

CHAPTER-1
INTRODUCTION

1. INTRODUCTION

In the present day world for all our queries, our homework etc we have become dependent on the use search engines. Our demands do not stop with direct answers we want faster response times, related results etc. Users are increasingly interested in getting related documents and finding their different questions answered instead of just being given a keyword based Boolean retrieval based result. To support better retrieval, researchers are turning towards the semantic web, where the potentials for answering such queries are high. To achieve this, representation of semantic information is of primary importance. Knowledge representations are used as a backbone for the semantic web.

Among the different data structures used for knowledge representation, graphs are being widely used to represent knowledge because of its simplicity, flexibility, understandability, and expressivity (e.g. if two nodes are linked in a graph then it means that the nodes are related). In these graphs each node contains the information about a data entry and the edges represent a relationship between two data entries. Data entries are usually key terms which are specific to a domain. Relationships between data entries capture the semantic connectedness between them. Two terms are semantically related if they are terms describing a common topic, or if they indicate a topic-subtopic relation. The relationships can be defined at different domain granularities. For instance, two terms may be related under the domain computer science but may or may not be connected under the domain 'Data Structures'. The power of the graph in terms of its knowledge expressivity is dependent on the accuracy and comprehensiveness of the relations.

Graphs that provide information about the type and nature of a relationship between two nodes are termed as concept graphs. One approach that has been widely used to represent knowledge is ontologies. Ontology represents knowledge as a set of concepts, and also provides the relationships between these concepts. But they are quite a few disadvantages with this presently existing technique of concept graphs.

One of the main disadvantages with it is that relationships in the ontology are a result of the views of different people involved in the domain. Different views of people make any general purpose ontology impossible. In addition this involves more manual intervention, which is impractical for creating concept graphs from large data sets. The other approach is to use statistical techniques to generate graphs. They use limited knowledge and employ no pre-defined assumptions to produce a graph capturing certain concepts and relations between these concepts. However, statistically generated graphs while being faster and easier to generate, lack the expressivity of ontologies. The primary reason being, currently there are no known techniques to analyze these graphs for semantic relationships.

Between the statistical graphs that provide less information and the concept graphs that can't be generalized and require human intervention the goal is to strike a balance between them. Statistically generated graphs can be used as a foundation, where relationships that are defined ontologically may be incorporated later. Statistical techniques can help generate concept graphs by easily identifying co-occurring terms in documents, and their proximity to each other. These features can help provide high level semantic insights. Using natural language processing over the generated graph, generating ontologies will become easier and involve lesser manual intervention.

Towards this goal, we have explored existing statistical representations (co-occurrence graphs[4], word distance graphs[2] for generating concept graphs. We also integrate these two graphs to generate a hybrid representation. The primary contribution of this project is to analyze the three representations for semantic properties, so that we can determine the power of such statistically generated graphs for expressivity. Such studies have not been done so far on these graphs to the best of our knowledge, and are an essential exercise to use the power of these easily able to generate graphs. We first study explicit graph theoretic properties like edge-weight, betweenness, degree etc for semantic meaning. Next, we apply graph algorithms to these graphs to analyze clustering properties, hierarchies, etc.

In addition we study the efficiency, scalability, and reliability of these different graphs over different data sets, and spot the advantages/disadvantages of each method. Finally we develop a GUESS based tool to support the concept graph creation and analysis. The Guess based tool helps study both the explicit graph properties and supports functions for clustering, identifying cluster to cluster link, related terms for a search term, level based search, clustering, and minimum spanning tree generation. The next section will explain the literature survey and motivate the need for this work.

CHAPTER –2
LITERATURE SURVEY

2. LITERATURE SURVEY

The value of time is increasing especially in the field of searching.” The faster the better” becomes our motto but it does not give us the authority to allow unwanted results. Hence a statistically created concept graph to speed up search and to allow deeper term relationships seems to be the way out.

The author of [1] tries to create a fast and adaptive layout algorithm to visualize graph layouts. The author uses the spring-embedded paradigm and introduces several new heuristics to improve factors like the convergence, local temperatures, gravitational forces and the detection of rotations and oscillations. The author believes that the proposed algorithm achieves visualizations of high quality on a wide range of graphs on standard settings. The algorithm used seems to reduce time, thus it may also be applicable on general undirected graphs of substantially larger size. Our project uses the Gem layout as one of the visualizations to help extract semantic data. The author also provides proof for the fact that it is indeed faster than kamada kawaii and fruchterman-Reingold algorithms.

The authors of [2] try and understand the framework behind the term occurrence graph. The author takes into account all the past notions of distance playing a factor in the information retrieval of different graphs. The authors compare different works that use distance based relevance calculations and try to bring up a theoretical framework. He explains the importance of term distance which even now is one of the key factors in information retrieval. For our project we have also created a graph using term distance and have been able to see quite a few relations and clusters being formed in it which proves the efficiency of the graph.

The authors relate to the problem of graph drawing or graph layout where there are a set of nodes and the edge weight between them. They have taken the Eades and modified it to provide better results .This layout is now known as the fruchterman-

Reingold layout. They have mapped the graph system with physical forces to create a heuristic to simplify graph visualization. They assume the graph to be like that of the universe or a molecule where there are both attractive and repulsive forces. These graphs are commonly known as force directed graphs. The author was able to model it and compare it with the previous models of kamada kawaii and Eades which provides good results. This is one of the few layout visualizations used for our project to understand simple statistical graphs [3].

The Key graph is the name the authors of [4] have given their co-occurrence graph. It is a graph created based on the co-occurrence of two or more words in a collection of documents. The algorithm is based on the segmentation of a graph, representing the co-occurrence between terms in a document, into clusters. Each cluster corresponds to a concept on which author's idea is based, and top ranked terms by a statistic based on each term's relationship to these clusters are selected as keywords. This is another statistical graph that we have taken into account in our semantic pattern search.

The author of [5] speaks of ways of improving the already present association algorithms such as apriori and concept lattice. The transitive path of association relation is discovered, by building a cyclic graph model. The keywords are divided into Document keywords, Domain keywords. The initial phase is to discover semantic relations such as “if A then B”. Using association relations between two keywords semantic relations are determined using their semantic degree which is intersection of the two keywords by the union of them.

- Ex:-If A and B are two keywords and if A is associated with {C,D} , and B with {E,F} the $A \cap B$ will be nothing or '0' and $A \cup B$ will be {C,D,E,F} if $\alpha = \frac{N(A \cap B)}{N(A \cup B)}$ [α is the semantic degree] then $\alpha=0$ which in this case would imply that A and B are not related.[N implies the number of items].

Then a semantic matrix which will be an nxn matrix of the keywords and their

relations determined above will be created. Different properties of the relations are studied like transitivity etc. The author says that it becomes hard to extract long length transitive paths between documents using the document level discovery algorithm and that the number of documents influences the transitive path.

The author of [6] speaks of a higher studies degree map. The map is to relate professional profiles, subjects required for the profile, and competencies required for subjects and acquired from subjects and the relationship between these. The author proposes a concept map wherein each subject will be mapped with its previous competencies and what competencies it contributes to. The mapping helps explain precedence and dependency relationships. Graphviz is the tool used. The relationships between entities are stored in the database which the graphviz tool uses to generate the map. The problems faced were adding new relations to the system which had to be done directly to the database, adding an extra entity becomes impossible.

The authors of [7] say that the apriori algorithm is used for item set mining and association rule generation. It is used on databases that contain transactions. It is a bottom-up approach. First it identifies frequent individual items in the database and then extends to larger item sets provided these sets appear often in the db. The item sets are then used to determine association rules. Breadth-first search and Hash tree are used to count the candidate sets (Item sets that occur frequently. They can also contain only one element).

- Apriori property says that a subset of any item set that is frequent must also be frequent .E.g:- if $\{A, B\}$ is frequent the $\{A\}$ should be frequent and $\{B\}$ should be frequent.If k is the number of items in a set then any $(k-1)$ item set that is not frequent cannot be a subset of the frequent k -item set.

Various approaches have been employed by systems in past years to develop techniques/methods that aid in the extraction of relationship from concept graphs. The

essential software tool for our project is the open source graph visualization software, **GUESS**, which helps visualize graphs in different layouts. Our project employs statistical methods to extract information from concept graphs that are visualized by considering the nodes as keywords and the edges representing the relation between them in statistic terms. The existing approaches that aids the statistical approach we used is discussed in this section.

The two major statistical approaches employed across all systems used for knowledge representation are Co-occurrence [4] and Word-distance [2].

Co-occurrence is based on the occurrence of two or more words together in multiple documents. Using this technique Yukio [4] came with a co-occurrence graph . It is a graph created based on the co-occurrence of two or more words in a collection of documents. The algorithm (Key-graph) is based on the segmentation of a graph, representing the co-occurrence between terms in a document, into clusters. Each cluster corresponds to a concept on which author's idea is based, and top ranked terms by a statistic based on each term's relationship to these clusters are selected as keywords. This strategy comes from con-considering that a document is constructed like a building expressing new ideas based on traditional concepts.

Since scanning through entire set of documents is time consuming the author employs a small set of terms to serve as representatives for the entire set of documents. This indexing process reduces the human effort in sifting through vast amounts of information. These keywords are used as a basis not for only for information retrieval but also as an abstract for a document. The terms that represents the view of the document is considered as keyword by Yukio [4].

The author compares his key-graph method to that of a construction of building as a metaphor this building has foundations (statements for preparing basic concepts), walls, doors and windows (ornamentation). But, after all, the roofs (main ideas in the document), without which the building's inhabitants cannot be protected against rains

or sunshine, are the most important. Here the authors refer to the keywords as the roofs as they are the main terms in the document. Co-occurrence of two or more terms is checked across multiple documents with help of a count. So more the count of the terms occurring together more they are related is what could be observed from the authors point of view.

