Master Degree Project

COMMUNITY-BASED INFLUENCE MAXIMIZATION FRAMEWORK FOR SOCIAL NETWORKS

Master Degree Project in Informatics with specialization in Data Science

One year Level, 15 ECTS
Spring term Year 2018

Tshering Wangchuk

Supervisor: Yacine Atif
Examiner: Birgitta Lindström

Abstract

COMMUNITY BASED INFLUENCE MAXIMIZATION FRAMEWORK FOR SOCIAL NETWORKS

Tshering Wangchuk

Maximizing Influence (IM) in social networks has a considerable role to play in the phenomenon of viral marketing, targeted advertisements and in promoting any campaigns. However, the Influence Maximization Problem is a very challenging research-problem due to it being NP-Hard and scaling with the social networks with millions of nodes and edges becomes very tough due to the computational complexities concerned with it. Recently, solving this problem through the use of community detection based methodology is becoming very popular since, it reduces the search space by dividing the network into smaller and more manageable groups called "communities." As part of the larger research work, we reiterate a framework which has been inspired by collection of different work done by Alfalahi et al. (2013) that we can implement to solve the IM problem and its limitation through community detection and fuzzy logic inspired approach. Since the work is still under development, for this project, we report on understanding the IM field through literature reviews and in communicating a design of IM framework as inspired by the previous works. We also present our version of the blueprint (Algorithm design) of the framework as a five step approach. For the purpose of this report, we implement and evaluated the step 1 and step 2 of the framework. Step 1 is about preprocessing the input network with a similarity measure, which according to previous study by Alfalahi et al. (2013a) aids the algorithms in detecting better community structure (clear and accurate distinction of the nodes into communities in networks). We test it to see if it holds true. Step 2 is about implementing the community detection in social network. We benchmark three candidate algorithms, chosen based on their performance, from the previous studies in community detection field and we report on which algorithm should we consider to use in the proposed framework through experimentation on the simulated data. We use Normalized Mutual Information (NMI) and Modularity (Q) as evaluation metrics to measure the accuracy of the community detected by the candidate algorithms. Our results show that similarity based preprocessing does not improve the community structure and thus may not be required in the framework. We also found out that Louvain should be the algorithm that use to detect communities in social networks since it outperforms both CNM and Infomap on Q and NMI.

Acknowledgments

Firstly, I would like to thank my supervisor Yacine Atif for always motivating me to strive for the best and for keeping things simple and exciting. I truly appreciate the luxury of time that he has provided to me. Secondly, heartfelt “thanks” to my examiner Birgitta Lindström, for all the timely feedback, which has played a very big role in this report shaping in its current form. My family, for always standing by me in hard times and for telling me that the learning begins to happen when you acknowledge the fact that you know nothing. I would also like to thank all the lifelong friends I have made while studying at the University of skövde. Last, but not the least, Swedish Institute Scholarship (SISS) for providing me with this opportunity to study in Sweden.

Dedicated to my late uncle Kuentsho Samphel Dhendup, who has passed onto the other side recently.

Tshering Wangchuk, Skövde, May 2018

Table of Contents

List of Figures **Page No.**

List of Tables

List of Algorithms

1. Introduction

Social networks have gained massive popularity as millions of people are actively using them through mediums such as Facebook, LinkedIn, Instagram, WhatsApp, Twitter, and Youtube. These social networks have had huge impacts on people's lives in so many different ways, which is illustrated by the sheer number of people participating in it. People are increasingly using social networks to communicate with each other and to spread information by creating contents on these platforms. This has resulted in a massive amount of data being propelled by these social-network platforms which, if adequately captured and analyzed could develop into insights on how people interact with and influence each other in a way that derives new substantial commercial values for businesses. The process of investigating social structures through the use of networks and graph theory is called Social Network Analysis (Otte & Rousseau, 2002). Hence, a social network is modeled as a graph with vertices representing people and the edges between two adjacent vertices representing the relationship which models the interaction between people in the network.

When people communicate with each other in these networks, a person's emotions, opinions, or behavior are inherently affected by others within the network (Khousa & Atif, 2017), resulting in what is called as *social influence*. This means that people form their own opinions or embrace a personal action or change behaviors based on the opinions of other people whom they know and look up to. Hence, an individual's decision-making process is being influenced by people whom they are connected to in their social network or clique (Anagnostopoulos, Kumar, & Mahdian, 2008). It is also observed that different people in the network have different influence power, i.e., some people can influence a larger group of individuals than others, which results in some people acting as *Influencing agents* in the network. Businesses are now using these agents in the process of Viral Marketing, where a business uses social-network platforms to market their products by using the word-of-mouth phenomenon, spreading attractive information about their products (Ferguson, 2008) to advertise them. In recent times, the use of social media for marketing has become a hot-trend in affecting purchasing decisions of potential customers (Boon-Long & Wongsurawat, 2015). The constant online-presence of billions of users, expressing their interests, provides the opportunity for businesses to capture a global customer-base for their products or services.

This trend has shifted the focus from traditional marketing strategies to creative strategies which use social networks for maximum customer engagement (Ashley & Tuten, 2014). One of these

strategies consists of finding agents with good online-presence, who could engage potential consumers and advocate brands. To reduce cost and maximize ROI, businesses want to find the minimum set of these agents (within a social network) who can influence the maximum number of consumers in the network. This business strategy gave rise to a fertile research domain centered on **Influence Maximization problem (IM)** in contemporary social networks. Generally, it means to short-list some agents within a social network, who can spread information about business product or services to billions of potential consumers. This requires the agent to be someone who is known by and followed by a considerable extent of network users. In the jargon of IM, shortlisted candidates are termed as members of a *seed set* whose elicitation process consists of two steps: (1) identifying them and then (2) rank their influence-power to derive the optimal agents that match the allocated budget of a prospective marketing campaign.

There are many different methodologies such as surveys and interviews through which the problem can be approached, but we concentrate on analyzing social network data[1] as the data-driven approach.

Earliest literature includes optimizing marketing funds spent on attracting customers by using data collected from social sites (Richardson & Domingos, 2002). Since then almost all the other proceeding works referred to Kempe et al. (2003a) for solving the IM problem. In their paper, they formulated the influence maximization model as a discrete optimization problem where people in the network (vertices) are influenced (activated), only if, they adopt the behavior from influencing agents. The adoption of the behavior (also called activation) is modeled using assigned edge weights and once activated, the vertex remains in activated state. Different influence-weight assignment models were introduced which models the influence response of the people in the social networks. They fall under the two most-popular models: i) Independent cascade model (ICM) and ii) Linear threshold model (LTM), which have been introduced and discussed in almost all the preceding work in IM field (Arora, Galhotra, & Ranu, 2017a). These influence-weight propagation models are introduced and discussed in detail in Section 2 (Background and problem statement). The IM problem is typically NP-hard under both ICM and LTM but can give a decent approximation using the Greedy algorithm (Kempe et al., 2003a). Not until recently, Kim et al. (2013) studied the IM algorithm from the community structure perspective to reduce approximation difficulties by limiting the search for influencing

[1] https://snap.stanford.edu/data/

agents within communities, rather than across the entire network. Thereby, also reducing the time required to run the approximation. These existing methods use different edge weight models under their respective diffusion model like constant and uniform to propagate influence in the network (discussed in Section 2). In our project, we also address the influence maximization problem from the perspective of community-detection. However, unlike the existing method mentioned by Kempe et al. (2003a) and Kim et al. (2013), we propose to use real-world data in the form of common actions performed by users in social networks to assign edge weights and propagate the network. We advocate a five-phase approach to achieving this.

First, we generate an enriched synthetic-network by adding edges between similar nodes to the input social-network. For that, we use a judicious similarity-function based on common neighbors, which has been proven to support community-detection algorithms detect better communities (AlFalahi et al., 2013a). Then, we use this synthetic network to detect virtual communities using different community-detection algorithms[2]. After that, we suggest using the community structure to find the *important nodes'* within each of these virtual communities by proposing a fuzzy-logic based approach to calculate the influence weight for each node within the community based on both centrality measure[3] and common actions[4]. Followed by, calculating the social-network *reach* (number of nodes activated across entire network) of each of those *important nodes* using modified social-network diffusion models[5] as proposed by Alfalahi et al. (2013b). Finally, based on the number of nodes each important nodes activate in the entire network, the important nodes are ranked. A final set of these important users are then shortlisted as top-k seed set where *k* is a parameter that matches a given marketing-campaign budget **(check our community-based IM framework in Section 4).** Such research works to discover the most influential users (seed set) is paramount in performing different activities around the social network such as the targeted advertisement, and item recommendations to social network users, or supporting new behavioral lifestyle campaigns such as stop smoking. The aim is to involve as fewer influencing agents as possible while maximizing the influence spread.

[2] See section 2.2
[3] see section 4.3
[4] see equation 4.3
[5] diffusion Models (see subsection 2.1.1)

1.1 Project Contributions

We initially positioned the research along the timeline that starts from a given network and ends with the seed set nodes following the milestones indicated earlier. However, due to the shortage of time, through this project we make the following contributions:

i) We investigate the different works done by Alfalahi et al. (2013a; 2013b) and propose our version of Community-based IM framework which could be studied and improved by other researchers in the future.

ii) We begin experimenting on the proposed framework by designing an experiment to evaluate the step 1 and step 2 of proposed Community-based IM framework. Our experiments show that the step 1 (similarity-based preprocessing) has no effect in helping community-detection algorithms detect better communities. We also, extend the work done by Alfalahi et al. (2013a) by running the experiments on two more algorithms (Louvain and Infomap) which are then compared to the Fast Greedy (CNM) algorithm as proposed by Alfalahi et al. (2013a). We found Louvain to be better algorithm than fast greedy (CNM) for community detection, followed by Infomap.

iii) We also contribute an interactive Python-based implementation of the experiments through the use of different tools and techniques available. Use of tools like jupyter-notebook, networkX, i-graph, gelphi, numpy, and pandas are shared as Appendix in the report.

The rest of this document is organized as follows. In Section 2, we look at some background and gradually build our problem statement; then in Section 3, we present our literature review and discuss some related works. Section 4 reveals the design of our version of "Community-based IM Framework." In Section 5, we describe the experiment to evaluate the usefulness of similarity-based preprocessing of the input network to enhance community-structures used in subsequent phases of IM. Subsequently, Section 6, looks at ethics and validity of our study. Finally, Section 7 concludes this document with a summary, the results of our experiments and its implications on the framework under design, our contributions, and some future direction suggestions. There are also additional appendices, where we present the different tools used in our project and links to know more about using those tools that could help readers in similar projects. We also show the code listings developed throughout the project, so that the project is open to interested people involved in similar works.

2. Background and problem statement

In this section, we look at some preliminaries (concepts) that are required for understanding the Influence Maximization problem, and the proposed IM framework. We converge towards setting the scope of the project through a formal definition of the IM problem mathematically.

2.1 Influence Maximization

As discussed earlier, social networks are modeled as a graph with nodes representing the users and the edges representing the relationship between them quantified by the edge weights. We first define the social network as shown in the following Definition 2.1. As to build the understanding on a linear way, the following definitions are quoted from Arora et al. (2017) since it is structured very well.

