Addis Ababa University

College of Natural and Computational Science

School of Information Science

Community Detection in Facebook: A Big Data Analytics Approach

A Thesis Submitted to the School of Information Science for the Degree of Masters of Science in Information Systems

By

Badimaw Terefe

Advisor

Melkamu Beyene (Ph.D.)

June 2020

Addis Ababa, Ethiopia

Addis Ababa University

College of Natural and Computational Science

School of Information Science

Community Detection in Facebook: A Big Data Analytics Approach

By

Badimaw Terefe

Advisor

Melkamu Beyene (PhD)

June 2020

Addis Ababa, Ethiopia

Name and Signature of Members of the Examining Board

Name	Title	Signature	Date
Melkamu Beyene (PhD)	Advisor	_____	_____
_____	Examiner	_____	_____
_____	Examiner	_____	_____

DECLARATION

I declare that this thesis is my original work and has not been presented for any degree in any other university.

Signature: _____

Badimaw Terefe

The thesis has been submitted for examination with my approval as a university advisor.

Advisor's Signature: _____

Melkamu Beyene (Ph.D.)

ACKNOWLEDGEMENT

It would have been very difficult for me to complete this thesis if many people next to GOD weren't willing to give me their hands during most of my happy and sad days. The first of all was my advisor Melkamu Beyene (Ph.D.) whose support was vital throughout the courses of this thesis work. I must mention and appreciate his affection towards helping me by giving many constructive comments and suggestions from the beginning until the end of the thesis. He also deserves a great thank for treating me patiently during my hard times.

Secondly, I would like to thank all Facebook users especially my Facebook friends who allowed me to use their publicly available data for my thesis. Without them, the research would have been extremely difficult to achieve.

Thirdly, I am in great pleasure to thank all my classmates and teachers for their all-rounded support during my spell here at Addis Ababa University School of Information Science.

Finally, I would like to thank my family especially my mother for their sacrifice to support me from my childhood till this age.

LIST OF ABBREVATIONS

DSR: Design Science Research

GMC: Greedy-Modularity-Communities Algorithm

GN: Girvan-Newman Algorithm

NGMC: _Naive_Greedy-Modularity-Communities also known as Fast Greedy Algorithm

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This thesis is about designing a community detection model on Facebook using relational data. It is mainly aimed at helping companies by grouping Facebook users based on their interests which has many applications.

The researchers selected four algorithms based on the concept of modularity optimization/maximization. Modularity based algorithms were selected since modularity is the best quality measure of a community detection algorithm. The algorithms selected and then compared were the 'girvan_newman' algorithm, the 'greedy_modularity_communities' algorithm, the 'fast_greedy' algorithm, and the 'Louvain' algorithm. Each of these four algorithms was studied separately before, but no detailed study was done to show which algorithm best detects communities especially when it comes to relational learning.

Five datasets required to undergo an experiment (five experiments since it is difficult to ensure an algorithm as the best algorithm by doing a single experiment) were extracted from Facebook and prepared into matrix and graph data structures. An attempt was made to select the best algorithm among the aforementioned algorithms and the results gained were cheering.

The results gained in the experiment showed that the Louvain algorithm was the best in many aspects. First and for most, it is the one that detected communities with the best modularity value (average modularity of 0.491). Second, it is the algorithm that fulfilled all of the conditions to say an algorithm as the best algorithm such as ease of use, ability to detect overlapping communities, state of the art, support to the visualization of networks, and the possibility to run it on a single or distributed computing resource. Finally, it is also the one that best balances the speed quality tradeoff in community detection. That means, it is the algorithm that detected quality communities within an acceptable time.

There were many challenges the researchers faced during community detection. But two of them were the most difficult ones. The first challenge was scrapping data from Facebook and the second one was understanding the data type algorithms need to perform community detection. The first challenge was solved by understanding the HTML structure of Facebook pages and the second one was solved by studying graph data structures and file extensions in detail.

Keywords*: Facebook, community detection, modularity-based community detection*.

CHAPTER ONE

INTRODUCTION

This chapter introduces the research problem to be addressed in this thesis. It starts by giving background information about the problem under investigation. After giving background information, it tries to show the research gap available in the literature followed by research objectives that are aimed at filling the gap in the literature. Finally, it gives a clue on the significance that would be acquired by undergoing the thesis followed by the boundaries of the problems and areas that are planned to be studied.

1.1. Background

These days, web technology is growing rapidly. This rapid growth of web technology allows individuals to interact with other individuals in their society or other societies online. The interactions of an individual with the rest of society creates social networks (Punam & Chhavi, 2016). During the early days, social networks were more centered around user profiles where users compile a list of friends and search for others with similar interests. But today, more advanced features are added to these networks to allow interaction among users. The interaction among users has created a high influence in predicting shopping, relationships, and education behaviors (Siricharoen & Waralak, 2012).

Every day, a huge amount of data flows through social networks such as Facebook, Twitter, Instagram, and the like. For instance, 100 Terabytes of data are updated daily through Facebook and other activities on social networks leading to an estimate of 35 Zettabytes of data generated annually by 2020 (IBM, as cited in Geanina et al.,2012). This amount of data on social networks can be both in content as well as in structure.

Content in this context is used to refer to the data component that social network users use to interact with one another whereas, structure refers to relationships between social network users such as friendships, likes, and fellowships. Analyzing data available on both the contents and the structures found on social networks is important for various applications such as marketing plans in businesses, recommender systems, and user interface adaptations (Michel & Michel, 2013).

Social networks can represent a community or cluster structure (Punam & Chhavi, 2016). Communities are strong relationship patterns among social network users (Matin , Justin , & Shahab , 2016). A study by Zhang et al., (2018) mentioned community structure as one of the most important features of a social network or a real network in general. Another study by Didwania and Narmawala (2015) named humans as social animals and want to live in communities than in single in a way to describe the importance of communities.

The process of finding clusters or groups in social networks is called community detection (Punam & Chhavi, 2016). It is one of the many areas of big data analytics through which relationships between entities on a large scale (social media for example) are extracted (Chopade & Zhan, May 27-31, 2014). Community detection is believed to solve the problem of identifying groups of vertices that are more densely connected than to the rest of the network (Symeon , Yiannis , Athena , & Ploutarchos , 2012). Detecting these communities has great importance in sociology, biology, computer science, and other disciplines (Dhanya & Shini , 2016). It has also a huge importance in business (Siricharoen & Waralak, 2012).

Although community detection is believed to have advantages in these disciplines, achieving it from a dataset/s is usually a difficult task. This is because community detection is one area of big data analytics and the data for big data analytics is usually large in size and complex in a relationship. In addition to this, community detection algorithms available today base their work on a graph data structure which in turn needs higher processing capacities and higher processing time compared to flat data structures. This is mainly because graph data has high relationships between data and these relationships from data can be queried in real-time which in turn needs high storage and processing capacities (Chopade & Zhan, May 27-31, 2014).

1.2. Statement of the Problem

In the technologically developed world, social media analytics and using the analyzed result in different sectors are common. The analyzed data taken from social media can be used as input for various areas such as marketing (Sen , Mengjiao , & Deying , 2015), crime detection, political decisions (David & Chris , 2018), and the like. For instance, in the area of developing marketing strategies, organizations found incorporating their advertisements on social media as an important factor that can easily develop their business. Due to this fact, most organizations in the developed world have organized people who are working on researches to use social media data to the benefit

of their businesses. In contrast, Ethiopian companies are still far from integrating the gifts of social media to the growth of their businesses with the help of scientific research (Gizat, 2016).

Although social media is believed to have as many such advantages for many applications, Ethiopian organizations are not effectively utilizing its power to benefit their businesses. That means, they have no tools or models to guide them to use the power of social media for the sake of developing their business (Gizat, 2016). Some organizations are using social media for applications to some extent, but these organizations didn't have a mechanism to understand who their fanbases are, what behavior do they have, what features do they share with the fans of other organizations, apart from creating a Facebook page and counting the number of likes and followers the page has and sometimes adding information updates about their organization. In addition to this, these organizations consider each follower as a single customer and they usually forgot the advantages of considering their followers as groups or understanding the homophily relationship between customers.

It is very important to consider the homophily relationship that naturally occurs among social media users, a tendency of users to be related to other users with similar characteristics which in turn can have an impact on the way of doing business on social media. This has been widely observed in various social network analysis studies (Nowell & Klienberg, 2007). For example, knowing the age of his/her friends is an important input to predict the age of a person in a social network. In this case, knowing the age group a set of individuals fall may be used as an indicator of the presence of similarity of needs among those individuals and companies can use it as an input during setting up their businesses.

Another scenario that shows the existence of a homophily relationship is the party membership of the deputy prime minister of a certain country is more likely to be the same as the party membership of the prime-minister as the prime-minister most likely select his/her vice from the party s/he belongs to (Jensen & Nevile, 2002).

As tried to show in the above paragraphs, understanding and extracting the relationship between social media users has several advantages in many applications. But it is difficult to say many studies were done to help these applications by studying the relational nature of social media users. In this regard, big data analytics through its different areas can play a great role since it is usually concerned with uncovering different relationships between entities. Community detection is one

of those big data analytics areas where the relationship between entities is represented in the form of graphs and one can query the graph to use the relationships to applications.

Community detection in Facebook can use two types of features. The first feature is content while the second-second feature is structural i.e. the relationship between users such as friendships, likes, and fellowships (Blaise , et al., 2013). The proposed community detection technique in this research mainly used structural features of users i.e. the relationship between them and pages as a main source of information. This is mainly because these kinds of features give additional pieces of information to understand the complex structure of the underlying network.

Relationships (autocorrelation) of users are derived across boundaries and provide a unique opportunity to increase the accuracy of community detection algorithms. Besides, unlike the naturally language-dependent content features, structural features can be used in a language-independent way. However, community detection in Facebook is not an easy task and conventional machine learning approaches have several drawbacks when it comes to relational data (Fischer, 2011), (Perlich & Provost, 2003). They have limitations to transform the graph data and loss of important information or the addition of redundant information may happen. Traditional approaches only utilized explicit information. Due to this, relational learning techniques that model relational features came into existence.

We can get several pieces of research that tried to compare community detection algorithms in the literature. However, one can get the following gaps too:

1. Almost all of them didn't give a great emphasis on the structural aspect of social media data and they focused only on the content aspect of it.
2. When they compare algorithms to community detection, they usually use quantitative measures of communities such as modularity, conductance, and community distribution. But they ignored the qualitative aspects such as support to network visualization, ease of use, computing resource requirement, and the like.
3. There is no clear and commonly followed architecture to show the procedures for community detection.

Thus, in this research, an approach to compare community detection algorithms through a relational learning approach and using both quantitative and qualitative measures is proposed. A

community detection architecture is also developed as a result of all the activities performed in the research.

Hence, this research tries to answer the following questions.

RQ 1: How can we extract the structural features of Facebook users?

RQ 2: How can we represent structural features for community detection?

RQ 3: Which techniques (algorithms) are appropriate to detect communities?

1.3. Objectives

1.3.1. General Objective

The main objective of this study is to design a community detection model in the case of Ethiopian Facebook users.

1.3.2. Specific Objectives

1. To review the literature to understand the state-of-the-art approaches in community detection
2. To prepare the datasets required for community detection
3. To represent the data of Facebook users
4. To cluster the dataset into communities (detect communities)
5. To evaluate the procedure or model used for detecting communities

1.4. Significance of the Study

The study will have both theoretical and practical significances. Theoretically, it will contribute to the literature by providing a novel community detection mechanism using the relational learning (graph learning) approach. Practically, this model can help companies to know their target customers and they can structure their applications based on the characteristics and needs of their customers. In addition to knowing their target customers, companies can also rank the targeted customers. Ethiopian companies know the benefit of targeting customers for various applications but there is a lack of a scientific mechanism to identify the segments of customers to be targeted. Thus, this research will fill this gap.

1.5. Scope and Limitation of the Study

There are many social media sites and apps used for sharing content among people and form relationships with one another. These include Facebook, YouTube, Instagram, and the like.

Analyzing data on these social media is helpful for various applications like marketing, politics, crime detection, fake news detection, and so on. Due to time and resource constraints, this research focuses on community detection from social media specifically Facebook.

Community detection can be done from both the content and the structure of Facebook. Here, the main focus is on extracting structural communities through relational learning since there is a lack of research in the literature from this perspective.

Furthermore, the study is done in the Ethiopian case (the research aims to study users and companies which operate in Ethiopia) so that Facebook users from other countries were not targets of the study. This is because there is an idea to extend this research in the future so that the community detection approach (model) can be used for applications such as targeted advertising, or setting up different campaigns (maybe market or political). In addition to this, the researchers can ask for privileges to use the publicly available data of users making them aware of it on their pages (by posting a data usage appeal on Facebook) if they are Ethiopian so that there will not be privacy and security issues.

CHAPTER TWO

LITERATURE REVIEW

This chapter is intended to describe the basic concepts and related works relevant to the subject under investigation. The review is organized into four broad sections. The first section contains basic concepts related to the subject. It is organized into four subtopics namely community, community detection, modularity, and social networks. The second section is aimed at describing the algorithms which work based on the concept of modularity optimization. The third section is aimed at describing the researches related to community detection in various contexts. The final section is intended to show the applications communities can have taking advertising as an example.

2.1. Basic Terminologies

2.1.1. Community

A community is a group of individuals who agreed or asked to be together to achieve a certain task, socialize, etc. These groups can be families, neighbors, schoolmates, friends, employees working on a project, or any user with identical entities (Rokia & Idrissa , 2014). From the perspective of graph clustering, these communities can be referred to as clusters or vertices with a high density of connections among them and seldom connected with the rest of the graph (Girvan & Newman, 2000, as cited in Prat-Pérez et.al., 2014). Another study underwent by Blaise et.al (2013) defined communities as “a group of nodes more connected among themselves than with nodes outside of the group”.

Another study by Julien and Odent (2012) presented a definition of community which they called "a definition which has consensus by most researchers". In the study, they defined communities mathematically as follows. "Consider that a partition P = {C1,..., Ck} of the vertices of a graph G = (V, E) ($\forall i, Ci \subseteq V$) represents a good community structure if the proportion of edges inside the Ci (internal edges) is high compared to the proportion of edges between them.”

Traditionally, communities are formed based on their physical locations in workplaces, schools, homes, and the like. But these days, due to the development and advancement of communication technologies, they are formed virtually. Virtual communities can shrink and grow without real control. However, when you come to skills and influences, some are more skillful and influential

than the other community members. Therefore, analyzing these communities to identify and study key people, information proliferation, evolution, behavior, structure, etc. is essential for knowledge discovery leading to inform decision-making. Now, thanks to the advancement of information technology and computing, researchers can build scalable solutions capable of handling big data (Rokia & Idrissa , 2014).

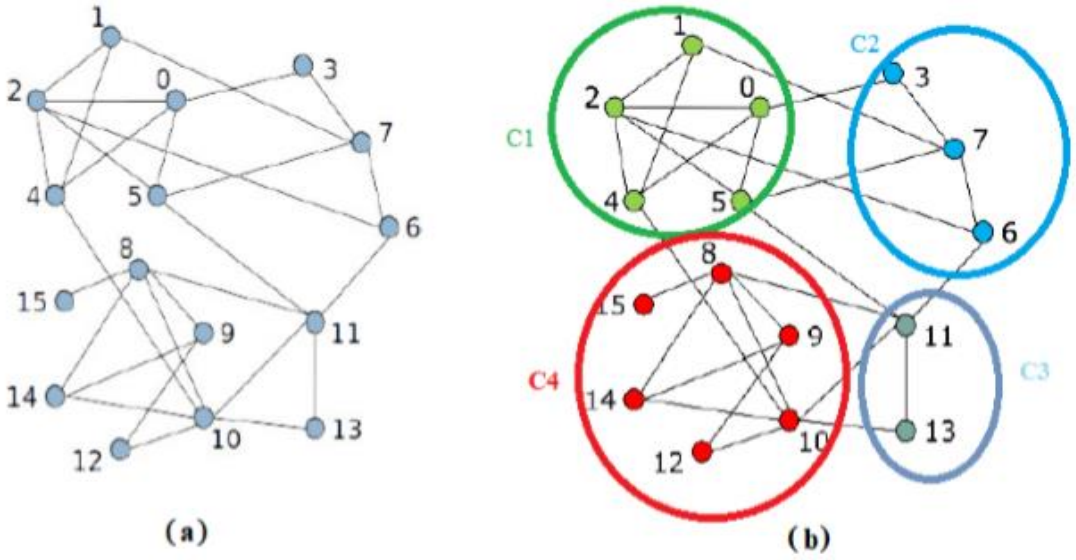


Fig 2.1: Two network representations. (a) A simple network. (b) A network of detected communities as presented in (Blondel et al.,2008, as cited in Fortunato, 2010)

2.1.2. Social Media

Social media is the term often used to refer to new forms of media that involve interactive participation. It is characterized by the following features. Social media involves a digital platform, but this does not mean everything digital is social media. Therefore, two main characteristics help to define social media. First, social media allow some form of participation. This characteristic is used to mean social media are never completely passive, even if sometimes social networking sites such as Facebook may allow passive viewing of what others are posting. At least a profile must be created that allows the beginning of an interaction. Second, social media needs interaction. This interaction can be with established friends, family, or acquaintances or with new people who share common interests or even a common acquaintance circle (Manning, 2014). Among the different families of social media, the researchers used Facebook in this study.

2.1.3. Community Detection

Community detection is the process of finding network nodes that are densely connected internally and sparsely connected between. These communities are created when individuals within the group interact more frequently with each other than with those outside the group (Parvathi & Monica, 2017). It is essential in social media because people usually form relationships based on their interests. Therefore, they share their political, economic, and social views of social media. In this process data related to their relationship as well as the content they share can easily be obtained from these social media and analyzed for different purposes such as crime detection, political decisions, and marketing planning (Sweta , Shubha , & Anurag , 2017). Punam and Chavi (2018) defined it as "the process of discovering cohesive groups or clusters in a network". They mentioned it as one of the key tasks of social network analysis which is useful in applications where group decisions are taken. Parvathi and Monica (2017) defined it as "it is a way to partition the network into dense regions of the graph, and those regions, in general, correspond to entities which are closely related, and can hence be said to belong to a community" (P. 159). Many complex systems in the real world can be modeled as graphs or networks. One of the most relevant features of graphs representing real systems is community structure, or clustering (Pravin & Justin , 2014).

Community detection aims at grouping nodes by the relationships among them to form strongly linked sub-graphs from the entire graph (Albert et.al., 2007 as cited in Wang et.al., 2015). In addition to grouping these members into clusters, it can also tie members because of their interaction (Bella & Xiaoou , 2016).

2.1.4. Modularity

The literature describes modularity as by far the most used and best-known quality function (Fortunato, 2010). It has the following definition. Given some partition, the metric used to measure or evaluate how well it is partitioned is known as modularity. It is used to indicate whether nodes in the partition are interacting with each other more frequently than with nodes in other partitions or not (Dubik, 2017). It has the following formula representation.

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij})\, \delta(v_i, v_j)$$

Fig 2.2: Modularity equation as presented in (Dubik, 2017)

Among the parts in the formula shown in the figure above, the first part is $\frac{1}{2m}$ which assures that Q is a normalized constant on the interval of [-1,1]. In this interval, positive values for Q indicate communities are detected and negative values of Q indicate no communities were detected ((Dubik,2017); (Julien M. O. & Odent , 2012)).

The second part of the formula is composed of two components; the connections of the nodes through an adjacency matrix, A, i, j represents two different nodes. A*ij* refers to an edge between the nodes in the adjacency matrix with value 1 if the nodes are connected and 0 if the nodes are not connected (Dubik, 2017).

The third part P*ij* is a random factor called the "null model" that is used as a value judgment when a particular density of edges is significant enough to define a community (Newman, 2006 as cited in Dubik,2017). The standard (Bernoulli) random graph or Barber's modularity for bipartite graphs can be considered as choices as null models (Barber, 2007 as cited in Dubik,2017).

The last term in the formula makes sure that only nodes in the given community are considered. The term is described in the form of Kronecker delta as follows:

$$\delta(v_i, v_j) = \left\{ \begin{array}{ll} 1, & \text{if node } i \text{ and } j \text{ are part of the same community} \\ 0, & \text{otherwise} \end{array} \right\}$$

Modularity has several advantages. The most important of them is it doesn't require the number and size of communities or partitions to be entered into algorithms manually (Julien M. O. & Odent , 2012) which is the drawback of most of the graph partitioning algorithms available today (Pons & Latapy , 2012). Modularity based algorithms follow a repetitive approach to determine the number of communities a particular network should be partitioned into by checking the modularity increase obtained in each repetition. But this creates a problem in extremely large networks. This is because modularity-based algorithms partition a network until no modularity increase is found and this requires considerably much computation time and resource (memory and processor) (Julien M. O. & Odent , 2012).

2.2. Modularity Based Community Detection Algorithms

As indicated above, modularity is considered by researchers as the best-known quality function for graph clustering problems. Therefore, the researchers of this paper have made their full

attention to understanding and experimenting with algorithms that are based on the so-called modularity maximization. So, let's first understand what is meant by modularity maximization before listing and describing each algorithm in detail.