But co-occurrence alone cannot give the exact semantic relation out of a concept graph, The authors of [2] try and understand the framework behind the term occurrence graph. The author takes into account all the past notions of distance playing a factor in the information retrieval of different graphs. The major benefit of this approach is that relevance scoring does not rely on collection frequency statistics and that distance between the terms as a primary estimate.

The author proposes that in addition to document relevance, distance measures have application in identifying phrases with a greater degree of tolerance for word substitutions and unexpected intervening words. The span length i.e. the length between two words is estimated using two possibilities here

1. The length of the intervening sequence
2. The sum of the length of the interval spans

Both of these approaches have their pros and cons. Though co-occurrence is able to find out the related terms across multiple documents, the terms that don't span across multiple documents and are related are omitted by this approach. The author of [2] seems to achieve the outcome of getting relations between terms but the word distance approach fails in certain cases. For example in index pages of different books unrelated terms may occur next to each other, so word distance will be taken as 1 which shows maximum relationship but they need not necessarily be related.

For our project we have considered both of these statistical approaches and tried

combining these approaches to get the precise relation between terms in a document. We came up with slightly different approach of Co-occurrence by considering an adjacency matrix with rows and columns as keywords and the matrix is updated each time the keyword pair gets a hit in the file. We have also created a graph using term distance and have been able to see quite a few relations and clusters being formed in it which proves the efficiency of the graph. The same matrix approach has been followed in this approach, the minimum distance between keywords is found and the matrix is updated. After analyzing the pros and cons of both the approaches, to help understand the graph better we came up with minimum spanning tree that consists of mostly important edges which was done by removing all unwanted edges based on the edge weight criteria.

CHAPTER – 3
DESIGN ISSUES

3. DESIGN ISSUES

This chapter includes the problem statement which is considered to develop this project; it also deals with the data sets taken as the input file for the various implementations in the code and algorithms. A few objectives of this module along with the assumptions made are following.

3.1 Problem statement

To study statistical and graph theoretical approaches for generating and analyzing concept graphs, and to develop a GUESS based tool for supporting this.

3.2 Objectives

The primary goal of this work is to study simple statistical approaches in order to reveal the potential expressivity of such graphs. This can help generate concept graphs with less complexity, while capturing the semantic features necessary for current applications. To do so, a comprehensive understanding of the graphs generated using the above said methods by observing key factors like betweenness, edge weight, degree, and cut edge etc to analyze for semantic features is essential. Once these factors explicit to any graph are analyzed, the goal is to apply algorithms like clustering and spanning tree generation that can reveal more insights into the graphs like topic based clusters, hierarchical clusters, topic subtopic hierarchies, links between classes etc.

CHAPTER - 4

SYSTEM DESIGN

4. SYSTEM DESIGN

The aim of the project is to design a system that automates the generation and analysis of graphs that represent knowledge extracted from documents containing data related to different domains.

Although there are various existing statistical approaches for generating concept graphs and extracting information from it, they have not been studied adequately for semantic patterns. Our project combines the current graph generating methods and tries to create graphs that has terms as nodes, depicts the type of relations between these nodes, analyzes it using different properties, forms clusters using these relationships and also describes the relationship between different clusters. This helps us to capture the semantic information with the combined statistical approach and is more efficient.

4.1 Proposed System Architecture

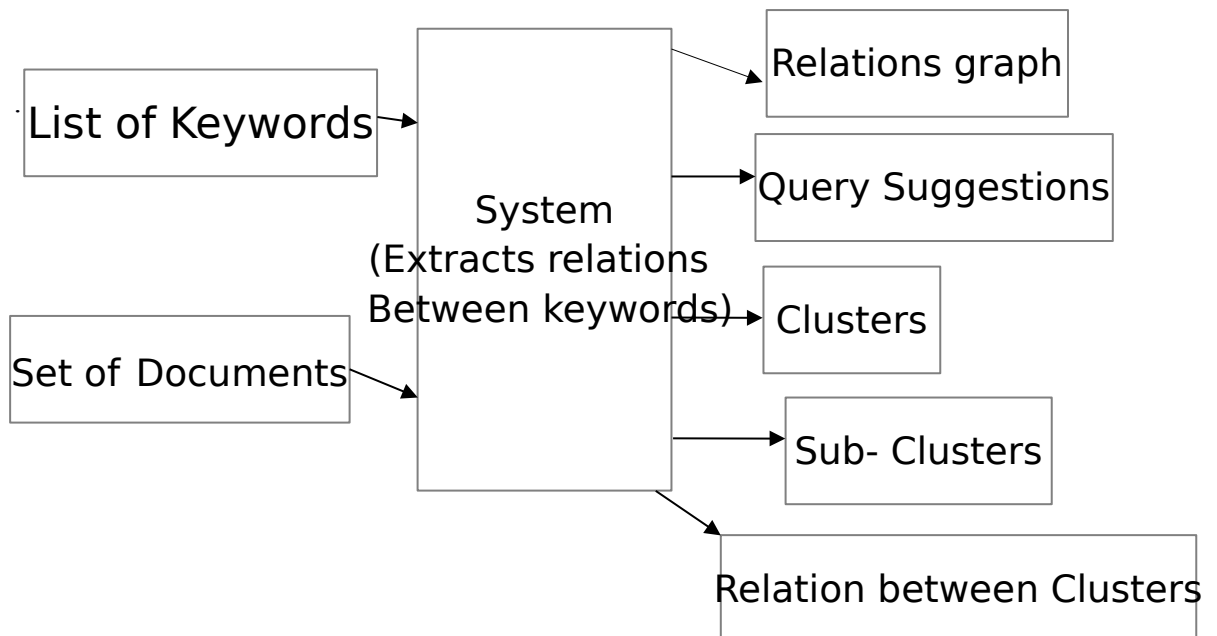


Fig.4.1.System Architecture

Inputs for this system are list of keywords and set of documents. Words from indices of different textbooks are taken as list keywords. Documents are taken from Wikipedia pages and different textbooks in corresponding domain.

4.2 Different approaches for graph generation

The Extraction of relations between keywords was done with the help of existing statistical approaches with a slight variation in the algorithm.

4.2.1 Co-occurrence Method:

Co-occurrence is a graph created based on the co-occurrence of two or more words in a collection of documents, in our approach we came up with an adjacency matrix that contains keywords as rows and columns. Every time a pair of keywords gets a hit in the document the matrix is updated. Each cell in the final matrix depicts the relation between the corresponding pair of nodes in statistical terms. The greater the value of the cell in the matrix the higher the relatedness value of the corresponding nodes.

Algorithm for Co-occurrence:

Input Set:

Master file – contains file names and Set of keywords corresponding to each file.
 Key file-File containing set of all keywords
 fp \equiv file pointer to master file
 w \equiv list containing the keywords of corresponding file
 W1,W2 \equiv pointers to the keywords in the file
 fn \equiv file pointer to the key file
 Size \equiv No of keys in the key file
 Array [Size][Size] \equiv Matrix of length Size \times Size
 i,j \equiv variables

Function (fp, fn, Array, Size):

```

for i in size:
    for j in size:
        do
            Array[i][j]  $\equiv$  0, j  $\equiv$  j+1
    i  $\equiv$  i+1
    for W1 in w:
        for W2 in w:
            if W1  $\neq$  W2:
                do
                    Array[W1][W2]  $\equiv$  Array[W1][W2]+1

```


4.2.2 Word distance Method

Another statistical approach we used is word distance, as the term indicates it is the distance between two or more words across multiple documents. The same adjacency matrix approach is followed here. A time line of keywords is created according to the order in which the keywords occur in a document and the distance between adjacent terms and every keyword to another is updated in the matrix. The process is done for every document and each time the average value of the distance between the two terms is updated in the matrix.

Algorithm for Word distance:

```
Input Set:
Master file - contains file names and Set of keywords
corresponding to each file.
Key file-file containing set of all keywords
fp≡file pointer to the master file
fn≡file pointer to the key file
index[][]≡ list of indexes of each keyword present in the
corresponding file
Size≡no of keys in the key file
w≡list containing keywords of corresponding files
Array[Size][Size]≡Matrix of length Size×Size
m,i,j,n≡ Variables used
min≡a minimum value is set as a base line
Function(fp,fn,index,Size,Array):
    for i in 0 to size:
        for j in 0 to size:
            do
                Array[i][j]≡0,j≡j+1

                i≡i+1
                n≡0
                m≡0
                for k in 0 to len(w):
                    for l in k+1 to len(w-1):
                        while m<len(index[i]) and
n<len(index[j]):
                            if abs(index[i][m]-index[j][n])<min:
                                do min=abs(index[i][m]-
index[j][n])
                                    if index[i][m]<index[j][n]:
                                        do m=m+1
                                        else.
```

4.2.3 Combination of Co-occurrence and word distance

The analysis of the above two approaches was done for two different data sets and it was found that both approaches had their own pros and cons. We used a hybrid of the two approaches and came up with a graph which helped in retrieving relation in an efficient manner.

Algorithm for Hybrid approach:

```
Input Set:
Key file ≡ File containing set of all keywords
Occur[] ≡ list containing Co-occurrence values
of keywords
Word[] ≡ list containing word distance values
of keywords
word ≡ list containing the keywords of
corresponding file
w1,w2 ≡ pointers to the keywords in the file
i,j,value ≡ Variables
fp ≡ file pointer to the keyfile
Size ≡ no of keys in the key file
Array[Size][Size] ≡ A matrix of length Size
×Size
Function(fp,Array,size,col,wd):
    for i in 0 to size:
        for j in 0 to size:
            Array[i][j] ≡ 0

    for w1 in word:
        for w2 in word :
            if w1!=w2:
                Array[i]
```

4.3 Different semantic parameters used to analyze the graphs:

The analysis of the above two approaches were carried out with the help of few

parameters that brought out the semantic relations from the graphs. The parameters are as follows:

Degree:

The number of links incident up on a node. It defines how many connections end up on a node or the edges that are directed towards the node. A high degree usually denotes that a node is potentially the key topic of a cluster.