Definition 2.1 Social Network (Arora et al., 2017)

"*A social network with 'n' individuals and 'm' social ties can be denoted as an edge-weighted graph G (V, E, W), where V is the set of nodes, |V | = n, and E is the set of directed relationships, $E \subseteq V \times V$, |E| = m, and W is the set of edge weights corresponding to each edge in E".*

Next, we define seed node, active node, and the two most popular diffusion models used i) Linear Threshold Model (LTM) and ii) Independent Cascade Model (ICM) in the following sets of definition.

Definition 2.2 Seed node (Arora et al., 2017)

"A node $v \in V$ that acts as the source of information diffusion in the graph G (V, E, W) is called a seed node. S denotes the set of seed nodes".

The set of seed nodes which are used to propagate influence and supposedly maximizes the influence in the network is called as the **seed set**.

Definition 2.3 Active node (Arora et al., 2017)

"*A node $v \in V$ is deemed active if either (1) It is a seed node ($v \in S$) or (2) It receives information under the dynamics of an information diffusion model I, from a previously active node $u \in V_a$. Once activated, the node v is added to the set of active nodes V_a".*

2.1.1 Diffusion Models

There are different information (influence) propagation models that have been postulated to propagate information. Two basic types are i) Viral propagation and ii) Decision-based

propagation. *Viral propagation* models the way the virus is spread in cells, which are thought to have the *broadcasting* effect. However, the spread of information in a social network depends on the edge weights; so decision-based propagation models, using the edge weights to make decisions regarding the activation of the nodes, are preferred over the viral propagation models (Zhukov, 2017). The two most popular decision-based propagation (diffusion) model which are used in social networks are Independent cascade model (ICM) and linear threshold model (LTM) (Kempe, Kleinberg, & Tardos, 2003a). We will define them as follows.

Independent Cascade Model (Arora et al., 2017)

"Under the IC model, time unfolds in discrete steps. At any time-step i, each newly activated node $u \in V_a$ *gets one independent attempt to activate each of its outgoing neighbors* $v \in Out(u)$ *with a probability* $p(u,v) = W(u, v)$*. In other words,* $W(u, v)$ *denotes the probability of u influencing v".*

Linear Threshold Model (Arora et al., 2017)

"Under the LT model, every node v contains an activation threshold θ_v*, which is chosen uniformly at random from the interval [0, 1]. Further, LT dictates that the summation of all incoming edge weights is at most 1, i.e.,* $\sum_{\forall u \in In(v)} W(u, v) \leq 1$*. v gets activated if the sum of weights* $W(u, v)$ *of all the incoming edges (u, v) originating from active nodes exceeds the activation threshold* θ_v*".*

For both ICM and LTM, we need to have a seed node (Definition 2.2) $s \in S$ activated and added to active nodes V_a. Also, in both the models, the activation of a node is based on the probability weights, $p(u, v)$ in ICM and the threshold value θv, in LTM. Different types of edge weights can be used for each of the different diffusion models.

2.1.2 Edge Weights assignment in Diffusion Models

The edge weight determines the spread of influence of the seed nodes we have selected in the network. There are different edge-weight models for both LTM and ICM. They are:

a) ICM based edge-weight model

- **Constant**: In this model, W (u, v) representing the weight between nodes u and v, has a constant probability p with values as either 0.01 or 0.1 (W. Chen, Wang, & Wang, 2010; Goyal, Lu, & Lakshmanan, 2011b; Kempe et al., 2003a).
- **Weighted Cascade**: In this model, the assigned weight is $\frac{1}{|In(v)|}$which means that all incoming neighbors of v influence v with equal probability. Thus, it is easier to influence low-degree nodes while using this model (Tang, Shi, & Xiao, 2015; Tang, Xiao, & Shi, 2014).
- **Tri-valency Model**: In this model, weights are chosen randomly from a set of probabilities [0.001,0.01,0.1] (Chen et al., 2010; Goyal et al., 2011b; Kempe et al., 2003a; Cheng et al., 2014).

b) LTM based edge-weight model

- **Uniform:** In this model,$\frac{1}{|In(v)|}$ is used as weights. It is similarly weighted as in the Cascade Model (Galhotra, Arora, & Roy, 2016).
- **Random:** Here, a random value between 0 and 1 is assigned (Tang et al., 2015, 2014).
- **Parallel Edges:** This weight-model is for multigraphs (nodes which have more than one edges coming in or going out from them). In such scenarios, there may be a node which is communicating to the other node more than one time. Such node with two or more edges directed towards the other node has what is termed as the "parallel-edges". In this model, the parallel edges are consolidated into a traditional graph. Here $W(u,v) = \frac{c(u,v)}{\sum_{\forall u' \in In(v)} c(u',v)}$ where $c(u,v)$ is the number of parallel edges from node u to v. It is said to be generalization of the uniform model used for the multi-graph case (Goyal et al., 2011b).

The diffusion models together with the different edge-weight models are used to propagate the influence in the network to see how much nodes the selected seed set activates. The seed set which maximizes the influence the most is selected. It is noteworthy to understand that the number of nodes in the seed set has to be set in advance. Figure 2.1 below shows the algorithm showing how the influence spreads in the network (Arora et al., 2017). It spreads in a discrete time-step, *i.* If S_0 is the initial seed nodes at the time, $i = 0$, at each time S_0, the

diffusion model, *I*, activates newer nodes at time S_1. The nodes cannot be deactivated once it has been activated. This continues until no node can be activated further.

Algorithm 2.1: **Spread (Arora et al., 2017)**

Input: Graph $G = (V,E,W)$, seed set S_0, diffusion model I

1: $i \leftarrow 0$

2: **repeat**

3: $i \leftarrow i + 1$

4: $A \leftarrow$compute the newly active nodes at time-step i under I

5: $S_i \leftarrow S_{i-1} \cup A$

6: **until** $S_i - S_{i-1} = \emptyset$

7: $V_a \leftarrow S_i$

8: **Return** V_a

Definition: Spread (Arora et al., 2017)

"Given an information diffusion model I, the spread $\Gamma(S)$ of a set of seed nodes S is defined as the total number of nodes that are active, including both the newly activated nodes and the initially active set S, at the end of the information diffusion process. Mathematically, $\Gamma(S) = |V_a|$ ".

Here, since the value of spread is calculated using some diffusion models, it is calculated in the form of *expected value* since these diffusion models are stochastic. This spread function is usually implemented using Monte-Carlo simulations of 10,000 (Kempe et al., 2003). With all these preliminaries defined, we can now finally define the Influence maximization problem formally.

2.1.3 Problem statement

Influence Maximization (IM) problem (Arora et al., 2017)

"Given *an integer value k and a social network G, select a set S of k seeds, $S \subseteq V \mid k = |S|$, such that the expected value of spread $\sigma(S) = E[\Gamma(S)]$ is maximized."*

As motivated in the Introduction section, the real-life use of the solution to this problem can be implemented in so many different ways. Our context in this project is to optimize the cost and

ROI for Businesses, who wants to market their product but wants to finds out which users should they be hiring for a minimal cost. For this purpose, the business need for identifying the seed set, *S* that can maximize the spread *σ(S)*, given a social graph *G*, a diffusion model *I*, and budget *k* is satisfied. We will now look at the other concepts, which are necessary for our proposed solution to the IM problem.

2.2 Community Detection

Dividing the social networks in terms of similar-user can help in reducing the search space when we propagate the influence in social networks (Kim et al., 2013). The problem that confronts researchers is to figure out how to group users who share similar interest with each other in a network. This can be achieved by detecting communities in social networks. There are many methods or algorithms which have been developed over the years for detecting communities. Alfalahi et al. (2013a) proposed to use the Fast Greedy (CNM) algorithm with the similarity based preprocessing-step since CNM is modularity-based and faster compared to other algorithms. In addition to CNM, we also select Louvain and Infomap in our experiments; since they are high performing community-detection algorithms (Bródka et al., 2010). **CNM** was developed by Clauset et al. (2004) and relies on the greedy optimization applied to a bottom-up agglomerative to combine and merge clusters in the network. Initially, every node will be their own community and it keeps joining these individual communities until it all becomes one and the modularity is maximized. The best partition of community is selected by comparing the modularity of different cluster with each other. Another modularity based approach is the **Louvain method** proposed by Blondel et al. (2008). It is an improvement of CNM introducing two-step hierarchical agglomerative strategy. First phase applies greedy approach to detect communities and during the second, it builds a network whose nodes are communities of the first. Within-community edges are given by self-loops, whereas the intercommunity edges are aggregated and given as edges between the new nodes. The process is repeated until only one community remains. **Infomap** is another method developed by Rosvall et al. (2008). The community structure has two levels to it. It is based on Huffman coding, where one of the level, is to differentiate communities in the network, and the other is to differentiate the nodes. Infomap is chosen to see the results for a complete random algorithm which uses a completely different approach of community detection from Louvain and CNM.

2.3 Fuzzy Logic

Fuzzy logic was introduced by Zadeh (1965). While traditional logic approach use binary values 1 for true and 0 for false values, it is not inclusive of the situations when the truth values are

not 0 nor 1 but has vagueness to it. Fuzzy logic using probability enables us to represent the values between 0 and 1. Hence, fuzzy logic is implemented in situations which requires gradual levels of affirmation on decisions like yes/no, or of truthfulness like true/false (Rojas, 2013). The process of reasoning is usually approximate rather than fixed or exact. This kind of reasoning is more human-like thinking (Zadeh, 1984) and is seen more in real-life situations where the decisions are not binary. We use the fuzzy logic inspired method in our proposed approach to determine the user of a social network with high probability to join the membership of "important nodes" which is used for influence propagation by selecting few of these important nodes as seed set. Fuzzy logic has a unique feature of being flexible and straightforward human language rule-based approach (Ahmed et al., 2013). A fuzzy-logic based system converts these affirmation into their mathematical equivalents which provides a more realistic behavior of the system in real world (Rahman & Ratrout, 2009). These Fuzzy systems requires a membership function that helps evaluate the correct value between 0 and 1 to conform to a given affirmation (Rojas, 2013). This approach is inspired by Wolfram et al. (2014) and it includes making fuzzy membership decision to find the important user in the network when two constraints are given and when one has to make decisions based on ranges of values representing each constraints. It is exemplified in the following subsection.

2.3.1 Fuzzy Logic inspired approach exemplified

Let's assume that we want to find the most important nodes in a social network as defined in Example 2.5. Let assume the following situation:

Example 2.5 : *Given a social network of five nodes, n ={1, 2, 3, 4, 5, 6} we want to find the nodes that will be the "most important", featured by the constraints that the node should be in a central location (represented as centralWeights) and has a high-influence weight (represented as Infl_Weight) on other nodes in the network.*

Let say we calculated the centrality of nodes in the network and the following values represent the first constraint using a fuzzy set of centrality values as given below:

centralWeights (CW) = {{1, 0.3}, {2, 0.6}, {3, 0.8}, {4, 0.4}, {5, 0.5},{6,0.4}} *Equation (2.5)*

Equation 2.5 shows that Node 3 has the highest membership grade with the value of 0.8, which means that Node 3 has the highest centrality in this network. While Node 1 has the

lowest membership grade of 0.4. This represents our first constraint hypothetically.

Then we represent the second constraint in a fuzzy set of average influence weight computed using common actions of users in the network and let's say the values are as given below:

Infl_Weight(IW) ={{1, 0.1}, {2, 0.9}, {3, 0.7}, {4, 1}, {5, 0.2}, {6,0.6}} *Equation (2.6)*

In the fuzzy set (*Equation 2.6*) the membership grades indicate the average influence weight of a node in the whole social network. The highest mean the most influential in the network in terms of user common actions. From that set, we notice Node 4 is the most influential with the grade of 1 while Node 1 is the least influential in the whole network with grade 0.1.

