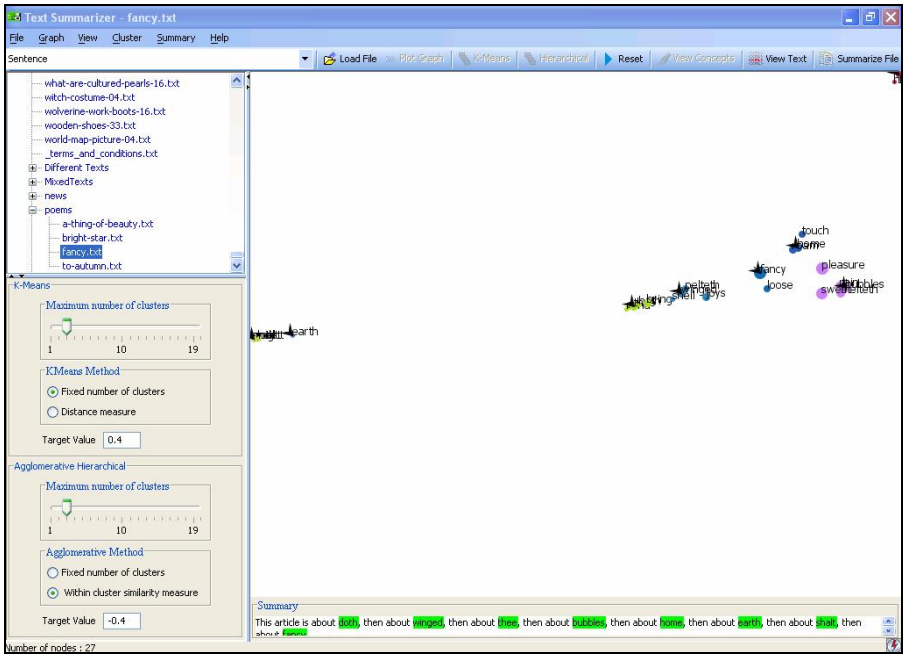

Figure 6.19: Clustered graph when the context is five words long

6.5. Novels

Novels are very lengthy pieces of text and it is expected that the main concepts are very hard to retrieve. The novel chosen is “Romeo and Juliet” and it is 6, 665 words long. The graph obtained is very dense and clearly separated clusters cannot be identified. The graph is clustered using K-means and assuming the number of clusters is chosen to be five, the result is shown in figure 6.20 below.