Betweenness:

Quantifies the number of times a node acts as a bridge along the shortest path between two other nodes and this parameter helps in finding out the connection existing along two nodes with less edges on the path. Betweenness along with degree identify the important terms in a graph.

Edge weight:

It is one of the important parameter for analysis. The above two approaches are related to this parameter as it helps in related the nodes based on its value. It quantifies the relation between two nodes, that is, how semantically related the two nodes are.

Cut edge:

It is an edge that connects two different topic clusters. The link or the edge that connects the two clusters usually indicates a pair of terms that are not closely related, but form links to different topics.

We analyze these properties over the three types of graphs and identify the patterns which indicate specific semantic properties.

4.4 Extraction and analysis of semantic clusters

Though the above approaches were able to retrieve relations from the corresponding graphs, the study of these graphs requires lot of effort. Since the nodes/terms are

connected to each other and formed a structure similar to that of a mesh network, it became a tedious task to analyze and experiment with the graphs. So we design an approach that can give us the hierarchical view of these graphs which will make analyzing and searching faster. A Minimum Spanning Tree is constructed for the connected weight graph with the Kruskal's algorithm. This gave us a hierarchical view of the graphs for both of the approaches helping us to better understand the relationships existing between nodes and for analyzing individual clusters, and their relationships with each other.

A minimum spanning tree or a maximum spanning for the co-occurrence graph, removes the weak links between terms, thereby clustering the closely related terms. In addition this captures the topic-subtopic relationships between terms. For clustering the graph, the minimum spanning tree is taken, and cut edges are identified based on the weakest links in the tree. Clustering can be done at different levels. Once a topic cluster is identified, it can be further clustered for subtopics. We use these clusters to identify related terms, terms serving as bridges between topics etc.

To identify cluster we develop a greedy approach over the minimum spanning tree.

An edge connecting two clusters connects vertices having high degree and betweenness, and its cost is high (word distance graph) (or low for co-occurrence graph).

Analyzing the different properties like betweenness and edge weight of the spanning tree we noticed that we can retrieve different types of relationship between keywords or terms in the clusters. The different types of relationship include topic-subtopic relation, inter-topic link and intra-topic relation. These algorithms for finding these relationships have been implemented and integrated into an application which is an extension of GUESS. The interface of this application has been done in JYTHON. This application not only helps us to retrieve the different kinds of relations but also helps in the analysis of these graphs. Details of this application along with the rest of

the implementation details and analysis will be discussed in the next section.

CHAPTER-5

IMPLEMENTATION, ANALYSIS AND RESULT

5 IMPLEMENTATION, ANALYSIS AND RESULT

In this section we analyze our three graphs, by first analyzing the overall graph, then their properties like edge weight. Then we analyze these graphs by extracting domain classes and subclasses using our clustering algorithm. We also analyze the topic subtopic hierarchies in these graphs by using the minimum spanning tree generated over these graphs. Finally we discuss the GUESS based extension and show how it provides functionalities to effectively study concept graph for potential applications.

5.1 Analysis of the CO-OCCURRENCE graph

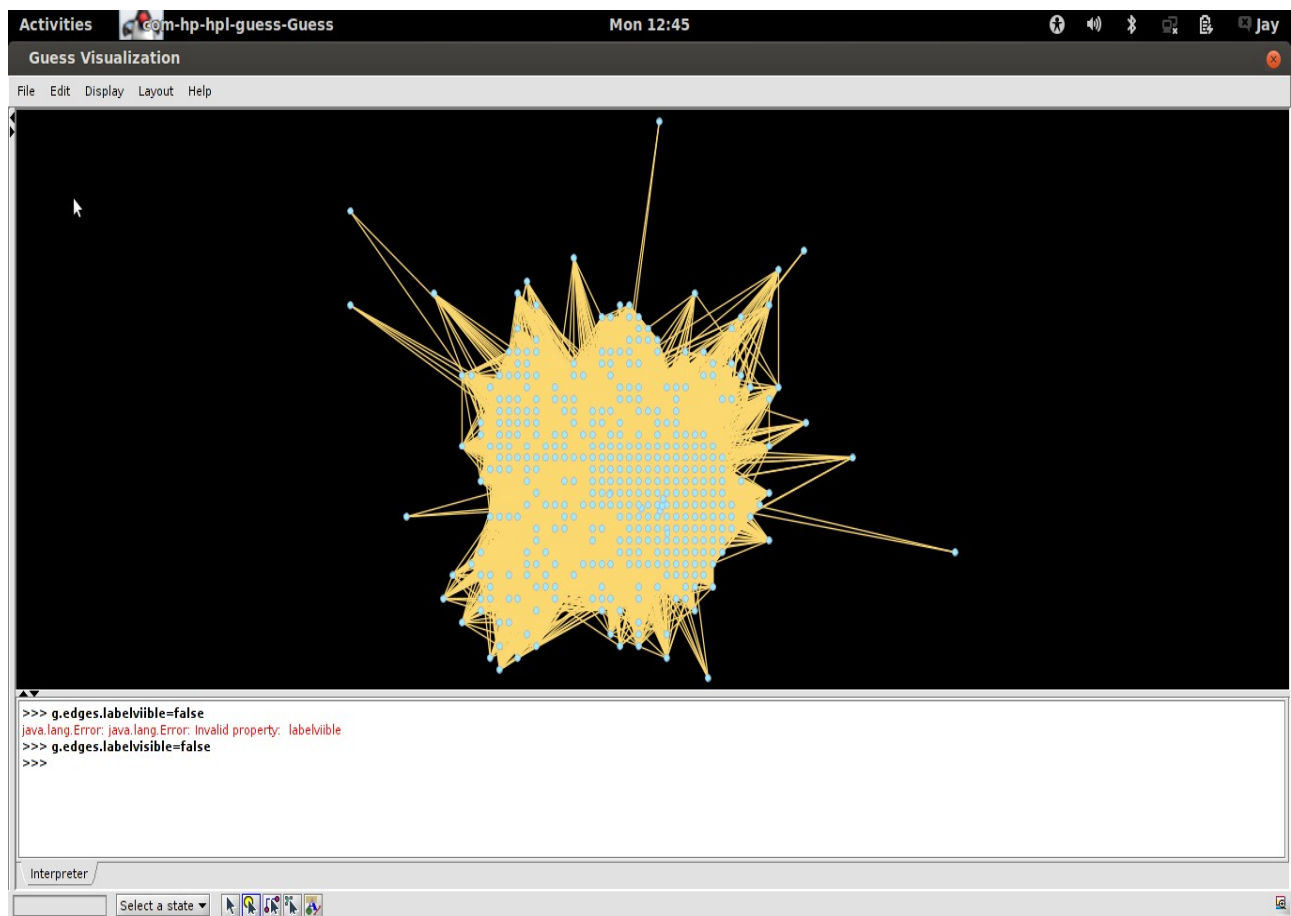


Fig.5.1. Co-occurrence Graph

Fig 5.1 shows a sample co-occurrence graph where nodes are terms in Engineering, and edges connect terms occurring in the same document. Each edge in co-occurrence

tells how related two key words are. Edge weight of co-occurrence is the number of times both the key words occurred together. More the value of edge weight more related the key words are. By visualizing this graph using the Kamada Kawai graph visualization technique [2], we can observe that the terms belonging to the same topic are closer in the visualization. This demonstrates that a simple co-occurrence graph can effectively capture semantic similarity among terms. Given this observation, we analyze this graph further for semantic information.

5.1.1 Analysis of Edge Weight

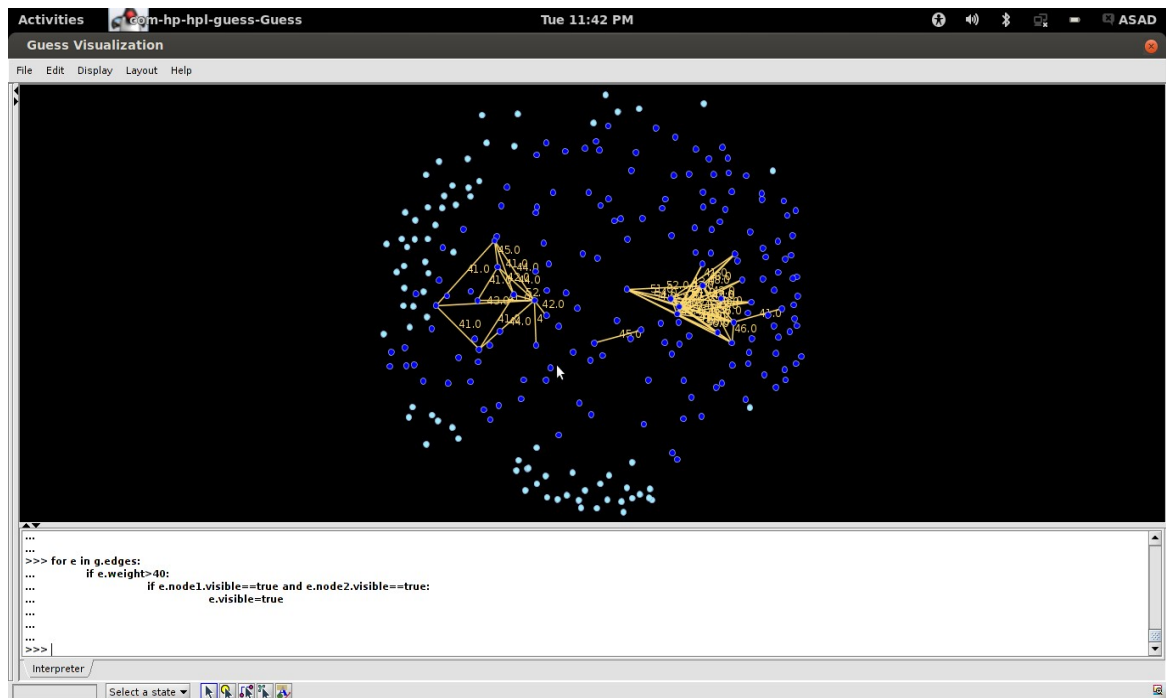


Fig.5.2 Edges with weight>80% maximum

In a co-occurrence graph, the large the edge weight, the closer the terms, since it indicates that these terms occur together in many documents, potentially indicating a topic similarity or relatedness. However since edge weights vary based on data set and size of data set, we perform an analysis to determine a common metric that can define the minimum edge weight which links two nodes which belong to a same topic, and are hence closely related. We do this by adding the edges in the graph in decreasing order of weight, till we reach a point when they do not play a role in intra-

topic relationships. As edges of a certain weight category are added, we observe topic based clusters being formed automatically.

