Detecting Rumors And Single Source Of Rumors In Social Network Subgraphs

Project report submitted to the Amrita Vishwa Vidyapeetham in partial fulfilment of the requirement for the Degree of

BACHELOR of TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Anand Patel AM.EN.U4CSE19206
Bharath Schneider AM.EN.U4CSE19213
Manas P P AM.EN.U4CSE19233
Nikhil Varghese AM.EN.U4CSE19039

AMRITA SCHOOL OF COMPUTING
AMRITA VISHWA VIDYAPEETHAM
(Estd. U/S 3 of the UGC Act 1956)
AMRITAPURI CAMPUS
KOLLAM -690525

MARCH 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM
(Estd. U/S 3 of the UGC Act 1956)

Amritapuri Campus

Kollam -690525

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **Detecting Rumors And Single Source Of Rumors In Social Network Subgraphs** submitted by **Anand Patel(AM.EN.U4CSE19206), Bharath Schneider(AM.EN.U4CSE19213), Manas P P(AM.EN.U4CSE19233) and Nikhil Varghese(AM.EN.U4CSE19039))**, in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology in Computer Science and Engineering from Amrita Vishwa Vidyapeetham, is a bonafide record of the work carried out by them under my guidance and supervision at Amrita School of Computing, Amritapuri during Semester 8 of the academic year 2022-2023.

Lekshmi S Nair

Subbulakshmi S

Dr. Prema Nedungadi
Chairperson
Dept. of Computer Science & Engineering

Reviewer

Place : Amritapuri
Date : March 23, 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

AMRITA VISHWA VIDYAPEETHAM
(Estd. U/S 3 of the UGC Act 1956)

Amritapuri Campus

Kollam -690525

DECLARATION

We, **Anand Patel(AM.EN.U4CSE19206), Bharath Schneider(AM.EN.U4CSE19213)), Manas P P(AM.EN.U4CSE19233) and Nikhil Varghese(AM.EN.U4CSE19039)** hereby declare that this project entitled **Detecting Rumors And Single Source Of Rumors In Social Network Subgraphs** is a record of the original work done by us under the guidance of **Lekshmi S Nair**, Dept. of Computer Science and Engineering, Amrita Vishwa Vidyapeetham, that this work has not formed the basis for any degree/diploma/associationship/fellowship or similar awards to any candidate in any university to the best of our knowledge.

Place : Amritapuri

Date : March 23, 2023

Signature of the student

Signature of the Project Guide

Abstract

Social Networks are a broad platform for sharing information, images, videos, and voice clips. Information spreads rapidly through social networks, but the authenticity of the information is of significant concern. Prior verification for authenticity is not done as the information propagates through the networks, and there are higher chances for the information to be false or misleading. Such incorrect information is termed Rumors. Identifying rumors has gained more attention to restrain the spread of rumors and reduce their influence. It is becoming increasingly important in today's age of social media and online communication where false information or rumors can spread rapidly and have significant consequences ranging from minor misinformation to major public panic or damage to reputation.In this paper, we deal with the identification of rumors and the source of rumors in snapshots of the social network Twitter. The subgraphs obtained from Twitter have an average of 45 nodes and 145 edges and represent smaller interaction networks. We only deal with nodes that are actively interacting and not the passive nodes which could be neighbors of these nodes. We explore graph properties to identify the source of information propagated in the subgraph and also aim to determine the presence of rumors in the subgraphs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Rumors In Social Networks

With the emergence of social networks, there has been a tremendous change in the way we interact, communicate, and share information. Various platforms like Facebook, Twitter, Snapchat, and Instagram help people worldwide communicate with each other. Despite these advantages, social networks raised many concerns regarding security, privacy, and the spread of misinformation. A rumor is a piece of information or a story that is spread around without any clear evidence or proof of its accuracy. Rumors often involve sensational or controversial topics, and they can spread more rapidly through social media than any other form of communication. Therefore it becomes crucial to come up with methods that can be used to identify rumors in networks and also detect the source of these rumors so that one can appropriately block the rumor- spreading node or inform the neighbors of the presence of rumor, so they can be alerted and thus reducing the effect of these rumors.The goal of rumor detection is to accurately identify

and classify rumors as quickly as possible, using a variety of techniques and approaches. This field draws on a range of disciplines, including natural language processing, machine learning, network analysis, and social psychology. All these techniques will help identify the patterns in rumor prop- agation, such as the source and the channels through which they spread, and helps assess the credibility of information.

1.2 Beneficiaries of Rumor and Source Detection

All users on social media can benefit from rumor detection as it improves awareness of the user who could otherwise have been misled into believing things that are fake. This will help people individually as the rumors might be targeted at certain individuals or it can also benefit society at large as well. Institutions can benefit from such false information being detected and users alerted. This can include institutions like banks or publicly traded companies as these are easy targets to produce the desired effect of the people spreading the rumor.