After representing both constraints as fuzzy sets, we then decide on nodes that are most influential by considering both of the constraints defined through the different values evaluated in terms of both centrality and common actions of a user. For that, we apply the intersection of these constraints to drive the fuzzy decision. It can be perceived as combining the given constraints to come up with the best overall decision (Wolfram, 2014). The fuzzy intersection between two fuzzy sets is computed by considering for each element (nodes in our example) the minimum of its membership in both sets (Rojas, 1996). This is done as a step to deal with network defects which might have given high values in both the constraints. Selecting the minimum as *intersection* and then applying the maximum of *intersection* ensures that the decision gives us the most important nodes (Alfalahi et al., 2013b). Thus the minimum and the maximum evaluation is as given below in Equation 2.7 and Equation 2.8 respectively:

Intersection = {{1, 0.1}, {2, 0.6}, {3, 0.7}, {4, 0.4}, {5, 0.2}, {6, 0.4}}. *Equation (2.7)*

Important-node = max (Intersection) *Equation (2.8)*

It is to be noted that the example was just a very basic implementing of fuzzy based membership function and in our actual implementation, the network given here would scale to different communities detected and finding the most influential nodes in the community using the given constraints. We now move on to our literature review and the related works in the IM field in the next section.

3 Literature Review and Related works

To understand what sort of work was done and what kinds of solution to the IM problem were proposed in the field, we see a requirement to study the field of IM to see which work in the IM field is considered the state-of-the-art. One question which needs to be answered is - which is the baseline approach which every researcher is benchmarking their proposed solution with. To answer this fundamental question, we thus design a Literature review, since Literature reviews are conducted to "*identify, analyze and interpret all available evidence related to a specific research question*" and it is a methodology which helps researcher understand the state-of-the-art in an area (Wohlin et al., 2012, p45). It is conducted based on a three-step process of *planning, conducting and reporting the review*. We thus present the planning step with the details on how we designed the study; followed by how we conducted it and reported on what we found in this section.

As part of the planning step, we choose to look at different approaches that were proposed within the IM field over the years (from 2002 to 2017). The year 2002 was chosen because the IM problem in its primitive form was first studied in that year (Richardson & Domingos, 2002). Based on the need to know the baseline approach (state-of-the-art) in the IM field, we formulate the following research question:

What are the different IM approaches that were proposed since the inception of IM problem and which baseline approach do they use to benchmark their approaches?

To this end, we design a very basic literature review to answer a very basic question. The selection of the different approaches to solve the IM problem were made based on the conferences (primary sources) in which the work (paper) was presented. This is done to ensure that the literature under consideration has undergone through the process of rigorous reviews and we feel that a conference papers validates the fact that the chosen work has undergone a rigorous review. We also took the help of comprehensive benchmarking study conducted by Arora et al. (2017) to shortlist the different approaches they think are the candidate for state-of-the-art as presented in their paper. The keywords used were the name of the "algorithm" the respective authors have called their approach (as shown in Table 3.1) and the quality of the approaches was controlled using the fact that the method should have been presented in well-known conferences or venues. We have selected 12 well-known algorithms to solve IM problem with their year and conference in which they were presented in. Then we proceeded to see which

baseline approach they compared the performance of their algorithm with and report our finding to answer the research question we framed for our literature review. We have summarized the whole literature review under the subsection titled "evolution in IM area." We also highlight some few works that are very close to our proposed solution of using community detection to solve the IM problem in our related works subsection.

3.1 Evolution in IM area

Influence in networks was first studied by Richardson et al. in (2002) in the context of a viral marketing approach by mining the networks from data and building probabilistic models to choose the best viral marketing plan. The data used to mine the network were from knowledge-sharing sites where customers reviewed products and advised each other. They claim to have successfully optimized the cost for each customer, instead of just considering the binary event of marketing or not-marketing to that particular customer. This work started the whole practice of studying the relationship between customers in a network by measuring influence among members of the network.

After Richardson et al. (2002), Kempe et al. (2003b) in its seminal paper studied influence maximization as an optimization problem through the use of “influence propagation” in social networks using LTM (Linear Threshold Model) or ICM (Independent Cascading Model). They approximated the influence spread using a Greedy algorithm. This work has since been driving the IM field forward as it defined different standards, properties or challenges in the IM field that one had to overcome to solve the IM problem. Some of the findings from Kempe et al. (2003b) needs to be highlighted here because all the other approaches which came after it was based on improving the greedy algorithm in its limitation of being not scalable on a (even) medium sized network.

3.1.1 Properties of IM problem

There are some challenges associated with IM problem as defined by Kempe et al. (2003b) shown as the theorem in the following:

Theorem 1. *Under both ICM and LTM, the IM problem is NP-Hard.*
For proof. Please see (Kempe et al., 2003b).

Fortunately, Kempe et al. (2003b) showed that the spread function Γ (·), and its expectation σ (·) = E [Γ (·)], is monotone and submodular while using both the ICM and LTM. This means that using this inherent property, the greedy algorithm can work around the problem of NP-hard, the process of iteratively choosing the element with maximal marginal gain approximates the optimal solution with a factor of $1 - 1/e$ (Fischetti & Williamson, 2007). Kempe et al. (2003a) implemented the Greedy algorithm (**Algorithm 3.1**) and validated it and therefore devising theorem 2.

Algorithm 3.1: **Greedy IM (Kempe et al., 2003b)**

Input: Graph $G = (V, E, W)$, k, diffusion model I

1: $S \leftarrow \emptyset$

2: $i \leftarrow 0$

3: **while** $(i<k)$ do

4: $i \leftarrow i + 1$

5: $v* \leftarrow arg\ max_{\forall v \in V} \{\sigma(S \cup \{v\}) - \sigma(S)\}$ under I

6: $S \leftarrow S \cup \{v*\}$

7: **end** while

8: **Return** S

Theorem 2 (Arora et al., 2017)

"The expected value of spread computed using the seed set returned by the Greedy algorithm is within $1 - \frac{1}{e} - \varepsilon$ *of the optimal. Mathematically,* $\sigma(S) \geq (1 - \frac{1}{e} - \varepsilon)\, \sigma(S*)$

Where S is the seed set computed ***by Algorithm. 3.1*** *and* $S*$ *is the optimal seed set. Furthermore,* $\sigma(S)$ *is the best possible approximation in polynomial time".*

Theorem 2 established the fact that Algorithm 3.1 can give an approximation guarantee of 63 % (Kempe et al., 2003) in polynomial time but it is not scalable, even for a network of medium size.

On these premise, we then went forward and conducted a literature review of approaches different authors have proposed over the years. As mentioned in the design of the literature

review, we chose the approaches that were proposed henceforth after Kempe et al. (2013b). The review to find out the "baseline approach used" (state-of-the-art) is as shown in Table 3.1.

Algorithm	**Author**	**Year and Conference**	**Baseline Approach**
CELF	(Leskovec et al., 2007)	KDD & 2007	Greedy
LDAG	(W. Chen, Yuan, & Zhang, 2010)	ICDM & 2010	Greedy
CELF++	(Goyal, Lu, & Lakshmanan, 2011a)	WWW & 2011	Greedy
SIMPATH	(Goyal et al., 2011b)	ICDM & 2012	Greedy
IRIE	(Goyal et al., 2011a;Jung et al., 2012)	ICDM & 2012	Greedy
Static Greedy	(Cheng et al., 2013)	CIKM & 2013	Greedy
PMC	(Akiba, Iwata, & Yoshida, 2014)	AAAI & 2014	not within the group
TM+	(Tang et al., 2014)	SIGMOD & 2014	Greedy
IMM	(Tang et al., 2015, 2014)	SIGMOD & 2015	Greedy
EaSyIM	(Galhotra et al., 2016)	SIGMOD & 2016	Greedy
ComPath+	(Bagheri et al, 2016)	ICNC & 2016	Greedy
Cl2	(Bozorgi et al, 2017)	KBS & 2017	Greedy

Table 3.1: IM algorithms introduced sorted by year and baseline algorithm

The literature review looked at 12 different approaches that were presented in well-known conferences as listed in Table 3.1, and each of the approaches was read to find out which baseline approach the authors of the respective approach compared their proposed algorithm to. We found that 11 out of 12 well-known algorithms that are listed in Table 3.1 compared their approach with a Greedy algorithm as the baseline approach. It was also seen that all the algorithm aimed to solve the limitation of greedy in terms of its scalability in the large network. The authors of the all the approaches mentioned in Table 3.1 claim that their approach performs better with faster time but does not mention how well their approaches approximates the spread

compared to Greedy. This establishes the fact that Greedy Algorithm is still relevant to these days and its approximation guarantee of 63% has not been bettered yet.

The greedy algorithm with all its limitation still seems to be the closest to the state-of-the-art, in terms of approximation guarantee. The other approaches that we studied claim to have lesser running time but does not have any claim on their approach improving the approximation guarantee. However, since the Greedy is still limited in its scalability, there is no state-of-the-art approach in IM problem and because of which, we think the field of IM problem is still a very relevant research problem to solve. We hope that our approach of using the Community based IM framework would solve the IM problem in terms of both running time and approximation of spread.

3.2.2 Related works

Solving the IM problem using Community detection was first worked on by Kim et al. (2013) who approached the IM problem using community detection. The rationale behind the approach was to limit the search space to within communities and reduce the running-time for scalability improvement. The communities are detected using the Markov clustering algorithm.

Another paper on IM problem which was studied from the perspective of community detection was done by Bozorgi et al. (2016), in which after they detect communities, they calculated the spread both locally within the communities and combined it with the global spread later on. This approach was also based on LTM model.

One such approach was proposed very recently by Bozorgi et al. (2017) again. In this approach, after detecting communities, like in our proposed framework, candidate nodes are selected from the communities, and the Greedy algorithm is applied by using Linear Threshold Model (LTM) to measure spread.

These approaches are similar, yet so different to our proposed approach because, instead of some edge-weight assignment models, we propose using external data to learn the edge weight using a fuzzy logic inspired approach to measure the spread. Once the important users have been selected, we run modified ICM diffusion model developed by Alfalahi et al. (2013) to propagate and return the spread given by the selected seed set.

4. Community-Based IM framework

In this section, we introduce our version of proposed community-based Influence Maximization framework to solve the IM problem for business marketing as we discussed previously; Figure 4.1 shows the overall view of our proposed framework. We will explain the framework and its steps that we designed to solve IM problem in the following subsections.

