CS5234 Assignment 4
Data Mining with Hadoop and Gephi

1) Extracting the Email Network

1.1) Implementation of the solution

Enron, largest email dataset is formatted as Hadoop Sequence file. A Sequence file is a collection of typed key/value pairs. In the provided dataset the key and value types are Text and BytesWritable respectively. The key keeps the name of mail file in the original dataset and the value holds the content of the single email message. The steps for solution is as follows:

Mapper Class

- The mapper class is used to read the email addresses in the dataset. Every email has a single sender, that is, the email address in the From: field, but may have one or more recipients. Recipient email addresses can be found in the To:, Cc:, Bcc: and also date from the Date: field.

- The stripCommand function is used to separate out the data from all the fields such as From:, To:, Cc:, Bcc: and Date: in the dataset.

- Then, we separate the emails that belong to only the enron dataset in the procFrom function after getting the emails separately.

- The map function of the Mapper formally scams each line of the dataset and calls the stripCommand and the procFrom functions to get all the data from the respective fields. Since the fields To:, Cc: and Bcc: means the receiver of the mail, they are listed in the same list known as the 'recipients' list. They are added to the list using the command 'recipients.add()'.

- The map function is also used to concatenate the timestamp or date to the recipient address and set it as the value.

- Summarizing the actions of mapper class, the mapper class iterate over the large set of records to obtain the values that are present in the fields 'From', 'To', 'Cc', 'Bcc' and 'Date'. Then, all the receivers' mail address is added to the recipient list and the timestamp is concatenated with the recipients. The 'from' mail address acts as the key and the recipients mail address with the timestamp will be the value.

Reducer Class

- Generally, shuffle and sort will generate intermediate results and the reducer will aggregate on the intermediate result to generate the final output. So, after shuffle and sort, the sender and the receiver address with the timestamp is brought to the reducer.
- An instance variable 'value' is used to stores the state between iterations of the reduce task.

- The reduce method iterates over the values and get the correct value that is required and this is written into the output file using the command context.write().
- As passed by the mapper, the ‘from’ mail address acts as the key and the receiver’s address with the timestamp acts as the value, and this is passed in the write function such as, context.write(key,value).

Finally, the main() function calls the Mapper and Reducer class to execute all the methods and thus provides a solution to the problem.

1.2) My Contribution to solving this section

The MailReader.java was extended further as follows:

- In the mapper class, inside map function, strip command and procFrom functions are used obtain the values of the fields ‘From:’, ‘To:’, ‘Cc:’, ‘Bcc:’ and ‘Date:’ using the command ‘procFrom(stripCommand(“From:”)))’. Similarly, we get the values for to, cc, bcc and date.
- After getting these values, it is checked whether it is null. If it is not null, then the values of to, cc and bcc are added to the recipients list.
- Then the datatype of values were changed to Text in the reducer to aggregate the intermediate values and get the final output.

2) Extracting The Social Network

2.1) Implementation of the solution

The values generated by the first question are used for this section.

Mapper Class

- The mapper class is used to read the email addresses in the dataset. Every email has a single sender, that is, the email address in the From: field, but may have one or more recipients. Recipient email addresses can be found in the To:, Cc:, Bcc: and also date from the Date: field.

- The stripCommand function is used to separate out the data from all the fields such as From:, To:, Cc:, Bcc: and Date: in the dataset.

- Then, we separate the emails that belong to only the enron dataset in the procFrom function after getting the emails separately.

- The map function of the Mapper formally scams each line of the dataset and calls the stripCommand and the procFrom functions to get all the data from the respective fields. Since the fields To:, Cc: and Bcc: means the receiver of the mail, they are listed in the same list known as the ‘recipients’ list. They are added to the list using the command ‘recipients.add()’.

- The map function is also used to concatenate the timestamp or date to the recipient address and set it as the value.

- Summarizing the actions of mapper class, the mapper class iterate over the large set of records to obtain the values that are present in the fields ‘From’, ‘To’, ‘Cc’, ‘Bcc’ and ‘Date’. Then, all the receivers’ mail address is added to the recipient list and the timestamp is concatenated with the recipients. The ‘from’ mail address acts as the key and the recipients mail address with the timestamp will be the value.

Reducer Class

Step 1: The duplicate entries are deleted that match the ‘to’ and ‘from’ entries in the values generated by the mapper class, that is, the table R. These duplicate values are deleted using the hash set ‘uniques’. This operation is done as follows:

- The values generated by the mapper class, that is the recipient email address is checked whether it is present in the hash set, if it is not present in that, then it is added to the hash set.
- So, this set contains unique values of a combination that includes the ‘from’, ‘to’ and ‘timestamp’ together. The set of values generated is stored in a new table ‘M’.

Step 2: The weight of an email is calculated between x and y where ‘x’ is the sender and ‘y’ is the receiver as follows:

$$\text{weight}(x; y) = \frac{\#\ \text{arc}(x; y)\ \text{in M}}{\text{total}\ \#\ \text{of arcs in M}}$$

This formula can be further simplified as follows:

$$\text{weight}(x; y) = \frac{\text{total}\ \#\ \text{of mails between x and y}}{\text{total}\ \#\ \text{of mails in M}}$$

This operation is done as follows:

- The count of mails between x and y where ‘x’ is the ‘from’ address and ‘y’ is the recipient address is found by first concatenating the x and y, so that the distinct pair of x and y is identified. The pair of all x and y is added to an array list
- The count of the mails between x and y is calculated using Collections.frequency(list,a), where ‘a’ involves a particular mail between x and y.
- The total count of distinct values is calculated by incrementing the variable ‘totcount’. This total count is the total number of mails in M
- The weight of an email is calculated by dividing the count of email between x and y by the total count of mails in M.
- As passed by the mapper, the ‘from’ mail address acts as the key and the receiver’s address with the weight acts as the value, and this is passed in the write function such as, context.write(key,value).

2.2) My Contribution to solving this section

After deleting the duplicates and storing it in a separate list, the MailReader.java was extended further as follows:

- The key and the value is concatenated for finding their unique pair, where key is the ‘from’ address and value is the ‘to’ address. The key-value pairs are added to an array list ‘list’.
- After adding the key-value pairs into the list, the number of times a particular ‘from-to’ email address pair has occurred in the list of ‘key-value’ pairs is calculated using the command Collections.frequency(list,a), where list is the set of ‘key-value’ pairs and ‘a’ is a particular key-value pair. This gives the total number of mails between a particular ‘from’ address ‘x’ and a particular ‘to’ address ‘y’.
- The weight of a particular email is calculated by dividing the count of email between x and y by the total count of mails.

3) Social Network Analysis

3.1) Implementation of the solution

Binomial Vs Poisson Distribution of Random graph (n and m as Enron graph)

The probability of an edge being formed between two nodes in the enron graph is p :

$$m / (nC_2) \text{ or } m / (^{n}{}_{2}).$$

Let X be any node among the n total number of nodes, and let it have k as indegree among the other (n-1) nodes. So, the binomial degree distribution is given by $P(k)={}^{(n-1)}C_k\ p^k\ q^{(n-1-k)}$. Similarly, the same formula is used for k as outdegree among the other (n-1) nodes. And the poisson distribution is given by $P(k)= e^{-lambda}\ (lambda)^k/k!$, where lambda= np. Similarly, the same formula is used for k as outdegree.

Indegree

Mapper Class

- The mapper class is used to read the email addresses in the dataset. Every email has a single sender, that is, the email address in the From: field, but may have one or more recipients. Recipient email addresses can be found in the To:, Cc:, Bcc: and also date from the Date: field.
- The stripCommand function is used to separate out the data from all the fields such as From:, To:, Cc:, Bcc: and Date: in the dataset.
- Then, we separate the emails that belong to only the enron dataset in the procFrom function after getting the emails separately.
- The map function of the Mapper formally scans each line of the dataset and calls the stripCommand and the procFrom functions to get all the data from the respective fields. Since the fields To:, Cc: and Bcc: means the receiver of the mail, they are listed in the same list known as the ‘recipients’ list. They are added to the list using the command ‘recipients.add()’.

