**TRACING RUMOURS BY ANALYZING SOCIAL MEDIA DATA**

**PROJECT DOCUMENTATION**

**LIST OF MODULES:**

1. Data Preprocessing
2. Contact Sequence Identification
3. Temporal Network Construction
4. Time respecting paths identification
5. Randomization of model
6. Simplification and spreaders identification

**INPUT:**

Social media data as csv file

**OUTPUT:**

List of influential and important spreaders

**SUBMITTED BY:**

**GOBINATH A L - 2018103025**

**KUTALA MANOJ - 2018103040**

**MOHAMED ABDUL MALIK J - 2018103043**

**INTRODUCTION**

Social media generates huge amount of metadata that could be used to understand social

information flow, identify how spreading happens, etc. Many kinds of such metadata could

be represented as temporal networks, networks that record when contacts happen, in

addition to who has been in contact with whom. The field of temporal networks is still under

rapid development. There are a number of techniques to analyse social media data sets.

These methods serve to identify important spreaders, characterize the behaviour of the

social media users, and map out temporal ad topological structures. In our project, we will

construct a temporal network using social media metadata and find the influential and

important spreaders. Influential spreader is the one who can influence many others.

Important spreaders are the ones who play a vital role in boosting the spreading process.

**AREAS THAT CAN BE ANALYZED:**

* Social media interaction
* Direct e-mails or e-mail like messages
* Facebook wall posts and friendships
* Internet forum posts
* Blogs and microblogging

**Social networks can be leveraged to:**

* Identify influential entities
* Detect communities with special interests
* Spread of ideas
* Entity disambiguation, etc

**Methods used to simplify the temporal network:**

* Randomized edges(RE)
* Randomly permuted times(RP)
* Random times(RT)
* Randomized Contacts(RCs)
* Equal-weight edge randomization(EWER)
* Edge randomization(ER)
* Time reversal(TR)

**LITERATURE SURVEY:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SNO** | **AUTHOR,PUBLICATION,YEAR** | **METHODOLOGY** | **ADVANTAGES** | **LIMITATIONS** |
| 1 | PETTER HOLME “ANALYZING TEMPORAL NETWORKS IN SOCIAL MEDIA”, 2014 | RANDOMIZING THE NETWORK | IMPORTANCE OF A NODE IN A NETWORK CAN BE STUDIED | MAY NOT BE SUITABLE FOR ANALYZING IF THE NETWORK CONTAINS MANY CYCLES |
| 2 | P. HOLME AND J. SARAMA, ‘‘TEMPORAL NETWORKS’’, 2012 | ALGORITHM TO IDENTIFY TIME RESPECTING COMPONENTS IN A NETWORK | MAKES IT EASY TO TRACK THE ORIGIN OF MESSAGE | MAY NOT WORK IF THE NETWORK DOESN’T HAVE ANY NODE WITH INDEGREE AS 0 |

**Randomization techniques:**

1) **Randomized Edges (REs)**: This method is similar to the edge swapping for static graphs mentioned above, with the additional fact that contact sequences of edges follow the edges when these are rewired.

2) **Randomly Permuted (RP) Times**: To understand the role of the order of the contacts, one performs the RP randomization. In this procedure, one permutes the contact times randomly while keeping the network structure and the numbers of contacts between all pairs of vertices fixed. Technically, this is much simpler than applying the edge-rewiring scheme discussed above, as it only requires randomly swapping the time stamps of all contacts. No checks similar to step 4) of the RE rule need to be performed for contact sequences. Like RE, this scheme also retains the overall rate of events in the network at every point in time, such as daily or weekly patterns in communication networks.

3) **Random Times (RTs)**: The ensemble defined by RE and RP randomizations conserves the set of times of the original contact sequence. Hence, although it destroys time structures of events related to individual vertices and edges, the rate of events in the entire temporal network is unchanged and will still follow the typical circadian and weekly patterns of human activity. This type of randomization needs a process where spreading depends on time, not only the order of contacts.In maximal speed spreading such as the one behind the latency discussion above, this randomization would have exactly the same effect as RP. Note that an alternative to uniformly random contact times is to generate them from a specific distribution or process, such as the Poisson process, with parameters set up so that the numbers of contact per each edge are, on average, conserved.

4) **Randomized Contacts (RCs)**: For this randomization scheme, one keeps the graph

topology fixed but redistributes the contacts randomly among the edges. After this

randomization, the number of contacts per edge follows the binomial distribution. It is

intended to test the effect of fat-tailed distributions of this quantity as typically

seen,especially in social media and other forms of human generated communication data. If

one would like to test the effect of the distribution of the number of contacts alone, keeping

the structure of the temporal order of the real data, then one would need a different

approach. For example, a vertex that is active primarily in the early stage of the data would

be so in the randomized data as well, but one would need to compensate for such effects.

5) **Equal-Weight Edge Randomization (EWER)**: Some times one would need to remove

correlations between the static network structure, and at the same time retain the temporal

structure of the edges. This is achieved by randomly swapping entire contact sequences of

edges with the same number of contacts. For example, all the contacts and their time

stamps are randomly exchanged between edges that have the same number of contacts.

Thus, single-edge patterns, such as burstiness of the contacts between two individuals, are

retained, together with other properties preserved by the RP model (like the number of

contacts of an edge, the system-wide contact frequency, and the topology of the network of

accumulated contacts). This null model requires a large enough system so that there are

enough edges with the same number of events.

6) **Edge Randomization (ER)**: This null model is similar to the EWER model with the

exception that the sequences can be exchanged between edges that have any numbers of

contacts. This corresponds to randomly exchanging the edge weights in the network of

aggregated contacts. The correlation between weight and topology is destroyed in this null

model. However, the intercontact time distributions of contact sequences of edges are

preserved; the sequences are just moved elsewhere in the network

.7) **Time Reversal (TR)**: This null model is designed for assessing the frequency and

importance of causal sequences of contacts, where, for example, contacts trigger further