Edge weight > 80% of maximum edge weight:

When edges with weight greater than 80 (Fig.5.2) were analyzed it has been observed that the edges of the nodes which are more related being selected. Internet and computer networks belonging to the network cluster being displayed is one such example. Many such cases have been found where nodes with more edge weight are being displayed related very closely to each other. Few of the cases are binary tree and heap, hash and tries etc. These belong to the cluster data structures and algorithms. This analysis shows that the relation between two nodes or keywords is directly proportional to their edge weight. The more the edge weight between two nodes the more they are related. We observe that in general, the edges indicate that the connected terms have a generalization-specialization relationship, or are categories under the same parent types.

Edge weight > 40% of maximum edge weight:

When edges having weight greater than 40% of maximum edge weight, the edges pertaining to the network clusters were displayed high in numbers leading to the fact that there are large number of keywords in network domain and that every node in the network cluster is related to at least another node in the cluster in one way or the other. To analyze this further let us take for example the edge between Ethernet and LAN. The term LAN defines a network that works over small distance and the term Ethernet defines a type of networking protocol used on LAN. Their semantic relationship is not that of similar terms describing same topic, or subtopics under the same topic. However here the relationships captured are of those between topics and subtopics like 'uses', 'is a medium of', 'similar to' etc. Ethernet and Internet, Television and Radio, radio and wireless are some examples that were found.

Edge weight>30% of maximum edgeweight:

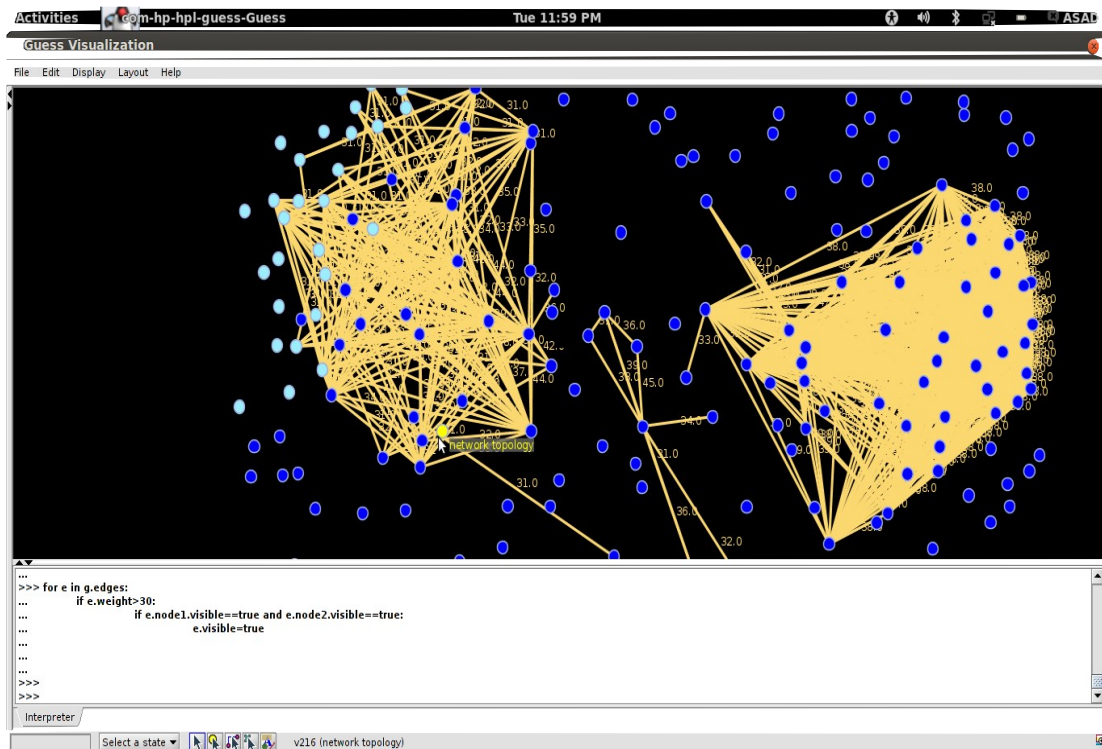


Fig.5.3. Edges with weight>30% maximum

When edge weight is further extended to include those with weight greater than 30% (Fig.5.3), more edges are added to the clusters, making them more defined. Inter-cluster edges do not appear in this level yet. However they do not add much to the semantic relationships except contributing to cluster formation. The relationships defined by these edges are similar to the one above. The figure shows that clusters are being more clearly defined and strong connections amongst the nodes are formed. The left side cluster shows the cluster of Computer Networks, right side is the algorithm and data structure cluster and the middle one shows database cluster. The clusters are completely disconnected and no inter-cluster relations were observed. Transaction and Query, query and database, internet-protocols and IEEE were some of the edges that were observed.

Edge weight > 20% of maximum edge weight:

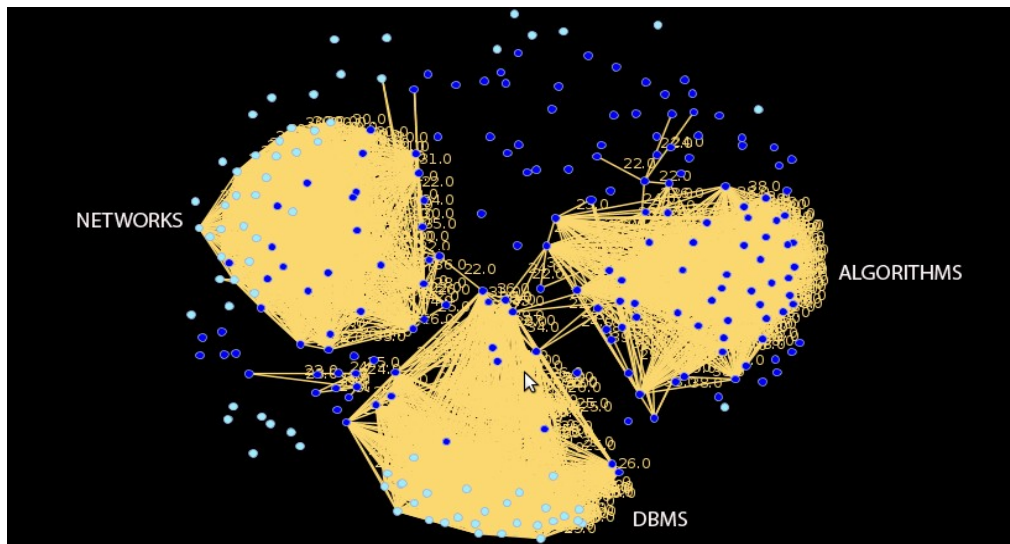


Fig.5.4. Edges with weight > 20% of maximum

More intense clusters were formed when edge weight limit was made 20%. As you could notice, in the graph, the cluster on the left side shows the networks, on the right is the cluster of algorithm and data-structure and on the middle is database cluster. At this level inter-cluster edges are appearing. For example, NULL (Database) and Binary tree (Data-structures). Null is a common term occurring in both database and data-structures. Another example is IEEE (Networks) and transaction (Database). So we can define the edge weight for edges connecting different clusters, and those that may potentially be cut edges as being in the range between 20-30% of maximum edge weight in the graph. This also helps find terms that link two topics, and this is very useful in many information retrieval applications.

Edge weight>5% of maximum edge weight:

As we keep analyzing we get to view new patterns being formed in the graphs. This analysis resulted in the formation of new edges like quick sort and shortest path algorithm. But edge weight being the factor it was found that the edges formed were less related and resulting in the formation of many unrelated edges. These edges were formed when they occurred in one document coincidentally. Hence edges below 20% of maximum edge weight are not considered.

5.1.2 Conclusions from analysis:

The foremost conclusion that could be drawn from the graphs is that higher the edge weight the more semantically related the terms. The closest connections were formed when top 80% of the edge weights were taken, for example Internet and networks. As the analysis was done by decreasing the edge weight, different relationships were identified till a point when the edges indicated relationship between different topics. The analysis of edge weight around 20% gave edges with completely unrelated terms or related terms with less number of related documents.