- The map function is also used to concatenate the timestamp or date to the 'from' address and set it as the value.

- Summarizing the actions of mapper class, the mapper class iterate over the large set of records to obtain the values that are present in the fields 'From', 'To', 'Cc', 'Bcc' and 'Date'. Then, all the receivers' mail address is added to the recipient list, and this acts as the key. The 'from' mail address and the timestamp is concatenated to be set as the value.

<u>Reducer Class</u>

- The values generated by the mapper class is checked whether it is present in the hash set, if it is not present in that, then it is added to the hash set.

- So, this set contains unique values of a combination that includes the 'from', 'to' and 'timestamp' together. The set of values generated is stored in a new table 'M'. This set of new values is stored in a hashset, namely, uniques.

- To find the binomial and the poisson distribution of a random graph, we need the values of n, m and k (as indegree as well as outdegree).

- The total number of unique connections is present in the 'uniques' hashset, the count of these unique connections is the 'm' total number of edges or arcs in the enron dataset

- The number of unique 'to' mail address is calculated to obtain the 'n' value for indegree, this is also calculated using a hashset, that is, uniqueList. First, all the 'to' mail addresses are added to an ArrayList, namely 'toList'.

- Then, the unique 'to' addresses are put in a set using the command uniqueList.addAll(toList), and the size of the set, calculates the value of 'n' for indegree

- Since we have obtained the values of 'n' and 'm', the value of 'k' or the indegree is found using the command Collections.frequency(toList, a), where 'a' refers each of the unique 'to' mail address.

- Using the values of n, m and k, the binomial and poisson distribution is found and is plot in a graph with the x-axis as the indegree and y-axis as the binomial and poisson deistribution obtained from this indegree.

<u>Outdegree</u>

<u>Mapper Class</u>

- The mapper class is used to read the email addresses in the dataset. Every email has a single sender, that is, the email address in the From: field, but may have one or more recipients. Recipient email addresses can be found in the To:, Cc:, Bcc: and also date from the Date: field.

- The stripCommand function is used to separate out the data from all the fields such as From:, To:, Cc:, Bcc: and Date: in the dataset.

- Then, we separate the emails that belong to only the enron dataset in the procFrom function after getting the emails separately.

- The map function of the Mapper formally scams each line of the dataset and calls the stripCommand and the procFrom functions to get all the data from the respective fields. Since the fields To:, Cc: and Bcc: means the receiver of the mail, they are listed in the same list known as the 'recipients' list. They are added to the list using the command 'recipients.add()'.

- The map function is also used to concatenate the timestamp or date to the recipient address and set it as the value.

- Summarizing the actions of mapper class, the mapper class iterate over the large set of records to obtain the values that are present in the fields 'From', 'To', 'Cc', 'Bcc' and 'Date'. Then, all the receivers' mail address is added to the recipient list and the timestamp is concatenated with the recipients. The 'from' mail address acts as the key and the recipients mail address with the timestamp will be the value.

Reducer Class

- The values generated by the mapper class is checked whether it is present in the hash set, if it is not present in that, then it is added to the hash set.
- So, this set contains unique values of a combination that includes the 'from', 'to' and 'timestamp' together. The set of values generated is stored in a new table 'M'. This set of new values is stored in a hashset, namely, uniques.
- To find the binomial and the poisson distribution of a random graph, we need the values of n, m and k (as indegree as well as outdegree).
- The total number of unique connections is present in the 'uniques' hashset, the count of these unique connections is the 'm' total number of edges or arcs in the enron dataset
- The number of unique 'from' mail address is calculated to obtain the 'n' value for outdegree, this is also calculated using a hashset, that is, uniqueList. First, all the 'from' mail addresses are added to an ArrayList, namely 'fromList'.
- Then, the unique 'from' addresses are put in a set using the command uniqueList.addAll(fromList), and the size of the set, calculates the value of 'n' for outdegree
- Since we have obtained the values of 'n' and 'm', the value of 'k' or the outdegree is found using the command Collections.frequency(fromList, a), where 'a' refers each of the unique 'from' mail address.

Using the values of n, m and k, the binomial and poisson distribution is found and is plot in a graph with the x-axis as the outdegree and y-axis as the binomial and poisson deistribution obtained from this outdegree.

Degree Distribution of Actual Enron Graph

Indegree

Mapper Class-1

- The mapper class is used to read the email addresses in the dataset. Every email has a single sender, that is, the email address in the From: field, but may have one or more recipients. Recipient email addresses can be found in the To:, Cc:, Bcc: and also date from the Date: field.
- The stripCommand function is used to separate out the data from all the fields such as From:, To:, Cc:, Bcc: and Date: in the dataset.
- Then, we separate the emails that belong to only the enron dataset in the procFrom function after getting the emails separately.
- The map function of the Mapper formally scans each line of the dataset and calls the stripCommand and the procFrom functions to get all the data from the respective fields. Since the fields To:, Cc: and Bcc: means the receiver of the mail, they are listed in the same list known as the ‘recipients’ list. They are added to the list using the command ‘recipients.add()’.
- The map function is also used to concatenate the timestamp or date to the ‘from’ address and set it as the value.
- Summarizing the actions of mapper class, the mapper class iterate over the large set of records to obtain the values that are present in the fields ‘From’, ‘To’, ‘Cc’, ‘Bcc’ and ‘Date’. Then, all the receivers’ mail address is added to the recipient list, and this acts as the key. The ‘from’ mail address and the timestamp is concatenated to be set as the value.

Reducer Class-1

- The values generated by the mapper class is checked whether it is present in the hash set, if it is not present in that, then it is added to the hash set.
- So, this set contains unique values of a combination that includes the ‘from’, ‘to’ and ‘timestamp’ together. The set of values generated is stored in a new table ‘M’. This set of new values is stored in a hashset, namely, uniques.

- The total number of unique connections is present in the ‘uniques’ hashset, the count of these unique connections is the ‘m’ total number of edges or arcs in the enron dataset

- The number of unique ‘to’ mail address is calculated to obtain the ‘n’ value for indegree, this is also calculated using a hashset, that is, uniqueList. First, all the ‘to’ mail addresses are added to an ArrayList, namely ‘toList’.

- Then, the unique ‘to’ addresses are put in a set using the command uniqueList.addAll(toList), and the size of the set, calculates the value of ‘n’ for indegree

- Since we have obtained the values of ‘n’ and ‘m’, the value of ‘k’ or the indegree is found using the command Collections.frequency(toList, a), where ‘a’ refers each of the unique ‘to’ mail address.

- After getting the indegree, the proportion for the degree distribution is found by dividing the indegree by the total number of nodes ‘n’ for indegree. This is done by using the command “double p=((double)indegree/n);”

- Using this the degree distribution plot is drawn by taking the x-axis as the indegree and the y-axis as the proportion obtained.

<u>Mapper Class-2</u>

- After calculating the indegree, it is given as the input to this mapper class(second mapper class).

- The mapper class thus has the value as the ‘to’ mail address and the indegree. This value is split into two halves to separate the indegree. This is done using the command: String str[]=value.toString().split("\t");

- Thus, this mapper class has indegree as its key and has IntWritable count which is initialised to 1.

<u>Reducer Class-2</u>

- The mapper class gives indegree with an IntWritable count initialised to 1.

- Since indegree is given as the key, for each unique value of the key, a counter ‘sum’ is set to keep track of the number of occurrence of each indegree or key. This is done using the command ‘sum+=val.get();’, where ‘val’ is the value of the IntWritable count (i.e, 1)

- The value of sum in then set as the ‘result’, and then the keys and values are written into the output file using the command ‘context.write(key, result);', where 'key' is the indegree and 'value' is the number of occurences of the indegree.