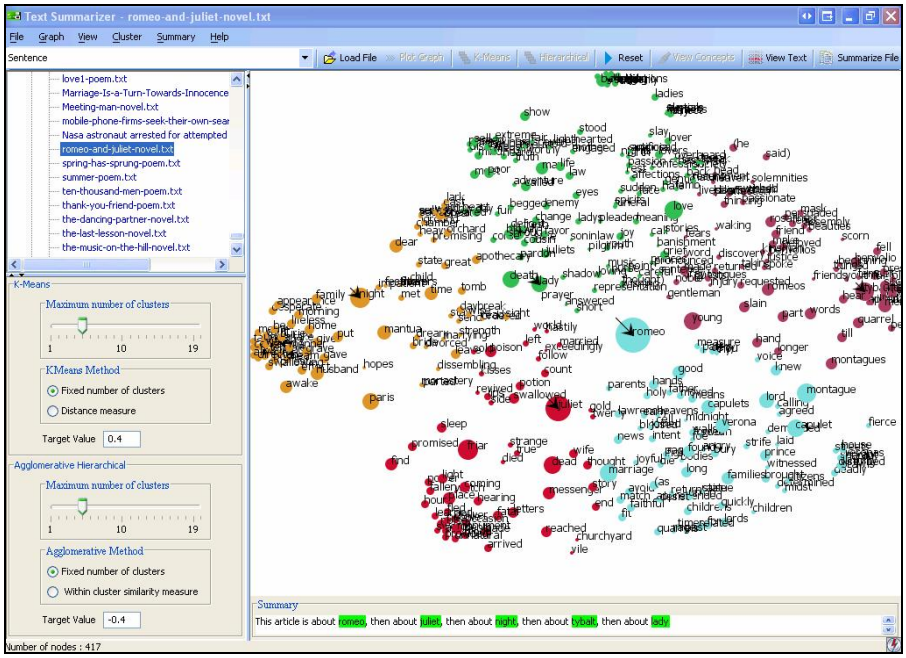

Figure 6.20: Using K-Means with K=5

The summary obtained is “This article is about romeo, then about juliet, then about night, then about tybalt, then about lady”. The tool did pick up the names of the main characters but it did not manage to extract what the novel is about. Therefore, we can deduce that the tool is not very effective at extracting the main concepts in novels.

6.6. Conclusion

It can be noticed that the tool is very effective at extracting the main concepts from structured texts such as articles and newspaper articles and not very effective with unstructured texts such as novels and poems. Another way to evaluate the tool is to test it with large corpora of text which have been tested with other similar applications. This will enable the comparison of the different outputs obtained and the relative effectiveness of the tool can be measured.

7. Conclusion

7.1. Achievements

A very effective tool has been implemented in this project for extracting the main concepts from a text document. From the evaluation carried out in chapter 6, it is noted that all the requirements described in chapter 3 have been met successfully. The unique feature of this tool is that it uses a completely new approach based on graph drawing and clustering techniques to identify the main concepts. This makes the tool very flexible because a lot of tuning is then possible. Tuning can take place in all the stages involved in the process of the concept extraction. The tool can therefore be adapted to suit the needs of the user and to analyse different forms of text.

A lot of research was involved in order to understand the different graph drawing techniques and decide which one would be best suited to the tool. Research was also carried out about the different clustering methods and existing methods of extracting main concepts from a document.

However, the tool has two main limitations which have been identified during the design and implementation. Words that originate from the same word stem for example “shield” and “shields” are treated as two different concepts. This can be eliminated if stemming is used when pre-processing the text (syntactic processing).

The other limitation of the tool is that it is very hard to understand large graphs because the nodes are too small and close together.

7.2. Future Work

Short-Term Improvements

The main improvement to the system is stemming and by using Porter’s algorithm [6] to implement stemming, the process of concept extraction will be improved considerably.

The other major improvement is zooming capabilities. This feature will be implemented on the graph and it will enable users to zoom into a particular area when they want to closely analyse a particular part of the graph or the text.

It has been noticed in the case of poems for example that the tool did not extract the main concepts properly. The main reason for that is that the choice of context is quite limited and the tool could not be adapted enough to cope with other contexts. More choice of contexts such as treating the word “and” as the start or end of a context can be implemented. This implies that sentences will be broken down into propositions using the word “and”. With more choice of context, the tool will be even more flexible and can be tuned more to analyse the different forms of text.

Adding more graph drawing techniques and clustering algorithms will also add to the flexibility of the tool by allowing more tuning.

Long-Term Improvements

The tool can be used in conjunction with internet searches. When a search is made in Google for example, a list of search results is returned and each of these results have a short text summary below it that summarises the main concepts in the text. It is often the case that these summaries are not relevant to the text or the main concepts in the summary are not the most important ones in the text. The tool implemented in this project adapted to work with web searches so that the content of the results (web pages) can be summarised in a more meaningful way.

The tool can also be adapted in order to be accessible to disabled people such as people who are visually impaired. This will imply incorporating sound effects in the tool and constant tracking of the mouse.

7.3. Final Thought

Overall, I have found that I have gained confidence from managing the project from conception to completion. This has enabled me to demonstrate many of the skills leant whilst studying at the University of Manchester whilst at the same time demonstrating my ability to learn new skills when required. This will provide me with skills that can be transferred to future personal or commercial projects.