1.3 Tasks Performed

In this project, we explore techniques with which we can detect the presence as well as the source of rumors in subgraphs of a network. These subgraphs contain the spread of information, which could be rumor or non-rumor and represent the snapshot of the nodes involved and their interactions across a certain period of time. We look into single-source detection of rumor in each of these subgraphs using various graph properties and perform a comparative

analysis of the same. We dive into various properties like the centrality measures, eccentricity, and cycles to detect the rumor source node. This is followed by a short discussion on the follow-up application of informing the neighboring nodes of this source of rumor. We also discuss the use of machine learning and graph neural networks in predicting the presence of rumors in the subgraph. We explore various Machine Learning algorithm along with dimensionality reduction techniques like PCA to model our problem as a binary classification problem. We also study the use of Graph Convolution Networks for our problem. The subgraphs only contain active nodes, which refer to the nodes that are interacting. This means that all the nodes in the neighborhood of the rumor node are not featured. Therefore we do not deal with passive nodes in this paper and the focus lies only on working on the interactions of active users in the subgraph. Also, there is no element of time as we only use the subgraphs as static graphs

1.4 Scope of the project

All the methods studied by us in this project can be leveraged for application on a much larger dataset, so that we can ascertain the capability of the methods discussed in being able to detect the rumors and their sources in real-life social networking sites like Twitter. We have not discussed multi-source rumor detection. However, this can be easily extended and included in the model of rumor and source detection that is studied in this project.

Chapter 2

Related Work

2.1 Content Based Approach

Existing approaches for fake news detection can be divided into three main categories, based on content, social context, and propagation [9-12]. Content-based approaches, which are used in the majority of works on fake news detection, rely on linguistic (lexical and syntactical) features that can capture deceptive cues or writing styles. The main drawback of content-based approaches is that they can be defied by sufficiently sophisticated fake news that does not immediately appear as fake. Furthermore, most linguistic features are language-dependent, limiting the generality of these approaches.

2.2 Social Context Based Approach

Social context features include user demographics (such as age, gender, education, and political affiliation), social network structure (in the form of connections between users such as friendship or follower/followee relations) and user reactions (e.g. posts accompanying a news item or likes) [13-15].

2.3 Propagation Modelling Based Approach

Propagation-based approaches are perhaps the most intriguing and promising research direction based on studying the news proliferation process over time. It has been argued that the fake news dissemination process is akin to infectious disease spread and can be understood with network epidemics models. There is substantial empirical evidence that fake news propagate differently from true news [16] forming spreading patterns that could potentially be exploited for automatic fake news detection. By virtue of being content-agnostic, propagation-based features are likely generalizes across different languages, locales, and geographies, as opposed to content-based features that must be developed separately for each language. Furthermore, controlling the news spread patterns in a social network is generally beyond the capability of individual users, implying that propagation-based features would potentially be very hard to tamper with by adversarial attacks.

2.4 Brief Discussion On Rumor Modelling

Rumor propagation has been widely studied in the literature. It has mainly been dealt with as a classification problem. Most of the literature deals with the individual content of each post and tries to classify it as either a rumor or non-rumor. This approach may also make use of features on social media that includes interaction by other users with the original post. For instance, considering the Twitter platform, one can like, retweet or comment on a post. These can be captured by a Twitter API and be included in the model. However, using a language processing system to identify rumors has its own issues as the texts containing misinformation are short, work with insinuation rather than explicitly stating a false claim, or resemble other postings that

deal with the same topic ironically. Therefore it becomes necessary to extend the study beyond a language processing system and study the characteristics of the users relationship with each other as well as the model that best represents the diffusion of misinformation.

There are many propagation models that can be used for this purpose. This includes models like Susceptible-Infected-Recovered(SIR) model, Independent Cascade(IC) model, Linear Threshold Model, Generalized Linear Threshold Model, etc.

In phase 1, we aimed to model the spread of rumours in a network based on entropy. For this, we analysed multiple information spread models like SIR, IC, etc. We also tried to create a model that can identify rumours and block the node that is spreading misinformation. While we had implemented a model that performed this function using entropy, we lacked ground truth to validate it and hence we couldn't proceed further with it.

Chapter 3

Dataset Details

3.1 Wico Graph Dataset

We use the Wico Graph Dataset [6], which is a labeled dataset of Twitter subgraphs. 3,000 manually extracted subgraphs were distinguished into three categories. The graphs are directed. Each edge is directed from the source node/user to the target node/user. Each node is given an id that is unique across all subgraphs and also reveals the number of friends and followers of the node/user. Friends and followers will be used in the Graph Convolution Network for graph classification. There are two files associated with each subgraph a)nodes b)edges. 'nodes' provide information about nodes, which node is the source node, and the friends and followers of the nodes.

'edges' file provides information on the interactions with each interaction directed from the source node to target node. Of the three categories of tweets, two categories include rumor tweets and the third category refers to tweets without rumors.

The dataset consists out of three classes.

- 5G-Corona Conspiracy: This class contains all tweets that claim or

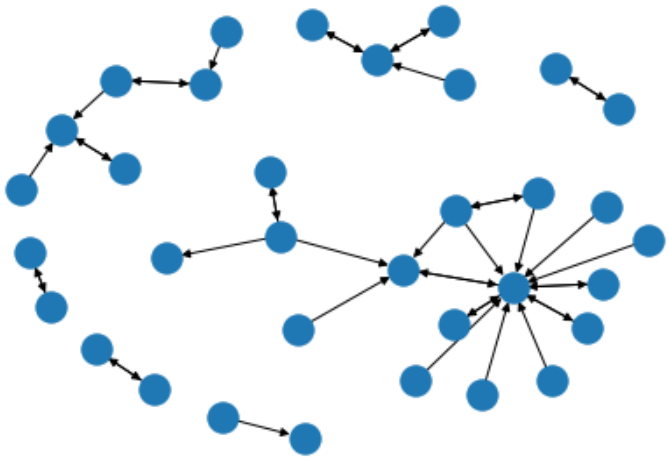

Figure 3.1: A sample subgraph from the dataset

insinuate some deeper connection between COVID-19 and 5G, such as the idea that 5G weakens the immune system and thus caused the current corona-virus pandemic, or that there is no pandemic and the COVID-19 victims were actually harmed by radiation emitted by 5G network towers. The crucial requirement is the claimed existence of some causal link.