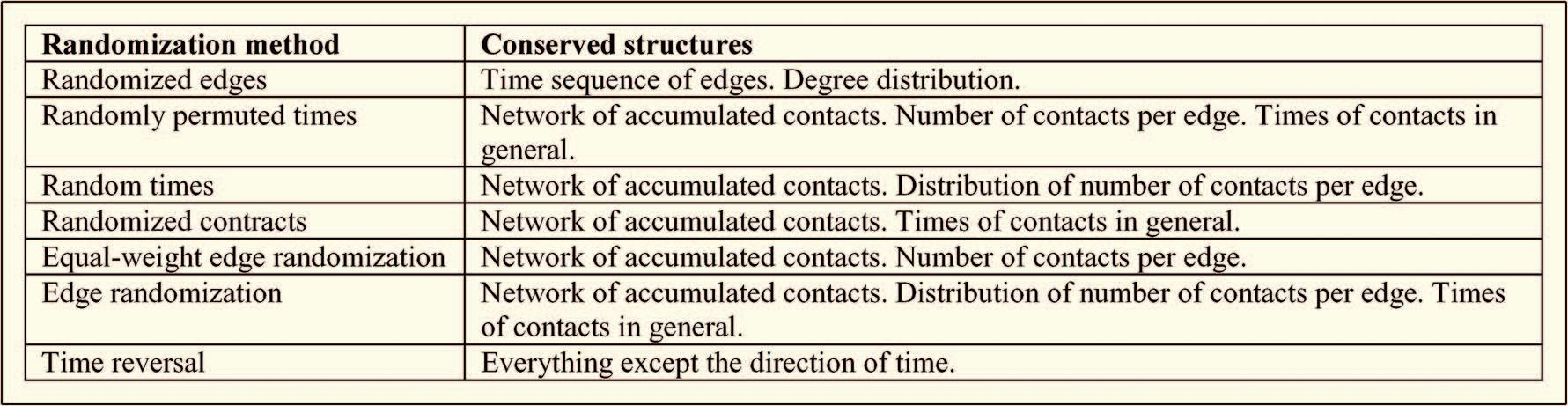
contacts. It simply involves running the original event sequence backward in time. If

sequences of consecutive contacts would be caused by temporal correlations alone, similar

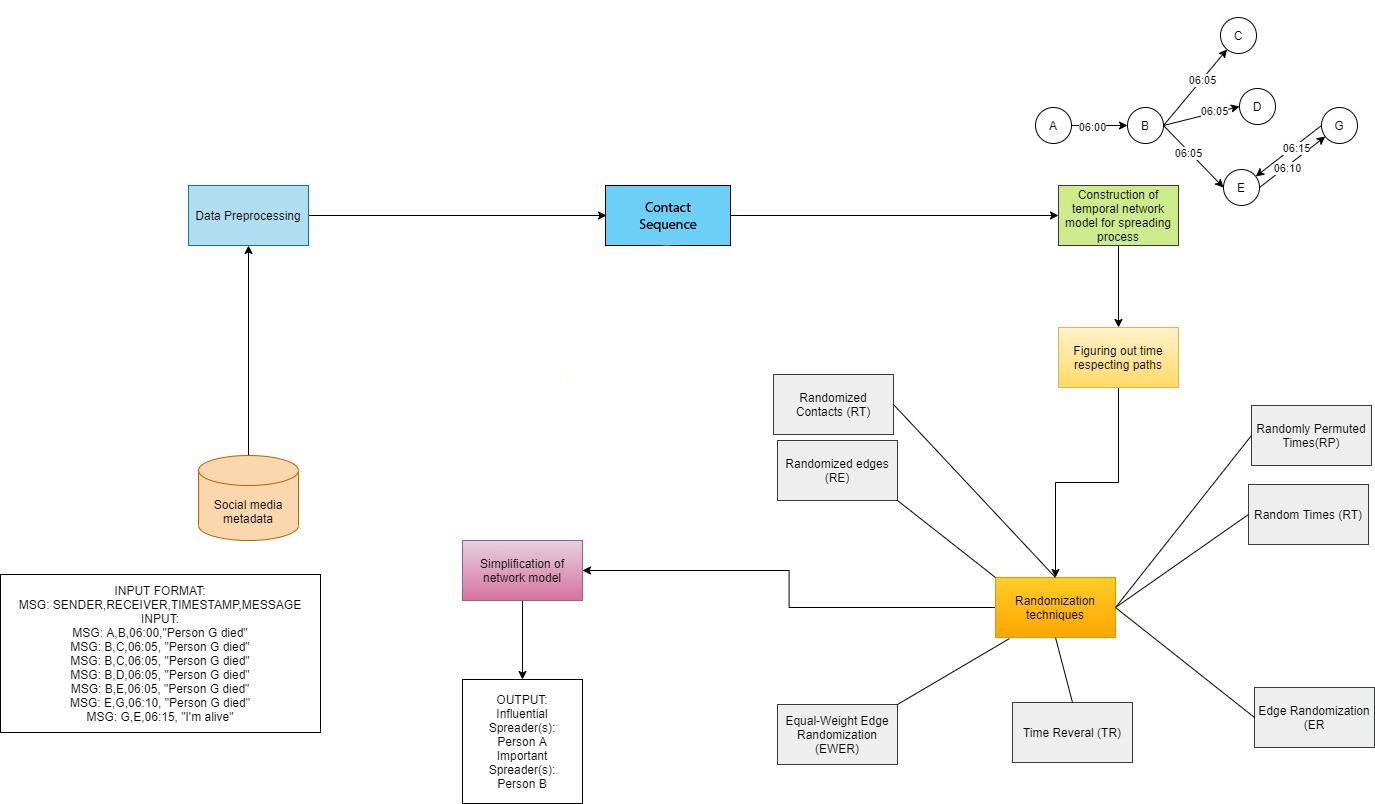
numbers of such sequences should be observed when time runs forward and backward. A

lack of such chains in the time-reversed null model compared to the original sequence could

be attributed to the arrow of time.



**ARCHITECTURAL DIAGRAM WITH INPUT AND OUTPUT MODULE:**

****

**MODULE 1 - Data Preprocessing**

**Input:** Social media metadata which is stored as csv file

**Output:** Preprocessed data

**Pseudocode:**

|  |
| --- |
| Read the CSV file and store it as multi-dimensional array object, D.  Maintain a set of unique and preprocessed messages, say S and also a message list, L.  for every item in D:  extract the message part, M  if M not in S:  add M to S  add M to L.  print all the unique messages  Ask the user to trace which message, say X.  Return all rows(subArrays) in D containing the message X. |

**Explanation:** The input which is given as a csv file to this module consists of the chats in the format: sender, receiver, timestamp, message sent. Now this file contains a lot of messages and the user is asked to select the message which has to be traced. This module extracts the tuples with this message and gives this list as output. It can also identify the slightly modified version of the message and extract it. It traverses through the rows and checks if the message value of the row is equal to the message given or almost similar to the message given and adds it in a set. When there are no more rows to traverse, it returns the set as the output.