The tool implemented will, hopefully provide the basis for future improvements and applications for example “Super Google” – a tool that will effectively summarise the contents of the web page results returned by Google.

8. References

[1] Berry, M. et al. SVDPACKC (Version 1.0) User's Guide, University of Tennessee Tech. Report CS-93-194, 1993 (Revised October 1996).

[2] Berry, M., Dumais, S.T., & O'Brien, G.W. (1995). Using linear algebra for intelligent information retrieval. SIAM: Review, 37(4), 573-595.

[3] Eades, Peter. A heuristic for graph drawing Stanton. Ralph G., editor. Congressus numerantium. Winnipeg: Utilitas Mathematica; 1984 May: 149-160. ISBN: 0-919628-42-7.

[4] Luhn's analysis, grids.ucs.indiana.edu/courses/xinformatics/searchindik/IRindexing.ppt.

[5] LUHN, H.P., 'The automatic creation of literature abstracts', *IBM Journal of Research and Development*, **2**, 159-165 (1958).

[6] Porter, M F, 1980, An algorithm for suffix stripping, *Program*, **14**(3) pp 130−137.

[7] Roberto Tamassia, Graph Drawing: Algorithms for the Visualization of Graphs (1999). Giuseppe Di Battista, Peter Eades
Roberto Tamassia, Ioannis G. Tollis.

[8] Graph drawings, http://www.yworks.com/en/products_yfiles_practicalinfo_gallery.htm.

[9] Graph drawings, http://www.informatik.tu-cottbus.de/~an/GD/ERLinLog/SouthernWomen.html.

[10] Andreas Noack. Energy-Based Clustering of Graphs with Nonuniform Degrees. Accepted for publication in: Proceedings of the 13th International Symposium on Graph Drawing (GD 2005, Limerick, Ireland, Sep. 12-14).

[11] System of forces, http://www.cs.brown.edu/~rt/papers/gd-tutorial/gd-constraints.pdf.

[12] Josh Barnes and Piet Hut. A hierarchical O (N log N) force-calculation algorithm. Nature, 324:446-449, 4 December 1986.

[13] Quad tree, http://www.cse.unsw.edu.au/~chak/dagstuhl/bh/problem.html.

[14] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. Technical report, Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, April 1991.

[15] Fruchterman.T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. Software -- Practic and Experience, 21, 1991.

[16] Clustering pictures, http://www.informatik.tu-cottbus.de/~an/GD/ERLinLog/Random.html.

[17] Jain, Murty and Flynn: *Data Clustering: A Review*, ACM Comp. Surv., 1999.

[18] Dendogram, http://www.resample.com/xlminer/help/HClst/HClst_intro.htm.

[19] Hirarchical clustering, http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/hierarchical.html.

[20] MacQueen J. B. MacQueen (1967): "Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability"*, Berkeley, University of California Press, 1:281-297.

[21] K-Means, http://www.autonlab.org/tutorials/kmeans11.pdf.

[22] Laudauer, T. K., Foltz, P. W., & Laham, D. (1998). Introduction to Latent Semantic Analysis. Discourse Processes, 25, 259-284.

[23] LSA application, http://lsa.colorado.edu/.

[23] Craig Larman , Applying UML and Patterns—An Intro to OOA/D and the Unified Process.

[24] GoF, Design Patterns: Elements of Reusable Object-Oriented Software(ISBN 0-201-63361-2).

[25] Shneiderman, Ben (1992). Designing the User Interface: Strategies for Effective Human-Computer Interaction: Second Edition. Addison-Wesley, Reading, MA.

[26] K Walrath, M Campione, A Huml, S Zakhour; “The JFC Swing Tutorial”, 2nd Edition, 2000.

[27] Fancy, http://www.poemhunter.com/john-keats/.

9. Appendix

Text of newspaper article

British scientists are planning to see whether a Star Trek-style deflector shield could be built to protect astronauts from radiation.