2.1.1. Modularity Maximization

Community detection algorithms based on modularity maximization generally assume that partitions having high modularity value are good partitions or strong communities even though it may not usually be true (Fortunato, 2010). So, a partition corresponding to its maximum value on a given graph should be the best or at least a very good one (Fortunato, 2010). This is the main motivation for modularity maximization followed by modularity-based algorithms. The maximum modularity value will, therefore, be obtained by partitioning the graph again and again until there is no increase in the modularity value. If a point in which no increase in modularity is reached which means modularity value remains the same with previous iterations, then it is an indicator of the conclusion that the maximum value of modularity is found so that no need of partitioning the network further.

According to Girvan, Newman (2004), who are the ones who used and defined the term modularity in the area of graph partitioning and clustering for the first time, a community is said a strong community if its modularity value falls between 0.3 and 0.7. Additionally, a community detection algorithm based on modularity maximization is said to be a good algorithm if it can cluster a network with a modularity value between 0.3 and 0.7.

What does a modularity value of 0.3 and 0.7 mean?

- ✓ A modularity value of 0.3 means the algorithm has put 30% of the edges inside communities and 70% of the edges between communities. i.e. 30% intracommunity and 70% inter-community relationships.
- ✓ A modularity value of 0.7 means the algorithm has put 70% of the edges inside the communities and the remaining 30% edges between communities. i.e. 70% intracommunity relationship and 30% inter-community relationships.

2.1.2. Algorithms Based on Modularity Maximization

2.1.2.1. Louvain Algorithm

Proposed by Blondel et al, (2008), the Louvain algorithm is one of the community detection algorithms which work based on the concept of modularity maximization. There are many reasons for it to be one of the most frequently used algorithms for community detection. Among these reasons, its ability to detect communities within a short time. In addition to this, many algorithms compromise the quality of communities while detecting them within a short time which is formally called the speed-quality tradeoff. But the Louvain algorithm is best known for balancing the speed-quality tradeoff (Blondel et al, 2008). The other reason to use this algorithm is its ideality for sparse networks.

However, the algorithm has also some problems. The first problem is its storage requirement is relatively high since the algorithm needs several iterations to detect communities. The second problem is called the resolution limit which means smaller communities are difficult to be detected by the algorithm. The third problem is its inability to find a global maximum modularity point for the communities and it is a problem that exists in all modularity-based community detection algorithms (Julien & Michael , 2012).

By principle, the algorithm starts clustering by assigning each node into a community and then move this community to its neighbor communities as far as it brought an increase in modularity (Arzum & Serap , 2018).

The Louvain algorithm performs community detection in two phases in each iteration. The first phase is known as modularity optimization via local heuristics. In this phase, the algorithm starts by assigning each node in the graph as a community that makes every node community (Vincent , Jean-Loup , Renaud , & Etienne , 2008). Then, it uses the local moving heuristic to get an improved community by moving individual nodes from one community to neighboring communities until no modularity increase is achieved (Arzum & Serap , 2018). It calculates the increase in modularity value by the following formula.

$$\triangle Q = \frac{\sum_C + k_i^C}{2m} - \left(\frac{\sum_{\hat{C}} + k_i}{2m}\right)^2 - \left[\frac{\sum_C}{2m} - \left(\frac{\sum_{\hat{C}}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)\right]$$

Where:

$\sum_C$ → is the sum of the weights of the edges inside C

$\sum_{\hat{C}}$ →is the sum of weights of the edges incident to nodes C

$\boldsymbol{ki}$→is the sum of the weights of the edges from i to nodes in C

$\boldsymbol{m}$→is the sum of the weights of all the edges in the network

The second phase is known as community aggregation. In this phase, the algorithm groups all the nodes belonging to the same community and construct a network where the nodes are the communities from the previous phase (Arzum & Serap , 2018).

In summary, the Louvain algorithm detects communities in the following two major steps (Julien & Michael , 2012):

The first step is called local modularity optimization and it can be done as follows:

A) Assign each node into a different community
B) For each node i,
    a. Evaluate change in modularity if we placed node i in the community of each of the neighbors. Modularity will be calculated using the above formula.
    b. Move node i to the community with the biggest modularity gain
    c. If no change in modularity value, make the node i stay in the original community
C) Repeat the above procedure until no further improvement in modularity value is obtained.

The second step is called community aggregation and works as follows:

A) Construct a new network. in this network, the communities found in the above first step will be considered as nodes.
B) Calculate the weights between the two nodes. The total weight of a community can be calculated by adding the total weights of the nodes in that particular community.

- Successful completion of the above two steps is called a single pass.

C) Repeat the steps above decreasing the number of communities with each pass until maximum modularity is achieved.

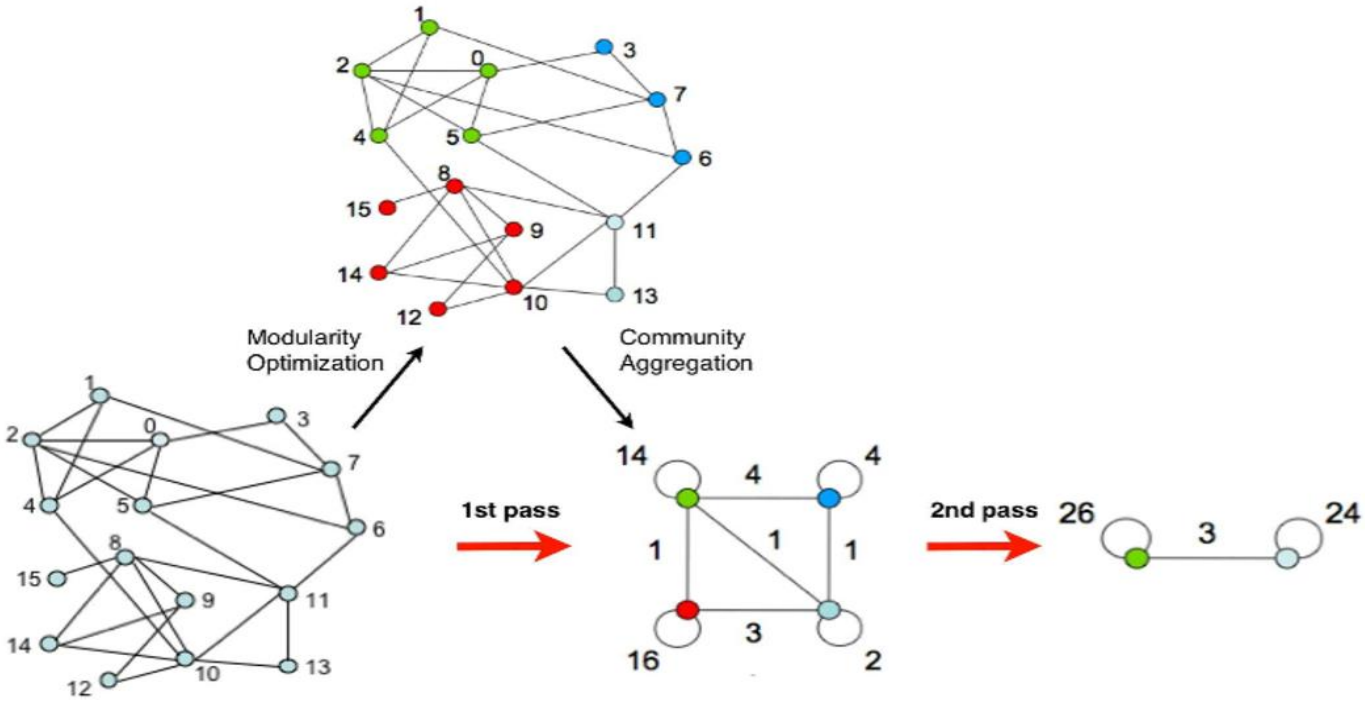


Figure 2.2: The Louvain algorithm. How does it work? As stated in Julien and Michael (2012).

2.1.2.2. The Girvan_Newman Algorithm

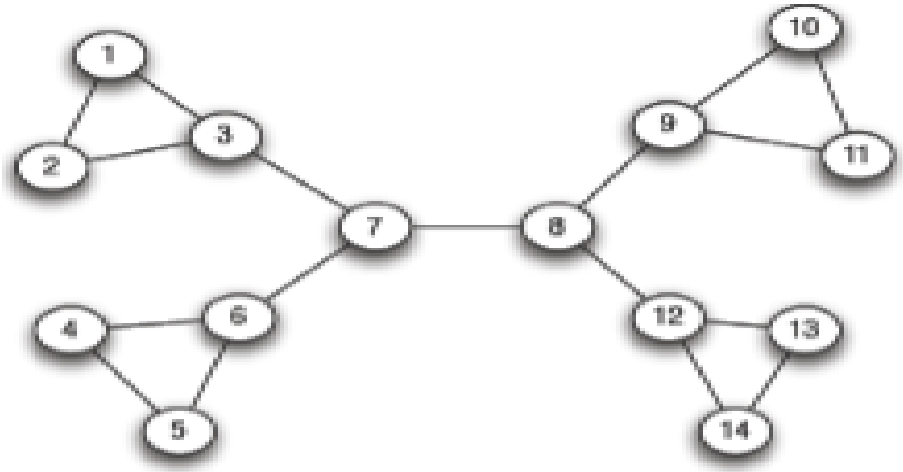
The Girvan_newman algorithm is a divisive community detection approach proposed by Girvan and Newman (2004). The algorithm uses edge betweenness as a metric to identify the boundaries between communities (Clauset, Newman, & Moore, 2004). It follows two major steps to detect communities from large network data. The first step is an iterative removal of edges from the network to find communities based on a measure called edge betweenness. The second step is to recalculating the edge betweenness measure after each removal (Girvan & Newman, 2004).

Using the divisive approach to partition a graph into communities, the algorithm first identifies vertices which are the most between other vertices and removes the edges between these vertices. The algorithm performs this operation repeatedly until some random point. This process divides a graph into smaller communities. After dividing a graph into smaller communities, the algorithm then represents these communities into a dendrogram (Girvan & Newman, 2004).

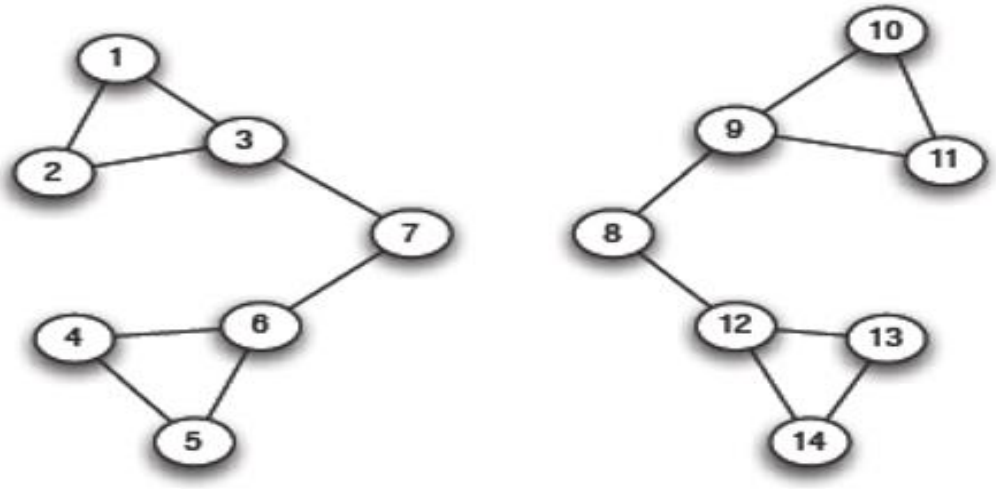
Generally, the algorithm follows the following pseudocode to detect communities from networks.

A) Compute betweenness centrality for all edges
B) Remove the edges with the largest betweenness centrality
C) Recalculate centrality on the running graph
D) Repeat the procedure from step B.

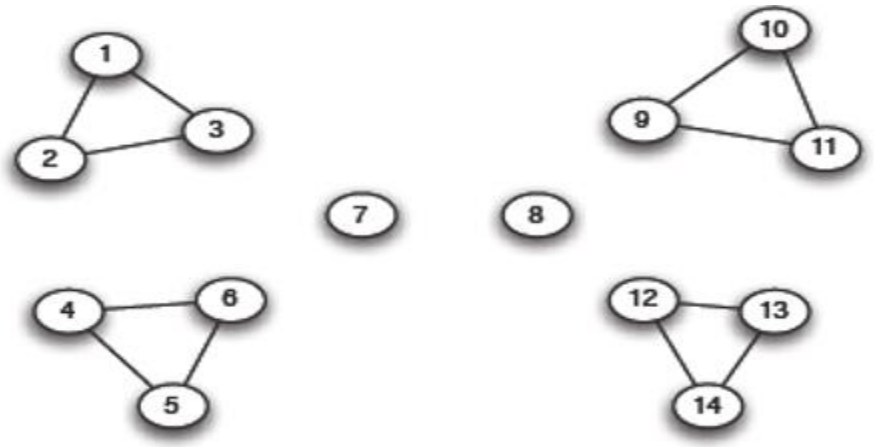
Assume a certain dataset has the following graph structure initially.



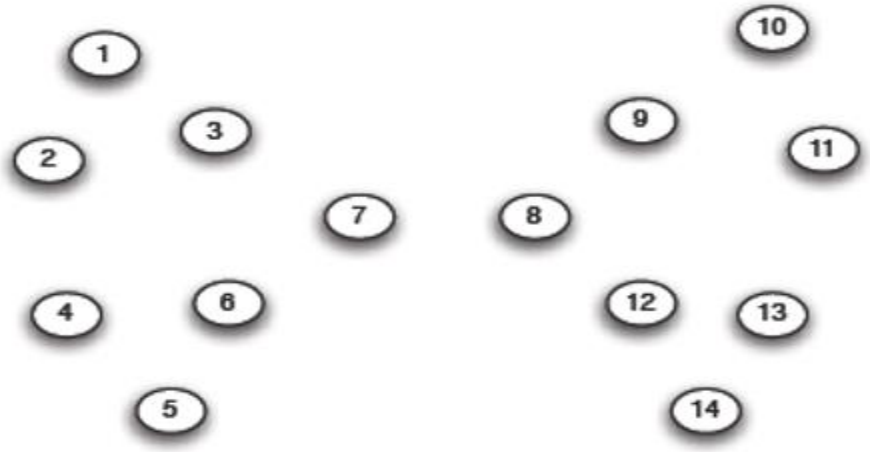
The Girvan_newman algorithm follows the following procedure to partition the graph into communities.



(a) *Step 1*



(b) *Step 2*



(c) *Step 3*

Figure 2.3 The procedure to remove betweenness followed by the Girvan_Newman algorithm

2.1.2.3. The Greedy Modularity Communities Algorithm

This is a greedy based community detection algorithm proposed by Newman and et al., (2004). It works based on modularity optimization and represented in a dendrogram which is a hierarchical decomposition of a network into communities. It starts detecting communities by first assigning each vertex as a community. Then, it repeatedly joins two communities whose consolidation produces the largest value of modularity value. The algorithm stops joining vertices after n-1 such joins assuming n is the number of vertices since this is a stage where the algorithm is left with a single community (Newman and et al., 2004).

The functioning of this algorithm has some similarities with that of the Louvain algorithm. For instance, both the algorithms start detecting communities by initially assigning each node as a community and they try to reduce the nodes by combining them as far as they produce an increase in modularity value. But in the process, they have differences as well. For instance, the greedy modularity algorithm creates edges during combining edges to form groups or communities whereas the Louvain algorithm creates them during the initial construction of the graph(Julien & Michael , 2012).

Generally, the greedy modularity algorithm follows the following procedure to detect communities from graphs:

A) Assign each node as a community i.e. there will be as many clusters as the number of nodes
B) Reduce the number of clusters by adding an edge and calculate modularity value i.e. when adding a single edge, the number of clusters will be minimized from 'n' to 'n-1'.
C) Construct the newly partitioned network.
D) Repeat the procedure from B until no unconnected node remains in the network.

2.1.2.4. The Fast-Greedy Algorithm

Proposed by J. Newman (2004), the fast-greedy algorithm detects communities through an agglomerative hierarchical clustering method. Like the greedy_modularity_communities algorithm, it starts by considering each vertex as a community and repeatedly join two communities. The main difference though is it joins two communities only if they can bring in an increase in modularity. This allows the algorithm to partition a network graph within a short time (Newman, 2004).

The algorithm is considered by many researchers as one of the fastest algorithms to detect communities, but it is also criticized for its inability to detect optimal communities under different circumstances (Zhang, Ma, Zhang, Sun, & Yan, 2018). Generally, the algorithm follows the step to detect communities.

A) **Initialization**: each point in the network is assigned into a separate community
B) **Merging communities**: choose community merging that increases the modularity value.
C) Iteratively execute step 2 until all communities are merged into a single community.

The community that produced the maximum modularity value during the process of consolidating all the communities will be selected as the final community.

2.3. Related Works

There are studies in the literature about community detection although it is difficult to say they are many in amount and quality. These studies proposed different algorithms and approaches to detect communities from social network graphs. But there lacks research that compares these approaches (algorithms) with an experimental approach and indicates the best one to follow too often. In addition to this, most of the researches focuses on the behavioral feature (contents) of users to identify communities from social networks. But the structural aspect of social media networks has got little attention from many of the researchers. Let's discuss them taking main concepts from each related study.

In 2013, a study about community detection and approaches to detect communities was proposed by Blaise and colleagues in France aimed at identifying the key functions to help social networks to monetize their audience. In the study, the authors started their study by defining the community and classified it into four categories. They defined communities as "a group of nodes more connected among themselves than with nodes outside of the group" (Blaise , et al., 2013). The authors not only categorized communities into four categories, but they also suggested better algorithms to detect each category of communities.

The first group is called a community of friends. These groups of communities refer to groups of users linked to each other with friendship links. They represent the social group a user belongs to. To detect these communities, the authors used the Louvain algorithm.

The second category of communities is called Local communities of friends. The methods to detect these groups of communities have the objective to identify the social group a node belongs to but using only local communities. These methods help to portray a user's neighborhood and are particularly interesting when a network is not entirely identified or is evolving. To detect these communities, the authors proposed a new algorithm called « IOLoco » which can detect overlapping local communities which is not possible through the Louvain algorithm. The local communities detected in this method can be used to discover and picture the neighborhood of a node, or for more sophisticated tasks such as social recommendation.

The third category of communities and which is the aim of this paper is called Communities of users with common interests. In this category of communities, what is taken into consideration is not the communication of users among themselves like in social links instead of a similarity measure based on common interests to link two individuals in a social graph. Here, the communities formed are called communities of interest. A community of interest is a group of users having common interests while not necessarily being socially connected in any way. We can consider a community of interest as a cluster resulting from applying a clustering algorithm on users' interest data. The study mentions the use of a bipartite graph to get these characteristics in the following way. First, a bipartite graph G (U, I, E) where two sets of nodes of different types (U the set of user nodes, I the set of interest nodes) are linked by edges (u, i) in E, linking user node u with interest node i if user u is interested in interest i is built. Then, a graph G1 (U, E1) is built by projecting graph G to the users' side: in this graph, a pair of user's nodes u1 and u2 will be linked if they are connected to at least n common interest nodes in graph G. Finally, a community detection algorithm is applied on graph G1, which will cluster users' nodes into communities of interest. The process is illustrated in the following figure.

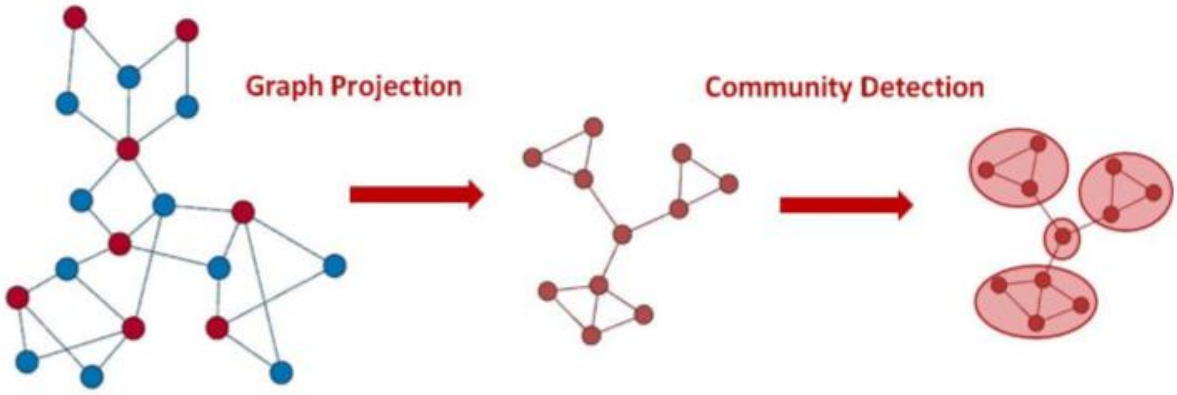


Figure 2.3: Constructing communities in bipartite graphs presented in (Blaise, et al., 2013)

The final category of communities is called communities of friends with common interests. These communities are created with friendship links and attributes that describe user interests. The Louvain method was used in the study to detect these types of communities (Blaise , et al., 2013).