**MODULE 2 -Contact Sequence Identification**

**Input:** Output of module 1

**Output:** Contact Sequence

**Pseudocode:**

|  |
| --- |
| Maintain a dictionary D with,  Key: Sender Name  Value: Receiver Name, Message sent time  for every message in preprocessed message list:  D[Sender name] += [Receiver Name, Message sent time]  for (key,value) pair in D:  print ( Sender name (key) : [Receiver name, message sent time] (value) ) |

**Explanation:** The preprocessed data from module 1 is given as input to this module. Now this preprocessed data is analyzed by this module. It maintains a dictionary D which is initially empty.

D = { }

Then it traverses the rows and add a new entry in the dictionary. The entry consists of sender name as key and receiver name and timestamp as the value.

D = { Sender : Receiver, Timestamp }

This step is repeated for all the rows in the preprocessed data. Finally we get the contact sequence which will be given as input to the next module. The contact sequence is just a better representation of preprocessed data and also easy to understand.

**MODULE 3 - Temporal network construction:**

**Input:** Preprocessed data

**Output:** Temporal Network

**Pseudocode:**

|  |
| --- |
| Create a graph G  Initially, first Sender as root node R of G  Make recipients of first Sender as child nodes of the root node R  Draw an edge between root and child nodes, labelling message sent time.  for every child node, C:  make all recipients children of C  draw edge E between C and children (C1…Ck), labelling message sent time  repeat the process until no more child nodes |

**Explanation:** The preprocessed data is given as input to this module. Now this module constructs the temporal network which is the basis for this project and this temporal network is required for the next 3 modules. The output of this is the graphical representation of the network. To represent the network graphically python libraries such as matplotlib, networkx and pygraphviz are used.

It reads the contact sequence and identifies all the persons involved. These persons are considered as nodes. Then an edge is formed between two nodes if the message is sent from one node to the other. The time at which the message is sent becomes the weight of the edge. So at last we get a weighted directed graph

**MODULE 4 - Time respecting paths identification:**

**Input:** Temporal network

**Output:** List of time respecting paths

**Pseudocode:**

|  |
| --- |
| Maintain an adjacent matrix which consists of adjacent nodes to a particular node  Maintain a set called Timerespectingpath which is initially empty  Find indegree of nodes  For each node x whose indegree=0 in temporal network:  For each node in adjacent(x):  DFS Traversal  UpdatePathOrder()  Timerespectingpath += traversedPath  Return timerespecting path |

**Explanation:** This module returns a list of time respecting paths which are the sequences of links that can be traversed in a time-varying network under the constraint that the next link to be traversed is activated at some point after the current one. Like in a directed graph, a path from A to B does not mean there is a path from B to A. It has a matrix called adjacent matrix in which adjacent(i) gives the nodes which are adjacent to node i. It also maintains a set Timerespecting path which is initially empty. Then for nodes whose indegree is zero, it finds all the possible paths from that node. The path should also be time respecting meaning increasing order of time should be maintained in the path.

A->B->C is said to be a timerespecting path if the timestamp in the edge (A,B) is less than the timestamp in the edge (B,C)

**MODULE 5 - Randomization of model:**

**Input:** Temporal Network, Time-respecting paths

**Output:** Randomized model

**Pseudocode:**

|  |
| --- |
| 1)Go over all edges sequentially.  2)For every edge (i,j):  Pick another edge (i’,j’)  3)With a probability 1/2, replace (i,j) and (i’,j’) by (i,j’) and (i’,j) otherwise, replace them by (i,i’) and (j,j’).  4)If the move in step 3) created a self-edge or a multiple-edge, then undo it and start over from step 1). |

**Explanation:** This module studies the temporal network to randomize the model and returns this randomized model as the output. As it can be seen from the pseudocode, it picks the first edge then it picks a random edge and the edges are randomized according to the algorithm. If it has been done for all the edges, then we get a edge randomized model of the temporal network.

The purpose of randomizing the edges is that we have to study the importance of each node in the network. Thus by randomizing the edges, we can analyze how the network behaves and finally this module gives a score to every node based on the importance of the node in the network. Higher score means that the node is important in the network and lower score means that it is not significant in the network. Also, this module generates 4 edge randomized models to calculate the score.

**MODULE 6 - Simplification and spreaders identification:**

**Input:** Calculated score

**Output:** Influential and important spreaders

**Pseudocode:**

|  |
| --- |
| Percentagelist = calculatepercentage(module5output)  For each node in Temporal Network:  findindegree(node)  findoutdegree(node)  for i,element in enumerate(indegree):  If element == 0:  print(“influential spreader”, i+64)  maxElement = findMaxElemet(percentagelist)  For i,element in enumerate(percentagelist):  If element == maxElement && if element > 20%:  print(“Important spreaders”,i+64)  End for |

**Explanation:** This is the final module of this project. This module is named “simplification of network model” because it studies the network as if it was a static model to obtain quicker results. This module studies the temporal network and finds the influential and important spreaders. Influential spreader is the one who starts the rumour. Important spreaders are the one who boosts the spreading process. The output is “influential and important spreaders”.

This module takes the calculated score as the input and convert them into percentage for simplicity. It classifies the node with indegree 0 as influential spreader and the nodes whose contribution to the network is >20% as important spreaders.