Using the values of n, k and the number of occurences of each k, the proportion for degree distribution is obtained by dividing the number of occurences of each indegree by the total number of nodes (n). Then a graph is plot with the indegree in the x-axis and the proportion in the y-axis to obtain the degree distribution for the actual enron graph.

Outdegree

Mapper Class-1

- The mapper class is used to read the email addresses in the dataset. Every email has a single sender, that is, the email address in the From: field, but may have one or more recipients. Recipient email addresses can be found in the To:, Cc:, Bcc: and also date from the Date: field.

- The stripCommand function is used to separate out the data from all the fields such as From:, To:, Cc:, Bcc: and Date: in the dataset.

- Then, we separate the emails that belong to only the enron dataset in the procFrom function after getting the emails separately.

- The map function of the Mapper formally scams each line of the dataset and calls the stripCommand and the procFrom functions to get all the data from the respective fields. Since the fields To:, Cc: and Bcc: means the receiver of the mail, they are listed in the same list known as the 'recipients' list. They are added to the list using the command 'recipients.add()'.

- The map function is also used to concatenate the timestamp or date to the recipient address and set it as the value.

- Summarizing the actions of mapper class, the mapper class iterate over the large set of records to obtain the values that are present in the fields 'From', 'To', 'Cc', 'Bcc' and 'Date'.Then, all the receivers' mail address is added to the recipient list and the timestamp is concatenated with the recipients. The 'from' mail address acts as the key and the recipients mail address with the timestamp will be the value.

Reducer Class-1

- The values generated by the mapper class is checked whether it is present in the hash set, if it is not present in that, then it is added to the hash set.

- So, this set contains unique values of a combination that includes the 'from', 'to' and 'timestamp' together. The set of values generated is stored in a new table ‘M’. This set of new values is stored in a hashset, namely, uniques.

- The total number of unique connections is present in the ‘uniques’ hashset, the count of these unique connections is the ‘m’ total number of edges or arcs in the enron dataset

- The number of unique ‘from’ mail address is calculated to obtain the ‘n’ value for outdegree, this is also calculated using a hashset, that is, uniqueList. First, all the ‘from’ mail addresses are added to an ArrayList, namely ‘fromList’.

- Then, the unique ‘from’ addresses are put in a set using the command uniqueList.addAll(fromList), and the size of the set, calculates the value of ‘n’ for outdegree

- Since we have obtained the values of ‘n’ and ‘m’, the value of ‘k’ or the outdegree is found using the command Collections.frequency(fromList, a), where ‘a’ refers each of the unique ‘from’ mail address.

- Using the values of n, m and k, the binomial and poisson distribution is found and is plot in a graph with the x-axis as the outdegree and y-axis as the binomial and poisson distribution obtained from this outdegree.

- After getting the outdegree, the proportion for the degree distribution is found by dividing the outdegree by the total number of nodes ‘n’ for outdegree. This is done by using the command “double p=((double)indegree/n);”

- Using this, the degree distribution plot is drawn by taking the x-axis as the indegree and the y-axis as the proportion obtained.

Mapper Class-2

- After calculating the outdegree, it is given as the input to this mapper class(second mapper class).

- The mapper class thus has the value as the ‘from’ mail address and the outdegree. This value is split into two halves to separate the outdegree. This is done using the command: String str[]=value.toString().split("\t");

- Thus, this mapper class has outdegree as its key and has IntWritable count which is initialised to 1.

Reducer Class-2

- The mapper class gives outdegree with an IntWritable count initialised to 1.

- Since outdegree is given as the key, for each unique value of the key, a counter 'sum' is set to keep track of the number of occurrence of each outdegree or key. This is done using the command 'sum+=val.get();', where 'val' is the value of the IntWritable count (i.e, 1)

- The value of sum in then set as the 'result', and then the keys and values are written into the output file using the command 'context.write(key, result);', where 'key' is the outdegree and 'value' is the number of occurences of the outdegree.

Using the values of n, k and the number of occurences of each k, the proportion for degree distribution is obtained by dividing the number of occurences of each outdegree by the total number of nodes (n). Then a graph is plot with the outdegree in the x-axis and the proportion in the y-axis to obtain the degree distribution for the actual enron graph.

3.2) My Contribution to solving this section

Degree Distribution of random graph with same number of nodes (n) and edges (m) as Enron graph

Indegree

- The number of unique 'to' mail address is calculated to obtain the 'n' value for indegree, this is calculated using a hashset, that is, uniqueList. First, all the 'to' mail addresses are added to an ArrayList, namely 'toList'.

- Then, the unique 'to' addresses are put in a set using the command uniqueList.addAll(toList). The addAll() function of the hashset checks each value of the array list, whether it has already entered the value or the value encountered is distinct from the values entered before. If it is a distinct value, then it is taken as a new value into the hashset or it is discarded. So, there won't be repetition of email ids. Thus, the size of the set, that is, the uniqueList hashset calculates the value of 'n', the total numbe of nodes for indegree. This is found by iterating the integer 'n' for the entire length of uniqueList hashset.

- The value of 'k' or the indegree is found using the command Collections.frequency(toList, a), where 'a' refers each of the unique 'to' mail address.

- After calculating the indegree, it is converted to a string and is concatenated with the 'to' mail address. This concatenated string is used for checking whether there are duplicates, that is, the ‘to’ mail address and the indegree pair should not be repeated in the output. So, this is checked for duplicates and is written into the output file.

- For writing into the output file, the key is set as the ‘to’ mail address and the value is the indegree. This is written into the output file using the command ‘context.write(key1,value1)’.

Outdegree

- The number of unique ‘from’ mail address is calculated to obtain the ‘n’ value for outdegree, this is calculated using a hashset, that is, uniqueList. First, all the ‘from’ mail addresses are added to an ArrayList, namely ‘fromList’.

- Then, the unique ‘from’ addresses are put in a set using the command uniqueList.addAll(fromList). The addAll() function of the hashset checks each value of the array list, whether it has already entered the value or the value encountered is distinct from the values entered before. If it is a distinct value, then it is taken as a new value into the hashset or it is discarded. So, there won’t be repetition of email ids. Hence, the size of this set, that is, the uniqueList hashset calculates the value of ‘n’, the total number of nodes for outdegree. This is found by iterating the integer ‘n’ for the entire length of uniqueList hashset.

- The value of ‘k’ or the outdegree is found using the command Collections.frequency(fromList, a), where ‘a’ refers each of the unique ‘from’ mail address.

- After calculating the outdegree, it is converted to a string and is concatenated with the ‘from’ mail address. This concatenated string is used for checking whether there are duplicates, that is, the ‘from’ mail address and the outdegree pair should not be repeated in the output. So, this is checked for duplicates and is written into the output file.

- For writing into the output file, the key is set as the ‘from’ mail address and the value is the outdegree. This is written into the output file using the command ‘context.write(key1,value1)’.

Degree Distribution of Actual Enron Graph

Indegree

- The number of unique ‘to’ mail address is calculated to obtain the ‘n’ value for indegree, this is calculated using a hashset, that is, uniqueList. First, all the ‘to’ mail addresses are added to an ArrayList, namely ‘toList’.

- Then, the unique ‘to’ addresses are put in a set using the command uniqueList.addAll(toList). The addAll() function of the hashset checks each value of the array list, whether it has already entered the value or the value encountered is distinct from the values entered before. If it is a distinct value, then it is taken as a new value into the hashset or it is discarded. So, there won’t be repetition of email ids. Thus, the size of the set, that is, the uniqueList hashset calculates the value of ‘n’, the total numbe of nodes for indegree. This is found by iterating the integer ‘n’ for the entire length of uniqueList hashset.

- The value of ‘k’ or the indegree is found using the command Collections.frequency(toList, a), where ‘a’ refers each of the unique ‘to’ mail address.