5.2 Initial WORD-DISTANCE graph

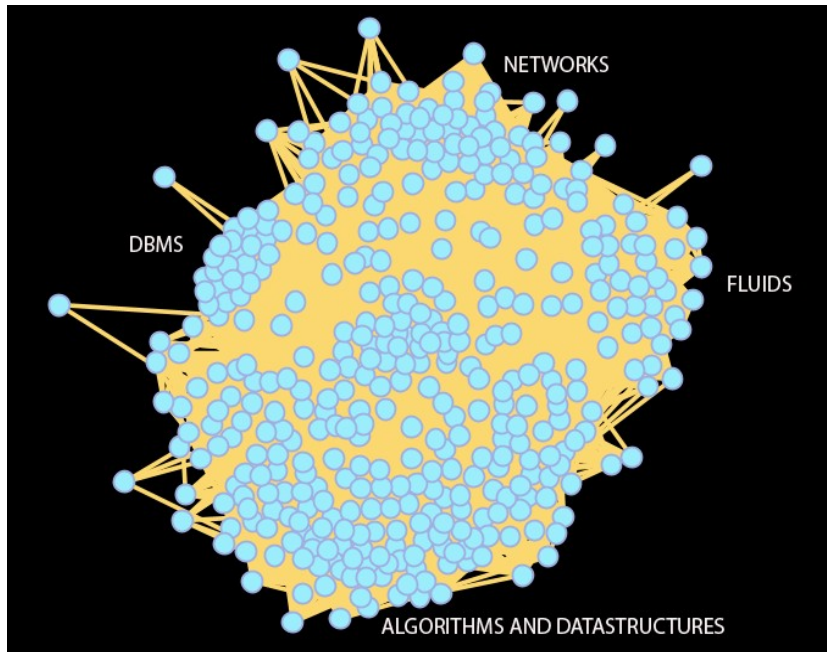


Fig.5.5 Word-distance graph

Fig 5.5 shows a sample word distance graph where nodes are terms in Engineering, and edges connect terms occurring in the same document. Each edge in word-distance tells how related two key words are. Edge weight of word-distance is the number of words between two keywords. More the value of edge weight more distance between

the keywords and hence less related. By visualizing this graph using the GEM graph visualization technique [1], we can observe that the terms belonging to the same topic are closer in the visualization. For example all the nodes related to algorithms like sorting algorithm, complexity are together, similarly networks database and fluids. This demonstrates that a simple word-distance graph can effectively capture semantic similarity among terms. Given this observation, we analyze this graph further for semantic information.

Analysis of edge weight:

In word-distance graph the edge weight is inversely proportional to the semantic relationship, ie the lesser the edge weight the closer the terms. On analysis of edge weight on word-distance graph we found that when edge weight is greater than 80%, relations like speed and radio, speed and Ethernet were present. We know that speed is not related to radio and Ethernet, hence these links are weak. Edge-weight less than 5% may have terms present in index which are close but not related, for example worst-case complexity and database. Hence maximum relations lie in the range of 5% to 80% of maximum edge-weight.

5.3 Hybrid graph:

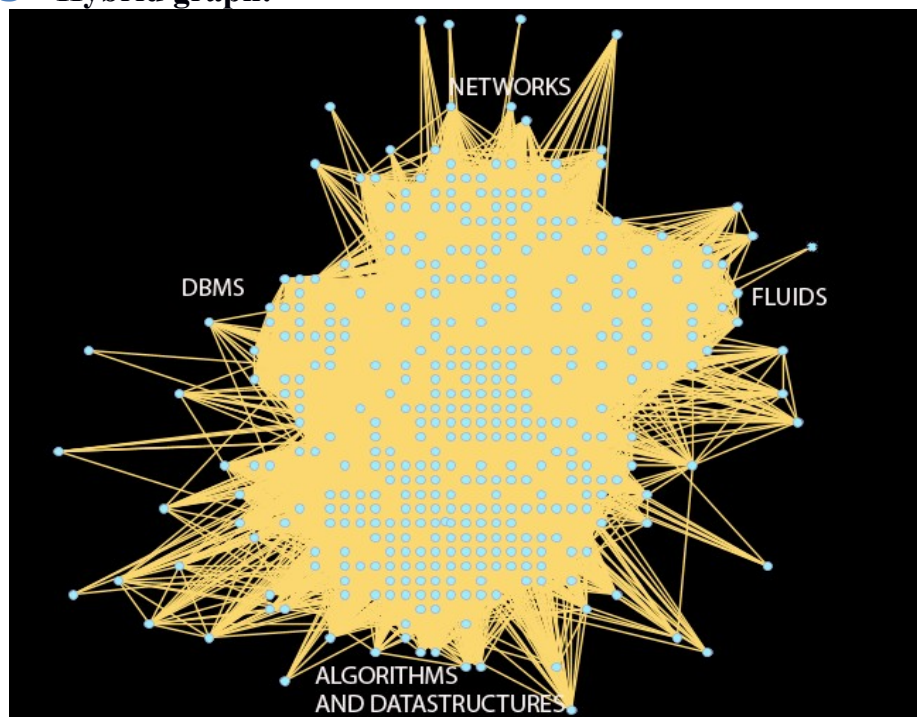


Fig.5.6. Hybrid graph

Co-occurrence graphs fail to capture the relationships between terms if their frequency of occurrence is low in multiple documents. Word distance based graphs may not capture the relationships when indexes are used for building the graph, since in indexes terms that occur close together need not be related. In hybrid graph (Fig.5.6) we combine the co-occurrence graph and word-distance graph by considering the co-occurrence of two terms over a fixed word distance. To determine the new edge metric, we combine the metrics generated over the two graphs as follows. Let $R(a,b)$ be the relationship between two terms a,b , $C(a,b)$ be the co-occurrence value of a,b and $W(a,b)$ the word-distance value of a,b . The combined relation $R_c(a,b)$ is given by $R_c(a,b) = k * C(a,b) / W(a,b)$, where k is the proportionality constant.

5.3.1 Analysis of Hybrid graph:



The following analysis explains the edges with maximum and minimum edge-weights.

Fig.5.7 edges with weight>90% of maximum

Edge-weight>90%of maximum edge-weight:

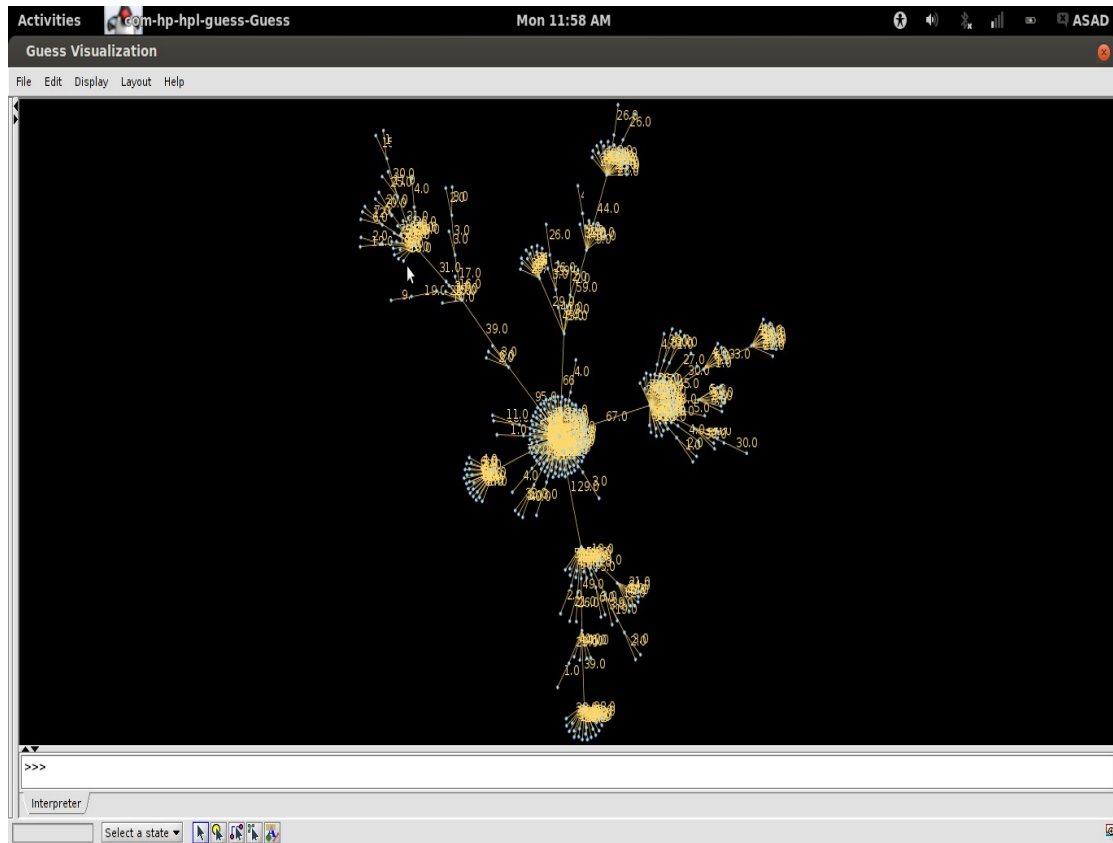
When edges with weight greater than 90 were analyzed (Fig.5.7) it was found that the edges displayed more related edges. Edges connecting array, sparse array, suffix array, variable length array were present. We see that these edges belong to the same topic array. There were also edges connecting algorithm to Dijkstra's algorithm, Kruskal's algorithm, greedy algorithm etc. hence we see that there are less insignificant edges compared to word distance in top 90%.

Edge-weight <10% of maximum edge-weight:

On analysing the bottom 10% edges we found that the relation represented by these edges are also related. Edges connecting heap to binary heap, AF-heap were displayed. There were also edges connecting algorithms to hill-climbing algorithm and edges linking candidate key foreign key, Armstrong axioms etc. hence unlike co-occurrence the lower edge weights also has significant relations.

On completion of the analysis it is observed that hybrid graph unlike word-distance and co-occurrence graph represent significant edges in all range of edge weight. It includes the important edges from co-occurrence and word distance graph.