- Other Consparacy: This class contains all tweets that spread conspiracy theories other than the ones discussed above. This includes ideas about an intentional release of the virus, forced or harmful vaccinations, or the virus being a hoax.

- Non-Conspiracy: This class contains all tweets not belonging to the previous two classes. Note that this also includes tweets that discuss COVID-19 pandemic itself, tweets claim that 5G is not proven to be absolutely safe or even can be harmful without linking it to COVID-19, as well as tweets claiming that authorities are pushing for the installation of 5G while the Publicis distracted by COVID-19. In addition, tweets pointing out the existence of conspiracy theories or mocking them fall

Graph Property Averages	5G - Conspiracy	Other Conspiracy	Non-Conspiracy
Nodes	35	52	45
Edges	142	198	132
Degree	1.16	1.13	1.22
Closeness Centrality	0.29	0.25	0.19
Core Number	3.9	3.9	3.0
Clustering Coefficient	0.23	0.24	0.19
Transitivity	0.23	0.24	0.27

Table 3.1: Properties of the Dataset used.

into this class since they do not spread the conspiracy theories by inciting people to believe in them.

We have 413 subgraphs in the first category, 598 in the second class, and 2502 subgraphs in the second class. Clearly, there is a class imbalance and hence we need to use appropriate scores during evaluation, preferably, precision and recall. The average subgraph contains nearly 45 nodes/users and 145 edges, with each edge being directed and pointing from the source of the tweet to the target node/user which interacts with it. Hence, we are mainly dealing with smaller size snapshots/subgraphs rather than a large network. Previous research has often dealt with larger networks or networks varying with time. We summarize the features of the graphs in each class in Table 3.1. Also, a sample subgraph from the dataset is depicted in Figure 3.1.

Chapter 4

Problem Definition

4.1 Social Network Analysis

- Perform Social Network Analysis to study the characteristics of graphs containing misinformation and those that don't.
- Social network analysis (SNA) tools are used to analyze patterns of relationships among people in groups. They are useful for examining the social structure and interdependencies (or work patterns) of individuals or organizations. SNA involves collecting data from multiple sources (such as surveys, emails, blogs and other electronic artifacts), analyzing the data to identify relationships, and mining it for new information—such as the quality or effectiveness of a relationship.

4.2 Graph Classification

- Classify sub-graphs extracted from Twitter into those containing misinformation and those that do not contain it.

- A graph classification task predicts an attribute of each graph in a collection of graphs. For instance, labelling each graph with a categorical class (binary classification or multiclass classification), or predicting a continuous number (regression). It is supervised, where the model is trained using a subset of graphs that have ground-truth labels.

- Graph classification can also be done as a downstream task from graph representation learning/embeddings, by training a supervised or semi-supervised classifier against the embedding vectors.

4.3 Single Source Detection

- Study single-source detection of rumor in each of these subgraphs using various graph properties and perform a comparative analysis of the same.

- Source detection is to find the person or location from where the entities like a viral disease, a virus in the network or a misinformation in the social network started initially. Different domains are represented by a network like the Network of computers, Network of peoples, Social Network, etc. where the source identification has been performed to find the root or origin.

Chapter 5

Single Source Detection of Information Propagation

Each subgraph in our dataset has a single source of information across the three classes. This is in contrast to situations with multi-source graphs. In this paper, we only deal with methods for single-source detection. We explore different approaches to locate the source in each subgraph and present the results in a tabular format. In each of the methods we use, only the nodes contained in the largest weakly connected component are considered. This is based on the assumption that the rumors are spread rapidly and more rigorously in a connected component and then spreads to nodes outside it. Also, this will mean that the source node is present in the connected component. Apart from the assumption mentioned, we will also see that certain methods require us to use the largest weakly connected component only.

5.1 Rumor Centrality

Rumor Centrality is the common measure used to detect the source of rumor. Like most centrality measures the higher the rumor centrality [7], the higher the probability of a node being the source of the rumor. It takes a graph G as input and assigns a positive number or score to each of the vertices. Then the estimated source is the one with the maximal score or rumor centrality. The estimated source is called the rumor center. It can also be considered as the number of propagation paths starting from the source node. In this method, it is not necessary to take the largest weakly connected component as we are calculating the BFS tree at each node. The score is then calculated based on the BFS tree and hence it ensures the avoidance of infinity or other undefined scores that we would come across using other methods mentioned. The steps to calculate the rumor centrality of nodes are as follows:

- Source node is chosen, let it be u.
- Convert the graph into a BFS tree with root u, using the Breadth First Search Technique.
- For the root node u, calculate the rumor centrality as per Equation 1.

$$R(u, G) = \frac{|V|!}{\prod_{w \epsilon V} T_w^u} \tag{5.1}$$

  where $|V|$ is the size of the tree and T_w^u is the size of the subtree of G that is rooted at w and points away from u.