- After calculating the indegree, it is given as the input to the next mapper class. This mapper class has indegree as its key and has IntWritable count to calculate the number of occurences of that particular indegree as the value.

- This value is initialised to ‘1’ in the second mapper class and in the reducer class, the outdegree is taken as the key. For particular outdegree, the count is incremented for each occurrence of that outdegree.

- This outdegree and the number of occurrence is written into the output file using the command ‘context.write(key,result)’, where ‘key’ is the unique outdegree values and ‘result’ is the number of occurrence of each outdegree value.

<u>Outdegree</u>

- The number of unique ‘from’ mail address is calculated to obtain the ‘n’ value for outdegree, this is calculated using a hashset, that is, uniqueList. First, all the ‘from’ mail addresses are added to an ArrayList, namely ‘fromList’.

- Then, the unique ‘from’ addresses are put in a set using the command uniqueList.addAll(fromList). The addAll() function of the hashset checks each value of the array list, whether it has already entered the value or the value encountered is distinct from the values entered before. If it is a distinct value, then it is taken as a new value into the hashset or it is discarded. So, there won’t be repetition of email ids. Hence, the size of this set, that is, the uniqueList hashset calculates the value of ‘n’, the total number of nodes for outdegree. This is found by iterating the integer ‘n’ for the entire length of uniqueList hashset.

- The value of ‘k’ or the outdegree is found using the command Collections.frequency(fromList, a), where ‘a’ refers each of the unique ‘from’ mail address.

- After getting the outdegree, the proportion for the degree distribution is found by dividing the outdegree by the total number of nodes ‘n’ for outdegree. This is done by using the command “double p=((double)indegree/n);”

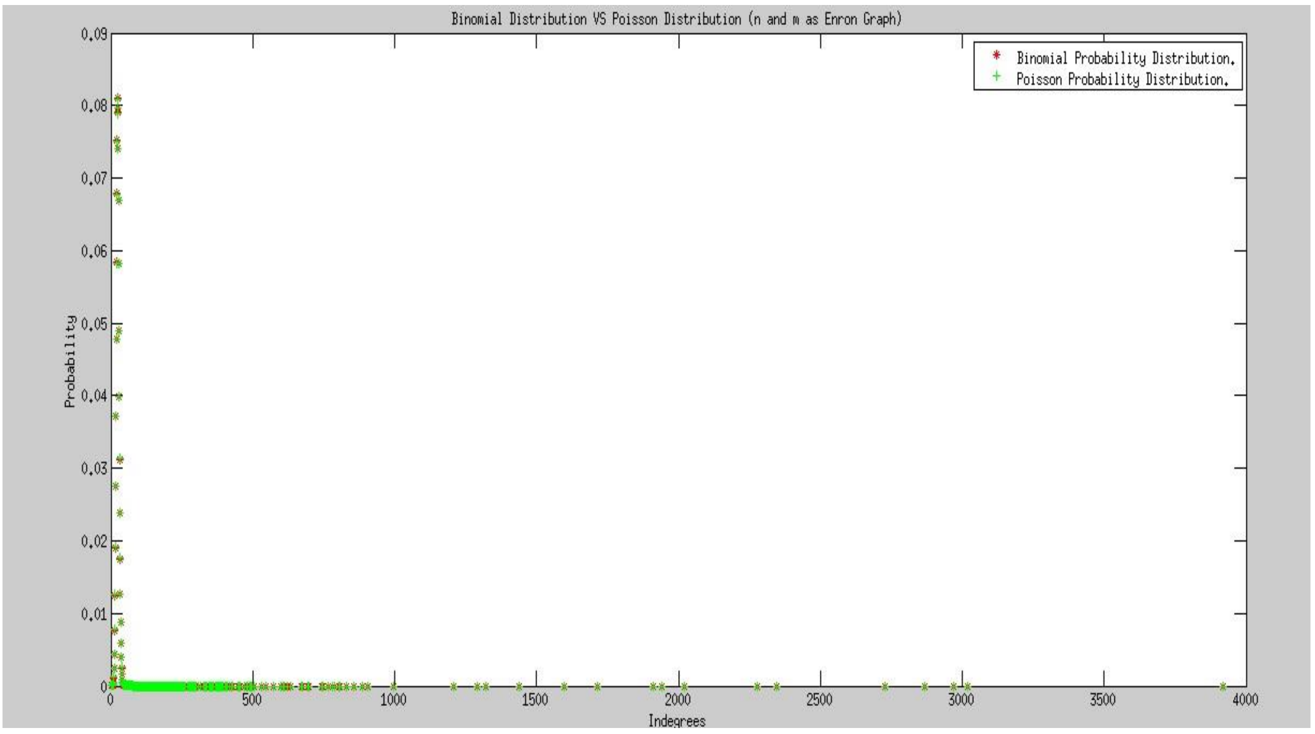
- After calculating the outdegree, it is converted to a string and is concatenated with the ‘from’ mail address. This concatenated string is used for checking whether there are duplicates, that is, the ‘from’ mail address and the outdegree pair should not be repeated in the output. So, this is checked for duplicates and is written into the output file.

- For writing into the output file, the key is set as the outdegree and the value is the proportion of outdegree with the total number of nodes ‘n’ for outdegree. This is written into the output file using the command ‘context.write(key1,value1)’.

3.3.1) <u>Degree Distribution of random graph with same number of nodes (n) and edges (m) as Enron graph</u>

<u>Indegree</u>

By plotting the indegrees with the binomial distribution and poisson distribution of the original data, that is, with the x-axis as Indegrees and y-axis as the Binomial Distribution and Poisson Distribution of the ‘from’ and ‘to’ mail addresses, we will obtain the following graph:



The above graph shows the plot between the indegrees in x-axis and the probability (P(k)), which can be Binomial Probability Distribution (or) Poisson Probability Distribution in the y-axis. The red (*) denotes the Binomial Distribution and the green (+) denotes the Poisson Distribution. The comparison between the distribution is done as follows:

Comparison of Results

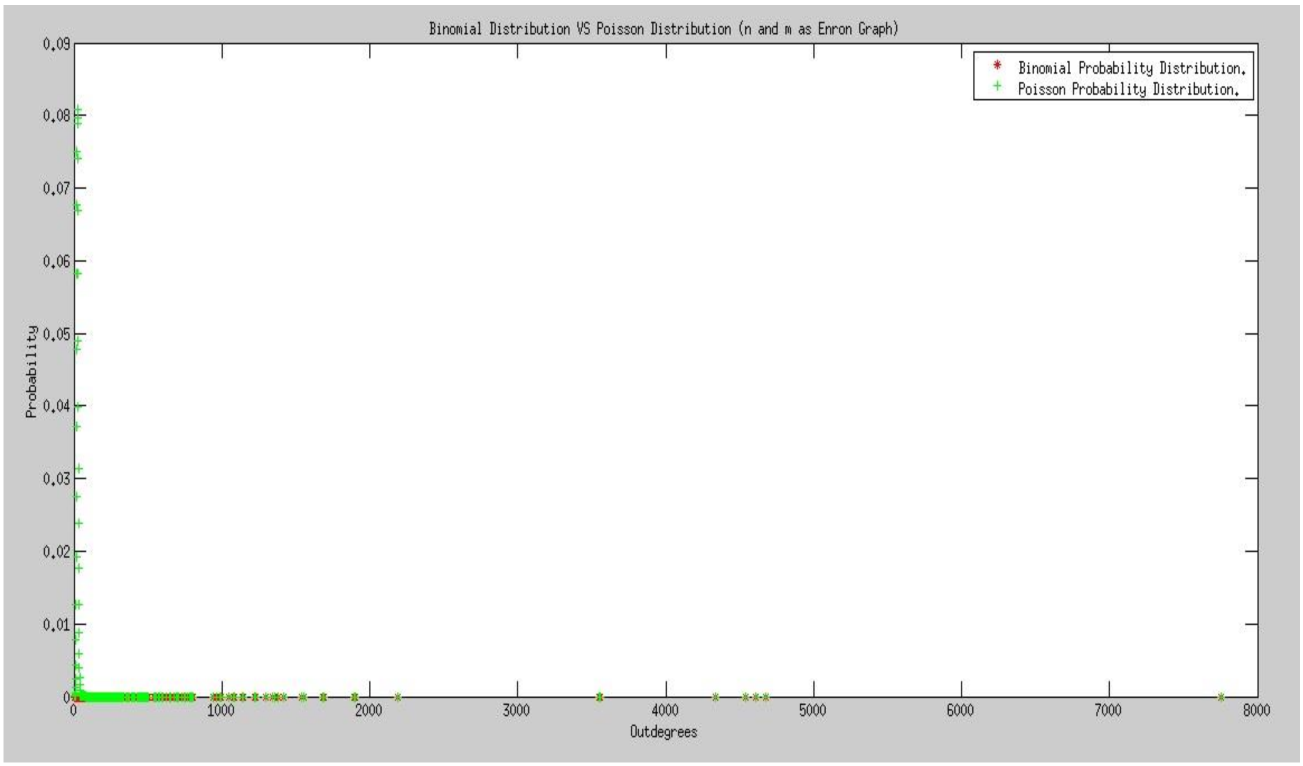
- The graph clearly shows that the binomial and the poisson distribution looks similar and overlaps at many instances. This is because binomial and poisson distribution are the same but poisson distribution varies with large values of nodes(n). Since the values of n are the same in both the distributions, both binomial and poisson distributions mostly overlap with each other but there are a few differences in the distribution.