5.4 Clustering the graph based on topic similarity



Clustering of a graph in general is partitioning the graph into smaller components each with its own specific properties. The partitioning is said to be good when the number of links/edges between two partitions is small. To obtain such partitioning in a graph we use heuristics and approximation algorithms. In case of clustering conceptual graphs each cluster becomes a topic and our goal is to determine which key term belongs to which topic. This structuring itself forms the primary relationship between topics and subtopics. The clustering mechanism can further be continued inside every cluster taking it to be a separate graph giving us multilevel clustering and relationships.

At the graph level every node seems to be connected to most other nodes among which a lot of these link are weak or unimportant or incorrect. To cluster terms based

on their relation to a particular topic, most of the weak links have to be removed to eliminate inter-topic relationships, to do so we analyzed the graph and employed the minimum spanning tree which was most effective in removing the weak edges. In addition the minimum spanning tree is fast and easy to create, provide a better topic subtopic understanding, provide the terms that may relate one topic to another, and a tree based structure enable faster search than a graph. We used the Kruskal's algorithm to generate the graph. The minimum spanning tree gives us topic- subtopic relationship. The figure depicts the minimum spanning tree formed.

5.4.1 Analysis of betweenness and degree:

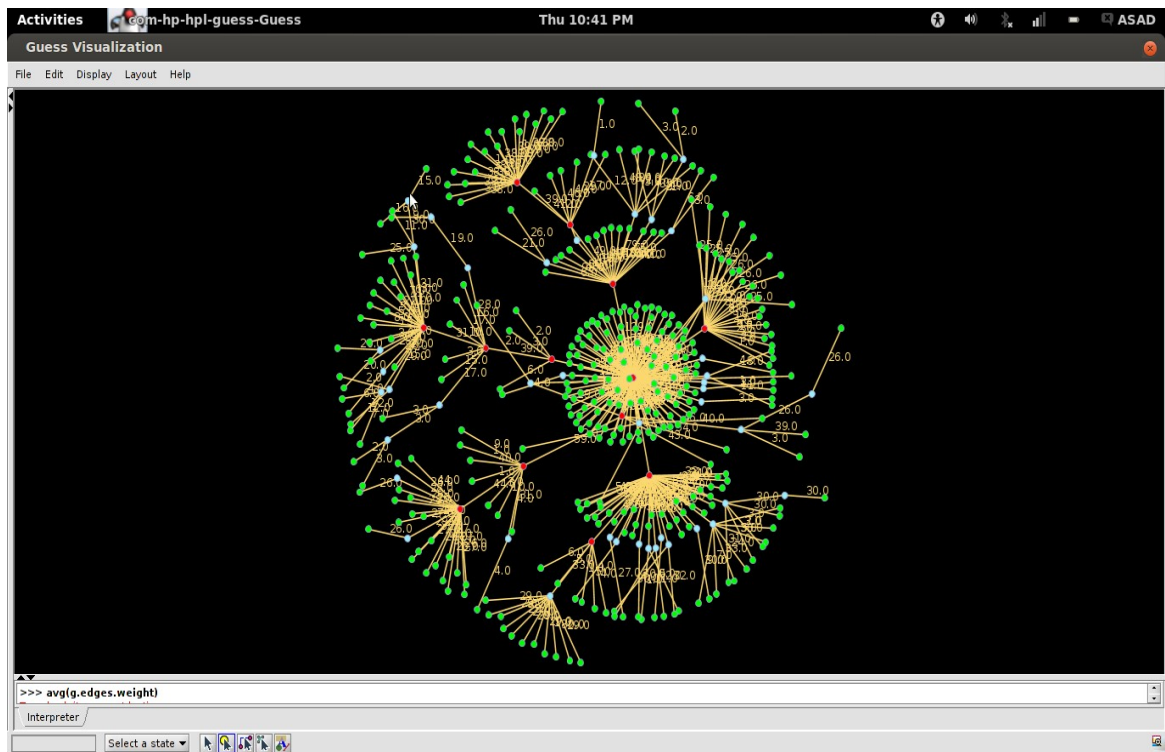


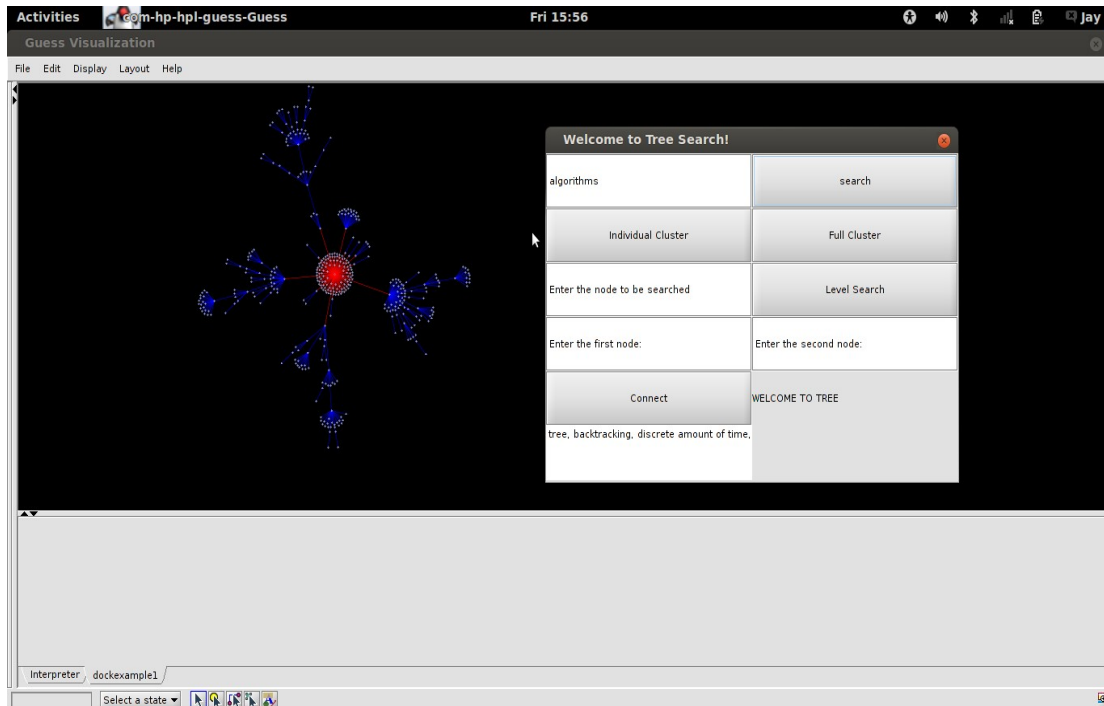
Fig 5.8 Analysis of degree and betweenness

Betweenness of a node gives the number of shortest path passing through a node. Greater betweenness for a node implies more significant is the node. Degree of a node is the number of nodes connected to the node.

In Fig.5.8 red nodes are the nodes with high betweenness and degree, green nodes are with less betweenness and blue nodes are nodes with high betweenness and less degree. By analyzing the degree and betweenness it was observed that nodes with degree and betweenness greater than 20% of the maximum were general topics representing clusters. Hence topic-subtopic relationships can be identified using these.

5.5 GUESS BASED TOOL

Fig.5.9 GUESS based tool



Analysis and visualization of graph become easy when there is a tool for supporting these functions. We developed an application(Fig.5.9) which is an extension of existing GUESS (open source tool for graph exploration), for this purpose. Several functions including clustering, topic wise individual clustering, level wise search, path between two clusters are integrated with this tool. The algorithms are designed using analysis of graph theoretical properties like degree and betweenness. These functions are explained in detail in next chapter. The platform used for development is JYTHON

CHAPTER-6
POTENTIAL APPLICATIONS OF OUR GRAPHS

6 APPLICATION OF THE CONCEPT GRAPHS

This section shows the applications of the graphs we created and the information they provide.

6.1 Query Suggestion

On searching a term, related terms of the corresponding node will be displayed

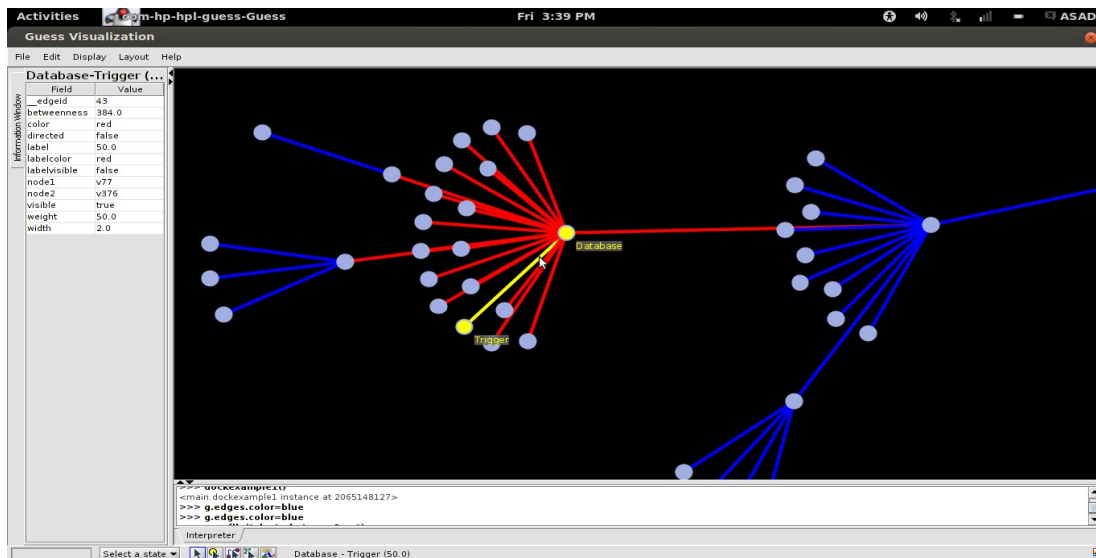


Fig.6.1 Query suggestion

The user will be given the option to search for any specific term. When he enters the term in the text box provided there will be a list of related terms given as output (Fig.6.1). These related terms are the nodes which are connected to the term given by the user through edges. For example the user gives a term data structures for search then terms like stack, pop and push are given as output. This application eases out the task of searching for the user. The user is provided with more choice to choose from.