  The node with the maximum rumor centrality will be the source of rumor.

5.2 Jordan Center

Jordan Center consider the nodes in the largest connected component. Jordan center uses eccentricity which is a distance measure and hence we would then have infinite distances for the nodes due to the presence of nodes outside its connected component. Rumors tend to spread to nodes completely outside a local connected community on social media because of recommendations. Once we take the largest connected component, for each vertex v, we calculate the eccentricity which refers to the largest distance from the node to any other node in the component. After which, we find the node with the least eccentricity. This node is the Jordan center of the graph and is estimated to be the source node in this method. Equation 2 sums up the Jordan Center.

$$J(G) = argmin_u(argmax_{w \epsilon V - \{u\}} dist(u, w)) \tag{5.2}$$

where V is the set of all vertices, dist(u,w) is the shortest path between vertices u and w

5.3 Max Cycle Presence

In this method, we assume that rumors tend to reach multiple nodes and most of them have a direct interaction with the source node. Let's consider a scenario in which we have three nodes a, b, and s. Let s be the source of the rumor. If node a interacts with the post of s and gets infected it could spread it to node b. So, there is an interaction between nodes a and b. Now it is quite likely that node b has already interacted with the post of s or could do so after it gets infected as well. This will lead to a circle between

the three. In order to implement this method, we first convert the largest connected component into an undirected graph. This is because we are only concerned with the presence of a cycle between nodes without worrying about the direction in which the interactions took place. Then we find the cycle basis of the connected component. A cycle basis of an undirected graph is a set of simple cycles that forms a basis of the cycle space of the graph. That is, any cycle in the graph can be constructed from the fundamental cycles. The fundamental cycle basis of a graph is calculated from a minimum spanning tree of the graph. For each edge that is not in the minimum spanning tree, there exists one fundamental cycle which is composed of that edge, its end nodes, and the path in the minimum spanning tree that connects them. Together, they are a compact representation of the set of all cycles,Then we look at the node that is involved in the most number of cycles and estimate it to be the source node. This is directly based on our assumption that nodes are mainly interacting with our source node in cycles. We will use the closeness centrality as a tie-breaker in case there is a small number of cycles, let's say one cycle in the cycle basis, and need to choose the right node. We will discuss the use of closeness centrality in the next section.

5.4 Closeness Centrality

This method is based purely on a graph property.For each node in the subgraph, we calculate the closeness centrality and estimate the source as the node with the highest closeness centrality. Once again we only take the largest weakly connected component. If we take the entire subgraph, it is quite likely that the average distance of a node to other nodes will become infinite as there are disconnected components. Closeness centrality is a way

of detecting nodes that are able to spread information very efficiently through a graph.Closeness centrality is the inverse of this average distance and hence higher the closeness centrality, the closer the node is to the other nodes in the network. We expect a source node to have this feature most other nodes end up interacting with the source node in the largest weakly connected component and then further spread the rumor to their neighbors. The closeness centrality of a node is calculated using Equation 3.

$$C(x) = \frac{N-1}{\sum_{y \epsilon V} d(x,y)} \tag{5.3}$$

Where N is the number of nodes and d(x, y) is the distance or path length between the two nodes. Here, we are normalizing the score. The subtraction with 1 in the numerator can be avoided for large networks, but since we are using smaller networks we will retain it.

5.5 Estimation using Voronoi Cells

This method is a modification of [8] which was actually used to detect multiple sources in a network. We adapt it to detect a single source instead.

In this method, we attempted to improve the predictions over the closeness centrality method. We first begin by taking the largest weakly connected component and finding the two nodes with the highest closeness centrality.

After this, we consider these nodes as the center nodes for the Voronoi partition. If C is a set of nodes in the graph and c is an element of C, the Voronoi cell centered at a node c is the set of all nodes v that are closer to c than to any other center node in C with respect to the shortest-path distance metric. Then, we take the center node with the largest Voronoi cell as the estimated source node.

5.6 Betweenness Centrality

Betweenness centrality (equation 4) is a measure that captures the extent to which a certain vertex lies on the shortest paths between other vertices. In other words, it helps identify individuals who play a "bridge-spanning" role in a network. We try to find out if the source node can be distinguished by a large information flow through it compared to other nodes.

Similar to the method using closeness centrality, we estimate the source node using the betweenness centrality and choose the node with the largest value.

$$B(v) = \sum_{s \neq v \neq t} \frac{\delta_s t(v)}{\delta_s t} \tag{5.4}$$

Where $\delta_s t$ is the total number of shortest paths from s to t, and $\delta_s t(v)$ is the total number of shortest paths passing through v.

5.7 Eigenvector Centrality

Eigenvector Centrality is an algorithm that measures the transitive influence of nodes. It indicates how important an entity is, based on how important the entities in contact with it are. In other words, the eigenvector centrality reveals which entities are better connected to important entities in the network. It does not matter how many connections an entity establishes, like the degree centrality, or what "linking power" these connections have, like betweenness centrality, or how easy they make to reach (or be reached by) others, as in the closeness centrality. What matters is the importance of the other entities connected to the entity that you are studying. Relationships originating from high-scoring nodes contribute more to the score of a node

than connections from low-scoring nodes. A high eigenvector score means that a node is connected to many nodes that themselves have high scores. Therefore, the node with the highest eigenvector centrality is the most influential among all the high-scoring nodes and is hence estimated to be the source.