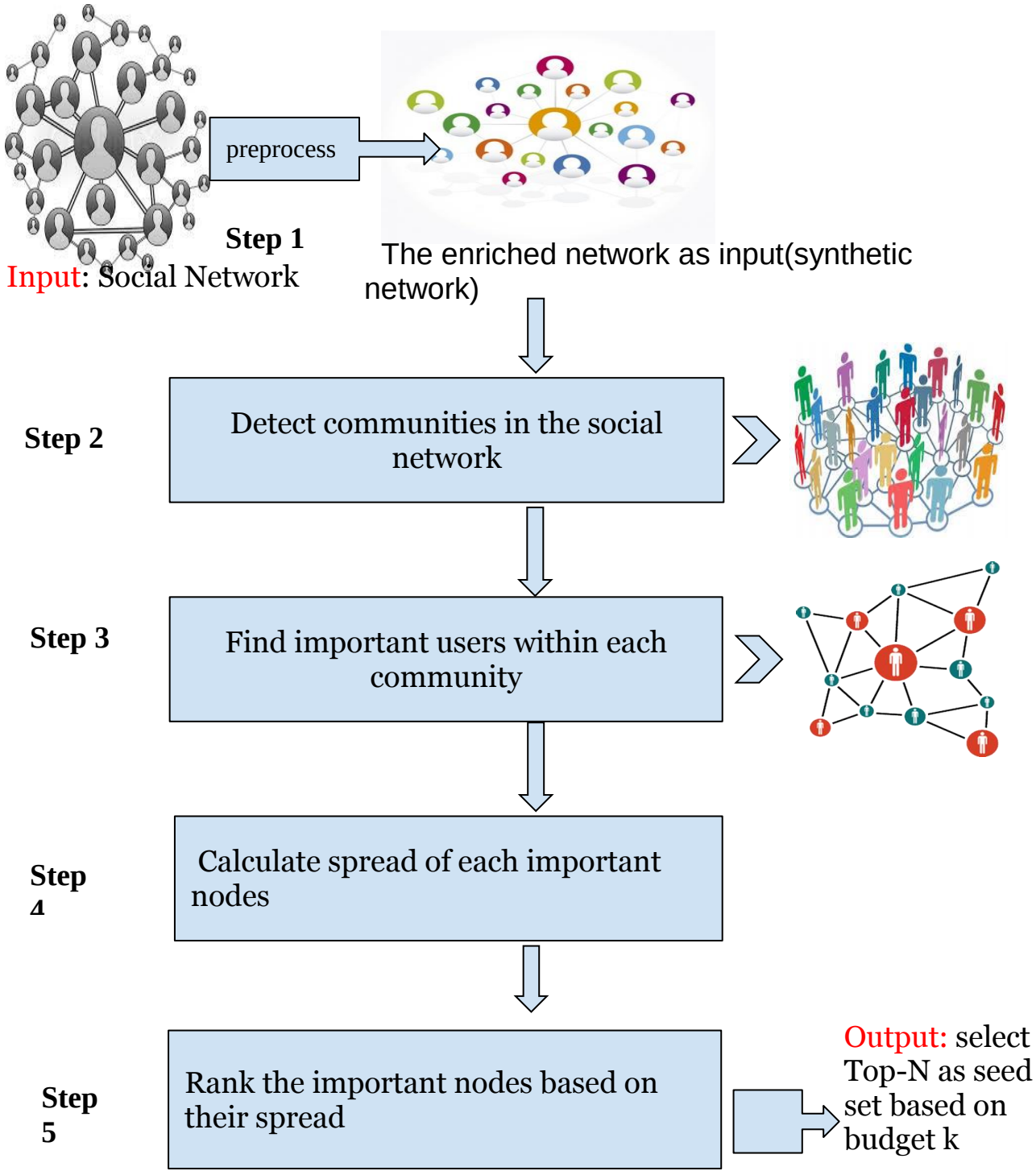

Figure 4.1: Proposed community-based IM framework

As shown in Figure 4.1, we propose a 5-step solution for Influence Maximization. We begin by performing similarity-based preprocessing step (Step 1) on the social network to enrich its edges, resulting in a new enriched network. The enrichment of the network takes place by applying Equation 4.1, which calculates the similarity coefficient based on the common neighbors for each node in the network. This gives us a way to find the most similar node for each node in the network which doesn't have an edge already existing between them. After evaluating the most similar node for each node, adding the edge between the two most similar nodes results in enrichment of the network (Alfalahi et al., 2013). Then, we detect communities in the enriched network (Step 2) using community detection algorithm. Followed by, finding important nodes within the detected-communities (Step 3) using fuzzy-logic approach. These spread for each of the important node is approximated (Step 4) and ranked based on their spread (number of nodes it activates). Finally, in step 5, the seed set is selected based on the budget k. Algorithm 4.1 shows the Algorithmic view of the framework is inspired from Alfalahi et al. (2013b).

Algorithm 4.1: Community based IM (modified from Alfalahi et al. (2013b))

1) Create an enriched network using similarity function, S(G) ->G' (**Algorithm 4.2**)
2) Detect communities C of G' (community detection algorithm)
3) For each C do:
    a) Find central user:
        i) centralUsers <- Find central user(centrality algorithms)
        ii) for each centralUsers do:
            A) centralWeight <- nodeDegree/totalEdges **(Eq. 4.2)**
    b) Find influence weight :
        i) for each centralUsers do:
            A) infl_Weight(jaccard coefficient (**Algorithm 4.3**)
        ii) for each centralUsers do:
            A) avg_infl_weights= sum(influenceWeight)/total nodes **(Eq. 4.3)**
    c) Find important users(Fuzzy Decision)
        i) $intersection_n$ = min($centralWeight_n$, $avg\_infl\_weights_n$) (**Eq 4.4)**
        ii) important_users <- max($intersections_n$) for all u in set CN **(Eq 4.5)**
4) For each important_users :
    a) S <- Apply propagation model **(Algorithm 4.4)**
    b) S <- sort important nodes based on their reach
5) Select the Top-N seed set based on Budget k

4.1 Similarity-based preprocessing

As shown in Figure 4.1 and Algorithm 4.1, the first step of our framework is to enrich the network based on a similarity measure. The similarity function *S* reads an input network *G* and transforms it into network G' which is what we feed to our IM framework. This step adds new edges between nodes that are most similar to each other (implemented using Algorithm 4.2). The reason for doing this is rationalized in Alfalahi et al. (2013a), where they found out that applying this similarity-based preprocessing to the input network and transforming it into what they call "virtual networks" optimizes the community structure. To evaluate this finding, we performed our experiment (in Section 5) to test; if the similarity-based preprocessing improves the quality of community detected in the context of the IM problem. The similarity function implemented by Alfalahi, Atif, & Harous (2013a) is given as follows:

$$\boldsymbol{Similarity}(\boldsymbol{a},\boldsymbol{b}) = \frac{\boldsymbol{adj_{ab}}+\boldsymbol{cn_{ab}}}{\boldsymbol{n_a}+\boldsymbol{n_b}} \qquad \textit{Equation (4.1)}$$

Where, $\boldsymbol{adj_{ab}}$ represents the intersection of node '*a'* and node '*b'* in the adjacency matrix, equal to 0 if there is no edge between the nodes and 1 if there exists an edge. $\boldsymbol{cn_{ab}}$ represents the number of common neighbors of node *'a'* and node '*b'*. n_a and n_b are the number of neighbors of node *a* and number of neighbors of node *b* respectively.

In our experiment, we replicated the similarity function to transform our input network using the following algorithm in 4.2.

Algorithm 4.2: Similarity preprocessing (Alfalahi et al., 2013a)

1) Transverse each node in the given graph *G*
2) For each node *a* and *b* do:
    i) calculate similarity based on ***Equation 4.1***
    ii) for each node *a* find the highest similar node *b*:
        a) add an edge between nodes *a* and *b*
3) Save the similarity enriched network as *G'*

Thus, at the end of step 1, as given in Algorithm 4.1; we generate a new enriched network *G'* from the input social network *G*. The time complexity of the algorithm is of order O (n^2), but

since it is performed as the (offline) preprocessing step, the complexity doesn't really affect the time complexity of the framework greatly, since it is used to transform the input network and only have to be run once, which greatly reduce the time when running again.

4.2 Detecting Communities

The next step in our framework is to start detecting the communities in the enriched network *G'*. We also look at three high performing community detection algorithms which performed well from experiments conducted by Bródka et al. (2010) and Orman, Labatut, & Cherifi (2011). We use and evaluate those algorithms' performance on the network *G'* (see section 5 for details) by conducting our experiments. Using these algorithms, we detect communities C in the social network.

4.3 Finding important-users within each community

After we detect the communities in the network graph G', the next step is to find the most important user within each community. To help us achieve this, we perform the following sub-steps.

4.3.1 Find central user

As shown in Algorithm 4.1, after we have detected community structure C_n in network G', for each community we detect; we try to find the central user within each community. There are diverse categories of centrality measure we can use to find central node like page rank (Nathan, Zakrzewska, Riedy, & Bader, 2017), betweenness centrality (Brandes, 2001), degree centrality (Diestel, 2010) to name some few. A further study may be required to choose the most appropriate centrality algorithm to find the central user in each community in our framework. After we find the central user in each community, for each central user we assign the centrality weight for each central user by using the Equation 4.2.

$$centralWeight = \frac{nodeDegree}{Total\ Edges} \qquad \textit{Equation (4.2)}$$

Where $nodeDegree$ represents the variable which holds each node's in and out degrees, calculated using a centrality algorithm which is to be studied in the future, and $Total\ Edges$ is the total number of edges in the network. The resulting values vary between [0, 1] which reflects centrality level of each node in the network. This measure is used to assign $centralWeight$ to each central users detected, which would then be used later in a selection of the important node

through the Fuzzy Logic inspired approach to making fuzzy membership decision(see Subsection 4.3.3).

4.3.2 Find influence weight

In our framework, we propose to calculate an additional weight called "influence weight," which shall be based on using external data collected on the users. This is the step that is unique in comparison to other existing approaches to solving IM problem. We devise a way to include external user-data to help determine the important user in our network. This ensures that we assign weights based on, not just the structure of the social network (centrality), but also based on external data from users' action. We hope that assigning edge weight based on the users' common action, rather than just using some random models to assign weights to the edges (as mentioned by Kempe et al. (2003b)) would result in better accuracy in finding the most influential user.

Here, we propose to collect data on the interaction between each node. For example, if the network is a social media, then we could capture information like the number of common likes each user has, number of common friends, number of shares shared by the nodes within each community and based on those "common actions" ; we can then assign influence weights to each candidate central user in each community. For extraction of common actions, we propose to use of *"Jaccard Coefficient Based on Common Actions Algorithm"* as developed by AlFalahi, Atif, & Abraham (2013b)**.** The algorithm 4.3 is as follows.

Algorithm 4.3 Jaccard Coefficient Based on Common Actions (Alfalahi et al.,2013b)

1. Find all $actions_u$;
2. Find all $actions_v$;
3. For all a ∈ $actions_u$ do:
   (a) if a is in $actions_v$ AND $time_{au} < time_{av}$ do:
      i) $common_{u,v} = common_{u,v} + 1$;
4. $JC_{u,v} = \frac{common_{u,v}}{common_u + common_v - common_{u,v}}$
5. return $JC_{u,v}$

After we extract Jaccard-coefficient for common actions, we assign it as the influence weight of each node in the community. We further process to find the average influence weight to discriminate nodes with highest historical influence actions in the network. This is done using the following Equation 4.3.

$$avg\_infl\_Weights = \frac{sum\_infl\_weights(node)}{totalNodes} \qquad \textit{Equation (4.3)}$$

Where $sum\_infl\_weights(node)$ gives the sum of all influence weights that a specific node has on all other nodes in the network and $totalNodes$ represents the total number of nodes in the social network. So, this is how we calculate the influence weight, which would then be used later in the selection of the important user through the Fuzzy Logic inspired approach to making fuzzy membership decision (see Subsection 4.3.3).