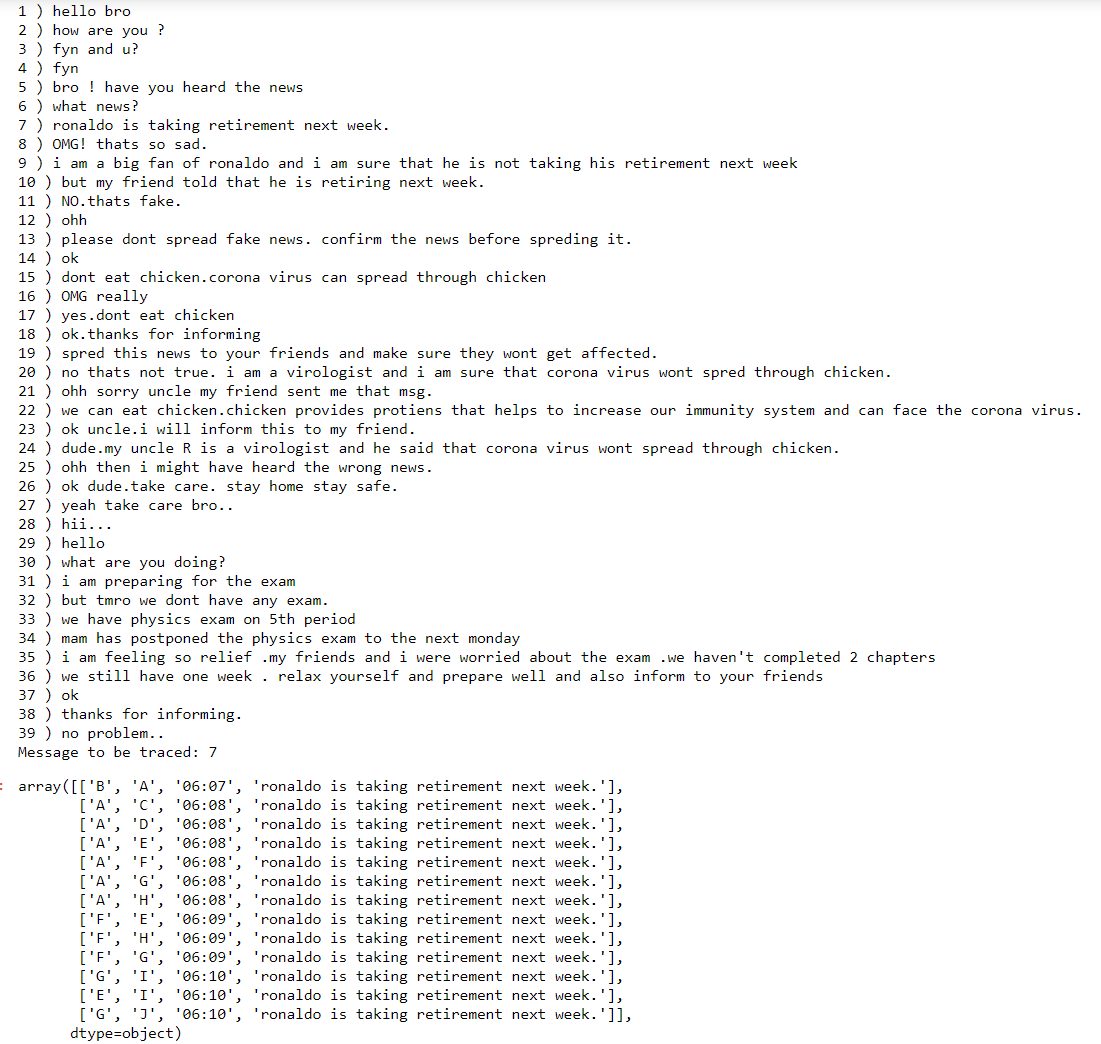
**RESULT AND DISCUSSION (100% IMPLEMENTATION):**

**INPUT FILE:**

* The first column contains the row number.
* The second column contains sender’s name.
* The third column contains the receiver’s name.
* The fourth column contains the timestamp of the message.
* The fifth column contains the actual message which was sent.
* This is a csv file and it is given as input.

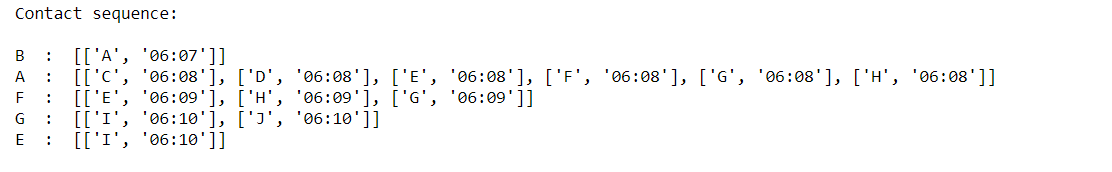
**TRACE 7**

**MODULE 1 OUTPUT:**



* Module 1 is **data preprocessing**.
* It displays the list of messages with a unique number given to it.
* Now the user is asked to enter the number of the message which is to be traced.
* Now this module processes the user input and extracts the message(in this image the message is “ronaldo is taking retirement next week”) which is associated with this number(in this image it is 7). It can be seen that only those tuples which contains the message(in this image it is “ronaldo is taking retirement next week”) are extracted successfully.

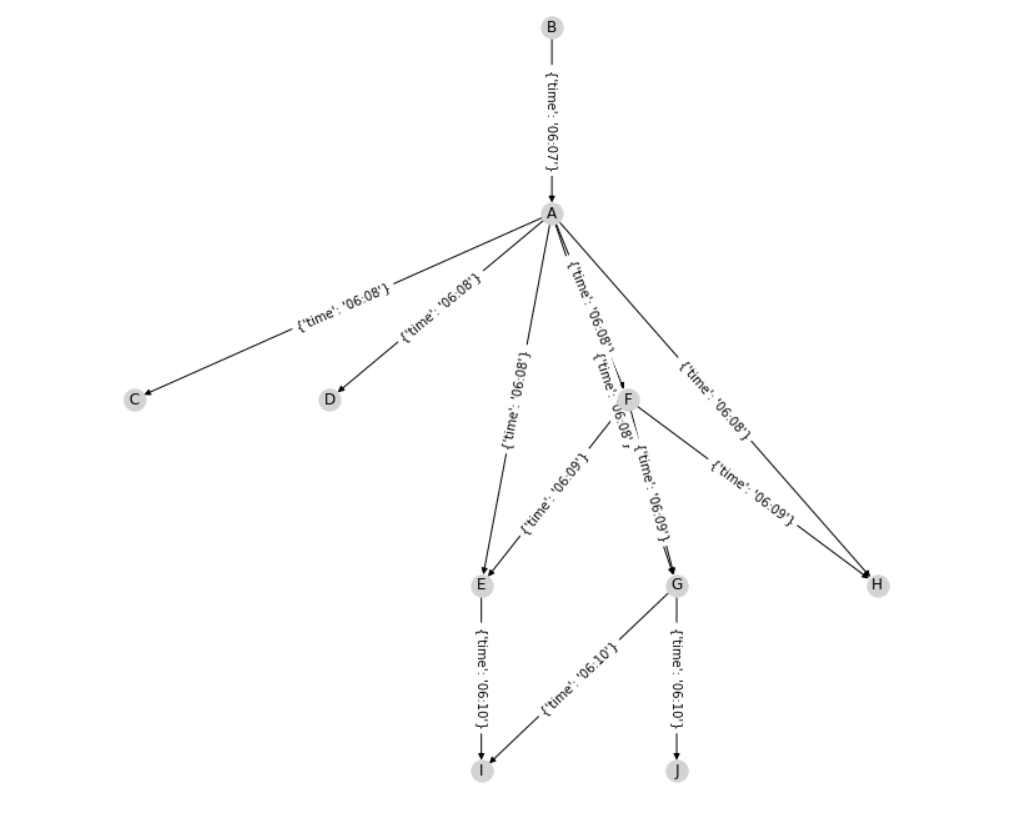
**MODULE 2 OUTPUT:**



The data after preprocessing by module 1 is given as input to this module. It can be seen from the output of the last module that person B is sending the message(“ronaldo is taking retirement next week”) to person A and person A forwards to C,D,E,F,G,H and person F forwards to E,H,G and person G forwards to I,J and person E forwards to I

Hence this module represents them as a contact sequence with the timestamp attached to it.