We use the eigenvector centrality to estimate the source of information in the subgraphs. Similar to the previous method, the node with the highest eigenvector centrality is considered to be the source node. This gives us an idea if the source node is surrounded by influential nodes as well and if so it would be the most influential node as per this measure. Therefore, a source node is assumed to have the maximum influence in our subgraphs.

5.8 Degree Centrality plus Closeness Centrality

The degree is a simple centrality measure that counts how many neighbors a node has. If the network is directed, we have two versions of the measure: in-degree is the number of incoming links; out-degree is the number of outgoing links. Typically, we are interested in in-degree, since in-links are given by other nodes in the network, while out-links are determined by the node itself. This is also consistent with our assumption that a lot of nodes that are infected interact with the source node after coming across a rumor.

In this method, we take the nodes with the highest in-degree centrality. Then these nodes are compared for their closeness centralities and the node with the highest closeness centrality is taken as the source node. Therefore, we are looking for nodes that receive a lot of interactions and are also capable of spreading rumors quickly in the network as indicated by a high closeness

centrality.

5.9 Eccentricity plus Closeness Centrality

The eccentricity of a graph vertex v in a connected graph component G is the maximum graph distance between v and any other vertex u of G. For a disconnected graph, all vertices are defined to have infinite eccentricity. We take the nodes with the least eccentricities and check them for maximum closeness centrality.

Chapter 6

Single Source Detection - Results

6.1 Findings

The results of the application of the algorithms are shown in table 6.1.

- We see that closeness centrality in itself is the best measure to estimate the source in a small subgraph of interactions when compared to all the other methods used. This indicates the possibility that the source node possesses the property of being easily able to reach all the other nodes in the largest component. The spread of rumors beyond the largest connected component does not seem to have a great effect on the final result.

- Eccentricity does not seem to be a useful measure. On the contrary, it registered poor performance when used in the Jordan center method. Also, it clearly reduced the efficiency down to zero of closeness centrality which otherwise did a very good job by itself. Therefore, it seems

rather clear that a source node is not required to have the lowest eccentricity, which means it need not have the shortest maximum distance to other nodes in the largest component.

- Eigenvector Centrality gave a decent result. This could indicate the possibility of the presence of spreader nodes near the source nodes that are influential in spreading the rumor. Therefore, we need to inform the neighbors of all the nodes directly in contact with the source of possible rumor spread.

- Degree Centrality itself did not have any great impact in adding to the efficiency of closeness centrality. But because degree centrality was used to shortlist the nodes to be checked for closeness centrality, we could infer that the nodes that have a possibility of spreading rumors also tend to have a higher degree in the interaction subgraph captured over time as a snapshot.

- The method using the presence of source nodes in many cycle bases did not give very good results compared to closeness centrality. While this probably indicates the inefficiency of this method, we still may not be able to conclude if source nodes really are included in a large number of cycles. This is because the actual number of cycles can be much larger than the number of cycles on a cycle basis. This also poses an issue for computation as the smaller-sized graphs in the wico dataset needed large amounts of time, making it infeasible for real-life situations. Therefore, cycles themselves may not be a very useful feature in detecting the source.

- Betweenness centrality is not a good measure for spotting sources of information. It indicates that the ability of a node to control the flow

of information across it does not necessarily mean that it is the source.

- Interestingly, using Voronoi partitions to check for information seems to work to some extent on the rumor subgraphs, but gave an accuracy of zero for non-rumor subgraphs. One can further study if Voronoi partitions are simply incapable of detecting the source of information in non-rumor class or if non-rumor sources behave differently in this respect when compared to rumor sources.

- Also, rumor centrality gave a very low accuracy. This could indicate that the metric is not suitable for small sized graphs.

Method	5G Class Accuracy	Other Class Accuracy	Non-Conspiracy Class Accuracy
Rumor Centrality	27	21	18
Jordan Center	34	34	50
Max Cycle Presence	41	57	51
Closeness Centrality	72	79	81
Estimation Using Voronoi Cells	56	52	0
Betweenness Centrality	49	48	37
Eigenvector Centrality	60	63	56
Degree + Closeness	72	80	81
Eccentricity + Closeness	7	3	3

Table 6.1: Accuracy of each method in estimating the source of subgraphs.

6.2 Application

We see that sources of information whether it be rumor or non-rumor can be estimated using basic graph properties. Therefore, we will take the approach of first detecting the presence of rumors in the subgraph followed by an estimation of the source. Hence, we will first detect the presence of rumors in these small subgraphs using machine learning algorithms, which will be discussed in the next section. After this, we estimate the source node and attempt to inform the neighbors of the infected nodes about the presence of rumors in the network. This will help in alerting these users and make them less susceptible to being influenced by the rumors.

Chapter 7

Rumor Detection In Subgraphs Using Machine Learning

7.1 Graph Classification

As mentioned in the previous section, there wasn't an explicit way to find out the presence of rumor and the source itself based on the method we discussed. The estimation of sources by our methods worked equally well for all three categories. This brings us to the need of classifying the subgraphs into those that contain rumor and those that contain non-rumor. The problem can thus be modeled as a binary classification problem followed by source detection for those subgraphs that have been identified as rumor-containing.