4.3.3 Find important user using the fuzzy Logic inspired approach

Till this point, we have centrality weight of each central users in the community, and average influence weight of all central users in the community, now we proceed to find the important users using the two weights that we inferred. To do this, we propose a Fuzzy Logic inspired approach (as discussed in Section 2). Important users are found based on a minimum Fuzzy-Set Intersection value for each central user (node n) using both centrality Weight and average Influence Weights as shown in the following Equation 4.4

$$intersection_n = min(centralWeight_n, avg\_infl\_weights_n) \qquad \textit{Equation (4.4)}$$

Where $intersection$n finds the minimum of the central weight and the average influence weight of a node (central user).The minimum intersection for each node is implemented in Equation 4.4; to minimize the effect of having a defect in either the central weights or the influence weight (Alfalahi et al., 2013b). Then, we proceed to select the maximum generated from the intersection above to finally decide on the important users having the maximum of the intersection in equation 4.4. This is shown by the following Equation 4.5.

$$important\_users = max(intersection_n) for\ all\ n \in CN \qquad \textit{Equation (4.5)}$$

Where CN represents the set of central nodes in the social network

4.4 Selecting the seed set

Once we found the set of important users based on location (central weight) and common actions (influence weight), we can then run influence propagation model, which is a modified ICM or Independent Cascade Model (Alfalahi et al., 2013b) for each important users and extract their reach which is basically the number of nodes the particular important user activates. It is shown in Algorithm 4.4 below, which returns the list of important nodes, which has the best coverage or the spread, as S. As shown in Algorithm 4.1, we then rank the seed set S based on the number of nodes it activated and then select the nodes based on the budget k. For example, if k=5, we see that we need 5 influence agents to market for us, so we choose the top 5 as the seed set.

Algorithm 4.4 Modified ICM (Alfalahi et al.,2013b)

(1) $\forall u \in$ important_users do:

(a) At step t = 0, activate u ∈ important_users and add it to $Coverage_0$

(b) At each step t > 0, ∀u ∈ $Coverage_{t-1}$ do

(1) $\forall v_{inactive}$ if $Infl\_weight_{u,v}$ > InfluenceThreshold

(A) Activate v

(B) ActiveList = ActiveList ∪ {v}

(C) TotalCoverage = TotalCoverage + 1

(2) All the nodes activated at this step are added to $Coverage_t$

(3) This process ends at a step t if $Coverage_t = 0$

(2) Add nodes u with highest TotalCoverage to S

(3) Return S

Thus, this is how we propose to solve IM problem through our community-based IM framework. For this project and due to the shortage of time, we were not able to implement the whole framework, so we report the work that we did on the first 2 steps in our experiment section.

5. Experiment

This section deals with the experiment that we conducted to evaluate the community detection approach we introduced in *community-based IM framework* in section 4. Due to the limited time, we only implemented the step 1 and step 2 of the proposed framework. As mentioned in the Algorithm 4.1, the step 1 of our algorithm proposes to use a similarity based preprocessing step to enrich the input social network before applying step 2 which detects communities in the social network. As discussed earlier, step 1 is implemented because the previous study conducted by Alfalahi et al. (2013a) showed that the similarity-based preprocessing step helps in detecting better community structure (measured using Modularity and NMI as evaluation metric to measure community quality) in the context of community detection. They also proposed to use CNM as a community detection algorithm due to two reason, CNM being modularity based algorithm and CNM being faster than other alternative algorithms. Since we are implementing a community-based framework for solving IM problems, we have to make decisions on which algorithm to use for community detection in our proposed framework and also make the decision on whether to use similarity-based preprocessing step as proposed by Alfalahi et al. (2013a) in our proposed framework. To help us make an informed decision, we conduct experiments to help us make these decisions because *"experiments are a valuable tool for all software engineers who are involved in evaluating and choosing between different methods, techniques, languages, and tool"* (Wohlin et al., 2012). Here, since we have to make choices between different community detection algorithms and also have to find out effectiveness of the preprocessing step is, we use experiment as our methodology for achieving our objectives (defined in next subsection). Other methods like case study, literature survey, qualitative doesn't help our cause due to its generalizability issue.

In this section, we first state the objective of the experiment through experiment objectives. Then we explain the design of the experiment with subsections on dataset used and the candidate community-detection algorithms that we considered to compare. Followed by the evaluation metrics used to measure the quality of community detected by the candidate algorithms and finally, we present the results and discuss them to report our findings against our objectives that we set.

5.1 Experiment objectives

In our approach, rather than detecting communities directly using some already available community detection algorithm, we proposed to apply a preprocessing step[6] to enrich the network by calculating the most similar nodes for each node within the network and assign the edge between those similar nodes. This is implemented because as shown by Alfalahi et al. (2013a), additional preprocessing step when applied (before applying community detection algorithm) results in detecting better community structure (measured through Modularity and NMI). Our experiment has thus two objectives.

a) Analyze the application of similarity based preprocessing on social network G to measure the effectiveness of preprocessing-step with respect to the quality of community from the point of view of a developer in the context of solving IM problem using community detection.
b) Analyze which candidate community detection algorithm should be used for the purpose of detecting communities in social network G with respect to the quality of community from a point of view of a developer in the context of solving IM problem using community detection.

Hence, we design an experiment to realize the objectives as stated above.

5.2 Experiment design

Our experiment takes place in three steps. **First**, we use the LFR benchmark network generator developed by Lancichinetti, Fortunato, & Radicchi (2008) to generate six different networks with different mixing-parameter values to simulate networks which are close to real-world networks as the dataset. This is done to randomize the network that we generate with all possible community structure (represented by different mixing parameter values) is included and make sure that the results are generalizable. Six networks are generated for each mixing parameter values ranging from [0, 1]. **Second**, we then apply our pre-processing step and create a new *enriched network* from each of the original ones. This results in a transformation of our input network, and we obtain six enriched networks. **Third**, we apply a selection of community-detection algorithms (Clauset-Newman-Moore, Lovain, and Infomap) onto both the sets of the actual network and the enriched networks. We term them "Actual" and "Similarity-based" in front of the name of the algorithm respectively. And we compare the performance of the algorithms on both sets of networks to see the effectiveness of the preprocessing step on the

[6] similarity (see equation 4.1)

quality of the community structure detected (measured through NMI and Modularity). We also compare which candidate algorithm does the best in detecting higher quality communities. We do this by plotting their performance on both modularity (Q) and NMI scale together and benchmark them by comparing their Q and NMI values.

The experiment was conducted on a MacBook Pro with 4GB RAM with 2.7 GHz Intel Core i7 processor. A wide array of Python-based tools were used to experiment. We make use of Jupyter Notebook as our IDE; NetworkX and Python-igraph package to analyze the social graphs; matplotlib and gelphi to visualize and graph our output; and (Python) pandas and numpy packages to deal with preprocessing of the dataset.

5.3 Dataset

Many experiments to evaluate the community detection algorithms' use simulated LFR benchmark networks as their dataset (Y. Chen et al., 2016; Emmons et al., 2016; Orman et al., 2011; Cao et al., 2015; Hafez, Hassanien, & Fahmy, 2014). Simulated network is used because it is very difficult to evaluate the communities detected in a real-world network due to an absence of community ground-truths and also because community structure is only as good as the algorithm that detects it (Cao et al., 2015). Thus, for this reasons, we make use of LFR Benchmark networks, which simulate networks that are very close to real-world social networks' data (Bródka et al., 2010) and is the new standard network generator for evaluating the performance of different community detection algorithms and benchmark them based on their performance (Lancichinetti & Fortunato, 2009a) .

5.3.1 LFR Benchmark

LFR benchmark (Lancichinetti et al., 2008) was introduced to provide more realistic benchmark networks for unweighted and undirected graphs that capture two critical characteristics of real-world social networks; *heterogeneous nodes degrees* and *community size* (Clauset, Newman, & Moore, 2004; Newman, 2005). Both of these characteristics conform to the type of distribution usually seen in real networks. It is an extension of Girvan-Newman benchmark which is limited by the fact that it can only generate networks with constant size (Bródka et al., 2010; Girvan & Newman, 2002). Thus, LFR benchmark has become a de-facto method to generate real-world like networks with community-structures (Lancichinetti & Fortunato, 2009a) since it enables the researchers to control the network generated by providing control through different parameters which can be tuned as per researchers' likings and requirement.

5.2.2 LFR Benchmark parameters

The community size and the degree distribution of the real-world social network were found to follow the "Power-law-distribution" which states that a relative change in a variable changes another variable by an exponential value. Through studies, they found these exponent values in the social network to be between [2, 3] for degree distribution and [1, 2] for community size distribution (Lancichinetti & Fortunato, 2009a). This phenomenon is implemented in the LFR benchmark graph generator through two parameters, $\tau 1$ is for *"power law exponent of degree distribution for the created graph"* and $\tau 2$ is for *"power law exponent for the community size distribution."* The most important parameter which gives us inherent diverse kind of networks is μ, known as the *mixing parameter* and it represents the fraction of intra-community edges incident to each node. Its value ranges from 0 to 1. Choosing 0 would result in graphs that have high community structure, and 1 would result in graphs that have no community structure. The mixing parameter generates this connection based on (1- μ) for intra-community edges and (μ) for inter-community edges. Thus, the values between 0 to 0.5 yields proper community structures and values 0.5 to 1, less community structure are generated. Hence, we will use the benefit of using different μ values to create a set of networks which builds all kinds of node distribution possible for generalization purpose. Not only the parameters already introduced ($\tau 1$, $\tau 2$, μ), the LFR also provides some other parameters like average_degree, max_degree, minimum_community, maximum_community, number of nodes in the network '*n'* to help users have control on the generated network according to the requirements of the researcher which is a very good control to have to generate exactly the kind of network which is required.

For our experiment, to evaluate the community structure detected using the different algorithm, we generate 6 LFR benchmark networks of size n= 1000; corresponding to different μ values: [0, 0.2, 0.4, 0.6, 0.8, 1] with $\tau 1$ and $\tau 2$, as 2 and 1.5. The average degree is set to 15 and maximum degree as 50. Also, we set the community size between 20 and 60. These values were set to be as close as possible to the values that were used by Lancichinetti & Fortunato (2009b)

5.4 Candidate algorithms

Due to the existing diverse number of community detection algorithms, it is challenging to categorize these algorithms into different groups (Orman et al., 2011). Nevertheless considering the size of the network that we would be dealing with in IM problem, we select a minimum set of algorithms which are represented in the context of our goal, to eventually develop a community-based IM framework. Alfalahi et al. (2013) proposed to use CNM as the community detection algorithm since CNM is modularity-based and very fast. However, in our framework,

we are considering another alternative to CNM called the Louvain, which is also fast and is modularity based and has been found to perform better than CNM (Orman et al., 2011). We also consider a random control algorithm in our experiment called the Infomap which is not modularity-based but found to be high performing in detecting communities (Orman et al., 2011).

The community is informally defined as the group of nodes which are densely interconnected compared to other nodes (Fortunato, 2010; Leskovec et al., 2009). The candidate community detection algorithms we compare are i) **Louvain** (Blondel et al., 2008) and; ii) **Clauset Newman Moore or fast greedy** (Clauset, Newman, & Moore, 2004) which are based on maximizing the modularity measure; iii) **Infomap** (Rosvall & Bergstrom, 2008) which is based on Huffman Coding as discussed in section 2.2. We check our enriched network (pre-processed based on similarity) on these algorithms and also benchmark them for future use in the proposed framework.