**MODULE 3 OUTPUT:**

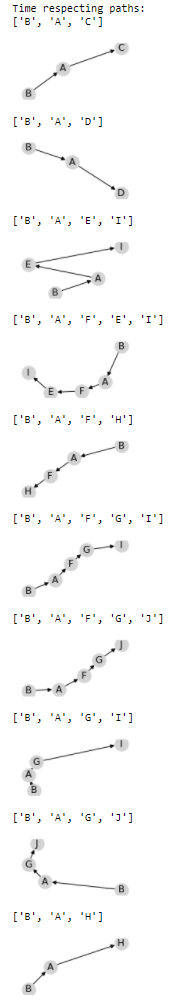


The contact sequence from the previous module is processed by this module to make a temporal network.

In this graph,

* Nodes represent persons
* Directed edge from x to y represents that x has sent message to y
* The weight of the edge represents the time in which the message was sent

**MODULE 4 OUTPUT:**

****

This module displayed the list of time respecting paths that are present in the network.

The constraint is that increasing order of time should be maintained.

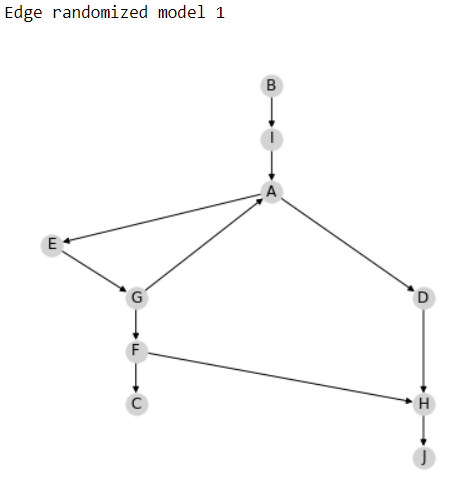
In B->A->C, B messages A in 06:07 and A messages C in 06:08

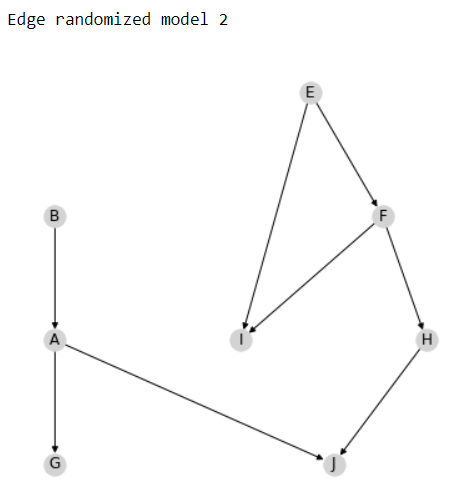
In B->A->D, B messages A in 06:07 and A messages D in 06:08

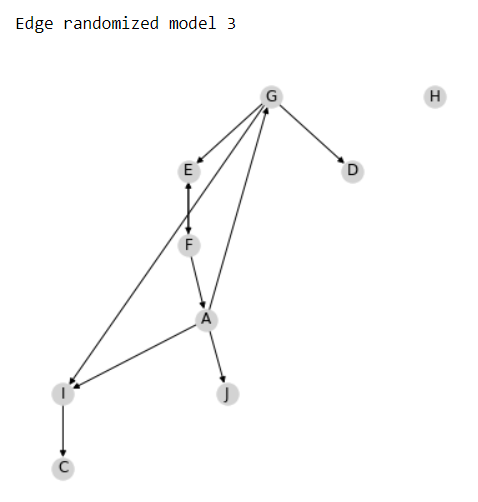
In B->A->E->I, B messages A in 06:07 and A messages E in 06:08 and E messages I in 06:10

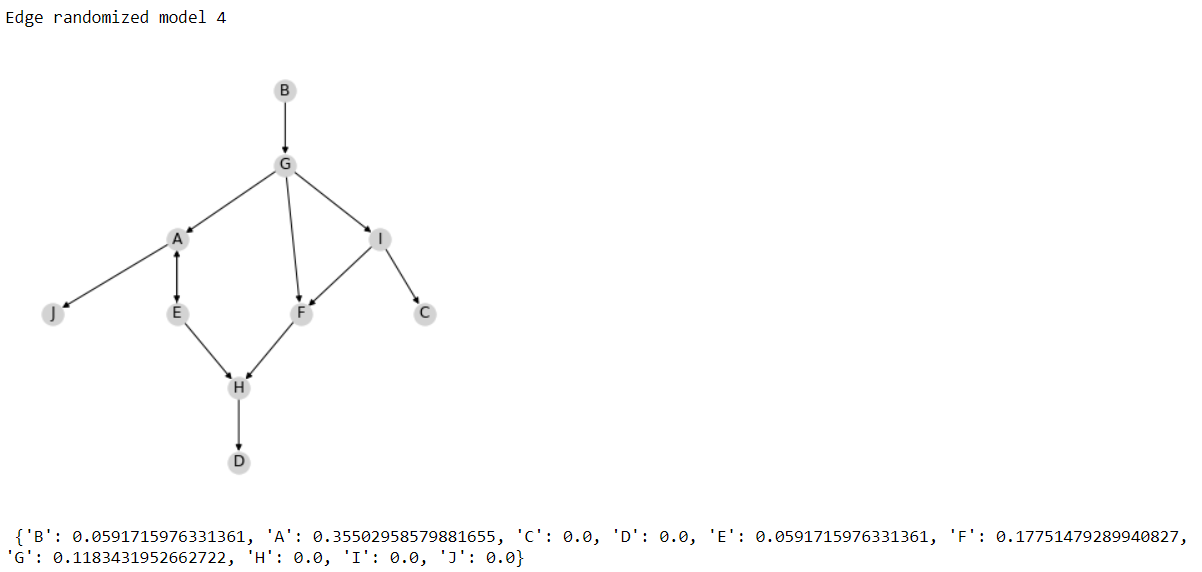
Similarly for other time respecting paths, we can see that it has increasing order of time

**MODULE 5 OUTPUT:**

****

****

****

****

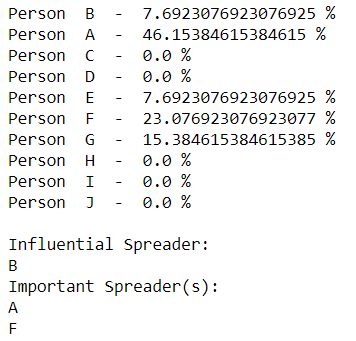
We are using the randomized edges technique here since we want to analyze the importance of each node(persons) to the network. So by randomizing the edges between the nodes, this module studies how the network behaves and calculates a score for each node based on the importance of the node in the network.