6.2 Clustering

On searching a term, the cluster to which the term belongs to will be displayed

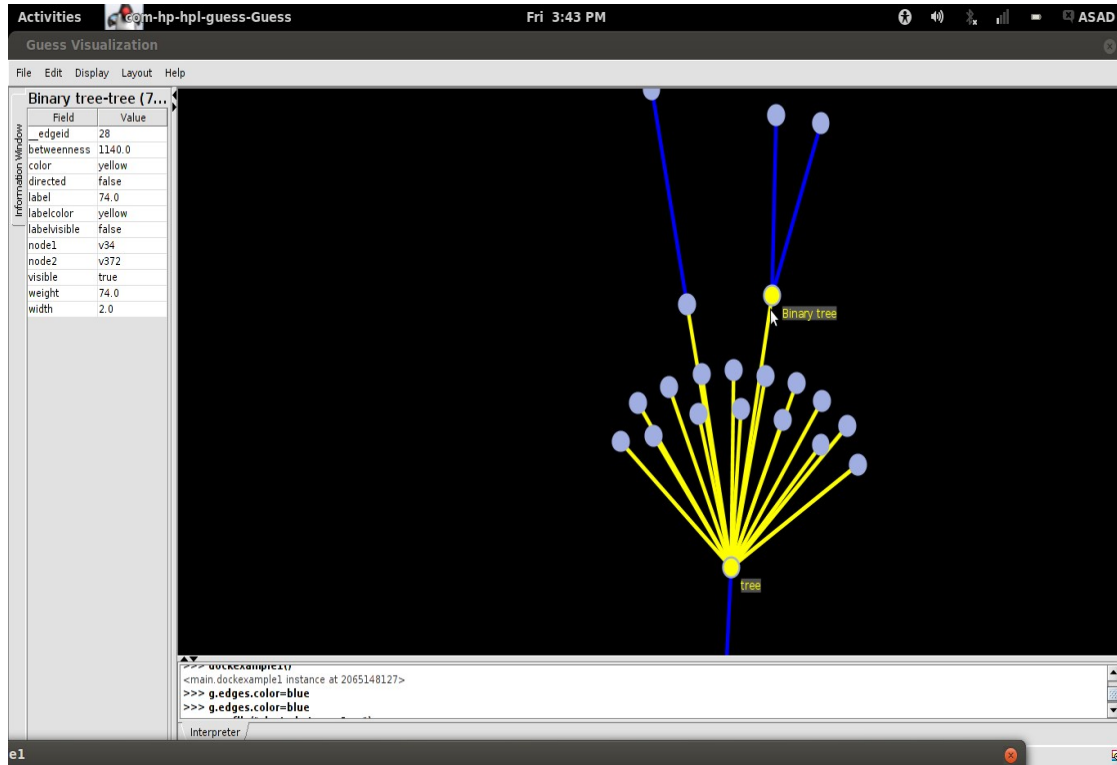


Fig.6.2 Identifying clusters

If the user wants to know to which topic does the term he is searching for belongs to, then he needs to enter the term in the text box named as cluster then the cluster name will be displayed(Fig.6.2). This is implemented in the following way: when the term is given as input it will point out towards its root node (traverse back to its parent node) and this process of traversing back is repeated until we reach the top most root node of the cluster and that root node will be displayed as the result which is indeed the cluster name.

6.3 Sub Clusters

The system identifies clusters, within clusters

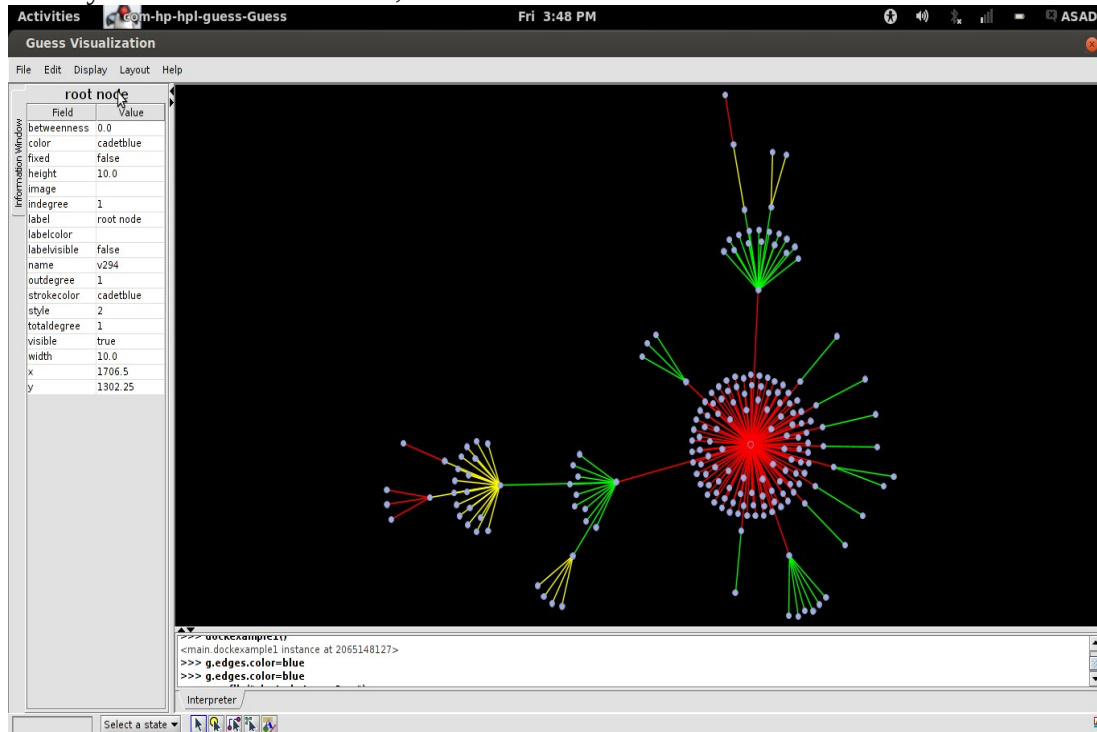


Fig.6.3 Identifying sub-clusters

This application will help the user to know what the different levels are and sub clusters present in a cluster. It provides a concept of hierarchy to the user. The user also gets to know about how a certain term is related to other terms and how important or generalized is that particular term which he is looking for. In this all the different levels present in a cluster are highlighted using different colors (Fig.6.3). The user can get to know to which level does the term he is searching for belongs to and how many levels are present in a particular cluster.

6.4 Relation between two nodes

Relation or path between nodes and clusters were identified

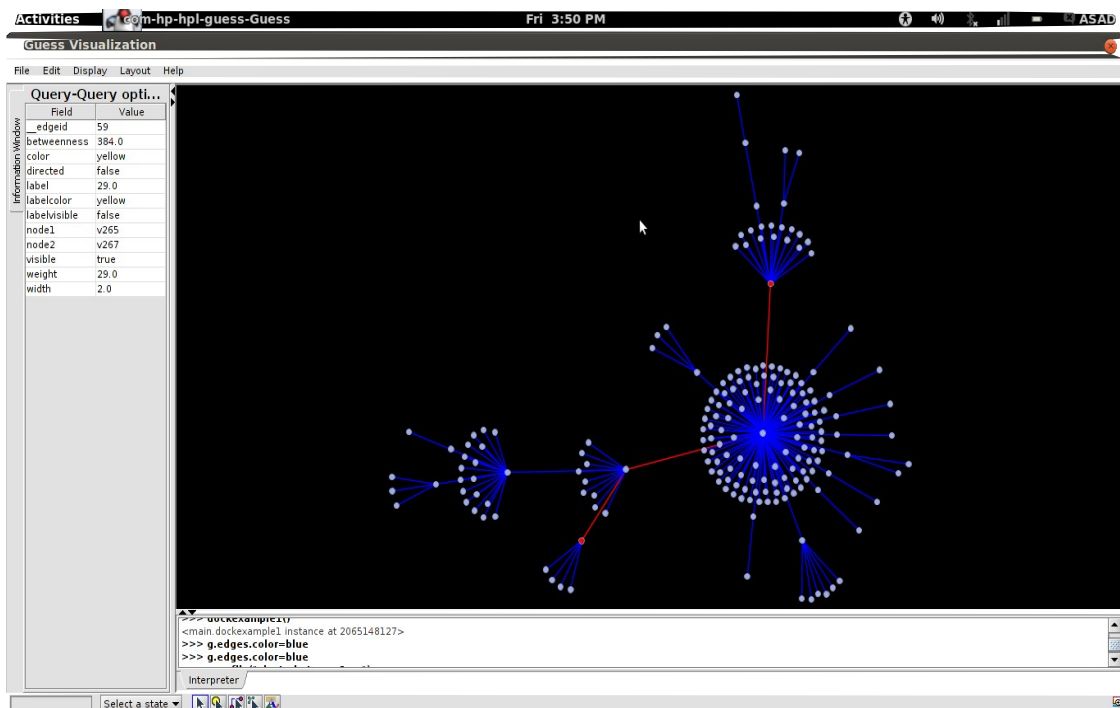


Fig.6.4 Identifying path between nodes

This application helps the user to know the path through which two nodes are connected to each other. The nodes can be present anywhere in the graph i.e.. they may be present in two different clusters or within the same cluster etc..