5.5 Evaluation metrics

Two evaluation metrics are used for evaluating the community structure that we detected through implementing three community detection algorithms. They are Modularity (Q) and Normalized Mutual Information (NMI).

5.5.1 Modularity (Q)

Modularity is one of the most popular evaluation metrics to evaluate the community structure. It is based on finding the difference of edges within the communities to the expected number of edges when edges are randomly generated (Alfalahi et al., 2013). Better communities are detected when the difference is large. According to Clauset et al, (2004), the value of Q above 0.3 is considered as a significant community structure. The following formula represents it:

$$Q = \sum_i (e_{ij} - a_i^2) \qquad \textit{Equation (5.1)}$$

Where e_{ij} is the fraction of edges which connects nodes in group i with nodes in group j and $a_i = \sum_j e_{ij}$. It is to be noted that high modularity does not always associate with best partitions of the community. The value ranges between [0, 1] and 0 means there is less community structure and 1 means community structure are very well partitioned (structured). High modularity partitions are not optimal due to limitations with the resolution (Good,

Montjoye, & Clauset, 2010) and hence, we use another metric called the Normalized Mutual Information (NMI) to consolidate our evaluation.

5.5.2 Normalized Mutual Information (NMI)

NMI is also a metric to evaluate the quality of the community that was proposed by Danon et al. (2005) and which looks at how similar are the communities detected by the algorithms to the ground-truth communities. The values of the metric are usually between 0 and 1. If the value is 0, then the similarity between the actual community and the detected community does not match at all. On the other hand, if the value is 1, it means that the actual communities found in the network is exactly the same as the one detected by algorithms.

$$NMI(A,B) = \frac{-2 \sum_{i=1}^{C_A} \sum_{j=1}^{C_B} N_{ij} \log(\frac{N_{ij} N}{N_{.i} N_{.j}})}{\sum_{i=1}^{C_A} N_{.i} \log\left(\frac{N_{.i}}{N}\right) + \sum_{j=1}^{C_B} N_{.j} \log\left(\frac{N_{.j}}{N}\right)} \qquad \textit{Equation (5.2)}$$

Where N is a confusion matrix where the rows i shows the actual communities and the column j shows the detected communities. The member of N, which is N_{ij} is the number of nodes in the actual community i which appears in the detected communities j. The number of actual communities is shown by C_A, and the number of detected communities is shown by C_B. The sum over in row i of matrix N_{ij} is represented by $N_{.i}$ (note the dot) and likewise for column j of matrix N_{ij} is represented by $N_{.j}$ (note the dot)

5.6 Results and discussion

We will first look at the effectiveness of similarity based preprocessing on the quality of community detected through the experiment and then proceed to see which candidate algorithm performs the best.

5.6.1 Analysis of the effectiveness of similarity based pre-processing step on quality of detected communities

No significant increase on Modularity (Q) nor NMI was seen for any of the set of the candidate algorithms selected for community detection as hypothesized and as shown by the following figures.

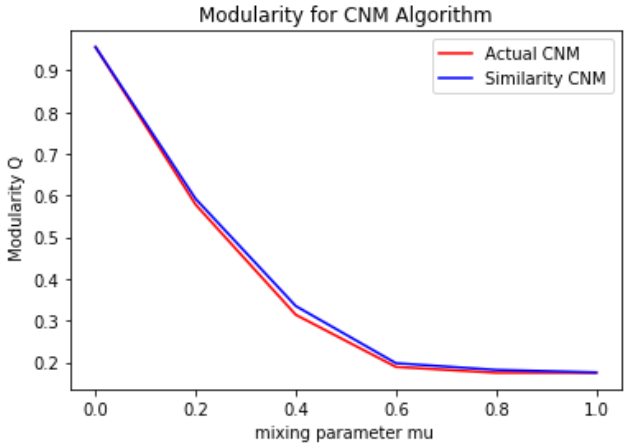

Figure 5.1: Modularity plot for CNM

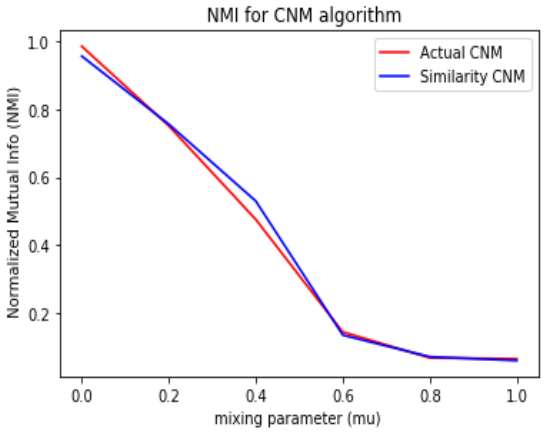

Figure 5.2: NMI plot for CNM

Figure 5.1 and Figure 5.2 shows that the modularity and NMI values for CNM algorithm on actual LFR benchmark network termed as "Actual CNM" and CNM algorithm on enriched-network termed as "Similarity CNM" against the different network we created with different mixing parameter values from 0 to 1. This was used to check if our pre-processing step helps in improving the community structure (increase in Q and NMI). Figure 5.3 and 5.4 show the modularity and NMI values for Louvain algorithm and Figure 5.5 and Figure 5.6, for Infomap algorithm respectively.

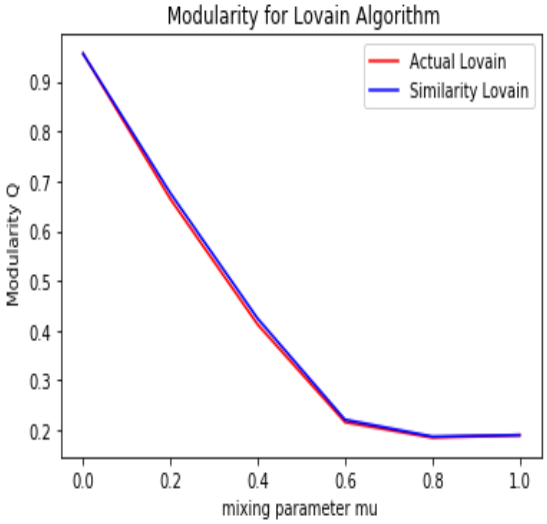

Figure 5.3: Modularity plot for the Louvain

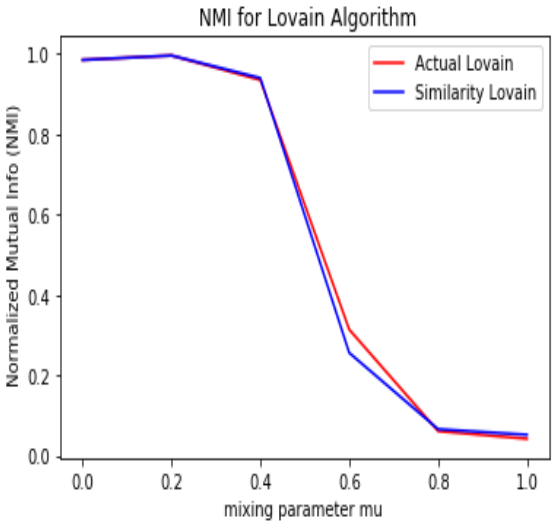

Figure 5.4: NMI plot for the Louvain

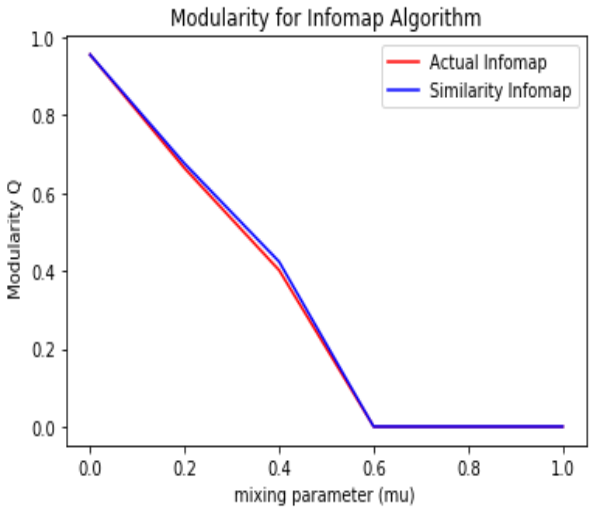

Figure 5.5: Modularity plot for Infomap

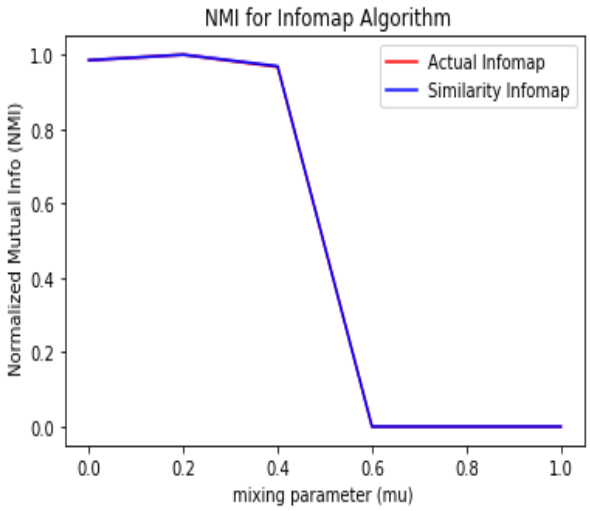

Figure 5.6: NMI plot for infomap

According to the results (Figure 5.1 through Figure 5.6) obtained, there is only a very slight or no difference on the Modularity and NMI values for all three algorithms, which suggests that the similarity based preprocessing does not increase NMI nor Q values significantly. Therefore the community structure detected using our pre-processing step is not any better than directly applying any Algorithm on the real networks without any preprocessing step. At this stage, we can only discuss the reason for such a result. There are only two factors which might be the probable culprit: *i) Network size used and; ii) The LFR benchmark parameter choice.* Running off-record experiments on the network size of 2500 and 5000 also produced very similar results. Beyond the network of size 10000; the computation time of running the preprocessing step was very high for the machine that we were using. However, we believe that the number of the nodes in the network (beyond a reasonable doubt) cannot be the reason for such results. There have to be some inconsistent variations if the size of the network played a significant role in the outcome of the experiment. As far as the LFR benchmark parameter choice are concerned, applying the community detection algorithm on those benchmark network yielded similar results to previously well-known studies by Bródka et al. (2010) and Orman et al. (2011) which also validates the fact that there are no inconsistencies in the network generated for the experiments. Thus we are confident of our result that the preprocessing step does not improve the community structure detected by different algorithms - as measured on Q and NMI measure.

5.6.2 Comparison of candidate algorithm

As per our objective, we plotted the modularity and NMI values for each algorithms to benchmark their respective performance on the simulated network. Figure 5.7 and Figure 5.8 show the modularity and NMI value for networks of different mixing parameters.