- We can observe that binomial distribution has the probability ‘0’ when its degree is 0 and for a few degrees between 500 to 3000, at 4000 but poisson distribution has the probability 0 at many degrees such as from 0 to 1000 and the values of degrees where the binomial distribution also had probability 0.

- Then almost all the values of binomial distribution overlaps with poisson distribution except a few point differences between the two distributions, there is only a small difference between the binomial and poisson distribution for each degree and binomial seems to exceed poisson distribution but by a very small value. This is because, as the value of n(total number of nodes) becomes larger, the binomial distribution is approximated as poisson distribution. As n increases, p value decreases and the computation value of ‘lambda’, which is used for the calculation of P(k) in poisson distribution decreases further.

Outdegree

By plotting the indegrees with the binomial distribution and poisson distribution of the original data, that is, with the x-axis as Indegrees and y-axis as the Binomial Distribution and Poisson Distribution of the ‘from’ and ‘to’ mail addresses, we will obtain the following graph:



The above graph shows the plot between the indegrees in x-axis and the probability (P(k)), which can be Binomial Probability Distribution (or) Poisson Probability Distribution in the y-axis. The red (*) denotes the Binomial Distribution and the green (+) denotes the Poisson Distribution. The comparison between the distribution is done as follows:

Comparison of Results

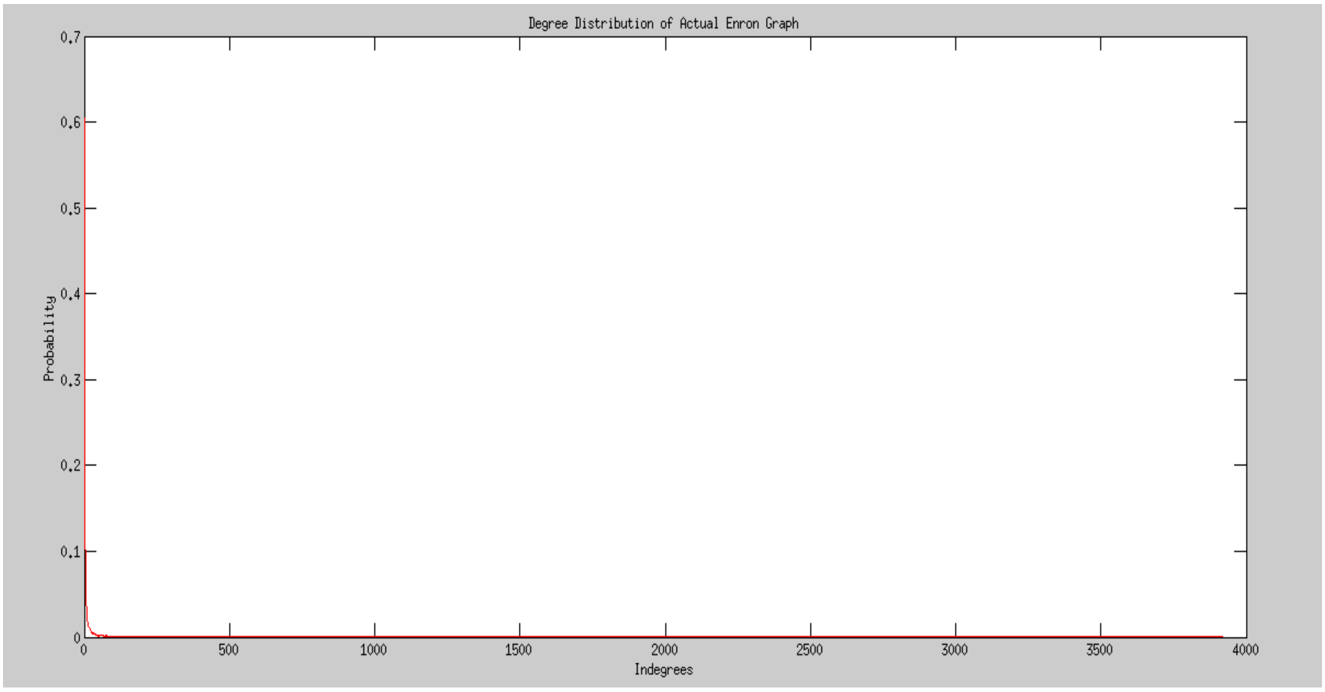
- The graph clearly shows that the binomial and the poisson distribution looks similar and overlaps at many instances. We can observe that both binomial distribution and poisson distribution has the probability '0' when its outdegree is 0 and for a few outdegrees between 0 to 2000, and at the degrees around 2200, 3600, between 4000 to 5000 and finally at 7800 whereas probability 0 for poisson at all degrees between 0 to 1000 and the values of degrees where the binomial distribution also had probability 0.

- The binomial distribution in the above graph is very low because the n or the total number of nodes is large. So, when the n is large, the binomial distribution is usually approximated using the poisson distribution. Hence, we can observe in the above graph that the peak can be observed in poisson distribution clearly than in binomial distribution. The poisson distribution reaches its maximum value of around 0.081 within few degree and then declines, and the binomial almost declines for higher values of n.

(3.3.2) <u>Degree Distribution of Actual Enron Graph</u>

<u>Indegree</u>

- The degree distribution of actual enron graph can be found by taking the indegrees that are present for each node in the enron graph. Then, taking into account all the indegrees in total, the number of occurrences of each 'to' address is taken as the count.
- This count is divided by the total number of nodes to find the proportion of the degree occurrence in the enron graph. This proportion of each degree can be calculated using the command "double p=((double)indegree/n);" , where 'p' is the proportion, 'n' is the total number of nodes for indegree.
- A graph is plotted with indegrees and the obtained proportion, this graph is the 'Degree Distribution of Actual Enron Graph'.



The degree distribution of actual twitter graph is formed by plotting the indegrees in the x-axis and the proportion of the indegrees' occurrence, that is, (number of times the indegree has occurred across all the nodes in the graph)/ (total number of nodes(n)).

<u>Interpretations</u>

- The indegrees are distributed in the actual enron graph is similar to the power law graph, that is, P(k) ~ x^{-alpha}, where alpha < 1, with a long tail. The power law, according to statistics, is a functional relationship between two quantities, where one quantity varies as a power of another. So, the degree distribution of enron graph looks similar to the linear scale of the Power law distribution. If we plot the degree distribution of enron graph in log-log scale, we get a straight line which joins both the axes.