Hence, We perform the task of graph classification on the subgraphs to detect the presence of rumors in them.

7.2 Using graph features to train machine learning classification models.

Data extraction and pre-processing: The data is present in the form of graphs. We will convert all the graphs into a tabular format that contains the features of the graph along with the class label of 1 for graphs containing rumors and 0 for those not containing rumors. We use networkx to extract the features of graphs and store them in a pandas data frame. Some graphs may have infinity or Nan for a few features. These values are replaced by the mean values in each category. We use 11 features extracted from the graphs. These features include:

- Nodes/Size: The number of network members in a given network. In our case it is the count of the number of users in the subgraph.
- Edges: This corresponds to the number of interactions in the subgraph.
- Degree Centrality: The degree centrality for a node v is the fraction of nodes it is connected to. The degree centrality values are normalized by dividing by the maximum possible degree in a simple graph n-1 where n is the number of nodes in G. We take the mean of all the degree centralities of nodes present in the graph. It is the normalized degree of the nodes.
- Closeness Centrality: Closeness centrality of a node u is the reciprocal of the average shortest path distance to u over all n-1 reachable nodes. The closeness centrality uses inward distance to a node, not outward. The average of this value over all the nodes in the subgraph is taken.
- Eigenvector Centrality: Eigenvector centrality computes the centrality

of a node based on the centrality of its neighbors. The eigenvector centrality for node i is the i-th element of the vector defined by the equation $Ax = \lambda x$ where A is the adjacency matrix of the graph G with eigenvalue λ. The average of eigenvector centrality over all the nodes is taken for each subgraph.

- Core number: We take the average core number of nodes in the subgraph. The core number of a vertex is the largest value of k of a k-core containing that node.

- Average Clustering Coefficient: Clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together. The local clustering coefficient of a vertex (node) in a graph quantifies how close its neighbours are to being a clique (complete graph). The average clustering coefficient of the subgraph.

- Transitivity: The graph transitivity is the fraction of all possible triangles present in G. Possible triangles are identified by the number of "triads" (two edges with a shared vertex). It is the overall probability for the network to have adjacent nodes interconnected, thus revealing the existence of tightly connected communities (or clusters, subgroups, cliques). The average value of transitivity of nodes in the subgraph.

- Square Clustering Coefficient: The squares clustering coefficient for nodes returns the fraction of possible squares that exist at the node. The mean of this value of all the nodes.

- Radius, Diameter: The diameter D is given by the maximum eccentricity of the set of vertices in the network and, analogously the radius R can be defined as the minimum eccentricity of the set of vertices.

We also include the average radius and diameter of the nodes in the subgraph to train our ML models.

7.3 Method

- One vs One Classification: We model our problem as a one vs one classification. We have three categories of data. Two of them for rumors and the third one for non-rumors. Since we want to avoid a model that has been trained purely on one dataset, we first train and test models in two situations. First, we work on the 5G-Conspiracy vs Non-conspiracy problem and look for the best models. Then we work on Other-Conspiracy vs Non-Conspiracy classification and compare the two problems to see which models gave the better results in either case. We choose the model that has done best and then use that model itself to train and test on the final combined Conspiracy vs Non-Conspiracy classification problem.

- Hyperparameter Tuning using Grid Search: Grid search is a tuning technique that attempts to compute the optimum values of hyperparameters. It is an exhaustive search that is performed on the specific parameter values of a model. The parameters selected are those that maximize the score of the left out data.

- Cross-Validation: We use five-fold stratified cross-validation to train and evaluate our models. In this approach, called k-fold CV, the training set is split into k smaller sets (other approaches are described below, but generally follow the same principles). The following procedure is followed for each of the k “folds”:

  - A model is trained using the k-1 folds as training data.
  - The resulting model is validated on the remaining 1 part of the data.

  The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. This approach can be computationally expensive but does not waste too much data. Also since there is a class imbalance, we use the stratified K-fold. This variation of K-Fold returns stratified folds. The folds are made by preserving the percentage of samples for each class. Therefore, we do not have splits that contain only the majority class.

- Dimension Reduction: We use Principal Component Analysis to reduce the dimension of data. This method preserves as much variance as possible from the original data. We reduce the dimensions to 7 as more than ninety percentage of information is captured at that number for all three categories of information that we have.

- Models Used: We make use of

  - Support Vector Machine (SVM): Support vector machines are a set of supervised learning methods used for classification, regression, and outliers detection. A simple linear SVM classifier works by making a straight line between two classes. That means all of the data points on one side of the line will represent a category and the data points on the other side of the line will be put into a different category. Also, different kernel functions can be specified for the decision function.
  - Random Forest: A random forest is a meta estimator that fits a

number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

- AdaBoost: An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.
- Logistic Regression: In statistics, the logistic model (or logit model) is a statistical model that models the probability of an event taking place by having the log-odds for the event be a linear combination of one or more independent variables.

Chapter 8

Rumor Detection - Results

8.1 Evaluation Metrics

8.1.1 Accuracy

The most straightforward way to measure a classifier's performance is using the Accuracy metric. Here, we compare the actual and predicted class of each data point, and each match counts for one correct prediction. Accuracy is then given as the number of correct predictions divided by the total number of predictions. From the spam classifier output above, we have 15 correct predictions and 5 incorrect predictions, which gives us an Accuracy of 75 percentage. Accuracy is often used as the measure of classification performance because it is simple to compute and easy to interpret. However, it can turn out to be misleading in some cases. This is especially true when dealing with imbalanced data, a scenario when certain classes contain way more data points than the others.