Another study by Liangxun and Bianfang (2015) discussed community detection algorithms based on statistical inference. In the study, the researchers broadly classified communities into two categories; vertex-based and link-based communities. For each category of communities, they

specified community detection models. For instance, the Planted Partition Model (PPM), Newman's Mixture Model (NMM), Mixed Membership Model (MMM), Mixed Membership Stochastic Block Model (MMSBM), and Degree-Corrected Stochastic Block Model (DCSBM) are listed under models for detecting vertex communities. On the other side, SPAEM (Simple Probabilistic Algorithm for Community Detection Employing Expectation Maximization), SBML (stochastic block model for link community), GSBM (general stochastic block model) are listed under models to detect link-based communities.

In the next year, another study by Chitra Devi and Poovammal (2016) discussed using community detection to understand trust and distrust relationships through opinion mining. To achieve this, they used machine learning, natural language processing, and text mining. In the study, the researchers classified communities as a node-centric community, link-centric community, network-centric community, and hierarchy-centric community. They mentioned the graph-based approach, link-based approach, agent and dynamic based approach, fuzzy-based approach, and modularity scoring based approach as approaches to detect communities. In their study, they followed the modularity-based approach (Chitra & Poovammal, 2016).

Communities in social networks can also be detected using node attributes in addition to the social links. Concerning this, Bella & Xiaoou (2016) proposed an attribute ranking algorithm for Facebook community detection. The algorithm first collects node attributes listed on user profiles, marketing events, and user behaviors. Then, it ranks the attributes using the RELevant Node Attributes (RELNA) algorithm and selects the top n relevant attributes to be used as input for rebuilding the attributed social network. Finally, community detection will be done from the attributed social network rebuilt previously as shown in the following figure.

Figure 2.4: A global view of community detection with pre-processing RELNA adapted from Bella & Xiaoou (2016)

Martin et al., (2016) wrote a paper on community detection and ranking. In this paper, they did community detection in two phases. First, they applied an attracter algorithm minimizing its complexity through loops and breadth-first search, and then they proposed a ranker algorithm to rank communities detected in phase one (Matin , Justin , & Shahab , 2016).

There are also community detection studies done based on the user's social activities. For instance, Amit and Nikita (2017) proposed an approach based on parameters called common social activity (CSA) and activity parameters. Activity parameters are given as input to the K-means algorithm which results in groups of users referred to as a community. In this process, the communities reflect groups of users having an active social relationship and a common interest (Amit & Nikita , 2017).

Sweta et al., (2017) classified community detection techniques using another approach as shown in the following diagram.

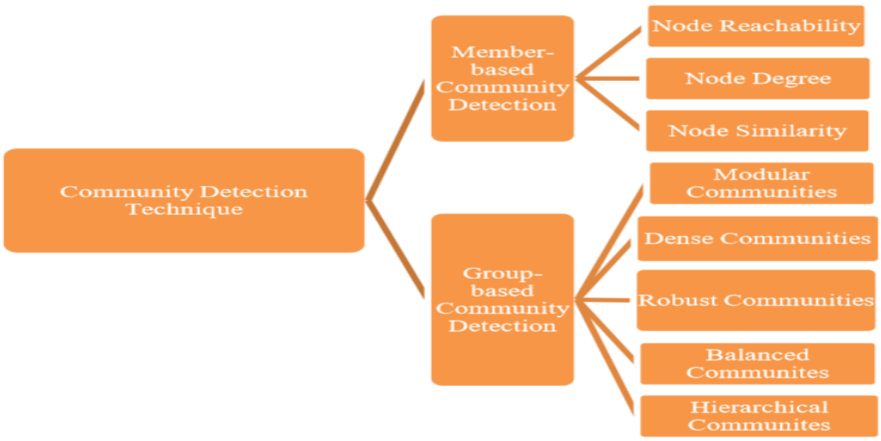


Figure 2.5: Hierarchical Classification of community detection Technique (Jiang et al., 2014, as cited in Rai et al., 2017)

Punam and Chavi (2018) undergone a detailed survey of the approaches for community detection on social networks. The study classified the approaches into five major classes; cluster-based community detection approaches, graph partitioning based approaches, genetic algorithm-based approaches, label propagation-based approaches, semantics-based community detection. It mentioned divisive clustering, modularity maximization, greedy optimization of modularity, hierarchical clustering, modularity optimization using simulated annealing, modularity optimization, and extremal optimization, agglomerative clustering, hierarchical fuzzy spectral

clustering, density-based clustering, Markovian clustering, entropy centrality-based clustering, and consensus clustering (random walk) under cluster-based approaches.

It also mentioned community score as the fitness function, multi-objective optimization, single objective, multi-objective optimization, community score and modularity as the fitness function, genetic algorithm and clustering, modularity optimization, and multi-population cultural algorithm as genetic algorithm-based approaches. Additionally, the study also mentioned iterative label propagation, propagation, inflation, cut-off, conditional update, label propagation, and overlapping communities as label propagation-based approaches. Finally, the authors mentioned clique and non-clique-based approaches for detecting overlapping communities (Punam & Chhavi, 2018).

2.4. Online Communities and Advertising

Advertising is not the point of focus in this research. The main reason for the inclusion of this section so is to show the role of communities by taking one of the several applications communities can have a great role. Advertising is chosen from those many applications because the relationship between business pages and users is relatively easy to understand for people.

Research by Spaulding (2010) describes revenues from communities that reach billions of dollars from software sales, subscriptions, and ad revenues. This indicates that advertisers are one of the highest beneficiaries of communities since they can allow them to generate such a huge revenue. The study also describes Facebook communities as interfaces for businesses and providers of tools for targeted advertising.

Communities have strength functions measured in various metrics. Hence, the various communities obtained by dividing a dataset based on some criteria can be ordered according to these strength functions. This can give a clue for advertisers in such a way that disseminating ads to those communities that can produce high response rates is advantageous than disseminating ads to communities that produce less or no response rates. Therefore, strong communities are more worthy of targeting for advertisement than weak communities. This is because the message contained in the advertisement has a high possibility to reach each member of a strong community than in a weak community since a strong community refers to a strong connection among community members.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1. Overview

This chapter describes the research methodology followed to undergo the research. It describes the phases of the research design such as the data collection techniques used in the research, data analysis, data presentation, and evaluation techniques.

3.2. Research Design

For identifying community structures in Facebook and designing a community detection model, researchers followed the design science research design. “Design science research offers an important paradigm for conducting applicable, yet rigorous, research” (Peffers, et al., 2006). It is best for researches aimed at producing artifacts such as models, algorithms, instantiations, and prototypes at the end (Peffers, et al., 2006) This paradigm is selected since it provides the workflow to achieve the objective of this research which is building a model for community detection.

Although design science research is commonly agreed to produce design artifacts, in the end, the processes to produce these artifacts are not similar in all researches. For instance, Peffers, et al. (2006) developed a six-step process to undergo design science research. These steps are problem identification and motivation, objectives of the solution, design and development, demonstration, evaluation, and communication (Peffers, et al., 2006).

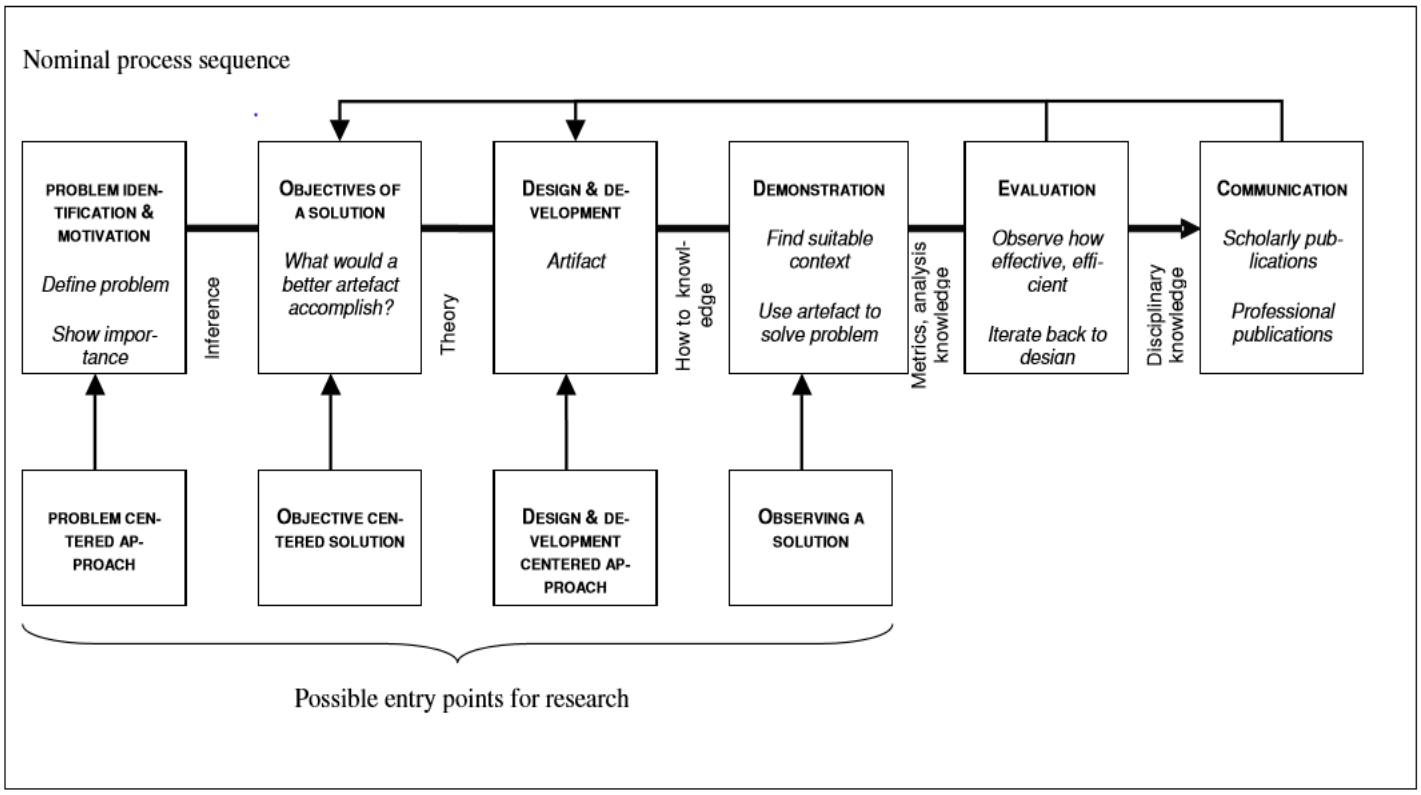


Figure 3.1: Design science research process (DSRP) model (Peffers, et al., February 2006)

On the other hand, (Vaishnavi, et al., 2004/19) proposed another process of undergoing the design science research containing awareness of the problem, suggestion, development, evaluation, and conclusion as depicted in the picture below.

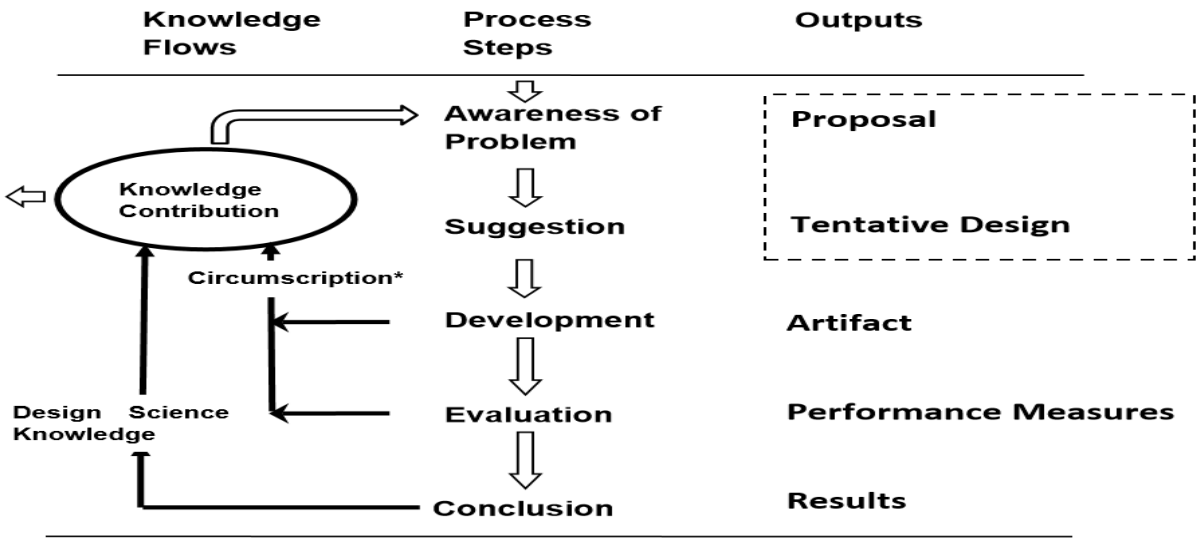


Figure 3.2: The Design Science Research Cycle, (Vaishnavi, et al., 2004/19)

In addition to the above researchers, another research by Philipp et al., (2009) had proposed another process for design science research. It is described in the figure below.

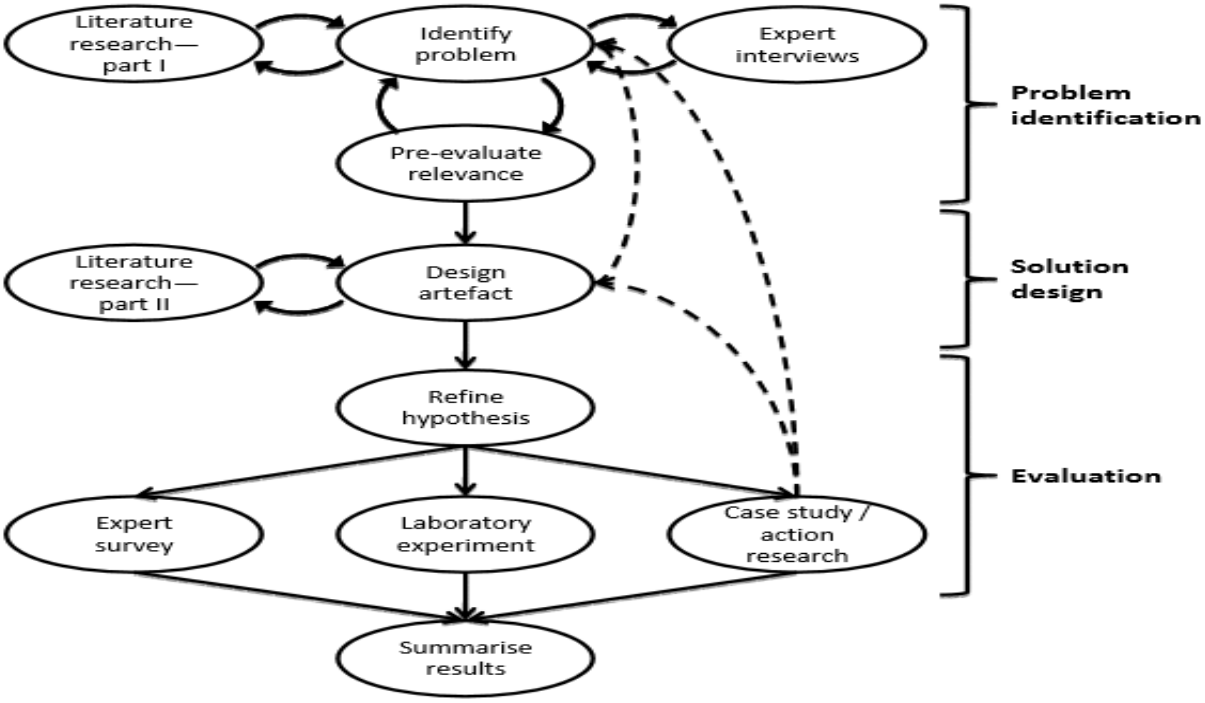


Figure 3.3: Proposed research process, (Philipp, Olga, Marten, & Udo, January 2009)

In this research, the researchers followed the methodology proposed in Vaishnavi, et al (2004/19) presented in figure 3.2. This is mainly because this process model is the most widely followed design science process model for this kind of researches (Baskerville, Baiyere, Gregor, Hevner, & Rossi, 2018) and easily fits the problem under study. The following are the phases of the selected DSR process model.

3.2.1. Awareness of the problem

Although social media especially Facebook is known to help companies and governments in their way to shape their operations and do their businesses, it is not well utilized by most of the companies which operate in Ethiopia (Gizat, 2016). This is not due to the negligence of those companies to disseminate their media releases via Facebook. Instead, it is because many media briefings and releases produced and disseminated by those companies went to the wrong pool of audiences on Facebook which in turn decreases the response rate intended to be gained by the briefings or releases.

Indicating a possible scientific mechanism to help companies in this process is therefore important and lacking in the literature. One of these mechanisms is a model to be used as a reference during setting up different campaigns for applications like marketing, election, policy-making, and others. In this regard, an attempt is made to design a community detection model (which uses structural or relational data) on Facebook. The model aims at first extracting data from Facebook, representing the extracted data, and then classifying users into communities.

As part of the problem awareness, the researchers had listed some problems and claimed a possible solution to the problem. To understand the problem under study which is mentioned in the problem awareness phase, the researchers had prepared two documentations; the proposal and the literature review. For more detail, one can refer to chapters one and two of this thesis documents.

3.2.2. Suggestion

As described in Philip et.al. (2009) the suggestion phase of design science research focuses on the novel configuration of existing or new elements.

The purpose of the suggestion phase in this research was to make sure community detection is achievable and to undergo community detection to understand and show the procedures to detect communities. This purpose was attained through three major steps; preparing a dataset, undergoing community detection, and evaluating the quality of detected communities. Unlike the other phases of the DSR, the suggestion phase is a complete phrase by itself. Meaning, undergoing the whole procedure is mandatory to solve a specific problem so that the researcher can produce a small but working version of the model in mind. Therefore, all activities done in the suggestion phase of this research were performed considering this concept.

In the first step, sample data (a graph with a relatively small number of nodes and edges) was prepared. To prepare the dataset, different Python libraries were used. The first library is the '*url.request*' library which was used to catch Facebook pages and store them into a handle so that the pages can be accessed and managed offline. The second library is '*BeautifulSoup'* which is used to show the pages extracted by the '*url.request*' library in a structured plain HTML structure so that one can easily identify the tags containing the exact information needed from the Facebook page. The third library is '*pandas'* and it was used for presenting and analyzing tabular data more specifically for constructing the data into a matrix format since presenting the data into a matrix format is necessary to construct the graph. The fourth library is '*Numpy'* and it was used to undergo

different numerical operations on the data. In addition to this, Numpy was used as a dependency to the various community detection algorithms used in the second step of this phase. The last library is '*Networkx'* and it was used to construct and analyze graphs since the graph is the ideal data structure for community detection.

Once the data is extracted and stored in an offline handle and presented into a graph, the second step is detecting communities. For doing so, four algorithms based on the concept of modularity maximization/optimization were selected taking the belief in the literature that says 'modularity is by far the best quality metrics to measure the quality of a community detection algorithm'. These algorithms are the '*girvan_newman*' algorithm, the the'*greedy_modularity_communities'*, the '*fast_greedy_algorithms*', and the '*Louvain*' algorithm. From the above four algorithms, three of them (except the Louvain algorithm) were imported from the networkx algorithm of Python. But the Louvain algorithm was imported from the 'community' algorithm of Python.

The details of the activities in this phase together with the development and evaluation phases are included in chapter four of this thesis document.

3.2.3. Development

According to Philip et.al. (2009), in the development stages of the design research, the tentative design developed in the suggestion phase is further developed and implemented. The type of artifacts developed and implemented can be design theories (Gregor and Jones, 2007, as cited in Philip et.al.,2009), concepts, models, processes, or instantiations (March and Smith, 1995; Hevner, et al., 2004, as cited in Philip et.al.,2009). The variation of the artifacts to be developed determines the technique to be used for developing and implementing these artifacts.

In this phase, the researchers implemented the tentative design developed during the suggestion phase. As part of the phase, they performed three main activities. The first activity was preparing the datasets. By their very nature, community detection algorithms need data in the form of graphs. So, five graph datasets were prepared for the experiment. Those five datasets were varying only by their size i.e. the number of nodes and edges. The need to prepare five datasets was to make sure the result gained during comparing algorithms was not by chance. The second activity was selecting algorithms to detect communities from the datasets prepared before. The algorithms selected were then compared against some criteria so that one can apply the best one to detect communities. The third activity was detecting communities using the algorithm that was found

best during the comparison. During the process, almost all of the tools mentioned during the suggestion phase were used here too since in the development phase of this research, the activities done in the suggestion phase were applied with more datasets and including additional concepts.

The detailed descriptions of what was done on the development of the artifact are included in chapter four of this thesis document.

3.2.4. Evaluation

Many metrics can be used to evaluate the quality of a community detection algorithm. Entropy, accuracy and precision, modularity, and using artificial data as a benchmark are the main ones. Among these metrics, many researchers believe modularity is the best quality function and the most used one (Fortunato, 2010) metrics. As a result, the quality function used in this research was the modularity metrics. To measure the modularity value, the researchers used the '*community.modularity*' library of Python.

In addition to the community detection algorithms, the model developed during the suggestion phase and further developed during the development phase must be evaluated. The model in this research was done through experimentation and the experimentation itself can be considered as an evaluation of the procedure to achieve community detection since one can apply the procedure by himself/herself.

To evaluate the quality of a community detection (to know which algorithm best detects communities) and the model, the researchers undergone experimentation as discussed in chapter four in detail.