Here, we can see that A has maximum score and H,I,J have minimum score.

The score given is correct and it can be verified by looking at the network. Since A has spreaded the message to many people, A has maximum score.

Note that this module gives this score by studying various randomized models. Here we are using 4 randomized models.

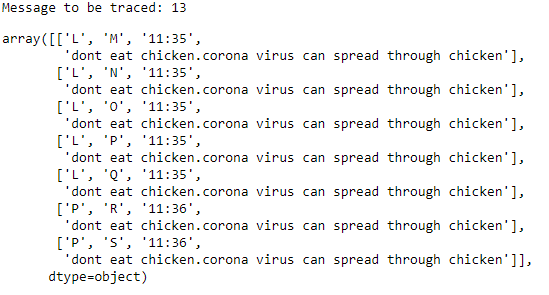
**MODULE 6 OUTPUT:**

****

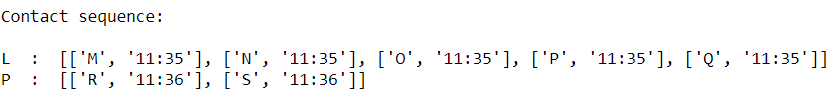
This module calculates the percentage of contribution of each person to the temporal network by converting the scores given by module 5 to percentage. Then this modules displays influential and important spreaders. As it can be seen in the result, B is the person who started this message and hence he is the influential spreader. Here we are displaying the spreaders who have >20% contribution to the network as important spreaders.

**TRACE 13**

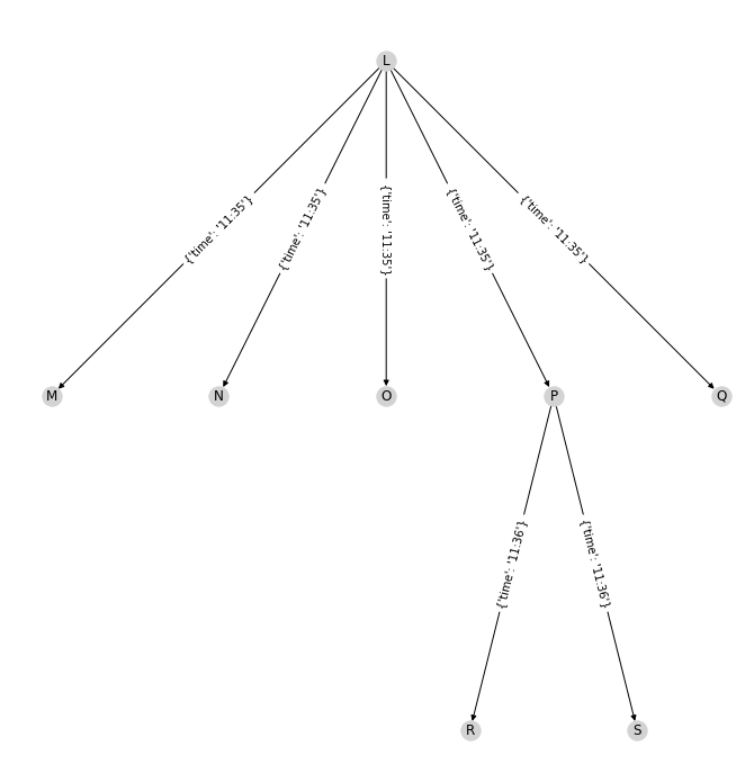
**MODULE 1 OUTPUT:**

****

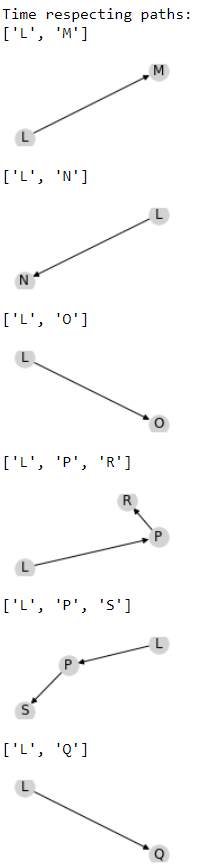
**MODULE 2 OUTPUT:**

****

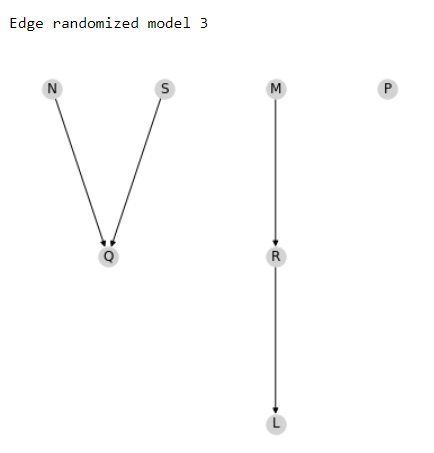
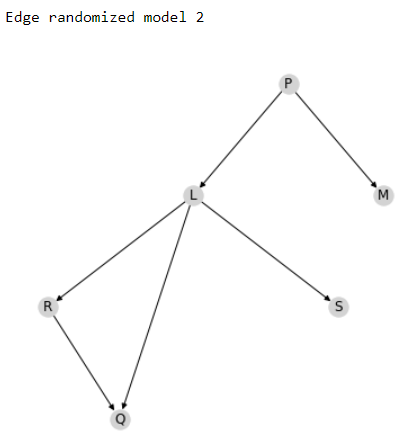
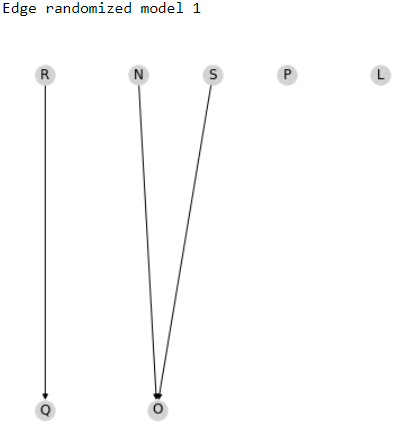
**MODULE 3 OUTPUT:**