- The degree distribution of enron graph does not exactly match the poisson distribution, whereas it looks similar to a graph plotted by power law distribution. The plot of power law shows that the initial values y-axis values are very high with small values of the term in x-axis and then the value in y-axis decreases gradually with an increase in x-axis values and we can observe that the above graph looks similar to the power law distribution. Hence, the plot on degree distribution of enron graph shows that the shape of this degree distribution matches most with the power law distribution.

- Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The above graph shows a positive skew or right skewed, that is, there is a long tail on the right side of the graph. The right tail is longer; the mass of the distribution is concentrated on the left of the figure. So, larger share of population rests within its tail, that is, many degrees are covered (from around 50 to around 3900) by the long tail on the right side of the graph. We can also observe that these degrees on the long tail have the probability 0 (from 50 to 3900), and with probability slightly above 0(less than 0.1) from 0 to 50 in the degree distribution and also there are many nodes with the degree 0, as mostly the proportion of nodes with the degree 0. Since the degree distribution is positively skewed, the mean is greater than the median reflecting the fact that the mean is sensitive to each degree in the distribution and is subject to large shifts when the sample is small and contains extreme degrees.

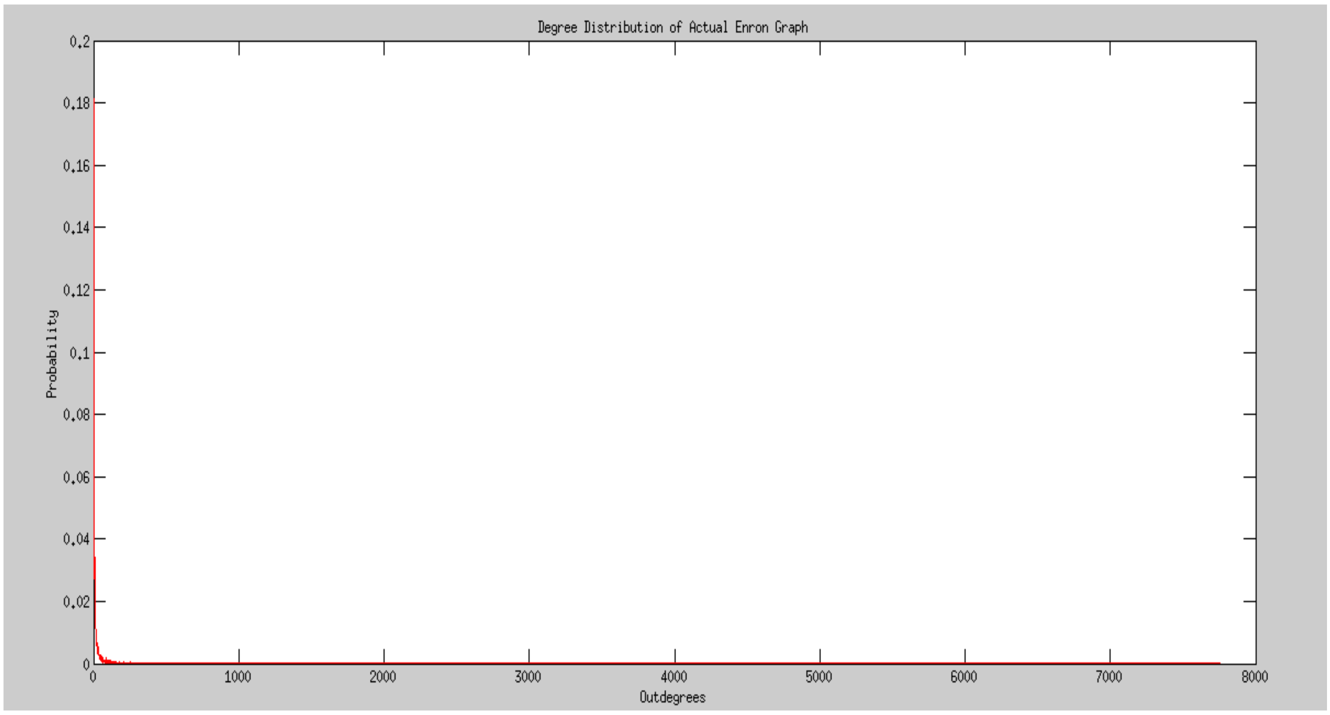
- There are a few hubs in the enron graph, that is, the nodes which are exceptionally well connected to the other nodes in the graph. The above degree distribution of enron graph is a plot between the indegrees and the proportion of nodes of each degree (count of occurrence of each indegree / total number of nodes). So, it can be observed from the above graph that the many nodes had the degree around 0 and then it increases to degress around 50, but declines after 50 to the probability 0 . The node which has the highest indegree(around 3900) is the hub as it will be well connected among various nodes in the graph.

<u>Outdegree</u>

- The degree distribution of actual enron graph can be found by taking the outdegrees that are present for each node in the enron graph. Then, taking into account all the outdegrees in total, the number of occurrences of each ‘from’ address is taken as the count.

- This count is divided by the total number of nodes to find the proportion of the degree occurrence in the enron graph. This proportion of each degree can be calculated using the command “double p=((double)outdegree/n);” , where ‘p’ is the proportion, ‘n’ is the total number of nodes for outdegree.

- A graph is plotted with outdegrees and the obtained proportion, this graph is the ‘Degree Distribution of Actual Enron Graph’.



The degree distribution of actual enron graph is formed by plotting the outdegrees in the x-axis and the proportion of the outdegrees’ occurrence, that is, (number of times a particular ‘from’ address has occurred across all the nodes in the graph)/ (total number of nodes(n)).

(2.1.4) <u>Interpretations</u>

<u>Outdegree</u>

- The degrees are distributed in the actual enron graph is similar to the power law graph, that is, P(k) ~ x^{-alpha}, where alpha < 1, with a long tail. The power law, according to statistics, is a functional relationship between two quantities, where one quantity varies as a power of another. So, the degree distribution of enron graph looks similar to the linear scale of the Power law distribution and so if we plot the degree distribution of enron graph in log-log scale, then it is almost a diagonal line connecting the x and the y-axis.

- The degree distribution of enron graph does not exactly match the poisson distribution, whereas it looks similar to a graph plotted by power law distribution. When power law is plotted, the initial probability values y-axis values are very high with small values of the term in x-axis and then the value in y-axis decreases gradually with an increase in x-axis values. Hence, the above graph on degree distribution of enron graph shows that the shape of this degree distribution matches most with the power law distribution.

- Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The above graph shows a positive skew or right skewed, that is, there is a long tail on the right side of the graph. The right tail is longer; the mass of the distribution is concentrated on the left of the figure. So, larger share of population rests within its tail, that is, many degrees are covered (from around 50 to around 7800) by the long tail on the right side of the graph. We can also observe that these degrees on the long tail have the probability 0 in the degree distribution. We can also observe that these degrees on the long tail have the probability 0 in the degree distribution and also there are many nodes with the degree 0, as mostly the proportion of nodes exists with degree 0. Then there were a few nodes above the degree 0 with also probability above 0(from degree 0 to degree 50), and then this declined. Since the degree distribution is positively skewed, the mean is greater than the median reflecting the fact that the mean is sensitive to each degree in the distribution and is subject to large shifts when the sample is small and contains extreme degrees.

- There are a few hubs in the enron graph, that is, the nodes which are exceptionally well connected to the other nodes in the graph. The above degree distribution of enron graph is a plot between the outdegrees and the proportion of nodes of each outdegree (count of each 'from' address / total number of nodes). The node which has the highest outdegree is the hub as it will be well connected among various nodes in the graph.

4) <u>Graph Visualization</u>

A community in a social network is a set of densely connected nodes with the nodes belonging to different communities being only sparsely connected. Communities play an important role in the social network analysis as they typically correspond to groups of individuals sharing a variety of common traits of interest.