CHAPTER FOUR

DEVELOPING A COMMUNITY DETECTION MODEL

4.1. Overview

This chapter presents steps and procedures undergone during experimenting as part of the development of a community detection model and the results gained from each procedure. These procedures were presented based on the design science research process chosen in the previous chapter. It describes three steps of the DSR process chosen in section 3.2 of the previous chapter. These are suggestion, development, and evaluation. However, the nature of the problem under study forced us to put the evaluation phase together with the development phase.

4.2. Suggestion

4.2.1. Designing a Prototype

A community detection model on Facebook is going to be developed and shown in the later stages of this chapter. But to check the achievability of the goal and to understand and indicate the procedures to undergo the experiment, developing a prototype was found important.

The prototype designed in this thesis was, intended to know the flow and implementation technique of the various concepts under community detection with a dataset of a limited size. Furthermore, the prototype was aimed at ensuring extracting data from Facebook, preparing it, and detecting communities from it, and mapping the communities detected to an application is something achievable. For designing the prototype, the researchers in this thesis performed the following activities.

First, the researchers installed Jupyter Notebook from Anaconda cloud and installed the libraries required to scrape data, structuring the data scraped, detect communities, and finally calculate modularity. For data scraping the researchers installed *urllib. request* library which is useful to request access to Facebook pages through URLs and downloading them to a handle. For structuring and presenting the downloaded data in a form easily understandable, they installed *BeautifulSoup* and *pandas* libraries. For detecting communities, two libraries were installed; networkx and community-Louvain.

Second, they used their Facebook accounts to understand the structure of data presented on Facebook pages and extracted the tags helpful to get the required data. Facebook pages are presented using HTML formats and each content on the page is assigned with a specific tag. Inspecting these tags to get the required content from the pages is one of the challenging tasks the researchers faced during the research.

Third, they scraped like information of 91 Facebook users containing 16 liked pages. The scraped pages were first put in an easily understandable format before saving them permanently. After understandably presenting them, the researchers created a matrix showing which Facebook users liked which pages putting 1 if there is a relationship and 0 if there is no relationship. Then, the data presented as a data frame dataset was exported to two formats; excel and CSV.

Fourth, they organized the user names and pages into nodes and edges of a graph which resulted in 108 nodes and 408 edges.

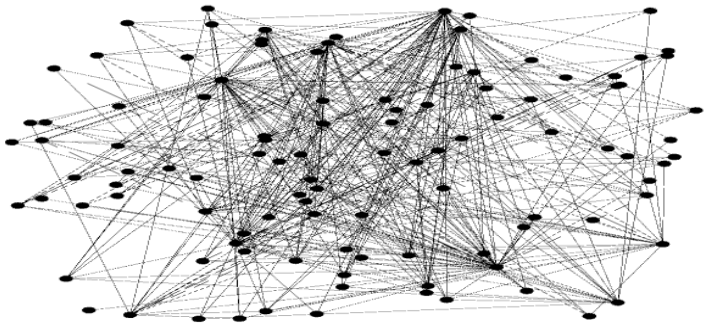


Figure 4.1: A graph structure of the dataset before partitioning

This was the structure of the network before partitioning it into communities using a community detection algorithm.

Once the dataset is prepared and is ready to run algorithms, the next task is detecting communities from it through an algorithm. To detect the communities from the dataset in the prototyping phase, the Louvain algorithm was chosen randomly. The following figure shows the structure of the communities detected using the Louvain algorithm.

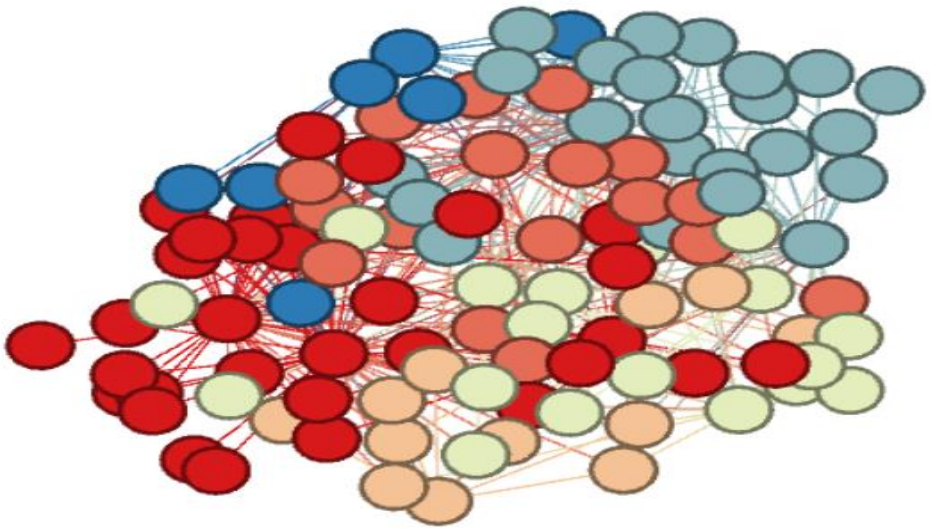


Figure 4.2: A graph structure of the dataset after partitioning

Each color shows the modularity classes of the network that tells us the network is partitioned into 6 communities each modularity class indicated by its color.

Finally, they applied the modularity function of *Networkx* to evaluate the level of strength the community function has brought to the dataset given above. Based on the result gained through the modularity function, the strength of communities is around 0.26.

Based on the descriptions given above, the researchers came up with a community detection process model on Facebook. The process model is shown in the figure below.

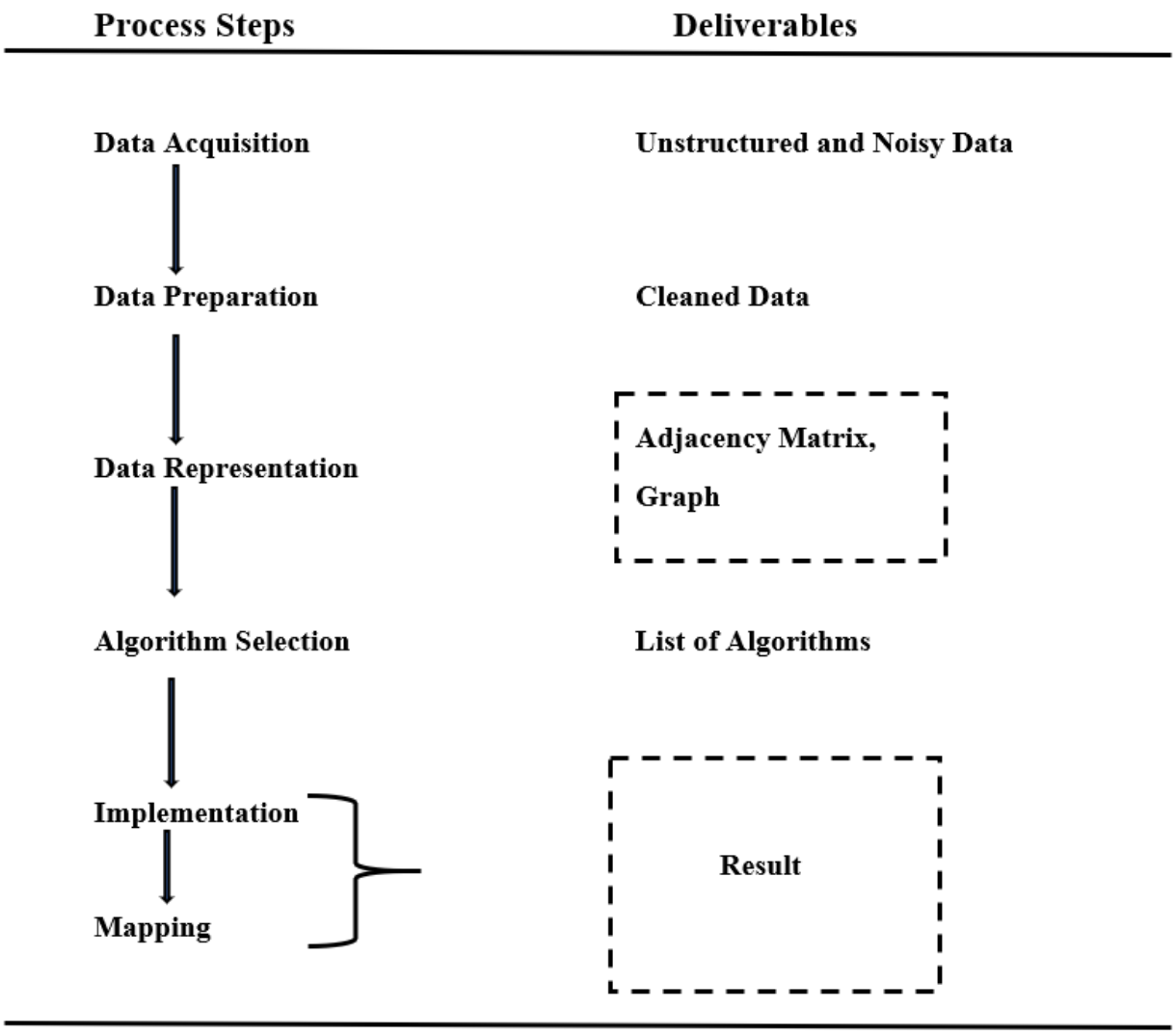


Figure 4.3: Community detection process model in Facebook

As shown in figure 4.3, community detection starts with procuring or acquiring data. In this case, the researchers crawled data from Facebook. The output of this phase is unstructured (noisy) crawled data.

The second phase shown in the model is data preparation. Data preparation is all about cleaning noises from the data. The prepared data then can be exported to any format that can easily be imported when a need arises.

The third phase is data representation. Community detection problems usually need data to be populated on graphs. The graph data structure, in turn, needs a matrix that shows the relationship

between vertices of a network or graph. Therefore, researchers need to construct both the matrix and the graph.

The fourth phase is selecting algorithms that could detect communities from graphs. Researchers may develop algorithms by themselves or can search for them from literature. The output of this phase is a list of algorithms selected for an experiment or an algorithm developed by the researcher/s. After selecting algorithms or developing them on their own, researchers can undergo their experiments. In the process, they should record and describe the results obtained from experiments. Implementation, in this case, means detecting communities from networks.

The final phase in our process model is mapping. Mapping is used in this context to mean identifying mechanisms to show or understand the relationships between various applications (they can be policies, advertisings, product, or service lists) and targeted communities. The output of this phase is a mechanism or way to map the application with a targeted community.

Once the prototype is designed and the required workflow to undergo the experiment is known, the next job is undergoing the main experiment to evaluate the concepts included in the model following the procedure indicated by the process flow model as shown in figure 4.3. In addition to this, the process model is important to help design the final model.

4.3. Development

4.3.1. Getting the Environment Ready

The researchers used python 3 to undergo the experiment. Python was selected because of its short learning curve and its support of many modules necessary for community detection. So, the first task they did was installing a platform to be used as a python work environment. For this, Jupyter Notebook from Anaconda cloud was chosen. Jupyter Notebook is chosen because it is user-friendly and supports many libraries to handle problems that require data science solutions. Community detection is one of those problems which need data science solutions.

After installing Jupyter Notebook, the next task was installing necessary libraries/modules. The modules installed were ‘request’ to scrape pages from Facebook, ‘BeautifulSoup’ to present the data understandably extracted from Facebook, ‘networkx’ to construct and manage graphs and to run community detection algorithms, ‘pandas’ to work with different data formats such as lists, dictionaries and data frames, community to run the modularity function. Once they installed the

necessary libraries, the next job was importing them to the points we need them during our work. All the necessary modules are imported means we can directly do the next major tasks such as data extraction, data preparation, and implementing community detection algorithms.

4.3.2. Data Acquisition

The data source for this research is Facebook. As we all know Facebook has closed extracting data from its page formally. The only formal way to extract data from Facebook pages is by creating projects using its graph API and exposing each phase of the project to the Facebook application management team for monitoring. This process may be helpful to extract data, but it is totally difficult to achieve in Ethiopia. It is because the project must follow all the requirements of a project demanded by Facebook. Here, we are doing research, not a project, so that undergoing research with a requirement of a project is difficult to achieve. Therefore, following other mechanisms is important to get the data needed to do the research. In this research, the required data was extracted by developing a python script that can crawl the required information from Facebook pages. To crawl the required information from Facebook pages, we followed the politeness policy of web owners. Figure 4.4 shows the snapshot of the sample crawling script.

```
my_url = "https://www.facebook.com/abebe.betemariam/likes_all"

uclient = req(my_url)

page_html = uclient.read()


uclient.close()

page_soup = bs(page_html,"html.parser")
```

Figure 4.4: Sample crawling script

In the above code snippet, the variable ‘my_url’ in the first line is used to store the URL of a specific individual's page which can show the pages he/she liked. The second line is used to request and download the data in the URL described in the first line to an offline variable. The third line is used to read the data from the variable in which the offline data is stored. The fourth line is simply used to close the connection between our scrapper code and Facebook pages. The fifth line is used to put the data in HTML format since the data to be extracted in later stages is better to be

displayed as content in HTML tags. The last line is prettification which is used to show the data as an exact HTML page for easy understanding.

The following is a sample page extracted from Facebook.

```
<a href="https://www.facebook.com/Ethiopian.DJ/">
 Ethiopian DJ የኢትዮጵያ ሙዚቃ
</a>
,
<a href="https://www.facebook.com/EthioVoice.net/">
 EthioVoice.net
</a>
,
<a href="https://www.facebook.com/yenetatubedotnet/">
 Yeneta tube የኔታ  ቲዩብ
</a>
,
<a href="https://www.facebook.com/mereja.media/">
 Mereja.com
</a>

<a href="https://www.facebook.com/Zehabesha/">
 Zehabesha
</a>
,
<a href="https://www.facebook.com/yegnatube/">
 Yegna Tube -  የኛ ቲዩብ
</a>
,
<a href="https://www.facebook.com/SocceeTV/">
 Soccee
</a>
,
<a href="https://www.facebook.com/Addidout/">
 Addisout.Com
</a>
,
<a href="https://www.facebook.com/mirwalimuhammadmufheri/">

<a href="https://www.facebook.com/HabeshaEntertainmentvideo/">
 Habesha - መዝናኛ
</a>
,
<a href="https://www.facebook.com/Habeshadish1/">
 Habesha Dish ሀበሻ ዲሽ እና ሪሲቨር ቴክኖሎጂ 1
</a>
,
<a href="https://www.facebook.com/bunamedia/">
 Buna Media - ቡና ሚዲያ
</a>
```

Figure 4.5: Sample pages extracted from Facebook

However, as shown in Figure 4.5, the extracted page is full of noise containing much of unnecessary tags together with the necessary ones. These unnecessary tags cannot give any additional meaning to our experiment. They even add noise to the data to be used in the analysis task planned in this research. Hence the whole page content is downloaded, the next task is extracting the exact text from the purified page content. To extract the data needed, first understanding the HTML structure of the data downloaded is necessary. Here, HTML components called tags play an important role since data is placed inside these tags.

As seen in the figure above, the name of each page is contained in an html tag called 'href 'or 'a' inside the main tag called 'div' which has a class value of 'fsl fwb fcb'. This tells us that to get the page names on Facebook, we should first locate our pointer to the ‘div’ tag hiving a class value of 'fsl fwb fcb' and then take our pointer further to the ‘href’ tag holding the page name needed. The researchers did the following to do so.

```
link = []

soup = page_soup.find("div",{"class":"fsl fwb fcb"})

for con1 in soup.find_all("a"):
    link.append(con1.text)
```

Figure 4.6: Sample python script to clean the noises in the crawled pages

As shown in the code snippet above, ‘find’ and ‘find_all’ functions of the python BeautifulSoup library are used as pointers to locate positions of the ‘div‘ and ‘href’ tags respectively. Since several ‘href’ tags are handled under the ‘div’ tag having a class value of 'fsl fwb fcb', accessing them through a loop data structure is necessary.

The output of the above code gives us the exact data extracted from the downloaded page content stored in a python data structure called the ‘list’.

Through this mechanism, we can extract lists of Facebook pages liked by a single individual at a single time of extraction. But our proposed system needs pages from more than one individual. Therefore, combining sets of pages liked by several individuals is required. To make this combination, we first stored data extracted from one individual in excel format, and then we appended the data of the rest individuals to data saved before.

Let's have a look at a sample list of pages liked by a single individual extracted in the process.

```
In [933]: link
```

Out[933]:

	Name
0	Kuku-Sebesebie
1	Geez lehulum - Abebe Betemariam
2	JTV Ethiopia - ጆሲ ቲቪ
3	መርካቶ hellomerkato.com
4	ማህሌት የአብነት ትምህርትን በቀላል መንገድ ይማሩ
5	Engineer Kitaw Ejigu
6	Yeneta tube የኔታ ቲዩብ
7	zikkir.org - Ethiopian Orthodox Tewahedo Church
8	ሰብለ ጎይለአብ
9	Tewahedo haymanote
10	Tewnet.com
11	Men's Journal + Fitness
12	ውስጥ አዋቂ
13	Elizabeth Dinku - Betengna
14	AhaduTube
15	English Conversation Club
16	Temesegen Amelache Temesegen Amelache
17	Jtv Ethiopia
18	Daniel Kibret Blog
19	Daniel Kibret Views
20	Reyot - ርዕዮት
21	Zemedkun Bekele
22	Daniel Kibret Views
23	Hilary Clinton for President 2016
24	Only for WRIE Enginers
25	Zehabesha
26	Haro Tube - ሐሮ ቱብ

Figure 4.7: Crawled Facebook pages after cleaning

The column heading 'Name' refers to the name of an individual whose name is anonymized for privacy reasons. It is a user on Facebook whereas the texts displayed under the header are the pages liked by the name given in bracket.

Lastly, the extracted data is exported into CSV files.

```
posts.to_csv('C:/Users/BadimawTerefe/Desktop/MyCrawling/Name.csv',encoding='utf-16')
```

4.3.3. Data Representation

4.3.3.1. Preparing the Matrix

The problem of community detection in this research is computed from one relation (i.e. the pages that the user liked and the user itself). This relation can be represented by a matrix which will be later converted into a graph data structure containing nodes and edges. A node can be any data used as a name to an entity (i.e. it can be any Facebook user or a Facebook page in this particular case). But an edge is formed among nodes if there is a 'like' relationship between one another. This needs creating a matrix to show there exists a relationship between nodes.

The matrix we created has Facebook users as row headings and liked pages as column headings and a value of 1 if there is an intersection between rows and columns and a value of 0 if there is no intersection. The code used to do so was:

```
r=len(tt.columns)
rr=len(tt)
```

```
for k in range(r):
    for kk in range(rr):
        m=tt.iloc[kk,k]
        mm=len(a1.columns)
        n=len(a1)
        if m in a1.columns:
            dd=a1.columns.get_loc(m)
            l=n-1
            a1.iloc[l,dd]=1
        else:
            a1[m]=0
            dd=a1.columns.get_loc(m)
            l=n-1
            a1.iloc[l,0]="Name"
            a1.iloc[l,dd]=1
```

Figure 4.8: Code snippet used to represent data into a matrix form

Where:

tt→ name of a new data file extracted as a link data structure and converted to a data frame for easily manipulation of tabular data having row and column values appended to a previous data frame

a1→ name of a data file read from excel and converted to data frame and new rows and columns added on to it i.e. the new data frame to work with

r → number of columns in the data frame to be appended on a1

rr → number of rows in the data frame to be appended on a1

m→ the location or index of the value in the intersection of the k^{th} column and the kk^{th} row

mm→ number of columns in the new data frame i.e. a1

n→ number of rows in the new data frame i.e. a1

dd→ the index of the column captured from the data frame to be appended in the new data frame (a1) if it exists

l→the last row of a1 since row and column index starts from 0 and end at n-1

iloc and get_iloc→ pandas functions to get the index of a certain value in a data frame

NB: we described two data frames here because we did matrix preparation and appending newly extracted page information simultaneously as described in subtitle 4.4.1.