The path through which the two nodes are connected is highlighted using a red color line(Fig.6.4). The line passes through a sequence of many intermediate nodes because of this the user will get to know what are the nodes present in the path and how they are connecting the two terms.

CHAPTER-7
CONCLUSION

7 CONCLUSION

After going through the results of the experiment we have come to the conclusion that a perfect statistical graph with 100% accurate relationships is not possible but it is possible to use these graphs to help generate the ontology/semantic graphs, to do so we need to improve the relations the statistical graphs display and extract certain conceptual information that lay dormant in the graphs. Our method of generation of graphs seems to be much more efficient than the currently existing techniques like manual generation and automatic generation using predefined relationships. Since our method needs no predefined knowledge and can be used for any domain of any size given the right data set, it already becomes a step up than the other methods. Here we use some standard statistical techniques, so there is no need for manual intervention and also unlike manual generation the problem of “different people having different interpretation” is non-existent. Our approach works well for large data sets with high standards of accuracy. Our method of automatic generation of concept graphs finds applications in various domains such as a data structure for various internet applications, for query suggestion etc.

REFERENCES

- [1] Arne Frick, Andreas Ludwig, Heiko Mehldau, “Monitoring A Fast Adaptive Layout Algorithm for Undirected Graphs”, Universität Karlsruhe, Fakultät für Informatik, D-76128 Karlsruhe, Germany
- [2] David Hawking and Paul Thistlewaite, “Relevance Weighting Using Distance Between Term Occurrences”, The Australian National University, August 1996
- [3] Thomas M. J. Fruchterman And Edward M. Reingold, “Graph Drawing by Force-directed Placement”, University of Illinois at Urbana-Champaign, “SOFTWARE—PRACTICE AND EXPERIENCE”, VOL. 21, November 1991
- [4] Yukio Ohsawa, Nels E. Benson and Masahiko Yachida, “KeyGraph: Automatic Indexing by Co-occurrence Graphbased on Building Construction Metaphor”, Graduate School of Engineering Science Osaka University
- [5] Xiangfeng Luo, Kai Yan and Xue Chen,”Automatic Discovery of Semantic Relations Based on Association Rule”, JOURNAL OF SOFTWARE, VOL. 3, NO. 8, NOVEMBER 2008
- [6] Miguel Riesco, Marián D. Fondón, Darío Álvarez, ”DESIGNING DEGREES: GENERATING CONCEPT MAPS FOR THE DESCRIPTION OF RELATIONSHIPS BETWEEN SUBJECTS”, Concept Mapping: Connecting Educators Proc. of the Third Int. Conference on Concept Mapping A. J. Cañas, P. Reiska, M. Åhlberg & J. D. Novak, Eds. Tallinn, Estonia & Helsinki, Finland 2008
- [7] Rakesh Agrawal, Ramakrishnan Srikant, “Fast Algorithms for Mining Association Rules” Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pages 487-499, Santiago, Chile, September 1994.

APPENDIX

Software Details under the following head:	
Language with version	Python 2.7., JYTHON
Operating System with version	Ubuntu 12.04
Graph Visualization	GUESS –Graph Visualization
Development Environment	Python 2.7 IDLE

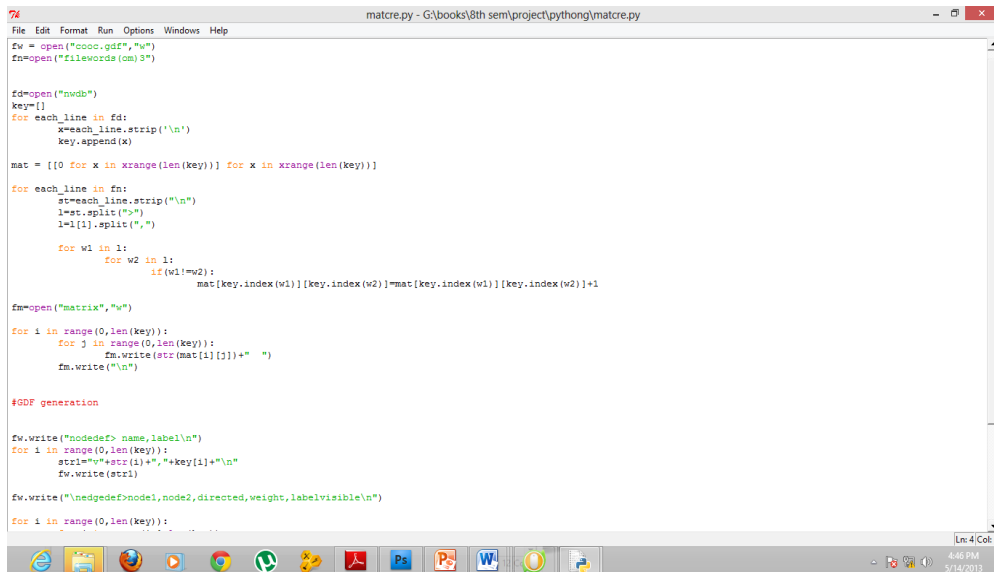
Installing GUESS

Prerequisites

You will need to:

1. Download and install python.
2. Download and unzip GUESS Graph exploration from GUESS official website.
3. Change and give executable permission of guess.sh file
4. Run the guess.sh file

Screenshots of the source code in the environment



```
matre.py - G:\books\8th sem\project\pythong\matre.py
File Edit Format Run Options Windows Help

fw = open("cocc.gdf", "w")
fm = open("filewords.ccm3")

fd = open("nwdb")
key = []
for each_line in fd:
    st = each_line.strip('\n')
    key.append(st)

mat = [[0 for x in xrange(len(key))] for x in xrange(len(key))]

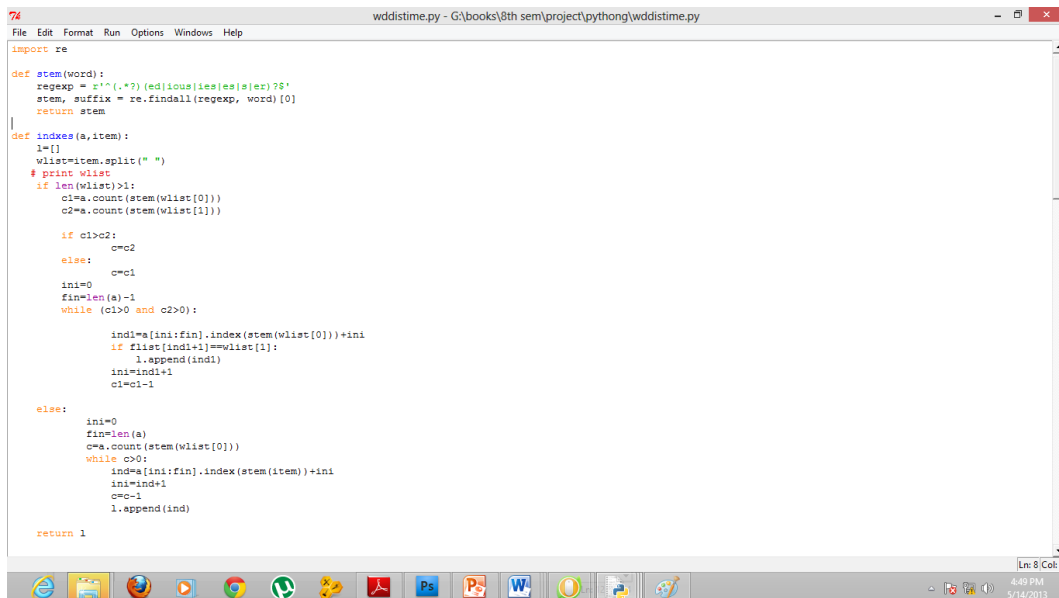
for each_line in fm:
    st = each_line.strip("\n")
    l = st.split(">")
    l = l[1].split(",")

    for w1 in l:
        for w2 in l:
            if w1 != w2:
                mat[key.index(w1)][key.index(w2)] = mat[key.index(w1)][key.index(w2)] + 1

fm = open("matrix", "w")
for i in range(0, len(key)):
    for j in range(0, len(key)):
        fm.write(str(mat[i][j]) + " ")
    fm.write("\n")

#GDF generation
fw.write("nodedef> name,label\n")
for i in range(0, len(key)):
    st1 = "v" + str(i) + ", " + key[i] + "\n"
    fw.write(st1)

fw.write("\nedgedef> node1,node2,directed,weight,labelvisible\n")
for i in range(0, len(key)):
```



```
wddistime.py - G:\books\8th sem\project\pythong\wddistime.py
File Edit Format Run Options Windows Help

import re

def stem(word):
    regexp = r'^(.*?) (ed|ious|ees|es|er)?$'
    stem, suffix = re.findall(regexp, word)[0]
    return stem

def index(a, item):
    l = []
    wlist = item.split(" ")
    # print wlist
    if len(wlist) > 1:
        c1 = a.count(stem(wlist[0]))
        c2 = a.count(stem(wlist[1]))

        if c1 > c2:
            c = c2
        else:
            c = c1
        ini = 0
        fin = len(a) - 1
        while (c1 > 0 and c2 > 0):
            ind1 = a[ini:fin].index(stem(wlist[0])) + ini
            if finst[ind1+1] == wlist[1]:
                l.append(ind1)
            ini = ind1 + 1
            c1 = c1 - 1

        else:
            ini = 0
            fin = len(a)
            c = a.count(stem(wlist[0]))
            while c > 0:
                ind = a[ini:fin].index(stem(item)) + ini
                ini = ind + 1
                c = c - 1
                l.append(ind)

    return l
```

Screenshots of the output