4.1) <u>Implementation of the solution</u>

The values generated by the second question are used for this section.

<u>Mapper Class</u>

- The mapper class is used to read the email addresses in the dataset. Every email has a single sender, that is, the email address in the From: field, but may have one or more recipients. Recipient email addresses can be found in the To:, Cc:, Bcc: and also date from the Date: field.

- The stripCommand function is used to separate out the data from all the fields such as From:, To:, Cc:, Bcc: and Date: in the dataset.

- Then, we separate the emails that belong to only the enron dataset in the procFrom function after getting the emails separately.

- The map function of the Mapper formally scams each line of the dataset and calls the stripCommand and the procFrom functions to get all the data from the respective fields. Since the fields To:, Cc: and Bcc: means the receiver of the mail, they are listed in the same list known as the 'recipients' list. They are added to the list using the command 'recipients.add()'.

- The map function is also used to concatenate the timestamp or date to the recipient address and set it as the value.

- Summarizing the actions of mapper class, the mapper class iterate over the large set of records to obtain the values that are present in the fields 'From', 'To', 'Cc', 'Bcc' and 'Date'. Then, all the receivers' mail address is added to the recipient list and the timestamp is concatenated with the recipients. The 'from' mail address acts as the key and the recipients mail address with the timestamp will be the value.

Reducer Class

Step 1: The duplicate entries are deleted that match the 'to' and 'from' entries in the values generated by the mapper class, that is, the table R. These duplicate values are deleted using the hash set 'uniques'. This operation is done as follows:

- The values generated by the mapper class, that is the recipient email address is checked whether it is present in the hash set, if it is not present in that, then it is added to the hash set.
- So, this set contains unique values of a combination that includes the 'from', 'to' and 'timestamp' together. The set of values generated is stored in a new table 'M'.

Step 2: The weight of an email is calculated between x and y where 'x' is the sender and 'y' is the receiver as follows:

$$\text{weight}(x; y) = \frac{\#\ \text{arc}(x; y)\ \text{in M}}{\text{total}\ \#\ \text{of arcs in M}}$$

This formula can be further simplified as follows:

$$\text{weight}(x; y) = \frac{\text{total}\ \#\ \text{of mails between x and y}}{\text{total}\ \#\ \text{of mails in M}}$$

This operation is done as follows:

- The count of mails between x and y where ‘x’ is the ‘from’ address and ‘y’ is the recipient address is found by first concatenating the x and y, so that the distinct pair of x and y is identified. The pair of all x and y is added to an array list
- The count of the mails between x and y is calculated using Collections.frequency(list,a), where ‘a’ involves a particular mail between x and y.
- The total count of distinct values is calculated by incrementing the variable ‘totcount’. This total count is the total number of mails in M
- The weight of an email is calculated by dividing the count of email between x and y by the total count of mails in M.
- As passed by the mapper, the ‘from’ mail address acts as the key and the receiver’s address with the weight acts as the value, and this is passed in the write function such as, context.write(key,value).
- The output obtained from this is saved as a .csv file, say, Result.csv and is uploaded into Gephi tool for community Detection

<u>Community Detection</u>

Gephi can be used to detect communities in a network. Graph modularity is a way of defining graph structure by its modules (also called clusters or communities). Graphs with high modularity have dense connections between the nodes within individual modules and sparse connections between nodes in different modules.

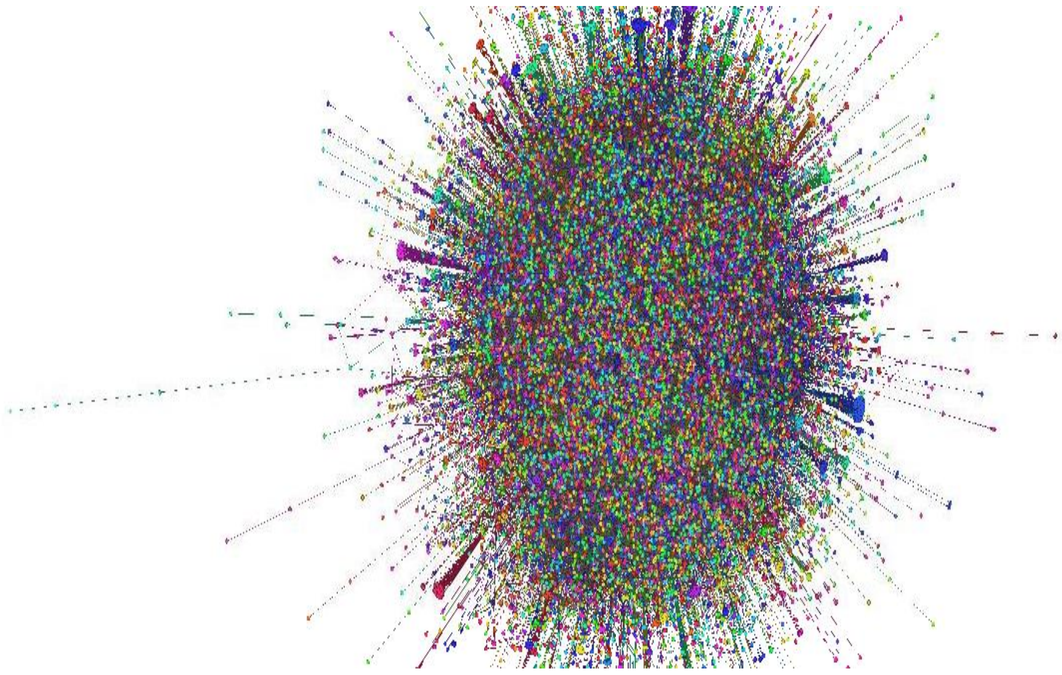
- From the Statistics tab, select the ‘Modularity’ under ‘Network Overview’ and run it.
- Then, from the ‘Partition’ tab, click the refresh button and in the drop-down list select the modularity option, click apply and your graph should now have distinct differently coloured communities.
- From the ‘Layout’ tab, choose ‘Force Atlas 2’ and click on ‘Run’ to run it.
- Save the modularity report obtained and export the graph as svg file naming it as ‘network.svg’
- Simlarly, from the Statistics tab, select the ‘Connected Components’ under ‘Network Overview’ and run it.
- Save the Connected Components Report obtained

4.2) <u>Community Structure Induced by Enron Email Communication</u>

The Community structure obtained from gephi is the following:

- The different colours community structure indicates the different communities. The communities which have similar opinions are grouped together.

- We can observe from the graph, that the blue colour dominates the other colours, so the community which belongs to the blue colour is the maximum.