Sample matrix representation:

Name	Injibara University Registrar Office	Wore Negari	ሂወት ያላንቺ	JTV Ethiopia - ጆሲ ቲቪ	Spymaster	Mesay buna	Bahirdar university	Adama Science And Technology University - ASTU	BULE HORA UNIVERSITY	...	Tuba Buyukustun Universal fans	Tuba Buyukustun - Bulgarian fans club (official)	E
AlehegnAbetie	1	1	1	1	1	1	1	1	1	...	0	0	
BirhanuEnyew	0	0	0	0	0	0	0	0	0	...	0	0	
MekonnenKassye	0	0	0	0	0	0	0	0	0	...	0	0	
YalemworkAbebe	0	0	0	0	0	0	0	0	0	...	0	0	
AmareAbewa	0	0	0	0	0	0	0	0	0	...	0	0	
DagimMelese	0	0	0	0	0	0	0	0	0	...	0	0	
MelisewBayu	0	0	0	0	0	0	0	0	0	...	0	0	
MareyeZeleke	0	0	0	0	0	0	0	0	0	...	0	0	
AbunuTesfaw	0	0	0	0	0	0	0	0	0	...	0	0	
AntenayehuAbawa	0	0	0	0	0	0	0	0	0	...	0	0	
GetahunEsubalew	0	0	0	0	0	0	0	0	0	...	0	0	
AlemayehuSemahegn	0	0	0	0	0	0	0	0	0	...	0	0	
BayushAlemayehu	0	0	0	0	0	0	0	0	0	...	0	0	
MulukenMinuye	0	0	0	0	0	0	0	0	0	...	0	0	
DessieGashu	0	0	0	0	0	0	0	0	0	...	0	0	
AnileyBekele	0	0	0	0	0	0	0	0	0	...	0	0	
AmareMekonnen	0	0	0	0	0	0	0	0	0	...	0	0	
GetawAyalew	0	0	0	0	0	0	0	0	0	...	0	0	
ኤደንየዱባይዋቃውጤ	0	0	0	0	0	0	0	0	0	...	0	0	
SemrtBitewo	0	0	0	0	0	0	0	0	0	...	0	0	
SelamDelie	0	0	0	0	0	0	0	0	0	...	0	0	
YibaTilahun	0	0	0	0	0	0	0	0	0	...	0	0	
ShumetArega	0	0	0	1	0	0	0	0	0	...	0	0	
BethelihemGetachew	0	0	0	0	0	0	0	0	0	...	0	0	
TesfaTilahun	0	0	0	1	0	0	0	0	0	...	0	0	
DejenTilahun	0	0	0	0	0	0	0	0	0	...	0	0	
BraManXII	0	0	0	0	0	0	0	0	0	...	0	0	
AyehuBerhan	0	0	0	0	0	0	0	0	0	...	0	0	
ErmiasYedngilLij	0	0	0	0	0	0	0	0	0	...	0	0	
LeyWUbalem	0	0	0	0	0	0	0	0	0	...	0	0	
MUlukenKasie	0	0	0	0	0	0	0	0	0	...	0	0	
ኦልፋኦሜጋአባጎዳ	0	0	0	0	0	0	0	0	0	...	0	0	
MusaYimam	0	0	0	0	0	0	0	0	0	...	0	0	
SurafelDagne	0	0	0	0	0	0	0	0	0	...	0	0	
AgerieLijalem	0	0	0	0	0	0	0	0	0	...	1	1	

Figure 4.9: Sample matrix representation

As shown in the figure above, the name of individuals under the 'Name' column is a list of individuals who have pages liked on Facebook and other column names such as 'Injibara University Registrar Office', 'Wore Negari' and 'Spymaster' are lists of pages liked by individuals. If individuals liked a page, 1 is placed at their intersection points and 0 is placed at their intersection points otherwise. This representation gives sufficient information about constructing the graph. It is in such a way that to construct graphs we need edges and edges can only exist if there is a relationship between nodes i.e. if the value of the relationship is 1.

4.3.3.2. Constructing the Graph

A graph in this research is a data structure that represents data in the form of nodes and edges as G = (N, C) where N stands for user names and page names and C stands for the like relationship between users and pages. It is the ideal data structure for network data representation which is the cause for the idea of community detection.

We created three graphs as follows:

```
G1=nx.Graph()
G2=nx.Graph()
G3=nx.Graph()
```

Figure 4.10: A sample code to create a graph in networkx

After creating the graphs, the next task is to populate the graph with node and edge data. Here, nodes hold Facebook users and the name of pages. The node and edge information to construct the graph/s is taken from the matrix structure prepared in the previous phase. As shown in the phase, the matrix constructed above is used to show the presence or absence of relationships between users and Facebook pages. A value 1 represents the presence of relationships between the nodes and a value 0 represents the absence of a relationship between the nodes. Therefore, edges will only be constructed from relationships having 1 as a value since and relationships with a value of 0 were discarded during the construction of the graph.

Since our dataset is presented in a matrix form, assigning user names and page names to nodes of the same graph is difficult to some extent. So, we separated the user name part of the matrix into

a separate dataset and assigned it into G1 and the page name part of the dataset into G2 as nodes. We did it as follows:

```
t=df['Name']
```

```
b=pd.DataFrame(t)
```

```
for i in b.columns:
    G1.add_node(i)
```

```
for j in df.columns:
    if(j=='Name'):
        a=1+2
    else:
        G2.add_node(j)
```

Figure 4.11: A sample code to add nodes to a graph

Now, we have two graphs G1 and G2 containing user names and page names as nodes respectively. Constructing graphs is not only about creating the graph and assigning node data to it. The main task though is creating a relationship or connection among the nodes. Here comes the concept of edges to show the connection between nodes. It is done in the following way:

```
a=0
for h in G1.nodes:
    for k in G2.nodes:
        mm=df.columns.get_loc(k)
        if(df.iloc[a,mm]==1):
            G3.add_edges_from([(h,k)])
    a=a+1
```

Figure 4.12: A sample code to add edges to a graph

As shown in the code snippet above, G3 was constructed from the two graphs constructed before each containing node information. But what is important to notice here is that the edges of the graph G3 were formed looking at the relationship features in the matrix. If we look at the fifth and sixth lines in the code snippet above, it says an edge between nodes ‘h’ and ‘k’ will be added to G3 only if their intersection value in the matrix is 1. So, nodes with intersection values 0 were not

used to construct the graph G3 which makes the graph contain nodes connected to other nodes at least once.

Now, we had constructed a graph G3 containing both nodes and edges. We can view various information regarding the graph from now onwards. For instance, the number of nodes and edges the graph G3 has can be viewed through the following code.

```
len(G3.edges()),len(G3.nodes())
```

```
(52570, 17095)
```

Figure 4.13: A sample code to show the number of edges and nodes in a graph

The output of the above code tells us the graph G3 has 52570 edges and 17,095 nodes. We can also view sample nodes and edges in the graph. For instance, viewing nodes can be done as follows:

```
G3.nodes()
```

```
NodeView(('Name', 'Injibara University Registrar Office', 'Wore Negari', 'ሂወት ያላንቺ', 'JTV Ethiopia - ጆሲ ቲቪ', 'Spymaster', 'M
esay buna', 'Bahirdar university', 'Adama Science And Technology University - ASTU', 'BULE HORA UNIVERSITY', 'Wolkite Univers
ity', 'Soccer Ethiopia - ሶከር ኢትዮጵያ', 'Secure Ethiopia', 'AfriSci Network', 'Happy valantine day', 'Bahirdar university.1', 'A
FP News Agency', 'FIFA World Cup', 'Shashemene FANA FM 103.4', 'Core Integrate', 'United Bank S.C.', '10X Growth Conference 2
018', 'Daymond John', 'CNBC Make It', 'Dawit alemayehu DAVE', 'Eyob Dawit', 'ከዕውቀት አድማስ 《 የኔ ገጽ 》', 'Mayor Office of Addis A
baba', 'ዘሪቱ ከበደ', 'Bnews Life', 'MTU school of computing and informatics', 'Office of Deputy Prime Minister of Ethiopia', 'Of
fice of the Prime Minister-Ethiopia', 'AbyssiniaDaily', 'IMT', 'China Xinhua News', 'Oromo Democratic Party /ODP', 'Ethiopian
Kids', 'Mizan tepi University Real page', 'Hawassa University', 'Energy Zone', 'SYG Jobs', 'ARIF PAGE አሪፍ ፔጅ', 'Beihang Unive
rsity', 'EuroEducation', 'Leaked News አፈትላኪ ዜናዎች', 'Malaysia Airports', 'IMF Finance & Development Magazine', 'Rift vally uni
versity', 'Ethio fm107.8', 'Computer Science', 'Cybrary', 'Ethiopian Yellow Pages', 'Rift Valley University', 'Ethiopian Yout
h Democrats', 'Eritrean Seaman Union', 'Best Locations for Vacations', 'Yared Negu', 'Gofere', 'Megabe Haddis Rodas Tadese',
'ሊቃውንትን እንጠብቅ', 'Wollo university', 'WDU Cloud Computing', 'Ethiopia university  Wachamo.', 'Vacancy in Ethiopia', 'Mizan te
pi university', 'Jinka University', 'Debre Tabor University', 'Jinka University.1', 'Arsi university- medical college', 'Wera
be universitiy', 'MTU', 'Worabe University students page  / ወራቤ', 'Arsi University', 'Wolkite University.1', 'Bule Hora Unive
rsity', 'Wallo university', 'Bahirdar university.2', 'Gambela university', 'Wolkite University Computing&informatic College',
'Mizan Tepi University', 'Jinka University Student Fellowship', 'Gonder university 2011', 'Ethiopian university', 'Kebir Deha
r university teacher association', 'Werabe University Procurement And Financce Administration Directorate', 'Injibara univers
ity', 'Bule hora university', 'Arbaminch university', 'Mekdela amba university comunication afrair', 'Ethiopian University En
trance Examination', 'Bule hora university.1', 'Bule Hora university', 'Biology Lovers', 'Arba Minch University', 'Dire dawa
```

Figure 4.14: Sample nodes from graph G3

The code written to view the list of nodes (G3. nodes) displays all the nodes contained in the graph. The above image only shows some part of the whole output to be shown as a sample.

4.3.4. Selecting the Best Algorithm for Community Detection

Researchers choose community detection algorithms based on their quality. The quality of a community detection algorithm can be characterized in many features (Fu, Huang, & Sun, 2017). The first feature is execution time which describes the amount of time required to group a given dataset into communities. The second feature is the type of processing system the algorithm can work on. Many community detection algorithms only work on a distributed processing system since they require much storage and processing capacity whereas other algorithms especially the newly developed ones can run on single laptop computers that have graphical processing units (GPUs). The third feature is the ability to detect overlapping communities. This feature is about the ability of an algorithm to assign a node to more than one community. The fourth feature is the ability to show a graph or network structure diagrammatically which is called network visualization. The last feature is its ability to support evaluation techniques (Fu, Huang, & Sun, 2017).

Among the features described in the paragraph above, all except the evaluation technique are used to compare each community in the literature. But the evaluation technique of the algorithms varies from one category of algorithms to the other categories (algorithms for community detection usually fall under categories such as clique based, distance-based, label propagation-based, modularity optimization-based, and the like). As mentioned in the literature review part (chapter two) of this thesis, modularity is considered by many as the strongest quality measure of a community detection algorithm. Therefore, in this research, four known algorithms from the modularity optimization category are chosen and compared against all the features mentioned above.

There are basically, four reasons why researchers usually prefer modularity-based algorithms to other categories of community detection algorithms (Karata¸s & S¸ahin, 2018).

1. They are prevalent
2. They are easy to implement
3. They need relatively low running time
4. They are good for systems that require big data such as social networks

The four algorithms are the 'girvan_newman' algorithm, the 'greedy_modularity_communities' algorithm, the '_naive_greedy_modularity_communities' algorithm, and finally the 'Louvain'

algorithm. The researchers experimented performance of these four algorithms on five datasets. The first dataset has 5439 nodes and 8230 edges. The second dataset has 11602 nodes and 21246 edges. The third dataset has 16924 nodes and 43043 edges. The fourth dataset has 17043 nodes and 50383 edges. The final dataset has 17095 nodes and 52570 edges.

The datasets mentioned in the above paragraph are only different in their size (the number of nodes and edges). That means, when we move from dataset one (1) to dataset five (5), the next dataset is created by adding a certain number of nodes and edges on top of the previous dataset. For instance, dataset two (2) is formed by adding 6,163 nodes and 13,016 edges on top of dataset one (1). The number of nodes and edges added in creating each next dataset was randomly formed and was decided by the amount of data extracted in each phase of the experiment. These five datasets were prepared to enable the researchers to undergo as many experiments as possible since the experiment involves comparing four algorithms and selecting the best one among those four algorithms which in turn needs more than one experiment to avoid the possibility of a certain performance of an algorithm by chance. This means the performance of algorithms may differ when the size of the dataset varies. Therefore, concluding the performance of algorithms with only a single experiment is not appropriate. In addition to this, since the structure of the data in all the datasets is similar, the size of the dataset is the criterion that can significantly differentiate the behavior of the dataset.

All these algorithms have some features in common. First, all can run with a computer system having at least 2 GB GPU but it is difficult to run them on a computer system with no GPU especially if the dataset is large. Second, all the algorithms support the so-called modularity function for measuring the quality of the partitioned graphs or communities. Third, all can be easily integrated with the ‘networkx’ and ‘matplotlib’ libraries of python to visualize the structure of the network or communities.

Although those algorithms are similar in the criteria mentioned above, there is one criterion that differs them. This criterion is the capability of the algorithm to detect overlapping communities. In this case, only the 'Louvain' algorithm can perform well.

Let’s first import the algorithms to our work environment.

```
from community import community_louvain

from networkx.algorithms.community import greedy_modularity_communities

from networkx.algorithms.community import _naive_greedy_modularity_communities

from networkx.algorithms.community import girvan_newman

from community import modularity


import time
```

Figure 4.15: Code used to import necessary modules

After importing all the required libraries to our working environment, we can directly go to implementing the algorithms selected to be analyzed in this paper. At each stage of the experiment, we have tried to read the datasets previously prepared and stored them on graphs having a varying number of nodes and edges. Later, we have run the four algorithms on the graphs.

Tasks such as reading datasets and storing them in graphs and running algorithms are believed better to be done at every stage of the experiment because the graphs used as inputs in each stage of the experiment are different. But some common tasks were required to be performed at each stage in implementing each algorithm. These are calculating the time required by an algorithm to detect communities from graphs, the number of communities detected by each algorithm, and the modularity value of the detected communities. Therefore, instead of rewriting the codes for these common issues, it is better to describe here once to avoid unnecessary repetitions.

The amount of time required for an algorithm to detect communities was calculated as follows:

```
start=time.time()

The code to detect communities from a graph by an algorithm

end=time.time()

print(end-start)
```

Figure 4.16: Code used to calculate elapsed time

Here, what is required is writing the code to run a community detection algorithm between the two-time functions stored on variables called '*start'* and '*end'*.

Furthermore, the modularity value of communities can be calculated by calling the 'modularity' function of the python 'community' module. The function takes two arguments namely the partition or community name and the original graph. It can be done as follows:

```
mod=community.modularity(part,G)
```

Figure 4.17: Code used to calculate modularity

Where 'part' is the name of the partition or the community and 'G' is the name of the graph before getting partitioned.

Finally, the number of communities detected by an algorithm can be calculated in the following way.

```
commCount=0
```

```
new_part = {}
for k, v in part.items():
            new_part[v] = new_part.get(v, [])
            new_part[v].append(k)
            new_part[v].sort()
strComms = [new_part[x] for x in new_part]
strComms.sort(key=len, reverse=True)
commCount+=len(strComms)
```

Figure 4.18: Code used to calculate the number of communities or modularity classes in a graph

Where 'part' indicates the name of the variable which holds the set of communities detected by an algorithm.

4.3.4.1. Experiment 1

In the first experiment, those four algorithms mentioned above experimented on a dataset having 5439 nodes and 8230 edges and the time taken to partition the graph into communities, the number of communities detected by each algorithm, and the modularity value of the communities detected by each algorithm were recorded. Let's first read the dataset so that we can directly use it in implementing each algorithm not to write it again and again.

```
G81=nx.read_gexf('C:/Users/Badim/Desktop/ResearchFiles/GraphFiles/Graph81.gexf')
```

```
len(G81.nodes()),len(G81.edges())
(5439, 8230)
```

Figure 4.19: Dataset for experiment 1

A. The Girvan-Newman Algorithm

```
pGN1=girvan_newman(G81)
```

Networkx module of python supports many of the community detection algorithms especially those algorithms that work based on the concept of modularity optimization. The 'girvan_newman' algorithm is not different. It can simply be imported from the networkx module and run taking a graph as an input. It took 2.260 seconds for the algorithm to detect communities 21 communities from the dataset containing 5439 nodes and 8230 edges with a modularity value of 0.522.

The above result especially the time taken by the algorithm to detect communities (2.260 seconds) and the modularity value (0.522) has something to tell us about the performance of the 'girvan_newman' algorithm. The first value (the processing time) tells us that the algorithm has detected communities with a very short period considering the complexity of graph analytics. Whereas, the second value (modularity value) tells us that the algorithm detected communities with a very good quality which is within the range of 0.3-0.7. This means the newly created graph structure has nodes that are more strongly connected inside communities and nodes weakly connected between different communities.

The other result gained in the above result was the number of communities detected from the graph (21). Unlike the other two results discussed above, the number of communities has no importance in measuring the performance of an algorithm. Therefore, no attention was given to this metric in each phase of the experiment.

B. The Greedy_Modularity_Communities Algorithm

We have read our dataset above and stored it on a variable called G81. Therefore, we can directly apply the algorithm instead of re-reading the file here again.

```
pGMC=greedy_modularity_communities(G81)
```

The ‘greedy_modularity_communities’ algorithm detected 33 communities from the dataset containing 5439 nodes and 8230 edges. It took 14.457 seconds to detect those 33 communities with a modularity value of 0.581.

As indicated by the numerical results shown above, the ‘greedy_modularity_communities’ algorithm is much slower than the ‘girvan_newman’ algorithm in terms of processing time. But it detected communities with better quality. However, detecting 33 communities compared to the 21 communities detected by its ‘girvan_newman’ counterpart doesn’t make it a better algorithm.

C. The _Naive_Greedy_Modularity_Communities Algorithm

Once again, we are not expected to re-read our dataset here. Let’s run this algorithm on the G81.

```
pNGM1=_naive_greedy_modularity_communities(G81)
```

The ‘_naive_greedy_modularity_communities’ (fast_greedy_algorithm) algorithm detected 19 communities from the dataset. It took 1.309 seconds to detect these communities with a modularity value of 0.523. As the name indicates, this algorithm is the fast version of the ‘greedy_modularity_communities’ algorithm and the processing time required proves that too as there is a huge difference between the execution times of the two algorithms and it is even faster than the ‘girvan_newman’ algorithm. But in terms of its quality to detect communities, it is more in the nearby to the ‘girvan_newman’ algorithm than to the ‘greedy_modularity_communities’ algorithm.

D. The Louvain Algorithm

```
pL1=community_louvain.best_partition(G81)
```

The 'Louvain' algorithm can allow a node to be a member of more than one community since it can detect overlapping communities. Here, the 'best_partition' function is one of the attributes of python's 'community_louvain' package. The algorithm took 4.510 seconds to detect 33 communities. The modularity value was 0.648. In terms of the processing time required to detect communities, it is faster than the ‘greedy_modularity_communities’ algorithm but it is a bit slower

than both the ‘girvan_newman’ and ‘_naive_greedy_modularity_communities’ algorithms. But in terms of the quality of the detected communities (which is the most important metric of all), it is the algorithm that detected the best quality communities among the four algorithms.

Summary of Experiment 1

Algorithm	Number of Communities Detected	Modularity Value	Time Taken (Seconds)
Girvan_Newman	21	0.522	2.260
Greedy_Modularity_Communities	33	0.581	14.457
_Naive_Greedy_Modularity_Communities	19	0.523	1.309
Louvain	33	0.648	4.510

Table 4.1: Summary of the first experiment

Let’s rank the algorithms based on the two important metrics used to compare them during the first experiment:

Modularity Value

1. Louvain
2. Greedy_modularity_communities
3. _Naive_greedy_modularity_communities
4. Girvan_newman

Computation Time

1. _Naive_greedy_modularity_communities
2. Girvan_newman
3. Louvain
4. Greedy_modularity_communities

Therefore, according to the first experiment, the ‘_Naive_greedy_modularity_communities’ is the best algorithm when it comes to computation time and the ‘Louvain’ algorithm is the best algorithm when it comes to the quality of detected communities (modularity value).

But for us, concluding the performance of these four algorithms with one experiment (using a single dataset) is not appropriate and the results might be caused by some chance. As a solution

for this, we proposed undergoing the other four experiments (using a new dataset for each experiment to see the reaction of each algorithm when the size of the dataset increases). Computing average values for computation time and modularity value at the end of the following four experiments together with the other metrics discussed in section 4.5 will be used to conclude. Now, let's undergo the remaining four experiments and record computation time and modularity values gained in each of the experiments so that we can compute the average values at the end.

4.3.4.2. Experiment 2

The second experiment was undergone on a dataset containing 11602 nodes and 21246 edges. We have a dataset prepared before on the desk. Before running the four algorithms, lets' first read the dataset and store it into a graph.

```
G261=nx.read_gexf('C:/Users/Badim/Desktop/ResearchFiles/GraphFiles/Graph261.gexf')
```

```
len(G261.nodes()),len(G261.edges())
(11602, 21246)
```

Figure 4.20: Dataset for experiment 2

Now we have read the dataset and stored it into a graph called G261 so that we can use G261 as input for each algorithm.

A. The Girvan-Newman Algorithm

```
pGN2=girvan_newman(G261)
```

The algorithm took approximately about 1.842 seconds to detect 28 communities from the dataset containing 11602 nodes and 21246 edges with a modularity value of 0.437.

It was a little bit surprising to see less execution time in this experiment than the previous one. This is because this experiment was done on a dataset having more size than experiment one. What we were expecting was that an increase in the size of the dataset will increase execution time, but what we got in the result was the reverse. However, whether or not it is usually true will be determined by looking at the experiments to come next.

Concerning the modularity value, although the value gained in the result falls in the accepted range, it is not comparable to the one gained during the first experiment. This indicates that the communities formed during this experiment have more nodes interacting with nodes in other communities than the communities formed in the first experiment.