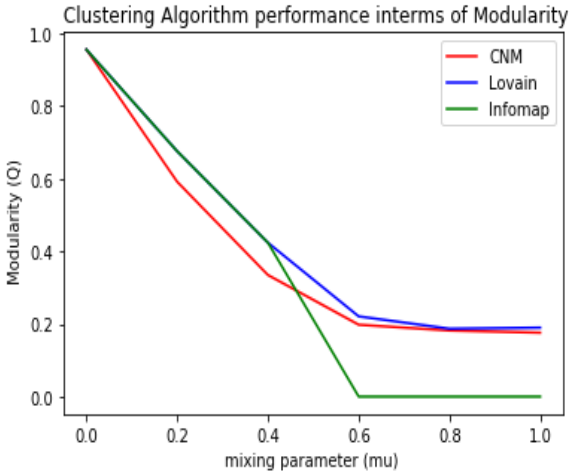

Figure 5.7: modularity plot for candidate algorithms

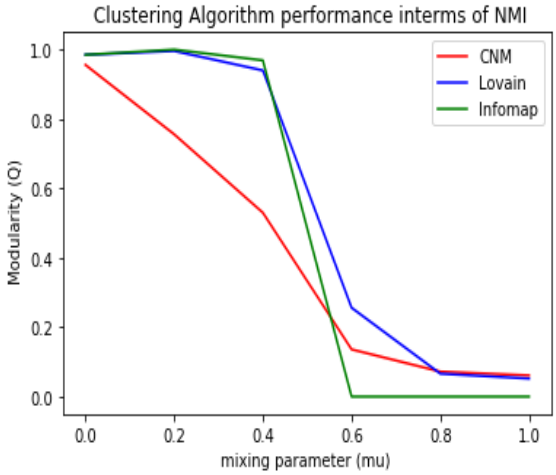

Figure 5.8: NMI plot for candidate algorithms

According to the results generated through Figure 5.7 and Figure 5.8; Modularity based algorithm CNM and Lovain perform the best across the candidate networks of different mixing parameter in both Modularity score and NMI score. As for Infomap, its ability to detect community becomes zero after exceeding the mixing parameter value of 0.6, which means that it cannot detect communities in networks where the community partition is noisy (represented by mu value of 0.6). On comparing Lovain with the CNM, based on both the NMI and Modularity score, the best algorithm which detects the best community structure is the Louvain method; it performs very well across all the different mixing parameter values. The results of the experiment were compared with other similar experiments conducted to compare the different community detection algorithms by Bródka et al. (2010) and Orman et al. (2011) and it was found that the performance of candidate algorithms that we selected based on NMI and Modularity score were conforming with these experiments, which found Louvain to be performing the best. Followed by CNM (also called fast greedy) since it detects communities even for very high mixing-parameter values. Finally Infomap, which performs almost as good as Louvain over the collection of different networks for lower **μ** values, but then almost plummets to zero for higher **μ** values above 0.6. Thus, we suggest using Lovain as our algorithm to detect communities in our community-based IM framework under development.

6. Ethics and validity threat evaluation

Any scientific (empirical) research involving human in the experimentation should be ethically moral (Wohlin et al., 2012) and should follow a guideline for the conduct of the empirical study as defined in (Singer & Vinson, 2002). As suggested in these guidelines, the basic principles in experiment design consist of the code of conducts like *informed consent of the participants*, the *scientific value of the work*, *data confidentiality*, and *beneficence of the study must outweigh risks or harms*. In our project, since we use an LFR Benchmark graph, which simulates the real-world social networks, we are in no way violating principles which are concerned with the research participant in experimentation process.

Never the less, there are other research ethics (code of conduct) that need to be upheld like *Honesty, Transparency/Openness, Objectivity, Integrity, and Respect for intellectual properties* as a researcher. **Honesty** in this context is to report result, data, method, and procedures honestly. In our project, we believe that we have explained and presented all the above components of our project clearly without any intention to deceive to uphold honesty. Next, **Transparency/Openness** stands for openness to the criticism by making data, method, tools, code open for others to see; In our project, we have provided every tools, techniques, data, and code, we have used in the form of Appendices. **Objectivity** stands for reducing bias in the study while applying the different components of research. We tried to reduce this by multiple validation steps we implemented to ensure that biases do not become the issue in our study. We have implemented validation principles like randomization and generalization, through the application of these validation principles on datasets, to eradicate biases**. Respect for intellectual properties** stands for giving the credits where it is due and in our project since we have used a lot of previous studies relating to the framework proposed. We uphold this principle by referencing the authors and crediting them for the work they have done. Moreover, since the supervisor of the project is the co-author of the previous-work used in this project, a clear distinction of the contribution of this project against the previous studies have been clearly defined and mentioned in section 7 in the conclusion part of the report.

Once ethics has been considered, next, we evaluate how we mitigated the validity threats in our experimentation process. Threats to validity in experiments according to Cook and Campbell (1979) are categorized as *conclusion, internal, external and construct*. To mitigate these validity threats, we generated six different LFR benchmark graphs using LFR parameter known as the "mixing parameter" to give back a set of graph for each values, [0, 0.2, 0.4, 0.6, 0.8, 1],

which generalizes and mimics all possible kinds of social network (graph) with different node degree distribution and community size possible in real-world scenario. We also keep all the other LFR parameters same for all the graphs generated to ensures that mixing-parameter is the only treatment which determines the graph generation process and thus improves the reliability of treatments implemented and the generated effect. We also used two evaluation metric (Q and NMI) to feel more confident about the results and maintain the reliability of the measures used, both of these measures the quality of communities detected. The LFR parameters, like *τ1, τ2, average_degree, maximum_degree, min_community, max_community, no. of nodes* were all set to values which are as close as possible to values that were seen in real-world social networks and as was used by Lancichinetti & Fortunato (2009a). The performance of the candidate algorithms conforms to other studies done by Bródka et al. (2010) and Orman et al. (2011); which also validates the fact that the algorithms were implemented correctly and instrumentation is not the culprit in obtaining the results that were obtained. We believe that our experiment follows the principle of randomization and generalization by generating a benchmark network using different mixing-parameter which produces six different networks with different node distributions. Thus we back our results with no reasonable doubt.

7. Discussion and conclusion

In this project, we first introduced the problem of Influence Maximization (IM) in social networks as the problem to address in this study. We further motivated the need to have a practical solution for IM problem; which could be used by businesses of all kinds to market products and services. We looked through different background concepts that are necessary to understand and solve IM problem using social network analysis modeled as graphs. We introduced different graph concepts like vertices, edges, edge weights, Linear Threshold Model and Independent Cascade Model as landmark diffusion models available in the literature, as well as social influence and influence spread concept. We also looked at community detection-algorithms, fuzzy logic theory to understand its practical use within the context of our project. We then looked at a collection of different works of literature which were based on existing approaches to solve IM problem as an optimization problem through different approaches. We also looked at some closely related work, which was also based on community-detection approach like our proposed approach. We motivated how our approach is different from the already existing ones and how it can be improved. We then explained our community-based IM framework in details as involving 5 steps: i) similarity-based preprocessing to enrich the input network ii) detection of communities iii) finding important nodes iv) calculate the spread for each important nodes and rank them based on spread value; and finally v) selecting the top-N nodes as seed set based on the budget. This framework is inspired by the previous collection of work done by Alfalahi et al. (2013a; 2013b), and we have reiterated the framework with some changes wherever we thought was necessary. We also discussed how we intend to use external user-data to assign influence-weights based on common actions and how it differentiates from existing methods which use some random models (probability values as edges) to assign the weights. This was also inspired by Alfalahi et al. (2013a; 2013b). We have also explicitly mentioned that due to the limited amount of time, the whole framework was not implemented and only step 1 and step 2 was implemented and validated. We have experimented the similarity based preprocessing and community detection part in the context of IM problem.

In our first experiment to test the effectiveness of similarity-based preprocessing step on the quality of community detected, we found that for different LFR benchmark graphs we generated using different mixing parameter, the similarity based preprocessing does not make much difference on the modularity and NMI score. This indicates that there is no improvement to the quality of the detected communities as measured by NMI and Q. This does not conform to the results obtained by Alfalahi et al. (2013a) in their implementation of similarity-based

preprocessing step as a network enrichment step. However, the size of the network used by them were 10,000 nodes. We could not run the experiments for 10,000 nodes because of the limitation of the system that we were using. However, our experiment of up to 5000 nodes obtained the same negligible result. The only possible justification that we can hypothesize at this point is that the choice of LFR Benchmark parameters are affecting the results obtained (if we consider Alfalahi et al. (2013b) to be correct). However, then again, the values that we used were used in a lot of previous studies related to benchmarking community detection algorithms by Lancichinetti & Fortunato (2009b), Bródka et al. (2010) and Orman et al. (2011). Evaluation of the experiments from Alfalahi et al. (2013a) is not possible due to the absence of their implementation details. The consequence of the results mean that the proposed IM framework does not need to have the preprocessing step, since it does not help algorithms' detect better community structure. Moreover the step has an overhead of $O(n^2)$ which affects the running time when it is run.

In other experiment which compared/benchmarked the candidate algorithms, our results show that Louvain performs the best of the three on both NMI and Q. Followed by CNM and then Infomap. Some may interpret Infomap to be performing better than CNM since it performs as good as Louvain for graphs with lower mixing parameter (distinct communities). However, in the context of our project, we interpret it to be performing rather extreme with the NMI and Q falling to 0 for mixing-parameter value 0.6 and higher. On the contrary, CNM shows a gradual decrease over the different mixing parameter values. When compared to other similar studies that benchmark the different community detection algorithms, our result conforms to the results obtained by Bródka et al. (2010) and Orman et al. (2011). The reason for the result obtained is quite straightforward because Louvain is an improvement of CNM introducing two-step hierarchical agglomerative strategy (Blondel et al., 2008). Infomap treats community structure as having two levels to it. It is based on Huffman coding, where one of the levels is to differentiate communities in the network, and other, to differentiate the nodes. It performs better on a well-partitioned network but fails to detect community when the distinctions become vague. As a consequence of this experiment, we propose the use of Louvain as the community detection algorithm in our IM framework.

To sum-up, we have made three contributions. Firstly we have studied different works done by Alfalahi et al. (2013a; 2013b) and reiterated different body of work to propose our version of the community-based IM framework through devising 5 steps. Secondly, we found out that the preprocessing step does not improve the quality of communities measured by Q and NMI and

that the Louvain is the best algorithm to implement in our IM framework. Lastly, by providing the source code for the entire project, we contribute a demonstration of the use of python language based libraries, tools, and techniques to analyze the social network which is modeled as graphs. The use of tools like networkX, i-Graph, Gelphi, numpy, python and Jupyter-Notebook for interactive social network analysis would benefit interested.

Future work would consist of pursuing the implementation of the proposed IM framework through different experimental setups to choose the best algorithmic alternatives like which centrality algorithms to use, which diffusion model would be better to calculate the network reach, all in the context of solving Influence Maximization problem. This approach aims to benefit businesses in optimizing the cost of marketing and Return Of Investment by identifying influential users (seed set) who have the maximum influence in social networks, which is becoming a de facto place for digital marketing.

References

Ahmed, Z., Ashraf, M. W., Baig, F., Imran, M., Tayyaba, S., Baryar, U., & Pervaiz, Z. (2013). Design and Simulation of Night Switch by Using Fuzzy Logic. *Open Journal of Artificial Intelligence*, *1*(2), 33.

Akiba, T., Iwata, Y., & Yoshida, Y. (2014). Dynamic and historical shortest-path distance queries on large evolving networks by pruned landmark labeling. In *Proceedings of the 23rd international conference on World wide web - WWW '14*. https://doi.org/10.1145/2566486.2568007

AlFalahi, K., Atif, Y., & Abraham, A. (2013a). Models of Influence in Online Social Networks. *International Journal of Intelligent Systems*, *29*(2), 161–183.