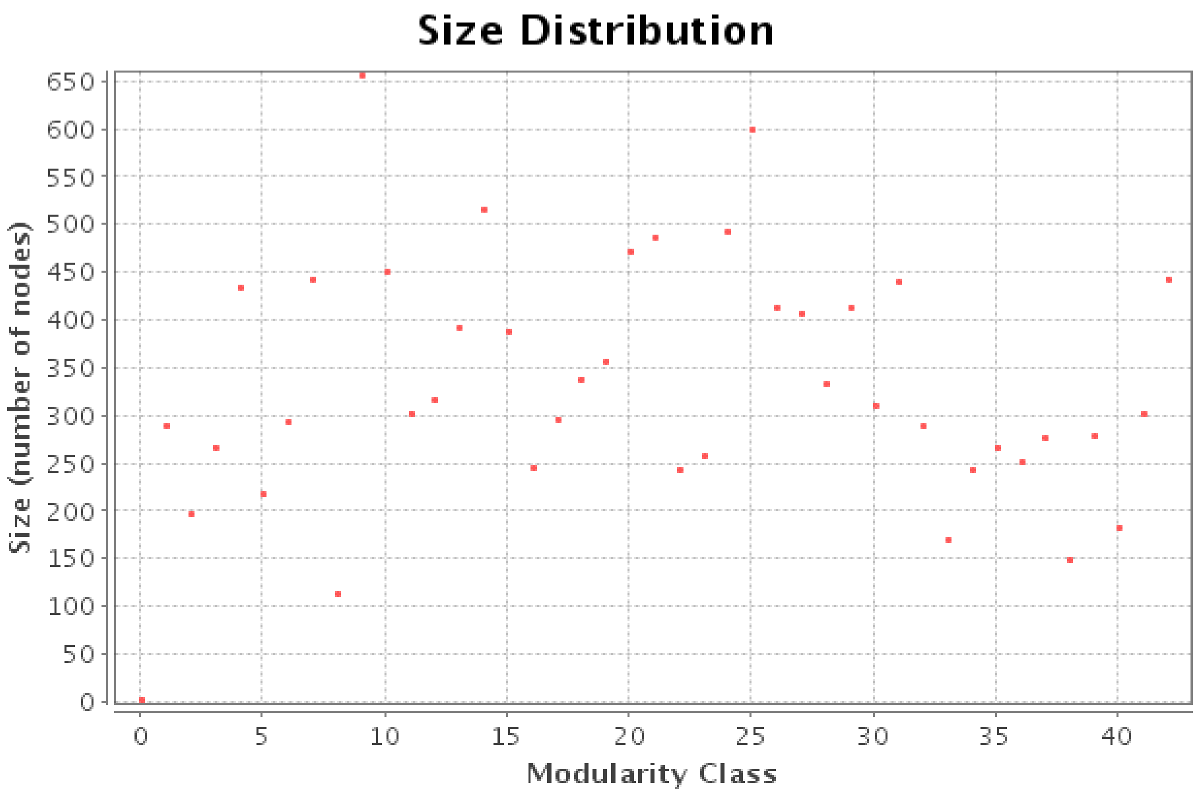
- In the ‘Partition’ tab, when refresh button is clicked, we obtain the ‘Modularity class’ in the drop down, this is selected to find the percentage domination of each community

- From this, we can observe that the 13th community which is represented by ‘blue’ colour has the maximum percentage of 4.78 % followed by the 26th community which is represented by ‘purple-pink’ colour with the percentage as 4.4%, then 15th community with 3.49% and the 39th community stands close to this with the percentage as 3.47%. This goes on with the least domination of the 0th community, which is represented by light green colour and has the percentage of 0.02 %.

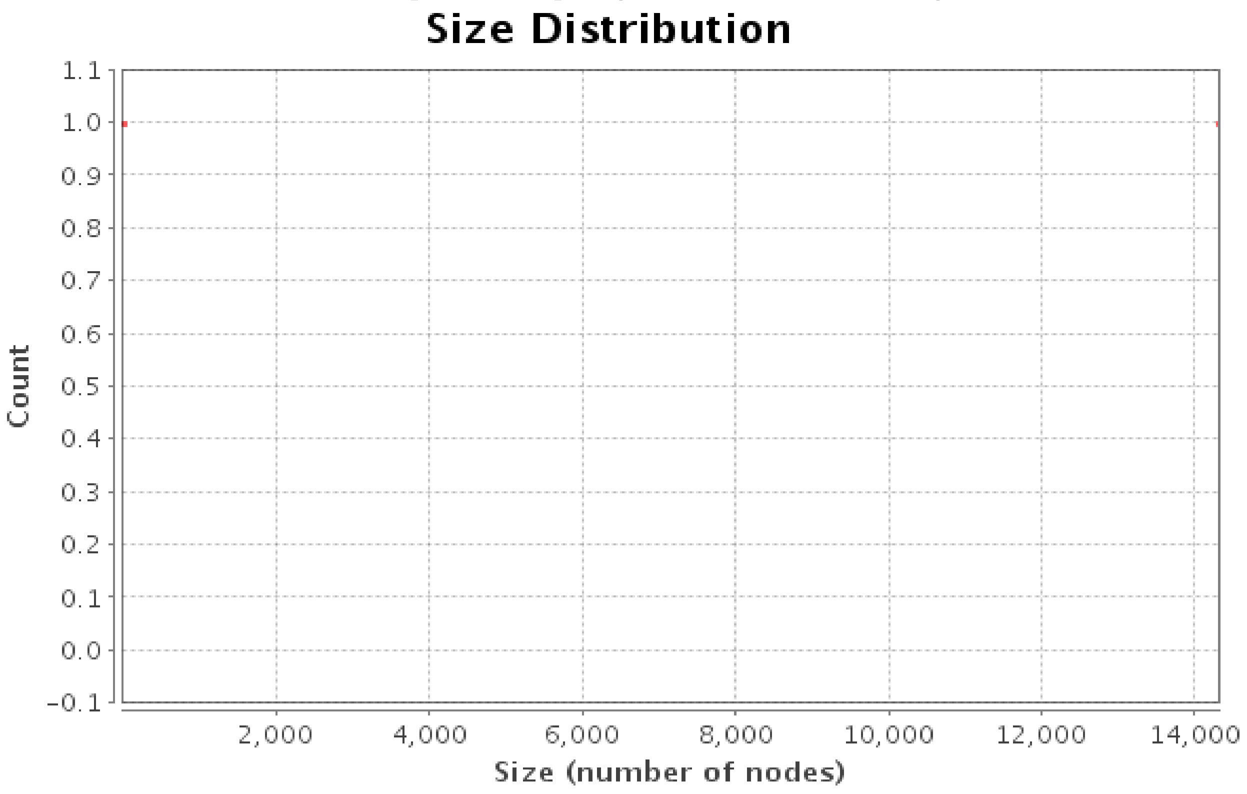
The Modularity Report generated is the following:

- The Modularity class shows the communities that have the same opinions, this is represented in the above graph with the modularity class community on the x-axis and the size or the number of nodes that belong to that particular class is present in the y-axis

- The Modularity class has the different communities numbered from 0 to 45 and it can be observed that 8th class has the maximum number of nodes of 650 followed by the 25th class with the size as 600 nodes and also the 0th class has the least number of nodes of 0.



The Connected Components Report generated is the following:



- The Connected Components report shows the strongly connected and weakly connected nodes in the graph

- In the above graph, the number of strongly connected nodes is 9563 and the number of weakly connected nodes are 2, so the strongly connected nodes contribute 99.98% of the total number of nodes and the weakly connected nodes contribute 0.02% of the total number of nodes

4.3) My Contribution to solving this section

- After running the 'Modularity', from the 'Layout' tab, I chose 'Force Atlas 2' and clicked 'Run' to run it to find the communities depicted in different colours and has their percentage in the Partitions tab.

- From the Statistics tab, select the 'Connected Components' under 'Network Overview' and ran it to find the strongly connected components and weakly connected components.

5) Conclusion

- The Enron email corpus remains one of the largest publicly available email datasets. This dataset is taken and the emails are extracted to split the 'from' address, 'to' address, 'cc', 'bcc', 'date'. The 'to', 'cc' and 'bcc' is added to the recipient list. The email addresses from a particular person to a particular person undergoes filteration according to the timestamp.

- The duplicate values are deleted and the dataset which has been modified to a table is extracted to a social network, that is the social interaction between a particular 'from' address and 'to' address is calculated by assigning a weight for each 'from-to' pair of mail address.

- After obtaining the social interaction among the 'from-to' pair, the social network analysis is made on the revised above data by drawing graphs between the indegree or outdegree with their binomial and poisson distribution and also a proportion of the indegree with the total number of nodes is taken as the degree distribution of the actual enron graph. This degree distributions gives the social network analysis of all the 'from-to' pairs that are present in the enron dataset.

- Finally, the number of communities that have the same opinion is found out using the Gephi tool, the community detection tells us which set of communities dominate over the other communities and which has the least domination.

Thus, the enron dataset is subject to many modifications for full data analysis and finally the data is visualized for social network analysis and communities among the dataset.