B. The Greedy_Modularity_Communities Algorithm

```
pGMC2=greedy_modularity_communities(G261)
```

The 'greedy_modularity_communities' algorithm took 151.809 seconds to detect communities from the dataset described above. The number of communities detected from the dataset with this algorithm was 51 with a modularity value of 0.488. Unlike in the case of the 'girvan_newman' algorithm, the time required to detect communities by the 'greedy_modularity_communities' algorithm was much greater when the dataset increased. But the modularity value dropped when the size of the dataset increased like in the case of the 'girvan_newman' algorithm.

C. The _Naive_Greedy_Modularity_Communities Algorithm

```
pNGM2=_naive_greedy_modularity_communities(G261)
```

The '_naive_greedy_modularity_communities' algorithm took 2.909 seconds to detect 44 communities from the dataset with a modularity value of 0.456.

It seems the 'girvan_newman' algorithm was the odd one in this experiment (experiment 2) with execution time since the other algorithms showed an increase in execution time when the size of the dataset increased. But when it comes to modularity value, it is similar in all the algorithms. i.e. there comes a decrease in modularity value as the size of the dataset increases. This seems because the graph dataset becomes sparser when a new number of nodes and edges are added to the previous one.

D. The Louvain Algorithm

```
pL2=community_louvain.best_partition(G261)
```

The 'Louvain' algorithm partitioned the graph into 51 communities with a modularity value of 0.565 which is a relatively good value. The time taken to do was 8.316 seconds. Similar to the other algorithms (apart from the 'girvan_newman' algorithm), the results gained from the 'Louvain' algorithm have also shown some differences. That is the execution time increased and the modularity value decreased as the size of the dataset increased.

Summary of Experiment 2

Algorithm	Number of Communities Detected	Modularity Value	Time Taken (Seconds)
Girvan_Newman	28	0.437	1.842
Greedy_Modularity_Communities	51	0.488	151.809
_Naive_Greedy_Modularity_Communities	44	0.456	2.909
Louvain	51	0.565	8.316

Table 4.2: Summary of the second experiment

Now, let's rank the performance of the four algorithms based on the results gained in experiment two.

Computation Time

1. Girvan_Newman
2. _Naive_Greedy_Modularity_Communities
3. Louvain
4. Greedy_Modularity_Communities

Modularity Value

1. Louvain
2. Greedy_Modularity_Communities
3. _Naive_Greedy_Modularity_Communities
4. Girvan_newman

As shown in Table 4.2 above, when we compare the four algorithms based on the time required to partition the graph given at the beginning of the experiment into communities, the 'girvan_newman' algorithm is the fastest among the four algorithms, and the 'greedy_modularity_communities' algorithm is the slowest one. In this regard, the result gained

in this experiment has a little different from the first experiment and shows the importance of doing more than one experiment before concluding a certain algorithm is the best. In the previous experiment, the fastest algorithm was the '_naive_greedy_modularity_communities' algorithm. Thus, it is not possible to say either 'girvan_newman' or '_naive_greedy_modularity_communities' is the fastest algorithm when it comes to community detection unless we do some additional experiments.

Concerning modularity value, the 'Louvain' algorithm is the best algorithm and the 'girvan_newman' algorithm is the worst algorithm.

4.3.4.3. Experiment 3

The third experiment was done on a dataset containing 16924 nodes and 43043 edges. The dataset was ready in a format that can easily be manipulated by community detection algorithms since it was first prepared in a graph format. So, let's first read the dataset so that it will be used by the four algorithms.

```
G543=nx.read_gexf('C:/Users/Badim/Desktop/ResearchFiles/GraphFiles/Graph543.gexf')
```

```
len(G543.nodes()),len(G543.edges())
```

```
(16924, 43043)
```

Figure 4.21: Dataset for experiment 3

As indicated by the code snippet above, we have read the dataset and stored it on a graph called G543. The graph has 16924 nodes and 43043 edges and it is now ready to run community detection algorithms onto it.

A. The Girvan-Newman Algorithm

```
pGN3=girvan_newman(G543)
```

The 'girvan_newman' algorithm partitioned the graph with the number of nodes and edges aforementioned into 61 communities within 1.815 seconds. It detected communities with a modularity value of 0.395. The time required to detect communities is still decreasing as the size of the dataset increases. Whereas the modularity value decreased when the size of the dataset increased which indicates as the size of the dataset increases, the graph becomes sparser and there happen nodes in communities that have more interaction with nodes in other communities than in the previous dataset.

B. The Greedy_Modularity_Communities Algorithm

```
pGMC3=greedy_modularity_communities(G543)
```

The ‘greedy_modularity_communities’ algorithm took 567. 639 seconds to partition the graph in to 64 communities with a modularity value of 0.402. Unlike the ‘girvan_newman’ algorithm, the ‘greedy_modularity_communities’ algorithm requires more time to detect communities as the siz e of the dataset increases. But the two algorithms show similar properties in the case of the modu larity value i.e. the modularity value decreases as the size of the dataset increases.

C. The _Naive_Greedy_Modularity_Communities Algorithm

```
pNGM3=_naive_greedy_modularity_communities(G543)
```

The _naive_greedy_modularity_communities’ algorithm took 1. 76 seconds to partition the graph into 57 communities with a modularity value of 0.400. In terms of computation time, this algorithm also displayed a deviation while experimenting three (3). In the previous experiments, the computation time was increasing as the dataset was increasing, but now it detected communities with less time than the previous experiment. But in terms of modularity value, it is similar to the previous steps i.e. the modularity value decreased as the size of the dataset increased.

D. The Louvain Algorithm

```
pL3=community_louvain.best_partition(G543)
```

The Louvain algorithm took around 12 seconds to partition the graph into 37 communities with a modularity value of 0.444. The algorithm continued to show similar properties both in terms of computation time and modularity value. The computation time increased as the size of the dataset increased and the modularity value decreased as the size of the dataset increased.

Summary of Experiment 3

Algorithm	Number of Communities Detected	Modularity Value	Time Taken (Seconds)
Girvan_Newman	61	0.395	1.815
Greedy_Modularity_Communities	64	0.402	567. 639
_Naive_Greedy_Modularity_Communities	57	0.400	1.760
Louvain	51	0.444	11.920

Table 4.3: Summary of the third experiment

Now, let's rank the performance of the four algorithms based on the results gained in experiment three.

Based on Computation Time

1. _Naive_Greedy_Modularity_Communities
2. Girvan_Newman
3. Louvain
4. Greedy_Modularity_Communities

Based on Modularity Value

1. Louvain
2. Greedy_Modularity_Communities
3. _Naive_Greedy_Modularity_Communities
4. Girvan_Newman

Based on the results of experiment three (3) and the ranking of the algorithms shown above, the '_naive_greedy_modularity_communities' algorithm is the fastest algorithm to detect communities as opposed to the previous experiment which indicated that the 'girvan_newman' algorithm as the fastest algorithm. However, the 'greedy_modularity_communities' algorithm is the slowest algorithm to detect communities in all three experiments.

Concerning the modularity value, no difference has been shown by all the experiments. In all the cases, the Louvain algorithm is the one that detected communities with the best modularity value and the 'girvan_newman' algorithm is the one that detected communities with the worst modularity value.

Having done three experiments using datasets of a different size, two things became certain to the researchers.

The first thing is it is difficult to say a certain algorithm is the best in terms of computation time since the results gained from the previous three experiments are not uniform. In two of the three experiments, the '_naive_greedy_modularity_communities' algorithm was the best. Whereas in one of the three experiments, the 'girvan_newman' algorithm was the best. Therefore, to tell one of the above two algorithms is the best, we have to do additional experiments computing the average computation time as a better option to consider.

The second thing was that we can tell the 'Louvain' algorithm is the algorithm to detect communities with the best modularity value and the 'girvan_newman' algorithm is the one that detects communities with the worst modularity value since the results gained from all the three experiments uniformly showed his fact.

4.3.4.4. Experiment 4

This experiment was undergone on a dataset containing 17043 nodes and 50383 edges. Let's read the dataset first and then we can run the algorithms on it.

```
G662=nx.read_gexf('C:/Users/Badim/Desktop/ResearchFiles/GraphFiles/Graph662.gexf')
```

```
len(G662.nodes()),len(G662.edges())
```

```
(17043, 50383)
```

Figure 4.22: Dataset for experiment 4

Now we can go to the implementation of those four algorithms taking graph G662 as input.

A. The Girvan-Newman Algorithm

```
pGN4=girvan_newman(G662)
```

This algorithm partitioned the graph with 17043 nodes and 50383 edges into 55 communities with a modularity value of 0.322. The time required to detect these communities was 1.160 seconds. The results gained from all four experiments tell us something about the 'girvan_newman' algorithm. The thing is that both the computation time and the modularity value decreases as the size of the dataset increases.

B. The Greedy_Modularity_Communities Algorithm

```
pGMC4=greedy_modularity_communities(G662)
```

The 'greedy_modularity_communities' algorithm, as usual, has again become the one that took several seconds. It required 625.742 seconds to detect 54 communities from the graph with a modularity value of 0.375. Like the ‘girvan_newman’ algorithm, the four experiments can also say something about the 'greedy_modularity_communities' algorithm. That is as the size of the dataset increases, the computation time increases but the modularity value decreases.

C. The _Naive_Greedy_Modularity_Communities Algorithm

```
pNGM4=_naive_greedy_modularity_communities(G662)
```

The ‘_naive_greedy_modularity_communities’ algorithm detected 49 communities from the graph (G662) having 17043 nodes and 50383 edges within 1.993 seconds with a modularity value of 0.343. This algorithm is the one that is showing a variable computation time as the size of the dataset increases. It increases one time and decreases the other time. Therefore, it is difficult to conclude whether it is best to use this algorithm when the size of the dataset is small or large.

But in terms of modularity value, the results gained in all of the four experiments are similar to the other algorithms i.e. it decreases as the size of the dataset increases.

D. The Louvain Algorithm

```
pL4=community_louvain.best_partition(G662)
```

The 'Louvain' algorithm detected 36 communities from graph G662 with a modularity value of 0.404. The time required to detect those 36 communities was 15. 437 seconds. It follows the trend of the ‘greedy_modularity_communities’ algorithm in both computation time and modularity value aspects. As the size of the dataset increases, the computation time increases and the modularity value decreases.

Summary of Experiment 4

Algorithm	Number of Communities Detected	Modularity Value	Time Taken (Seconds)
Girvan_Newman	55	0.322	1.160
Greedy_Modularity_Communities	54	0.375	625.742
_Naive_Greedy_Modularity_Communities	49	0.343	1.993
Louvain	36	0.404	15.437

Table 4.4: Summary of the fourth experiment

Now, let's rank the performance of the four algorithms based on the results gained in experiment four.

Based on Computation Time

1. Girvan_Newman
2. _Naive_Greedy_Modularity_Communities
3. Louvain
4. Greedy_Modularity_Communities

Based Modularity Value

1. Louvain
2. Greedy_Modularity_Communities
3. _Naive_Greedy_Modularity_Communities
4. Girvan_Newman

The results gained from the experiment above shows that the four algorithms produced similar results with the second experiment as far as their ranking concerning computation time and modularity value is concerned. That means the 'girvan_newman' algorithm is the fastest algorithm and the 'greedy_modularity_communities' algorithm is the slowest algorithm to detect communities. On the other way, the 'Louvain' algorithm is the algorithm that detected communities with the highest modularity value and the 'girvan_newman' algorithm is the one that detected communities with the lowest modularity value.

Till this stage, we saw how the four algorithms respond to varying size datasets. As mentioned during summarizing experiment three, concerning computation time, it is still difficult to conclude a certain algorithm is the fastest to detect communities since the 'girvan_newman' algorithm and the '_naive_greedy_modularity_communities' algorithm are taking the lead interchangeably. But

concerning the modularity value, all the four experiments have assured that the 'Louvain' algorithm is the best algorithm and the 'girvan_newman' algorithm is the worst one. Therefore, undergoing one additional experiment is found necessary to see which algorithm detects communities with the fastest time.

4.3.4.5. Experiment 5

This is the last experiment undergone in this paper. It was done on a dataset containing 17095 nodes and 52570 edges. Let's first read the dataset and store it onto a graph data structure.

```
G715=nx.read_gexf('C:/Users/Badim/Desktop/ResearchFiles/GraphFiles/Graph715.gexf')
```

```
len(G715.nodes()),len(G715.edges())
```

```
(17095, 52570)
```

Figure 4.23: Dataset for experiment 5

Now, we can run the four algorithms on the graph called G715.

A. The Girvan_Newman Algorithm

```
pGN5=girvan_newman(G715)
```

The 'girvan_newman' algorithm detected 56 communities from graph G715 within 1.600 seconds with a modularity value of 0.323

B. The Greedy_Modularity_Communities Algorithm

```
pGMC5=greedy_modularity_communities(G715)
```

The 'greedy_modularity_communities' algorithms took 447.951 seconds to detect 52 communities from the graph G715 with a modularity value of 0. 351.

C. The _Naive_Greedy_Modularity_Communities Algorithm

```
pNGM5=_naive_greedy_modularity_communities(G715)
```

This algorithm detected 45 communities from the graph G715 within 1.441 seconds. The algorithm found those communities with a modularity value of 0.332.

D. The Louvain Algorithm

```
pL5=community_louvain.best_partition(G715)
```

The 'Louvain' algorithm detected 35 communities from the graph G715. The algorithm detected these communities with a modularity value of 0.393. The time taken to detect the communities was 31.921 seconds.

Summary of Experiment 5

Algorithm	Number of Communities Detected	Modularity Value	Time Taken (Seconds)
Girvan_Newman	56	0.323	1.600
Greedy_Modularity_Communities	52	0.351	447.951
_Naive_Greedy_Modularity_Communities	45	0.332	1.441
Louvain	35	0.393	31.921

Table 4.5: Summary of the fifth experiment

Now, let's rank the performance of the four algorithms based on the results gained in experiment five.

Based on Computation Time

1. _Naive_Greedy_Modularity_Communities
2. Girvan_Newman
3. Louvain
4. Greedy_Modularity_Communities

Based on Modularity Value

1. Louvain
2. Greedy_Modularity_Communities
3. _Naive_Greedy_Modularity_Communities
4. Girvan_Newman

So far, we have undergone five experiments to see how algorithms react in computation time and modularity value when we expose them to datasets of increasing size. Based on the results gained from those experiments, we can say the following.

A) As the size of the dataset increases, the modularity value of detected communities decreases. This happened due to the following fact. As we add new nodes and edges to our dataset, the resultant graph becomes sparser and results in many nodes that have many connections outside their community to be assigned into similar communities by algorithms. If this happens in many communities, then the modularity value will be less since modularity is all about ensuring nodes must have more relations with other nodes in their community than nodes outside their community.
B) As the size of the dataset increases, it is difficult to tell whether computation time increases or decreases. This is because some of the algorithms detected communities from large data sizes in a faster time than small data sizes and the others showed the reverse.
C) We can say a certain algorithm is the best and a certain algorithm is the worst with modularity metrics based on the results we saw so far. However, we can't say the same with the time metrics since all the five experiments didn't produce uniform measures to computation time. Therefore, we have to come up with another mechanism to solve this problem. The mechanism we come with was computing the average computation time gained from the five experiments and ranking them based on the average result.
D) The number of communities detected from a graph cannot indicate the quality of an algorithm. Therefore, much attention was not given to it during the whole experiment.

In the experiments, we saw only two criteria to select an algorithm as the best one. But at the beginning of this section (section 4.5), we have mentioned the other five criteria. The reason we recorded only two of these criteria is not due to our negligence to the other five. Instead, they were not appropriate to describe them in numerical terms to record in each stage of the experiment. In addition to this, all of these five criteria can only be understood by the researchers during the process of undergoing the experiments and they cannot be measured with some metrics. Furthermore, the results gained from the experiments with these criteria are not varying from experiments to experiments. Therefore, writing the same thing again and again was not important apart from adding unnecessary repetitions to the document. As a result, the researchers preferred to present them as a summary together with the average computation time and modularity value metrics.

Summary of the Overall Experiment

Criteria	**GN**	**GMC**	**NGMC**	**Louvain**
Average processing time (Seconds)	1.735	361.520	1.882	14.4208
Average modularity value	0.400	0.440	0.411	0.491
Support visualization of the network structure	Yes	Yes	Yes	Yes
Computing resource required	Single with GPU or distributed	Single with GPU or distributed	Single with GPU or distributed	Single with GPU or distributed
Capability to detect overlapping communities	No	No	No	In some extent
Estimated ease of use	Difficult	Medium	Difficult	Easy
Is it a state of the art?	No	No	No	Yes

Table 4.6: Summary of the whole experiment

Where:

GN→Girvan_Newman

GMC→Greedy_Modularity_Communities

NGMC→_Naive_Greedy_Modularity_Communities (fast greedy modularity)

As shown in Table 4.6 above, the 'Louvain' algorithm is the best in many aspects. For instance, it is only worse than two of the four algorithms only with one criterion i.e. processing time. Although it has a larger processing time than the 'girvan_newman' algorithm and '_NGMC', the time taken by the Louvain algorithm is not that much worrying. Meaning an average of 14.4028 seconds to detect communities with higher modularity value is something worthy of considering it one of the best algorithms in this aspect. Among the four algorithms compared with the seven parameters mentioned above, the 'Louvain' algorithm is the best with four of the seven parameters (modularity

value, ease of use, state of the arts, and ability to detect overlapping communities) and it is joint best with two of the seven parameters (support of visualization and ability to run on a single or distributed computing resource) and it is the second slowest with one of the seven parameters (computation time). Therefore, among the four algorithms, it is the one that is the best in many aspects and we will proceed to detect and analyzing communities using the 'Louvain' algorithm.

4.3.5. Detecting Communities and Targeting Them for Applications

So far in the experiment, we have seen how the four algorithms reacted on five datasets and chosen the 'Louvain' algorithm as the best algorithm to continue detecting communities. Now onwards, we will perform our activities using one algorithm (the Louvain algorithm) and we will see the structures of the communities detected from a dataset.

We have seen how we can import a graph file to our Jupyter Notebook in the previous sections. Now, let's see how the Louvain algorithm detects communities.

```
p1=community_louvain.best_partition(G81)
```

P1 is a new graph formed by partitioning the graph G81 with the Louvain algorithm. Now let's see the structure of the graph p1.

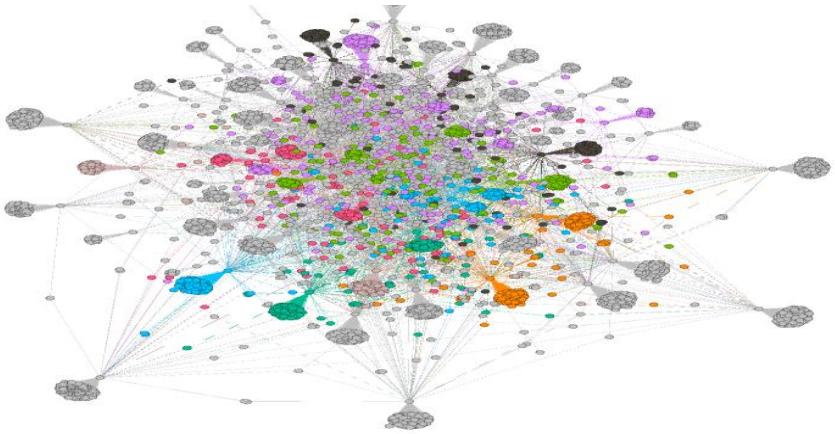


Figure 4.24: A network partitioned into communities each color showing the modularity class.

Since the purpose of this paper is to improve the effectiveness of different applications through the help of communities, understanding the properties of communities in graphs is very important.

The graph has several properties. The most important property of communities is their strength measured in modularity in this case. A community is called a good or strong community if its modularity value is between 0.3 and 0.7. But different modularity values between this range can differ the level of strength of the community. For instance, a community with a modularity value of 0.7 is stronger than a community with a modularity value of 0.5 even though both are within the acceptable range.

```
mod1=community.modularity(p1,G81)
```

```
mod1
```

```
0.6487052525434464
```

For instance, the above code snippet shows that the Louvain algorithm detected communities with a modularity value of around 0.65. Therefore, we can say the community is strong.

The second property of the graph is called modularity class. The graph is composed of many communities and each community in the graph is assigned with a modularity class. Thus, a modularity class is assigned to a community. The number of nodes (entities) is different from community to community and each modularity class holds some percentage of the total nodes in the total graph. This property of the modularity class is known as *community size distribution* and indicates that each community shares some amount of the nodes in the network. It also indicates that the final community structure of the network contains only a small number of large communities and most of the communities are small communities that only contain very few nodes.