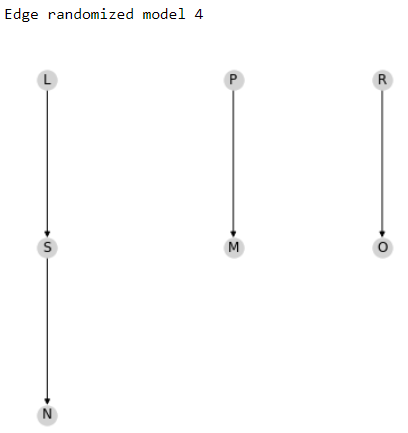
****

**MODULE 4 OUTPUT:**

****

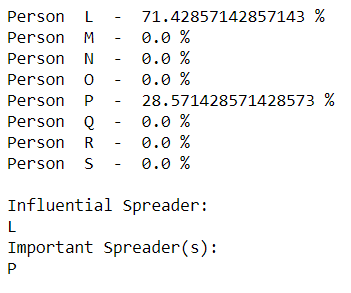
**MODULE 5 OUTPUT:**

****

****

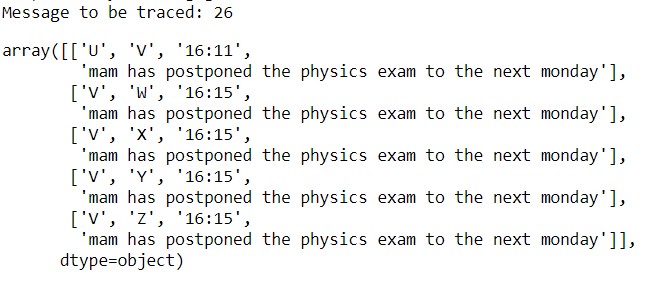
****

**MODULE 6 OUTPUT:**

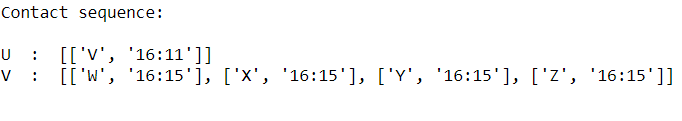
****

**TRACE 26**

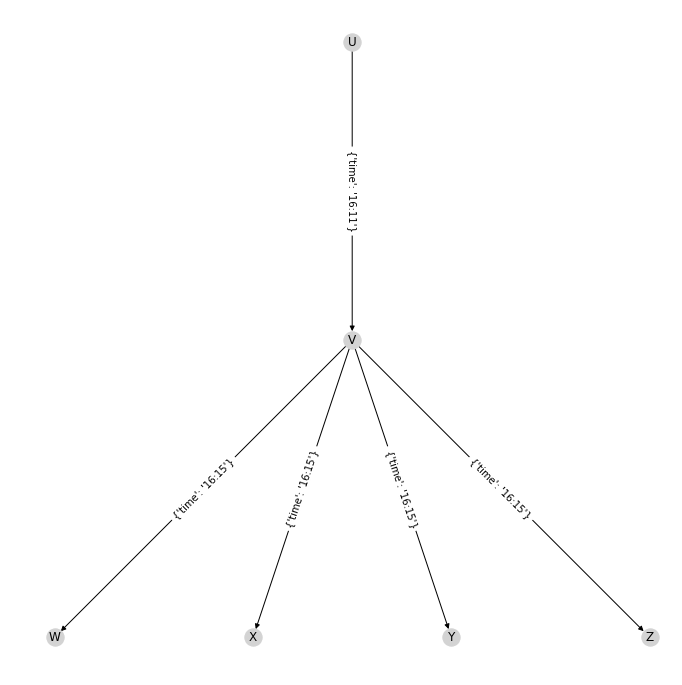
**MODULE 1 OUTPUT:**

****

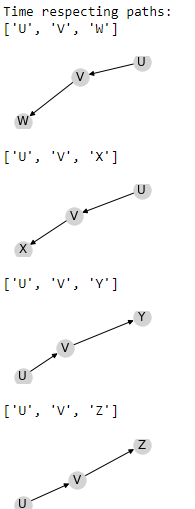
**MODULE 2 OUTPUT:**

****

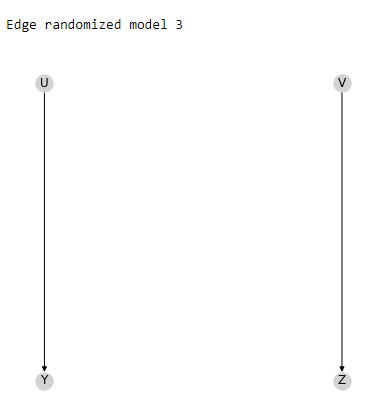
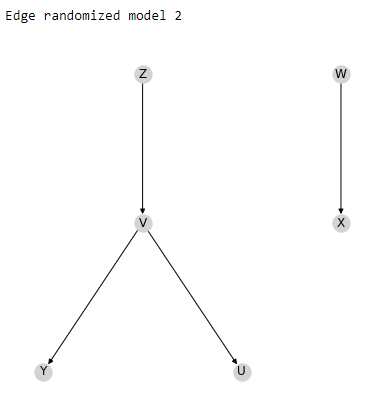
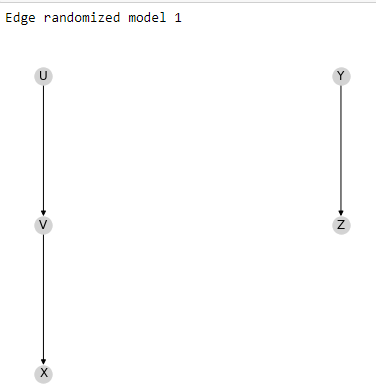
**MODULE 3 OUTPUT:**

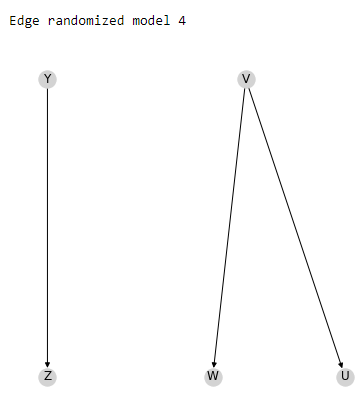
****

**MODULE 4 OUTPUT:**

****

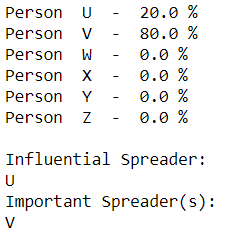
**MODULE 5 OUTPUT:**

****

****

****

**MODULE 6 OUTPUT:**

****

**TABULATION**

**MODULE 1:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TRACE 7** | **TRACE 13** | **TRACE 26** |
| **MESSAGE** | Ronaldo is taking retirement next week | Don’t eat chicken.corona virus can spread through chicken | Mam has postponed the physics exam to monday |
| **NO OF TIMES SENT** | 13 | 7 | 5 |
| **NO OF PERSONS** | 10 | 8 | 6 |
| **PERSONS INVOLVED** | A,B,C,D,E,F,G,H,I,J | L,M,N,O,P,Q,R,S | U,V,W,X,Y,Z |
| **ORIGIN TIME** | 06:07 | 11:35 | 16:11 |