They argue that magnetic shields could be deployed around spacecraft and on the surfaces of planets to deflect harmful energised particles.

Several countries' space agencies have announced their intentions to resume human exploration of the Solar System.

Scientists hope to mimic the magnetic field which protects the Earth.

Details have been presented at the Royal Astronomical Society's National Astronomy Meeting in Preston, UK.

There are a variety of risks facing future space explorers, not least of which is the cancer-causing radiation encountered when missions venture beyond the protective magnetic envelope, or magnetosphere, which shields the Earth against these energetic particles.

The nice thing is that magnet technology is really quite evolved here on Earth. The question is can you take it into space?

Mike Hapgood,

Rutherford-Appleton Laboratory

The Earth's magnetosphere deflects many of these particles; others are largely absorbed by the atmosphere.

Between 1968 and 1973, the Apollo astronauts were only in space for about 10 days at a time.

They were simply lucky not to have been in space during a major eruption on the Sun that would have flooded their spacecraft with deadly radiation.

Crew members on the International Space Station can retreat to a thick-walled room during times of increased solar radiation.

Stable field

But these protective shelters would not be practical on long-duration space journeys, since the "drip-drip" of energised particles is thought to be as harmful to the health of astronauts as large solar storms.

The harmful particles come from the Sun, in the form of the solar wind, and from sources outside our Solar System.

To create the deflector shield around a spacecraft or on the surface of a planet or moon, scientists need to generate a magnetic field and then fill it with ionised gas called plasma.

The plasma would held in place by a stable magnetic field (without the magnetic field, the plasma would simply drift away). This shield could be deployed around a spacecraft or around astronauts on the surface of a planetary body such as the Moon.

As energetic particles interact with the plasma, energy is sapped away from them and they slow down.

"You don't need much of a magnetic field to hold off the solar wind. You could produce the shield 20-30 kilometres away from the spacecraft," explained Dr Ruth Bamford, from the Rutherford-Appleton Laboratory in Didcot, UK, one of the scientists on the team.

Dr Mike Hapgood, from the Didcot-based research centre, told BBC News: "The nice thing is that magnet technology is really quite evolved here on Earth. The question is can you take it into space?"'

The team from Rutherford-Appleton plans to build an artificial magnetosphere in the laboratory. They would eventually like to fly a test satellite which would test the technology in space.

'Shields on'

The idea has been likened to the deflector shields which protect the USS Enterprise and other spacecraft in Star Trek. Like their fictional counterparts, these shields could also be switched on and off.

An artificial magnetosphere could come in handy anywhere in the Solar System where humans would need to be for long durations.

A permanent Moon base, of the type Nasa plans to build, could be buried under lunar soil to protect the occupants and equipment from space radiation. But inhabitants will still be vulnerable when venturing outside in their spacesuits.

"Our warning systems aren't very good [for solar flares]. You might be able to say: 'this is a dangerous period in terms of solar activity', but you might be on red alert for weeks," said Dr Hapgood.

"If you've got a problem, you might not want to wait a week to fix it. You might want a device to deploy on the surface as a shield that would blunt the effect of a flare at ten minutes' notice, it adds an extra level of safety."

The idea for the shields draws on technology pioneered in experimental nuclear fusion reactors. Nuclear fusion is not yet a mature technology.

It works on the principle that energy can be released by forcing together atomic nuclei rather than by splitting them, as in the case of the fission reactions that drive existing nuclear power stations.

At the Jet experimental fusion facility at Culham in the UK, magnetic fields were used to keep plasma away from the interior wall of the reactor.

This represents a reversal of that technology: "We want to use the same technique to keep an object in the middle away from plasma that's on the outside," said Dr Bamford.

But the plasma needed to protect against particles from the solar wind and elsewhere would actually be weaker than that generated in experimental fusion reactors like Jet.