Figure 4.25: The distribution of nodes in each modularity class by size

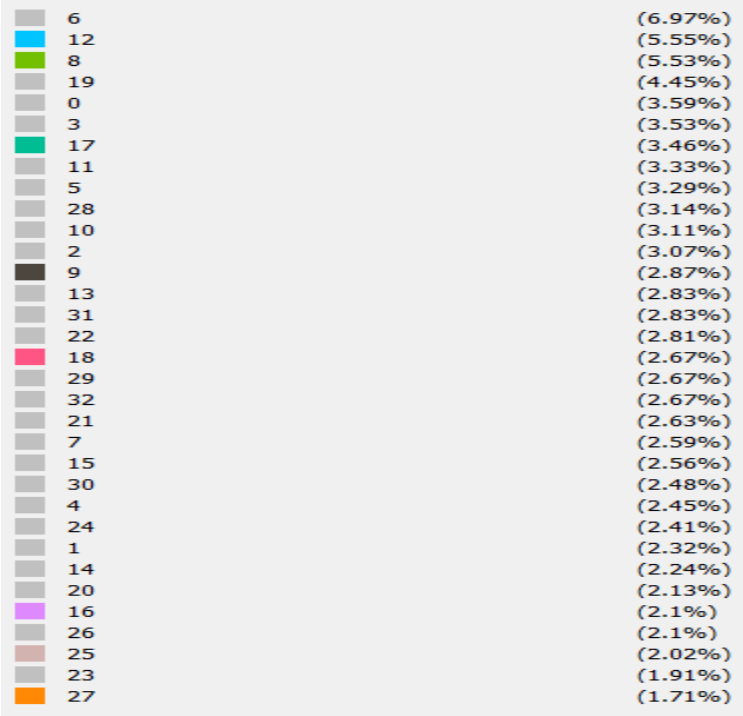


Figure 4.26: The percentage of the nodes contained by modularity classes (communities)

As shown in the figures above, the number of nodes in each community is different. For instance, three communities (community 6, community 8, and community 12 respectively) are the communities having the maximum number of nodes (more than 300 nodes each) followed by community 19 which contains around 250 nodes. Whereas, community 27 is the one having the lowest number of nodes (less than 100).

From the above description about communities and the number of nodes in each community, businessmen can understand that a post seen and liked by an individual in a certain community may be seen and liked by all other individuals in that community too. For instance, if a post is seen and liked by an individual in community six (6), then it would also have a high possibility to be seen and liked by the other more than 300 individuals in that community since members of the same community are expected to have many features in common. In opposite to this, if a post is seen and liked by an individual in community twenty-seven (27), then, there are no many individuals to be treated like that individual, and the effectiveness of the post may also be less.

In the above paragraph, we have seen how businessmen can benefit from targeting communities having many members. But the size of a community may not always be an indicator of success to

an application because the interaction between those many members in a community may not be as strong as needed. In this case, we need other metrics that can ensure that the connection between communities is strong enough to consider. Here comes the concept of modularity.

Communities can usually be ranked based on their modularity value. The community with the highest modularity value indicates that members of the community are strongly connected. Whereas, the community with the lowest value of modularity value indicates that members of the community are weakly connected. Therefore, posting a message on strong communities has a high possibility of success than posting it on weak communities.

The other property of the graph is the distance property characterized by three features; diameter, radius, and the average path length to go from a node to other nodes. These are 8, 4, and 4.00 respectively in this case. This property can also be called the centrality property which can be expressed in terms of betweenness, closeness, harmonic closeness, and eccentricity centrality. More importantly, these properties are always used to show the most important node in the community if there is any.

4.3.6. Mapping Communities for Applications

Companies can target a specific community from a partitioned network by simply checking the community number or modularity class they are assigned to and then by checking other members which are assigned into the same community as themselves. For instance, if the owner of the page called "*የኢትዮጵያ አየር መንገድ*" wanted to know its targeted customers, then he/she can do it as follows:

```
partition["የኢትዮጵያ አየር መንገድ"]
```

```
6
```

```
strComms[6]
```


```
AtlasView({'MekonnenKassye': {'id': '2206'}, 'TadesseMeskelu': {'id': '3137'}, 'TemuBogale': {'id': '3138'}, 'ኢይሰዉ': {'id': '31
39'}, 'AbrehamZBerta': {'id': '3140'}, 'DerejeKumie': {'id': '3141'}, 'FekieAddissWass': {'id': '3142'}, 'MebaWondimteka': {'i
d': '3143'}, 'EnseGetachew': {'id': '3144'}, 'EtsehiwoteMelese': {'id': '3145'}, 'መልካምስራናአርሰው': {'id': '3146'}, 'GetahunYarega
l': {'id': '3147'}, 'BetelhemDeribe': {'id': '3148'}, 'AbebawBelste': {'id': '3149'}, 'GezahegnYetewahidoLij': {'id': '3150'},
'TomeYeMaryamLj': {'id': '3151'}, 'ZelalemGetaye': {'id': '3152'}, 'AddisuZewdie': {'id': '3153'}, 'MebratuYetayeh': {'id': '31
54'}, 'NahuSenay': {'id': '3155'}, 'TilahunAddis': {'id': '3156'}, 'BádmîêShérãrõ': {'id': '3157'}, 'YaregalKassahunGrazmachamb
ay': {'id': '3158'}, 'EdmealemEsubalew': {'id': '3159'}, 'TesfayeYimer': {'id': '3160'}, 'DegelaFena': {'id': '3161'}, 'MulatSh
iferaw ': {'id': '3162'}, 'AbelSisay': {'id': '3163'}, 'AkliluTSh': {'id': '3164'}, 'YebelayMulu': {'id': '3165'}, 'AbelTesfay
e': {'id': '3166'}, 'AssefaBirku': {'id': '3167'}, 'FantuEshetu': {'id': '3168'}, 'ኤርሚሀይሉ': {'id': '3169'}, 'AbebaGizachew':
{'id': '3170'}, 'AlelignAschaleWudneh': {'id': '3171'}, 'SewaleAbate': {'id': '3172'}, 'MadibaAdis': {'id': '3173'}, 'YohannesS
endek': {'id': '3174'}, 'ZelekeAbebe': {'id': '3175'}, 'AmanuelBelewDelele': {'id': '3176'}, 'KisanetAbrha': {'id': '3177'}, 'K
elkiyasLemma': {'id': '3178'}, 'MakiYeHemiEniyew': {'id': '3179'}, 'WendosenZegu': {'id': '3180'}, 'SisayShibruNoru': {'id': '3
181'}, 'ElfarusEthiopiaRaziel': {'id': '3182'}, 'TedBekaluAya': {'id': '3183'}, 'BizuayehuMoged': {'id': '3184'}, 'HabtishMinwu
ye': {'id': '3185'}, 'FelekeMola': {'id': '3186'}, 'HabtamuMindaye': {'id': '3187'}, 'HeranieBerhanu': {'id': '3188'}, 'Alemtse
hayAbebe': {'id': '3189'}, 'AddisuGirma': {'id': '3190'}, 'YeabsiraGetahun': {'id': '3191'}, 'እንዳይላሉደጌ': {'id': '3192'}, 'Sisay
Hailu': {'id': '3193'}, 'EdenNigussie': {'id': '3194'}, 'LiigiiOdpGinjoo': {'id': '3195'}, 'KeruKerishaKere': {'id': '3196'},
'NebiyEyobWaktole': {'id': '3197'}, 'SatenaweYibekalGebrehiwot': {'id': '3198'}, 'TigistAyele': {'id': '3199'}, 'እንድለእናቱ': {'i
d': '3200'}, 'MisganaGirma': {'id': '3201'}, 'GetawAyalew': {'id': '3202'}, 'SemrtBitewo': {'id': '3203'}, 'ErmiasYedngilLij':
{'id': '3204'}})
```


Figure 4.27: Sample community mapped to an application

Where:

Partition→ a variable holding a set of communities

strComms→a variable holding set of sorted communities

As shown in the above code snippet, a page named "የኢትዮጵያ አየር መንገድ" is assigned into a community having a modularity class of 6. Then accessing other members assigned to the same modularity class with the page can be taken as possible customers of the company or individual owning the page.

The above example shows that a company owning a page on Facebook can map its products with a set of individuals for various applications (when the mapping is done at a company level). That means, the above way of mapping communities with applications is better when the dataset only contains the relationships between many company names and users who are following them on Facebook. However, mapping communities didn't end at the company level. When we come down to the various services a specific company delivers to users, first of all, the data extraction and community detection process should be limited to the users who are related to that particular company.

For example, the company mentioned above (Ethiopian Airlines) provides various services to its customers at its airport. As mentioned on the company's website, these services include arrival

services, lounges, stopovers, airport service, and connection with other airports and airlines with minimum time. Thus, if we want to map customers to each application provided by the airline, we have to know which users are interested in each application first through the process of community detection. After the community detection process, mapping follows a similar mechanism used above. But instead of the company name itself, what we feed to the function is the service or application name. For instance, if we want to map communities with the arrival service of the airline, then we will do the following.

```
partition["Arrival Services"]
```

The code returns the modularity class the service or application called ‘Arrival Services’ is placed into (let it be 2 for instance). After getting the modularity class (community number), the next task is finding other user or page names assigned to the third modularity class together with the service ‘Arrival Services’ (the second modularity class will be 2 since it starts at 0). It can be done as follows.

```
strComms[2]
```

The list of names returned by the code snippet will then be used to be targeted by the company as people interested in the ‘Arrival Services’ service.

The case indicated by the example above is an example of a community of page likes in which communities are detected by an algorithm from Facebook users who liked pages.

But communities can also be formed from other interactions on Facebook. The major ones apart from the ‘like’ relationship between users and pages are the friendship relationship (community of friends) and relationships through the comments they give to an advertisement or a post (semantic-based communities).

In the case of communities of friends, the relationship is formed between an individual who reacts to an advertisement and his/her friends on Facebook because if he/she liked the advertisement or post, he/she may share the post on his/her timeline which can allow the post to be seen by all his/her friends. In this case, one can map the product he/she wan to advertise to a community of

friends by looking at the community number (modularity class) of a person who liked the advertisement or post and then querying the other users in that modularity class.

Whereas in the case of the community of semantic-based communities, detecting the communities is a little bit intricate as one has to collect all the comments given about a specific ad or post and the comments should be classified as positive or negative to know the opinion of an individual about a product or service. Although the process of detecting the communities is intricate compared to detecting communities of likes, mapping an ad to an individual is almost the same as that of mapping a community of likes (or community of structures in general). This is because to do the task, one is required to identify the modularity class the product or service name is assigned into and then finding the users assigned into that modularity class. For example, assume if a company called 'Tecno' want to map an ad to its 'Tecno k7' brand phone he/she can do as follows:

```
partition["Tecno K7"]
```

Then, the code results in the community number or modularity class the product called 'Tecno K7' is assigned in (8 for example). After knowing the modularity class, the next job is viewing all the users who are in the same modularity class with the code displayed above.

```
strComms[8]
```

Now onwards, the owner of the product or the ad can stream every advertisement related to the product to those people under the modularity class 8.

4.3.7. Discussion of Results

Reaching this stage, we can answer the research questions raised at the beginning of the research. There were three questions the researchers raised at the beginning of this work. These are:

RQ 1: How can we extract the structural features of Facebook users?

RQ 2: How can we represent structural features for community detection?

RQ 3: Which techniques (algorithms) are appropriate to detect communities?

Now, let's answer these questions taking one at a time.

RQ 1: How can we extract the structural features of Facebook users?

Before answering this question, let's first understand what structural features are in social networks especially Facebook. In Facebook, structural features are features that express relationships between entities (the entities in Facebook are mainly users and pages either page of a famous person or a business company). These relationships can be expressed in the form of fellowships and friendships. Fellowships are ways in which a set of individuals follow a famous person or company page having the idea to be reached by every post made on the page in mind. Whereas, friendship is a way a user becomes a friend to other users even if he/she does not know them physically. Extracting these structural features (relationships) is important if one is working in relational learning where the relationships in data, not the content are the issue of concern.

Many approaches can be followed to extract the structural features of Facebook features. But almost all of them follow similar procedures although the internal techniques may be different. One is better to perform the following tasks to extract structural features from Facebook.

The first task is going to the part of the page which is planned to be extracted and copy the URL of the page and paste to the script written to extract the data and then run the script.

```
my_url = "https://www.facebook.com/badimaw.terefe.7/likes_all"
```

The second task is viewing the extracted data clearly and understandably so that one can easily identify the HTML tags which hold the needed piece of information. In the process, two basic python libraries are helpful; 'urlb.request' for extracting the page held by the URL and 'BeautifulSoup' for displaying the page in a clear format. The result of the above two activities is the data but with noise included. The data extracted in this phase looks like this.

```
<a href="https://www.facebook.com/Tewahedo-haymanote-1076877829030168/">
 Tewahedo haymanote
</a>
,
<a href="https://www.facebook.com/tewnetmedia/">
 Tewnet.com
</a>
,
<a href="https://www.facebook.com/mensfitness/">
 Men's Journal + Fitness
</a>
,
<a href="https://www.facebook.com/ውስጥ-አዋቂ-1045624338857926/">
 ውስጥ አዋቂ
</a>
```

The third task is inspecting the page so that the HTML structure of the page will be displayed. When the HTML structure is displayed, find the tag which holds the piece of information planned to be extracted and record the tag somewhere so that it will be used as an input to the script written to extract the useful data excluding the noise from it and store the clean data in a variable (handle) and export it to a file with any format (exporting it to excel and CSV file formats are helpful for later stages in community detection). With this data, extraction comes to an end.

For more detail, one can see section 4.3 of this thesis.

RQ 2: How can we represent structural features for community detection?

By their very nature, community detection algorithms work on a graph data structure. So, the data extracted from Facebook must be represented as a graph data structure. But since Facebook is all about relationships between entities, there must be a certain way to depict this relationship between entities. Here comes the concept of the adjacency matrix. In the adjacency matrix, we can represent the point of whether or not there is a relationship between two entities. Therefore, in this thesis, the extracted data was first represented through an adjacency matrix to show the relationship between entities (users and Facebook pages in this case). If two entities have relationships (if they are connected in a network context), then a value 1 was placed and if there is no relationship between them, then a value 0 was placed. Once the availability of relationships (connections) is known and represented through an adjacency matrix, the next task is transforming the data in the matrix into a graph form.

A graph is composed of nodes and edges where nodes represent entities and edges represent relationships between entities. In this case, nodes were Facebook users and Facebook pages and edges were the relationships between pages and users.

In conclusion, extracted data must be represented in a graph format to be used for community detection. But the graph format needs the data to be represented in a matrix format first. Therefore, both the matrix format and the graph format are equally important for community detection algorithms to work. For more detail, one can refer to section 4.4 of this thesis.

RQ 3: Which techniques (algorithms) are appropriate to detect communities?

Many algorithms for community detection are available in the literature. But their quality measures are entirely different. Many regard that algorithms that use the modularity metrics as their quality metric as the best algorithms. Due to this reason, four algorithms based on the concept of modularity optimization were selected in this research and compared against some criteria. As tried to be shown in the experimentation in section 4.5 of this thesis, the '***Louvain'*** algorithm was the algorithm that was found the best in many aspects and the researchers are convinced about the overall performance of the algorithm and identified it as the best one.

Now, let's see how the results shown in this research can be compared to similar researches in the literature.

It was impossible to get a research paper that took all of those four algorithms into consideration in an attempt to compare various algorithms and to choose the best one. As a result, we were forced to look into researches that included at least two of the four algorithms in their description and comparison to compare with our results.

A recent study compared eight algorithms based on internal and external characteristics. They included Louvain and Fast Greedy algorithms for comparison and modularity and processing time as characteristics as bases for comparison and we have limited our review on these features since others were not issues of our research (Mkhitaryan, Mothe, & Haroutunian, 2019).

Concerning modularity value, they found that the Louvain algorithm was the best in seven (7) of the eight (8) datasets they used. Like the result gained in our research, their research also indicated that it is better than the fast-greedy algorithm. As a result, we can say that the result they found is similar in this aspect. They also indicated that the Louvain algorithm detected communities with less processing time than the fast-greedy algorithm. In this aspect, the result they gained is not complimentary with ours since our result showed that the fast-greedy algorithm is much faster than the Louvain algorithm (Mkhitaryan, Mothe, & Haroutunian, 2019).

The other major deviation we observed from the research mentioned above was the relationship they put between modularity value and the size of a dataset. In their research, they said that both the algorithms (Louvain and fast greedy) detected communities from both small and large size datasets with almost equivalent modularity value. In our case, the modularity value of communities

detected by those algorithms together with the other two algorithms was decreasing as the size of the dataset was increasing (Mkhitaryan, Mothe, & Haroutunian, 2019).

Another study by Zhang and et al (2018) described five algorithms based on the concept of modularity maximization namely Girvan Newman, community fast detection algorithm, Clauset Newman, and Moore (CNM), Louvain and Smart Local Moving algorithms using their processing time as a way to compare them. The algorithms they chose for comparison were similar to the ones compared in this paper with some exceptions (greedy modularity was not there and CNM and SLM were not here).

The description of the algorithms and the way they described them shows that the algorithms were developed as an improvement to the other algorithm in the order they were listed here as the size of the dataset increases taking the processing time and community quality (although they didn't quantify community quality in their paper) criteria into consideration. But as shown in the experimentation, this property cannot be assured in this paper since the algorithms they described were not resulting in a sound improvement in the quality of communities when the size of the dataset was increasing. But the fact that modularity-based algorithms are the fastest from all the other categories of community detection algorithms is assured in all researches (Orman, Labatu, & Cherifi, 2012).

What seems more sound in their paper and is true in our case too is that balancing the quality of a community and an algorithm's performance is the main difficulty almost all community detection algorithms suffer from. As a result, many researchers become reluctant to say a certain algorithm is the best in balancing these two very important features in community detection. In this regard, our research had tried to pass that line and said the 'Louvain' algorithm is better (if not the best) than many of the other algorithms.

As far as processing time is concerned, many researchers concluded that the fast greedy (_naive_greedy_modularity_communities) algorithm is much faster than the Girvan Newman algorithm in detecting communities (Chen, Kuzmin, Member, & Szymanski, 2015). But it is difficult to agree with this conclusion after all the inconsistencies we saw from those two algorithms at the experimentation part of this paper. On average, the Girvan Newman algorithm was found faster than all of the algorithms compared in this paper. The fast-greedy algorithm is no

different although it was showing faster processing time at times than the Girvan Newman algorithm and the others too.

In their study, Singh and Garg (2017) mentioned that the Louvain algorithm is fastest from modularity-based algorithms such as Girvan Newman, fast_greedy, and greedy_modularity_communities (Singh & Garg, 2017). But the result gained in this research didn't witness that. Apart from the greedy_modularity_communities algorithm, no algorithm needed more time than the Louvain algorithm.

One of the major gaps regarding community detection algorithms is that most researchers completely ignore qualitative features and they only focus on the quantitative aspects of these features (Orman, Labatu, & Cherifi, 2012). In contrast to this, an attempt to see the qualitative aspects of the algorithms was made in this paper together with the usual qualitative aspects. As a result, characteristics such as ease of use, ability to detect overlapping communities, state of the art, support of visualization of networks, and resource needed to run were assessed and used to compare algorithms.

Now, everything is done and we have ensured that community detection is achievable. In addition to this, we have made all the procedures clear through experimentation. Therefore, we can design the final architecture to solve the problem under investigation having the knowledge we got and proved so far. The final architecture is:

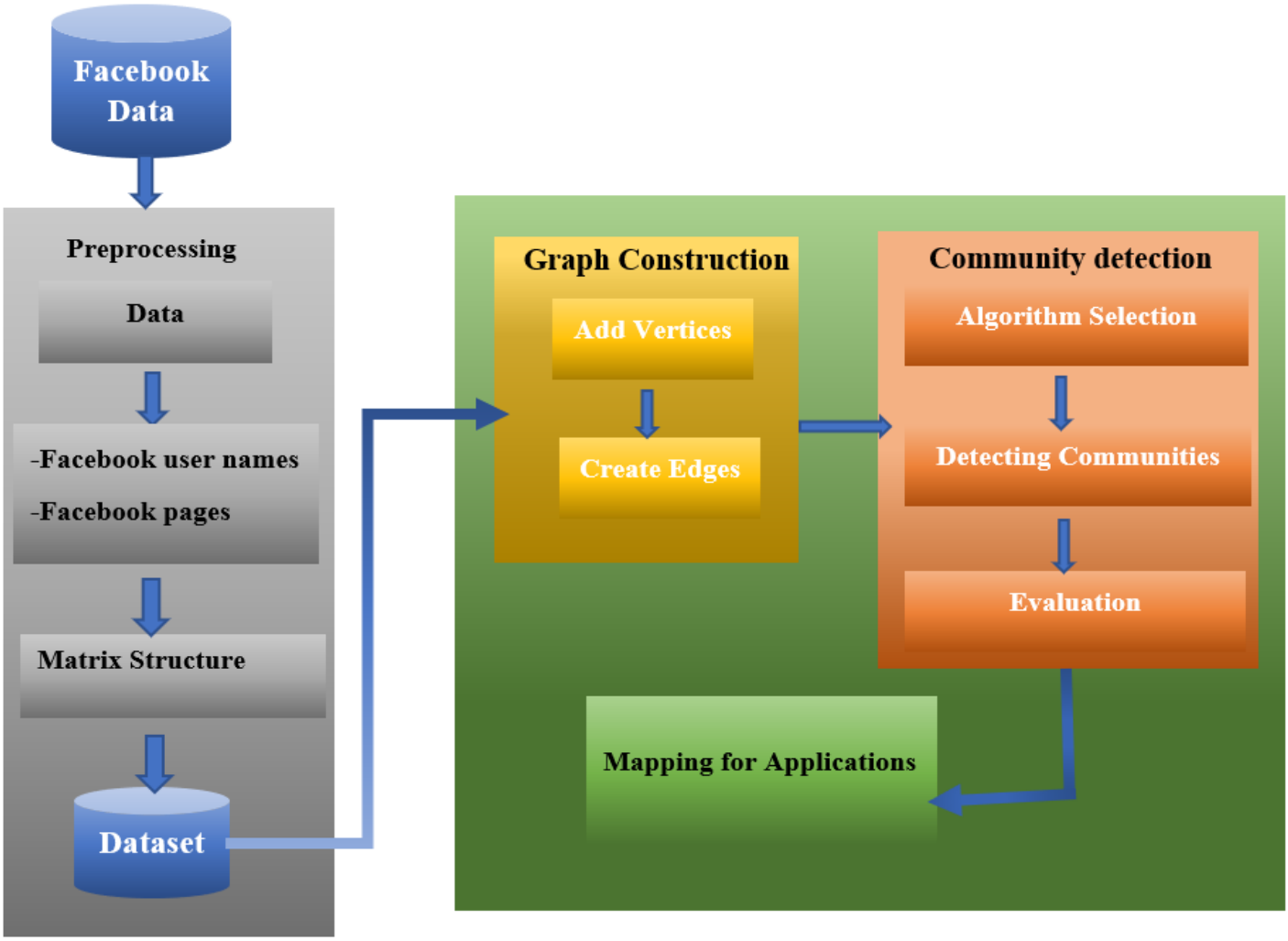


Figure 4.28 Community Detection Architecture from Facebook

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1. Overview

This chapter is organized into two subtopics; conclusion and recommendation. The conclusion part is a summary of the concepts discussed in the previous four chapters in a direction that leads to wrapping up the answers to the research questions raised at the starting of the research. The recommendation part discusses the issues that would be better if someone incorporates them with the issues discussed in this research in his/her future research. It is aimed to present the conclusion phase of the selected DSR model.