8.1.2 Precision

Precision explains how many of the correctly predicted cases actually turned out to be positive. Precision is useful in the cases where False Positive is a higher concern than False Negatives. In our case precision is the most appropriate technique for measuring performance as it is important that we do not have false positives. That means we are blocking the users that have not spread rumors.

8.1.3 Recall

Recall is a metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made. Unlike precision which only comments on the correct positive predictions out of all positive predictions, recall provides an indication of missed positive predictions. In this way, recall provides some notion of the coverage of the positive class. In our case this means the fraction of rumor nodes we are able to identify. While the task is to identify as many rumors as possible, precision still has the upper hand because we cannot afford to block innocent users. Hence while recall is a relevant measure we will still focus on precision.

Models	5G vs Non-Conspiracy		Other vs Non-Conspiracy		Non-Conspiracy vs Conspiracy	
	Precision	Recall	Precision	Recall	Precision	Recall
Without PCA						
SVM	0.46	0.15	0.30	0.22	-	-
Logistic	0.21	0.01	0.26	0.01	-	-
Random Forest	0.34	0.13	0.34	0.17	-	-
AdaBoost	0.43	0.07	0.51	0.11	-	-
With PCA						
SVM	**0.60**	**0.12**	**0.60**	**0.07**	**0.62**	**0.20**
Logistic	0.36	0.01	0.20	0	-	-
Random Forest	0.37	0.15	0.36	0.18	-	-
AdaBoost	0.36	0.01	0.20	0	-	-

Table 8.1: Results of Machine Learning Classification Task.

8.2 Results

- The results of the ML algorithms can be found in Table 8.1. We see that only SVM has performed consistently on all the three cases. It had a precision of 0.62 in the final dataset. This refers to the fact that of the classes that it predicted to be a rumor, 0.62 of them where actually rumor classes. While an algorithm to be implemented in a real-world scenario would need a higher accuracy, we believe it can be improved with a more extensive dataset. However, the algorithms had a low recall scores. This means that it was able to identify only a small proportion of actual rumors. While it may not be a major issue to not have a recall value that is very high, this can still be improved tremendously. In the next section we use a Graph Neural Network Architecture to model the same situation.

Chapter 9

Rumor Detection In Subgraphs Using Neural Network

9.1 Using Graph Convolution Network to perform the classification with the graph as input.

In this model, we use Graph Convolution Networks to classify whole graphs into two different classes, rumors, and non-rumors. The subgraphs are homogeneous, which means all the nodes are of the same type. The nodes have two features obtained from the dataset. The friends and followers count represents the number of friends and followers of the user associated with the node. This is not the in-degree and out-degree of the nodes. They are consistent for each user across the whole dataset.

We use the Stellar-Graph library, a library built on TensorFlow and Keras to build and train our model. A graph classification task predicts an attribute of each graph in a collection of graphs. For instance, labelling each graph

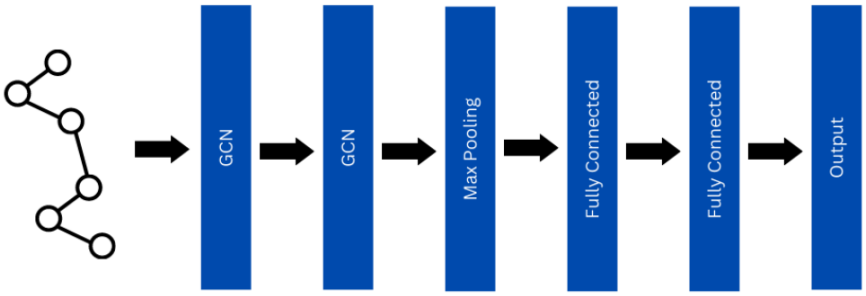

Figure 9.1: GCN Architecture

with a categorical class (binary classification or multi-class classification), or predicting a continuous number (regression). It is supervised, where the model is trained using a subset of graphs that have ground-truth labels.

9.1.1 Graph Convolution Network

The input is the graph represented by its adjacency and node features matrices. The first two layers are Graph Convolutional with each layer having 64 units and relu activation. The next layer is a mean pooling layer where the learned node representation is summarized to create a graph representation. The graph representation is input to two fully connected layers with 32 and 16 units respectively and relu activation. The last layer is the output layer with a single unit and sigmoid activation. The graph architecture is shown in Figure 9.1. This architecture gave an accuracy of 71.3% in repeated k-fold cross-validation with k = 10.

In order to improve the GCN model, it is required to have a much larger dataset. While we have discussed the architecture in this paper, a larger dataset will be required to fine-tune it and reduce any overfitting that might have occurred.

Chapter 10

System Design

The input to our task will be the subgraph/snapshot of the interaction that is taking place on the network. We then take the subgraph and extract the graph features associated with each subgraph. After this, the data of the graph features are used to classify the subgraph as one with rumor or one without rumor. If the subgraph is found to have no rumors then we end the process. Otherwise, we take the subgraph and find the single source of rumor that is present in the subgraph. We then alert the neighboring nodes about the possibility of rumors being spread in their networks by the identified user/source.