**MODULE 2:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TRACE 7** | **TRACE 13** | **TRACE 26** |
| **CONTACT SEQUENCE** | U -> V(16:11)  V -> W(16:15),  X(16:15),  Y(16:15),  Z(16:15) | L -> M(11:35),  N(11:35),  O(11:35),  P(11:35),  Q(11:35)  P -> R(11:36),  S(11:36) | U -> V(16:11)  V -> W(16:15),  X(16:15),  Y(16:15),  Z(16:15) |

**MODULE 3:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **TRACE 7** | | | **TRACE 13** | | | **TRACE 26** | | |
| **O**  **U**  **T**  **P**  **U**  **T** | **SOURCE**  B  A  A  A  A  A  A  F  F  F  G  E  G | **DEST**  A  C  D  E  F  G  H  E  H  G  I  I  J | **TIME**  06:07  06:08  06:08  06:08  06:08  06:08  06:08  06:09  06:09  06:09  06:10  06:10  06:10 | **SOURCE**  L  L  L  L  L  P  P | **DEST**  M  N  O  P  Q  R  S | **TIME**  11:35  11:35  11:35  11:35  11:35  11:36  11:36 | **SOURCE**  U  V  V  V  V | **DEST**  V  W  X  Y  Z | **TIME**  16:11  16:15  16:15  16:15  16:15 |

**MODULE 4:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TRACE 7** | **TRACE 13** | **TRACE 26** |
| **NO OF PATHS** | 10 | 6 | 4 |
| **PATHS** | B->A->C,  B->A->D,  B->A->E->I,  B->A->F->E->I,  B->A->F->H,  B->A->F->G->I,  B->A->F->G->J,  B->A->G->I,  B->A->G->J,  B->A->H | L->M,  L->N,  L->O,  L->P->R,  L->P->S,  L->Q | U->V->W,  U->VX->,  U->V->Y,  U->V->Z |
| **ORDER** | INCREASING TIME | INCREASING TIME | INCREASING TIME |

**MODULE 5:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TRACE 7** | **TRACE 13** | **TRACE 26** |
| **NO OF RANDOMIZED MODELS** | 4 | 4 | 4 |
| **SCORES FOR EACH PERSON** | A: 0.35502958579881655,  B: 0.0591715976331361,  C: 0.0,  D: 0.0,  E: 0.0591715976331361,  F: 0.17751479289940827, G: 0.1183431952662722,  H: 0.0,  I: 0.0,  J: 0.0 | L: 1.2755102040816326,  M: 0.0,  N: 0.0,  O: 0.0,  P: 0.5102040816326531, Q: 0.0,  R: 0.0,  S: 0.0 | U: 0.6666666666666666, V: 2.6666666666666665, W: 0.0,  X: 0.0,  Y: 0.0,  Z: 0.0 |
| **NO OF PERSONS WITH SCORE AS 0** | 5 | 6 | 4 |

**MODULE 6:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TRACE 7** | **TRACE 13** | **TRACE 26** |
| **PERCENTAGE** | A - 46.15384615384615 %  B - 7.6923076923076925 %  C - 0.0 %  D - 0.0 %  E - 7.6923076923076925 %  F - 23.076923076923077 %  G - 15.384615384615385 %  H - 0.0 %  I - 0.0 %  J - 0.0 % | L - 71.42857142857143%  M - 0.0 %  N - 0.0 %  O - 0.0 %  P - 28.571428571428573 %  Q - 0.0 %  R - 0.0 %  S - 0.0 % | U - 20.0 %  V - 80.0 %  W - 0.0 %  X - 0.0 %  Y - 0.0 %  Z - 0.0 % |
| **INFLUENTIAL SPREADER** | B | L | U |
| **IMPORTANT SPREADER(S)** | A, F | P | V |

**CONCLUSION**

Analyzing temporal networks has way more applications than one would even imagine.

In a world where the rumour can spread in a rapid rate, it is essential that we study and

analyse temporal networks to identify the spreaders involved in spreading of the rumour. It

is very useful during times of disease spread. Because that’s the perfect time for rumours to

spread faster than expected. Also, during elections times we can analyse the influential

and important spreaders that usually determine the result of the election.

**REFERENCES**

1. BASE PAPER:

<https://drive.google.com/file/d/1cc-Imr9Looh5t-ZH95UAUl5JX7bGoeBG/view?usp=sharing>

1. P. HOLME AND J. SARAMA¨KI, ‘‘TEMPORAL NETWORKS,’’ PHYS. REP., VOL. 529, PP. 97–125, 2012.
2. J.-P. ECKMANN, E. MOSES, AND D. SERGI, ‘‘ENTROPY OF DIALOGUES CREATES COHERENT STRUCTURES IN E-MAIL TRAFFIC,’’ PROC. NAT. ACAD. SCI. USA, VOL. 101, PP. 14333–14337, 2004
3. H. EBEL, H.-I. MIELSCH, AND S. BORNHOLDT, ‘‘SCALE-FREE TOPOLOGY OF E-MAIL NETWORKS,’’ PHYS. REV. E, VOL. 66, 2002, 035103
4. B. VISWANATH, A. MISLOVE, M. CHA, AND F. P. GUMMADI, ‘‘ON THE EVOLUTION OF USER INTERACTION IN FACEBOOK,’’ IN PROC. 2ND ACM WORKSHOP ONLINE SOCIAL NETW., 2009, PP. 37–42.
5. S. A. GOLDER, D. M. WILKINSON, AND B. A. HUBERMAN, ‘‘RHYTHMS OF SOCIAL INTERACTION: MESSAGING WITHIN A MASSIVE ONLINE NETWORK,’’ IN COMMUNITIES AND TECHNOLOGIES 2007, C. STEINFIELD, B. T. PENTLAND, M. ACKERMAN, AND N. CONTRACTOR, EDS. LONDON, U.K.: SPRINGER-VERLAG, 2007, PP. 41–66.