5.2. Conclusion

Web technology is growing rapidly which in turn allowed individuals to communicate with one another and with organizations. These communications are called networks or called social media and companies are using data on these networks for various applications. Among these applications, supporting businesses through social media advertising is being used in high volume in the developed world although it is at its starting age in our country Ethiopia. Facebook is the social media that is being highly used in Ethiopia.

Community detection is one of the frequently used techniques which is growing together with the growth of social media. It is being applied in recommendation systems, crime detection, making political decisions, and marketing. In this research, an attempt was made to understand community detection using Facebook relationships, and the mechanisms to achieve it were discussed in detail.

Communities are groups that are formed virtually by individuals having a common interest in some issues. They can help Facebook applications by allowing companies to target these communities as their potential customers. Community detection is all about finding these communities found virtually on social media (Facebook in this case).

Communities on Facebook can be found on structure or content-wise. 'like', 'follow', and 'friendship' are the forms a community can be found structurally. Whereas, 'comment', 'like' and 'dislike' and are the forms a community can be found in content-wise.

There are many algorithms today for community detection on Facebook data. Among these algorithms, those algorithms which work based on modularity maximization are believed as best in many aspects such as processing time, quality, ease of use, and the like. In addition to this, these algorithms are best because modularity is the best quality measure for community detection algorithms and they use it as a measure. Many algorithms work based on the concept of modularity maximization and among them the Louvain algorithm is the best algorithm in its capability to support the modularity function, it can support network visualization, it is state of the art and too easy to learn and work with. It can also be done using a single PC with a GPU or a distributed environment. As a result, we can say it is the best algorithm from the bunch of community detection algorithms that are based on the concept of modularity optimization.

Companies can target detected communities for applications in two simple steps. The first step is knowing the community number (modularity class) they are assigned to and the second step is knowing the list of members who are assigned in the same group as themselves.

5.3. Recommendations

Someone interested in community detection from Facebook can use this research as a starting point and further improve it by adding the following features into it.

I. This study used 'like' and 'follow' structures on Facebook as a source of data. Including 'friendship' structure and user-generated content such as comments, likes and dislikes are better to be studied in the future.
II. In this study, the researchers experimented with only modularity-based community detection algorithms on datasets of a varying number of nodes and edges. Including other algorithms apart from the modularity maximization concept in the experiment is recommended. In the process, if someone interested in this area arises, the researchers are voluntary to give him/her the dataset they used in the research provided that he/she promises to keep ethical issues.
III. The datasets used during the experimentation were taken only from Facebook. It is recommended if added datasets from other sources like Twitter and Instagram will experiment together with the ones taken from Facebook.
IV. It is also recommended if anyone can work on detecting communities from online Facebook data i.e. detecting communities from dynamic data apart from the static one.

REFERENCES

Amit , A., & Nikita , G. (2017). Community Detection in Social Network Based on User's Social Activities. *International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)* (pp. 625-628). India: IEEE.

5 Reasons Why Social Media Advertising is Better than Traditional Advertising. (2019, December 8). Retrieved from Adlg: https://adlgmarketing.com/blog/5-reasons-why-social-media-advertising-is-better-than-traditional-advertising/

Advantages of target marketing: criteria for effective segmentation - Marketing Management. (2020, March 18). Retrieved from wisdomjobs.com: https://www.wisdomjobs.com/e-university/marketing-management-tutorial-294/advantages-of-target-marketing-criteria-for-effective-segmentation-9347.html

Afolabi, A. (2015). *Social Media Marketing; The Case of Africa.* Ontario, Canada.

Ahmed, Q. M., & Raziq, M. M. (2017). The Social Media Advertising Model (SMAM): A Theoretical Framework. *Journal of Managerial Sciences* , 117-144.

Arzum , K., & Serap , S. (2018). *A Comparative Study of Modularity-Based Community Detection Methods for Online Social Networks.* Izmir, Turkey: CEUR-WS.org/vol-2201.

Baskerville, R., Baiyere, A., Gregor, S., Hevner, A., & Rossi, M. (2018). Design Science Research Contributions: Finding a Balance between Artifact and Theory. *Journal of the Association for Information Systems*, 358-376.

Bella , M.-S., & Xiaoou , L. (2016). Ranking Features in Facebook to Detect Overlapping Communities. *Proceedings of 2016 IEEE 13th International Conference on Networking, Sensing, and Control* (pp. 1-6). Mexico: IEEE.

Blaise , N., Emmanuel , V., Savaneary , S., Philippe , S., Françoise , F.-S., & Rémi , K. (2013). *Monetization and Services on a Real Online Social Network Using Social Network Analysis.* France: IEEE.

Chen, M., Kuzmin, K., Member, S., & Szymanski, B. (2015). Community Detection via Maximization of Modularity and its Variants. *IEEE vol.1 no.1*, 65-76.

Chitra, D., & Poovammal. (2016). An Analysis of Overlapping Community Detection Algorithms in Social Networks. *Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016)* (pp. 349 – 358). Chennai, India: Elsevier.

Chopade, P., & Zhan, J. (May 27-31, 2014). Community Detection in Large Scale Big Data Networks. *ASE BIGDATA/SOCIALCOM/CYBERSECURITY Conference* (pp. 1-7). Stanford University: ResearchGate.

Clauset, A., Newman, J., & Moore, C. (2004). Finding community structure in very large networks. *arXiv:cond-mat/0408187 vol.2,* 1-6.

David , R., & Chris , S. (2018). The detection of criminal groups in real-world fused data: using the graph-mining algorithm “GraphExtract”. *Security Informatics Vol. 7*, 1-16.

De Meo, P., Ferrara, E., Fiumara, G., & Provetti, A. (2014). Generalized Louvain method for community detection in large networks. *ResearchGate*, 88-93.

Deshpande, N., Shaikh, S. A., & Khode, A. (2014). Web based Targeted Advertising: A Study based on Patent Information. *Procedia Economics and Finance* , 522-535.

Dhanya , S., & Shini , R. (2016). Survey of Community Detection Algorithms to Identify the Best Community in Real-Time Networks. *International Journal of Scientific Engineering and Applied Science – Volume-2, Issue-1*, 529-533.

Dubik, M. (2017). *A comparative evaluation of state-of-the-art community detection algorithms for multiplex networks.* Uppsala : Uppsala Universitet.

Fischer, T. (2011). Relational learning and optimization in the semantic web. *Tagungsband Zum 14. Interuniversitaren Doktorandenseminar Wirtschaftsinformatik*, 16.

Five Reasons Why Social Media Advertising is Better than Traditional Advertising. (2019, December 8). Retrieved from Adlg: https://adlgmarketing.com/blog/5-reasons-why-social-media-advertising-is-better-than-traditional-advertising/

Fortunato, S. (2010). Community detection in graphs. *ArXiv, abs/0906.0612.*, 1-103.

Fu, Y.-H., Huang, C.-Y., & Sun, C.-T. (2017, November 9). *A community detection algorithm using network topologies and rule-based hierarchical arc-merging strategies*. Retrieved from PLOS|ONE: https://journals.plos.org/plosone/article/authors?id=10.1371/journal.pone.0187603

Girvan, G., & Newman. (2004). Community structure in social and biological networks. *Natl*, 7821–7826.

Gizat, A. (2016). *ASSESSEMENT OF SOCIAL MEDIA ON MARKETING STRATEGY: ETHIOPIA FACEBOOK USERS PERSPECTIVE.* Addis Ababa: Academia.

Jensen, D., & Nevile, J. (2002). Linkage and autocorrelation cause feature selection bias in relational learning. *Proceedings of the Nineteenth International Conference on Machine Learning (ICML2002)* (pp. 259-266). MA 01003 USA: Morgan Kaufmann.

Julien , M. O., & O. J. (2012). *Automatic detection of community structures in networks.* Leuven: Universite Catholique de Louvain.

Julien , O., & Michael , S.-G. (2012). Automatic detection of community structures in networks. *Théorie et algorithmique des graphes*, 1-13.

Karata¸s , A., & S¸ahin, S. (2018). A Comparative Study of Modularity-Based. *Izmir Institute of Technology Izmir Institute of Technoloy Izmir Institute of Technology Vol 2201*, 1-12.

Kunpeng , Z., Siddhartha , B., & Sudha , R. (2016). LARGE-SCALE NETWORK ANALYSIS FOR ONLINE SOCIAL BRAND ADVERTISING. *BIG DATA & ANALYTICS IN NETWORKED BUSINESS*, 849-872.

Liangxun, S., & Bianfang, C. (2015). Discussion of the community detection algorithm based on statistical inference. *Perspectives in Science*, 122-125.

Ludo , W., & Nees , v. J. (2013). *A smart local moving algorithm for large-scale modularity-based community detection.* Leiden University: Centre for Science and Technology Studies.

Ludo , W., & Nees , v. J. (2013). A smart local moving algorithm for large-scale modularity-based community detection. *THE EUROPEAN PHYSICAL JOURNAL* , 1-14.

Macskassy, S., & Provost, F. (2003). *A simple relational classifier.* DTIC.

Manning, J. (2014). Definition and Classes of Social Media. In Harvey, *Encyclopedia of social media and politics* (pp. 1158-1162). Thousand Oaks,: CHAP .

Matin , P., Justin , Z., & Shahab , T. (2016). An optimized approach for community detection and ranking. *Journal of Big Data . 3. 22. 10.1186/s40537-016-0058-z.*, 1-12.

Matin , P., Justin , Z., & Shahab , T. (2016). An optimized approach for community detection and ranking. *Jornal of Big Data vol.3*, 2196-1115.

Michel, P., & Michel, C. (2013). Survey on Social Community Detection. *Social Media Retrieval* , 65-85.

Mkhitaryan, K., Mothe, J., & Haroutunian, M. (2019). DETECTING COMMUNITIES FROM NETWORKS: COMPARISON. *DETECTING COMMUNITIES FROM NETWORKS: COMPARISON Vol. 26, Number 3,* 237-267.

Muhammad , R. M., & Qazi , A. M. (2018, June). The Social Media Advertising Model (SMAM): A Theoretical Framework. *Journal of Managerial Sciences* , 117-144.

Naidoo, T. (2011). *The effectiveness of advertising through the social media in Gauteng* .

Newman. (2004). Fast algorithm for detecting community structure in networks. *https://arxiv.org/abs/cond-mat/0309508 vol.1,* 066133-1-066133-5.

Nowell, D. L., & Klienberg, J. (2007). The link prediction problem for social networks. *Journal of American society for information science and technology Vil. 58(7),* 1019-1031.

Odhiambo, C. A. (2012). *Social Media as a Tool of Marketing and Creating Brand Awareness* .

Ohajionu, U. C., & Mathews , D. (2015). ADVERTISING ON SOCIAL MEDIA AND BENEFITS TO BRANDS. *Research Gate*, 335-351.

Orman, K. G., Labatu, V., & Cherifi, H. (2012). *Qualitative Comparison of Community Detection Algorithms.* Istanbul, Dijon: arxiv.org/abs/1207.3603 vol.1.

Parvathi, & Monica. (2017). Survey on Social Network Community Detection and Ranking. *International Journal of Control Theory and Applications* , 159-164.

Pasquale, M. D., Emilio , F., Giacomo , F., & Alessandro , P. (2011). Generalized Louvain Method for Community Detection in Large Networks. *11th InternationalConference on Intelligent Systems Design and Applications* (pp. 88-93). University of Oxford: ResearchGate.

Peffers, K., Tuunanen, T., Gengler, C. G., Rossi , M., Hui , W., Virtanen , V., & Bragge , J. (February 2006). The design science research process: A model for producing and presenting information systems research. *Proceedings of First International Conference on Design Science Research in Information Systems and Technology DESRIST*, 85-106.

Perlich, C., & Provost, F. (2003). Aggregation-based feature invention and relational concept classes. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 167-176). ACM.

Philipp, O., Olga, L., Marten , S., & Udo , B. (January 2009). Outline of a Design Science Research Process. (pp. 1-12). Berlin, Germany: January 2009 .

Pons , P., & Latapy , M. (2012). *Computing communities in large networks using random walks.* Paris: LIAFA – CNRS and University .

Pravin , C., & Justin , Z. (2014). *Community Detection in Large Scale Big Data Networks.* Stanford University: ASE.

Punam, B., & Chhavi, S. (2016). Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery Vol 6*, 1-23.

Punam, B., & Chhavi, S. (2018). *Community detection in social networks.* Delhi: ResearchGate.

Qazi , M. A., & Muhammad , M. R. (2018, July 03). The Social Media Advertising Model (SMAM): A Theoretical Framework. *ResearchGate*.

Rokia , M., & Idrissa , S. (2014). Lecture Notes in Social Networks. In M. Rokia, & S. Idrissa, *Social Network Analysis - Community Detection and Evolution* (pp. 121-159). Switzerland: Springer Publishers.

Sen , W., Mengjiao , T., & Deying , X. (2015). Network Structure Detection and Analysis of Shanghai Stock Market . *Journal of Industrial Engineering and Management 8. 10.3926/jiem.1314.*, 383-398.

Singh, D., & Garg, R. (2017). Some Observations on Community Detection Algorithms in Social. *International Journal of Engineering Science Invention (IJESI) Volume 6 Issue 12*, 2319 – 6726.

Siricharoen, & Waralak. (2012). Social Media, How does it Work for Business? *International Journal of Innovation, Management and Technology Vol.3*, 476-479.

Sobolevsky, S., & Campari , R. (2014). General optimization technique for high-quality community detection in complex networks. *PHYSICAL REVIEW E 90, 012811* , 012811-1-012811-8.

Sweta , R., Shubha , C., & Anurag , J. (2017). Community Detection on Social Media: A Review. *International Journal of Scientific Research Engineering Technology Vol. 3*, 29-33.

Symeon , P., Yiannis , K., Athena , V., & Ploutarchos , S. (2012). Community Detection in Social Media; Performance and application considerations. *Data Mining and Knowledge Discovery Vol. 24*, 514-554.

Tashrifa , H., & Shadman , S. (2016). A Study On The Influences of Advertisement On Consumer Buying Behavior. 1-12.

Vincent , B. D., Jean-Loup , G., Renaud , L., & Etienne , L. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics Theory and Experiment vol.2008*, 1742-5468.

Z. X., M. Z., Z. Z., S. Q., & Y. J. (2018). A Review of Community Detection Algorithms Based on Modularity Optimization . *Journal of Physics*, 1-6.

Zhang, X., Ma, Z., Zhang, Z., Sun, Q., & Yan, J. (2018). A Review of Community Detection Algorithms Based on. *Journal of Physics : Conf. Series 1069 (2018) 012123*, 1-5.

APPENDIX

I. User Permission

ውድ የፌስቡክ ጓደኞቼ በሙሉ በፈጣሪ ስም ሰላም እላችኋለሁ እንዴት ከረማችሁ።አብዛኞቻችሁ እንደምታውቁት ከ መስከረም 2010 ዓ.ም ጀምሮ በ አ.አ ዩኒቨርሲቲ የሁለተኛ ዲግሪየን (MSc) በመማር ላይ ነበርኩ። አሁን የመጨረሻ የምርምር ስራዬን እየሰራሁ ስለሆነ በግልጽ እንዲታይ የፈቀዳችሁትን መረጃችሁን (Basically your name and the pages you liked on FB) ለዚሁ ጉዳይ እንድጠቀምበት ትፈቅዱልኝ ዘንድ በአክብሮት እጠይቃችኋለሁ። ከዚህ በተጨማሪ ስማችሁ በመጨረሻው የወረቀት ጥራዝ ውስጥ ቢገኝ ለመጥፎ ጉዳይ አይደለም እና እንዳትቀየሙኝ በፈጣሪ ስም አደራ እላለሁ። በዚሁ ጉዳይ የማትስማሙ ጓደኞቼ በውስጥ መስመር እንድታሳውቁኝ እማጸናችኋለሁ። ካላሳውቃችሁኝ ዝምታችሁን እንደ መስማማት ቆጥሬ መረጃችሁን እንደምጠቀመው እንድታውቁ ይሁን።

በጣም አመሰናለሁ።
ባድማው ተረፈ።

II. Plain data extracted from Facebook (prior to cleaning)

```
<a href="https://www.facebook.com/mirwalimuhammadmufheri/">
 Sport zone Studio Ethiopia
</a>
,
<a href="https://www.facebook.com/EtYelp/">
 Andegna - EthioTime
</a>
,
<a href="https://www.facebook.com/ለመሳቅ-ብቻ-479874445540948/">
 ለመሳቅ ብቻ
</a>
,
<a href="https://www.facebook.com/ዶርማቸን-Dormitory-1474258612792336/">
 ዶርማቸን Dormitory
</a>
```

```
<a href="https://www.facebook.com/ethiodaily/">
 Ethio Daily ኢትዮ ዴይሊ
</a>
,
<a href="https://www.facebook.com/hatricksport1/">
 HatTrick sport ሀትሪክ ስፖርት
</a>
</span>
<span class="hiddenItem">
,
<a href="https://www.facebook.com/Woldia-University-598971513503655/">
 Woldia University
</a>
,
<a href="https://www.facebook.com/አማራ-ሚዲያ-አገልግሎት-Amhara-Media-Service-396615230875752/">
 አማራ ሚዲያ አገልግሎት - Amhara Media Service
</a>

<a href="https://www.facebook.com/ሎሬት-ፀጋዬ-ገብረ-መድህን-271243023615527/">
 ሎሬት ፀጋዬ ገብረ መድህን
</a>
,
<a href="https://www.facebook.com/ፋሲል-ከነማ-ከክለብም-በላይ-ነው-1538759856241442/">
 ፋሲል ከነማ ከክለብም በላይ ነው
</a>
,
<a href="https://www.facebook.com/ቅኝት-በኳስ-ሜዳ-823582698033581/">
 ቅኝት በኳስ ሜዳ
</a>
,
<a href="https://www.facebook.com/ZemedkunBekeleB/">
 Zemedkun Bekele
</a>
,
<a href="https://www.facebook.com/Bahirdar-Kenema-ባህርዳር-ከነማ-152863915371393/">
 Bahirdar Kenema - ባህርዳር ከነማ
</a>

<a href="https://www.facebook.com/Bure-Shikudad-preparatory-School-401992313325099/">
 Bure Shikudad preparatory School
</a>
,
<a href="https://www.facebook.com/bayliepage/">
 Baylie Damtie
</a>
,
<a href="https://www.facebook.com/ነጃም-ህብረት-ለኢትዮጵያ-አንድነት-በሲያትል-1275504575813669/">
 ነጃም ህብረት ለኢትዮጵያ አንድነት በሲያትል
</a>
,
<a href="https://www.facebook.com/Woldia-university-department-of-civil-Engineeringby-Abrham-165581738

 Woldia university department of civil Engineering.by Abrham
</a>
```

```
 <a href="https://www.facebook.com/theresilientmindproject/">
  Strong Mental Health
 </a>
 ,
 <a href="https://www.facebook.com/psyhelpk/">
  Psychology online advice - Katerina Balshaw
 </a>
 ,
 <a href="https://www.facebook.com/CR7Sneha/">
  Sneha CR7
 </a>
 ,
 <a href="https://www.facebook.com/theredsgomarchingon/">
  Red Glory
 </a>

<a href="https://www.facebook.com/ethiopiansportesp/">
 Ethiopian Sport
</a>
,
<a href="https://www.facebook.com/ethiosportPage/">
 Ethio - መዝናኛ
</a>
,
<a href="https://www.facebook.com/ethiopianteachers/">
 የኢትዮጵያ መምህራን ድምፅ ይሰማ
</a>
,
<a href="https://www.facebook.com/tribuna.united.en/">
 United Fans Live
</a>
```