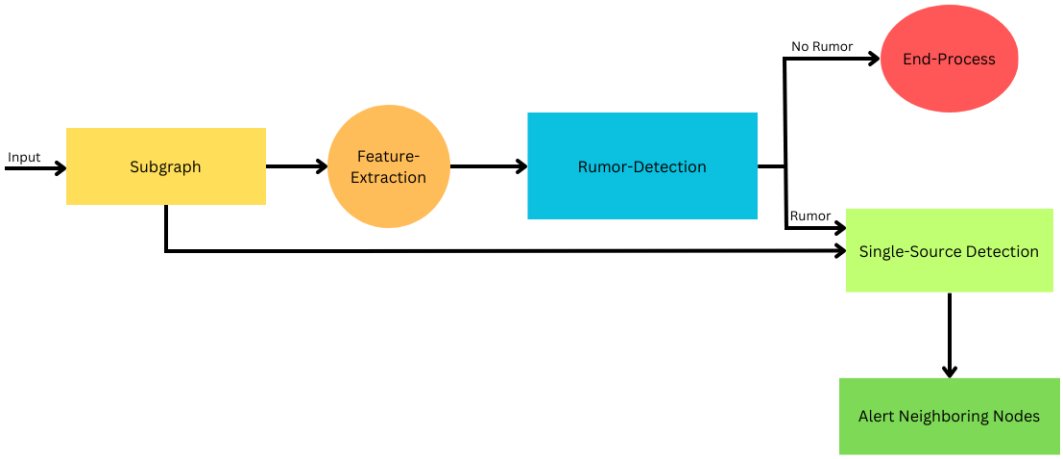

Figure 10.1: System Design

Appendix A

Libraries

A.1 Networkx

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. It provides:

- tools for the study of the structure and dynamics of social, biological, and infrastructure networks
- a standard programming interface and graph implementation that is suitable for many applications
- the ability to painlessly work with large nonstandard data sets

A.2 Pandas

pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has

the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way towards this goal.

A.3 Scikit Learn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

A.4 StellarGraph Machine Learning Library

The StellarGraph library offers state-of-the-art algorithms for graph machine learning, making it easy to discover patterns and answer questions about graph-structured data. It can solve many machine learning tasks like:

- Representation learning for nodes and edges, to be used for visualisation and various downstream machine learning tasks
- Classification and attribute inference of nodes or edges
- Classification of whole graphs
- Link prediction

Appendix B

Source Code

The code for the project can be found at https://github.com/Manas641/TwitterRumorDetectionUsingEntropy.

References

[1] Pew Research Center, November, 2021, "News on Twitter: Consumed by Most Users and Trusted by Many"

[2] Shelke, Sushila, and Vahida Attar. "Rumor detection in social network based on user, content and lexical features." Multimedia Tools and Applications 81.12 (2022): 17347-17368.

[3] Sun, Shengyun, et al. "Detecting event rumors on sina weibo automatically." Asia-Pacific web conference. Springer, Berlin, Heidelberg, 2013.

[4] Yu, Suisheng, Mingcai Li, and Fengming "Rumor identification with maximum entropy in micronet." Complexity 2017 (2017).

[5] Suthanthira Devi, P., and S. Karthika. "Rumor Identification and Verification for Text in Social Media Content." The Computer Journal 65.2 (2022): 436-455.

[6] Pogorelov, Konstantin, et al. "Wico text: a labeled dataset of conspiracy theory and 5g-corona misinformation tweets." Proceedings of the 2021 Workshop on Open Challenges in Online Social Networks. 2021. https://doi.org/10.1145/3472720.3483617

[7] Shelke, Sushila, and Vahida Attar. "Source detection of rumor in social network–a review." Online Social Networks and Media 9 (2019): 30-42.

[8] Luo, Wuqiong, Wee Peng Tay, and Mei Leng. "Identifying infection sources and regions in large networks." IEEE Transactions on Signal Processing 61.11 (2013): 2850-2865.

[9] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. ACM SIGKDD Explorations Newsletter, 19(1):22–36, 2017.

[10] Xinyi Zhou and Reza Zafarani. Fake news: A survey of research, detection methods, and opportunities. arXiv:1812.00315, 2018.

[11] Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. Truth of varying shades: Analyzing language in fake news and political fact-checking. In Proc. Empirical Methods in Natural Language Processing, pages 2931–2937, 2017.

[12] Victoria Rubin, Niall Conroy, Yimin Chen, and Sarah Cornwell. Fake news or truth? using satirical cues to detect potentially misleading news. In Proc. Computational Approaches to Deception Detection, pages 7–17, 2016.

[13] Kai Shu, H Russell Bernard, and Huan Liu. Studying fake news via network analysis: detection and mitigation. In Emerging Research Challenges and Opportunities in Computational Social Network Analysis and Mining, pages 43–65. Springer, 2019.

[14] Kai Shu, Suhang Wang, and Huan Liu. Understanding user profiles on social media for fake news detection. In Proc. Multimedia Information Processing and Retrieval, pages 430–435, 2018.

[15] Kai Shu, Suhang Wang, and Huan Liu. Beyond news contents: The role of social context for fake news detection. In Proc. Web Search and Data Mining, 2019.

[16] Adam Kucharski. Post-truth: Study epidemiology of fake news. Nature, 540(7634):525, 2016.