Alfalahi, K., Atif, Y., & Harous, S. (2013b). Community detection in social networks through similarity virtual networks. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining - ASONAM '13*. https://doi.org/10.1145/2492517.2500299

Anagnostopoulos, A., Kumar, R., & Mahdian, M. (2008). Influence and correlation in social networks. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08*. https://doi.org/10.1145/1401890.1401897

Arora, A., Galhotra, S., & Ranu, S. (2017a). Debunking the Myths of Influence Maximization. In *Proceedings of the 2017 ACM International Conference on Management of Data - SIGMOD '17*. https://doi.org/10.1145/3035918.3035924

Arora, A., Galhotra, S., & Ranu, S. (2017b). Debunking the Myths of Influence Maximization. In *Proceedings of the 2017 ACM International Conference on Management of Data - SIGMOD '17*. https://doi.org/10.1145/3035918.3035924

Ashley, C., & Tuten, T. (2014). Creative Strategies in Social Media Marketing: An Exploratory Study of Branded Social Content and Consumer Engagement. *Psychology & Marketing*, *32*(1), 15–27.

Bagheri, E., Dastghaibyfard, G., & Hamzeh, A. (2016). An efficient and fast influence maximization algorithm based on community detection. In *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. https://doi.org/10.1109/fskd.2016.7603422

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(10), P10008.

Boon-Long, S., & Wongsurawat, W. (2015). Social media marketing evaluation using social network comments as an indicator for identifying consumer purchasing decision effectiveness. *Journal of Direct, Data and Digital Marketing Practice*, *17*(2), 130–149.

Bozorgi, A., Haghighi, H., Zahedi, M. S., & Rezvani, M. (2016). INCIM: A community-based algorithm for influence maximization problem under the linear threshold model. *Information Processing & Management*, *52*(6), 1188–1199.

Bozorgi, A., Samet, S., Kwisthout, J., & Wareham, T. (2017). Community-based influence maximization in social networks under a competitive linear threshold model. *Knowledge-Based Systems*, *134*, 149–158.

Brandes, U. (2001). A faster algorithm for betweenness centrality*. *The Journal of Mathematical Sociology*, *25*(2), 163–177.

Bródka, P., Musial, K., & Kazienko, P. (2010). A Method for Group Extraction in Complex Social Networks. In *Communications in Computer and Information Science* (pp. 238–247).

Cao, X., Wang, X., Jin, D., Guo, X., & Tang, X. (2015). A Stochastic Model for Detecting Overlapping and Hierarchical Community Structure. *PloS One*, *10*(3), e0119171.

Cheng, S., Shen, H., Huang, J., Chen, W., & Cheng, X. (2014). IMRank. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval - SIGIR '14*. https://doi.org/10.1145/2600428.2609592

Cheng, S., Shen, H., Huang, J., Zhang, G., & Cheng, X. (2013). StaticGreedy. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management - CIKM '13*. https://doi.org/10.1145/2505515.2505541

Chen, W., Wang, C., & Wang, Y. (2010). Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10.* https://doi.org/10.1145/1835804.1835934

Chen, W., Yuan, Y., & Zhang, L. (2010). Scalable Influence Maximization in Social Networks under the Linear Threshold Model. In *2010 IEEE International Conference on Data Mining*. https://doi.org/10.1109/icdm.2010.118

Chen, Y., Zhao, P., Li, P., Zhang, K., & Zhang, J. (2016). Finding Communities by Their Centers. *Scientific Reports*, *6*(1). https://doi.org/10.1038/srep24017

Clauset, A., Newman, M. E. J., & Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, *70*(6). https://doi.org/10.1103/physreve.70.066111

Cook, T. D., & Campbell, D. T. (1979). *Quasi-experimentation: design & analysis issues for field settings*.

Danon, L., Díaz-Guilera, A., Duch, J., & Arenas, A. (2005). Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, *2005*(09), P09008–P09008.

Diestel, R. (2010). *Graph Theory*.

Emmons, S., Kobourov, S., Gallant, M., & Börner, K. (2016). Analysis of Network Clustering Algorithms and Cluster Quality Metrics at Scale. *PloS One*, *11*(7), e0159161.

Ferguson, R. (2008). Word of mouth and viral marketing: taking the temperature of the hottest trends in marketing. *Journal of Consumer Marketing*, *25*(3), 179–182.

Fischetti, M., & Williamson, D. P. (2007). *Integer Programming and Combinatorial Optimization: 12th International IPCO Conference, Ithaca, NY, USA, June 25-27, 2007, Proceedings*. Springer.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, *486*(3-5), 75–174.

Galhotra, S., Arora, A., & Roy, S. (2016). Holistic Influence Maximization. In *Proceedings of the 2016 International Conference on Management of Data - SIGMOD '16*. https://doi.org/10.1145/2882903.2882929

Girvan, M., & Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, *99*(12), 7821–7826.

Good, B. H., de Montjoye, Y.-A., & Clauset, A. (2010). Performance of modularity maximization in practical contexts. *Physical Review E*, *81*(4). https://doi.org/10.1103/physreve.81.046106

Goyal, A., Lu, W., & Lakshmanan, L. V. S. (2011a). CELF. In *Proceedings of the 20th international conference companion on World wide web - WWW '11*. https://doi.org/10.1145/1963192.1963217

Goyal, A., Lu, W., & Lakshmanan, L. V. S. (2011b). SIMPATH: An Efficient Algorithm for Influence Maximization under the Linear Threshold Model. In *2011 IEEE 11th International Conference on Data Mining*. https://doi.org/10.1109/icdm.2011.132

Hafez, A. I., Hassanien, A. E., & Fahmy, A. A. (2014). Testing Community Detection Algorithms: A Closer Look at Datasets. In *Intelligent Systems Reference Library* (pp. 85–99).

Hellmann, M. 2001. Fuzzy logic introduction. Retrieved from http://epsilon.nought.de/tutorials/fuzzy/fuzzy.pdf.

Jung, K., Heo, W., & Chen, W. (2012). IRIE: Scalable and Robust Influence Maximization in Social Networks. In *2012 IEEE 12th International Conference on Data Mining*. https://doi.org/10.1109/icdm.2012.79

Kempe, D., Kleinberg, J., & Tardos, É. (2003a). Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03*. https://doi.org/10.1145/956755.956769

Kempe, D., Kleinberg, J., & Tardos, É. (2003b). Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03*. https://doi.org/10.1145/956755.956769

Khousa, E. A., & Atif, Y. (2017). Social network analysis to influence career development. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-017-0457-9

Kim, C., Lee, S., Park, S., & Lee, S.-G. (2013). Influence Maximization Algorithm Using Markov Clustering. In *Lecture Notes in Computer Science* (pp. 112–126).

Kim, C.-M., Kim, Y.-H., & Han, Y.-H. (2013). A Community-Based Influence Measuring Scheme in Delay-Tolerant Networks. *The Journal of Korea Information and Communications Society*, *38B*(1), 87–96.

Lancichinetti, A., & Fortunato, S. (2009a). Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, *80*(1). https://doi.org/10.1103/physreve.80.016118

Lancichinetti, A., & Fortunato, S. (2009b). Community detection algorithms: A comparative analysis. *Physical Review E*, *80*(5). https://doi.org/10.1103/physreve.80.056117

Lancichinetti, A., Fortunato, S., & Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Physical Review E*, *78*(4). https://doi.org/10.1103/physreve.78.046110

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., & Glance, N. (2007). Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*. https://doi.org/10.1145/1281192.1281239

Leskovec, J., Lang, K. J., Dasgupta, A., & Mahoney, M. W. (2009). Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. *Internet Mathematics*, *6*(1), 29–123.

Nathan, E., Zakrzewska, A., Riedy, J., & Bader, D. (2017). Local Community Detection in Dynamic Graphs Using Personalized Centrality. *Algorithms*, *10*(3), 102.

Newman, M. E. J. (2004). Fast algorithm for detecting community structure in networks. *Physical Review E*, *69*(6). https://doi.org/10.1103/physreve.69.066133

Orman, G. K., Labatut, V., & Cherifi, H. (2011). Qualitative Comparison of Community Detection Algorithms. In *Communications in Computer and Information Science* (pp. 265–279).

Otte, E., & Rousseau, R. (2002). Social network analysis: a powerful strategy, also for the information sciences. *Journal of Information Science and Engineering*, *28*(6), 441–453.

Rahman, S. M., & Ratrout, N. T. (2009). Review of the Fuzzy Logic Based Approach in Traffic Signal Control: Prospects in Saudi Arabia. *Journal of Transportation Systems Engineering and Information Technology*, *9*(5), 58–70.

Richardson, M., & Domingos, P. (2002). Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02*. https://doi.org/10.1145/775047.775057

Rojas, R. (2013). *Neural Networks: A Systematic Introduction*. Springer Science & Business Media.

Rosvall, M., & Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of the United States of America*, *105*(4), 1118–1123.

Singer, J., & Vinson, N. G. (2002). Ethical issues in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, *28*(12), 1171–1180.

Tang, Y., Shi, Y., & Xiao, X. (2015). Influence Maximization in Near-Linear Time. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data - SIGMOD '15*. https://doi.org/10.1145/2723372.2723734

Tang, Y., Xiao, X., & Shi, Y. (2014). Influence maximization. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data - SIGMOD '14*. https://doi.org/10.1145/2588555.2593670

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in Software Engineering*.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353.

Zadeh, L. A. (1984). Fuzzy probabilities. *Information Processing & Management*, *20*(3), 363–372.

Zhukov, L. (2017, April 15*). Diffusion of Innovation and Influence Maximization*. Retrieved from http://www.leonidzhukov.net/hse/2015/networks/lectures/lecture15.pdf

Appendix 1: Python based tools

This additional section describes the different tools available for Python language that has been used to realize the goal of the project. Please note that the links to visit for further explorations of the tools are given in the footnotes.

A.1 Jupyter notebook

Jupyter notebook which is web-based computing environment developed for interactive computing in the programming language like Julia, Python and R. We use Jupyter notebook in our project because it allows for interactive analysis and visualization of data of any format. Not just the interactivity, it also provides the different style of parallelism [7] like task and data parallelism, single program multiple data (SPMD), and multiple programs, multiple data (MIMD), which is very handy and straightforward to implement. We use this as our Integrated Development Environment (IDE) for our project since we are reading and writing different graph network in and out of our experiment. The tool also supports all the Python libraries which we would be using for our project. For further information, please visit[8].

A.2 NetworkX

Networkx is a python library for studying graphs and network which is distributed under the BSD-new license release free of cost. It consists of different API's and functionalities which are very powerful to analyze the graph and network dataset. Our project we use it to generate the LFR benchmark networks to test the community detection algorithms. We also use this library to implement our Similarity-based pre-processing step to enrich the LFR benchmark network using its different functionalities available to transverse, search through the nodes and edges of the network graph. We also use its functionalities to write the graphs generated in an external file which could be read quickly and be used by different programs. Please refer to learning about different functionalities it provides.

A.3 Python-Igraph

Like NetworkX Python-igraph is also a python library for studying graphs and networks. Just like networkx it also has the vast array of functionalities that we can use to analyze graph. Dataset. In our project, we make use of the package to make use of selected community detection algorithm like CNM (Fast Greedy), Lovain and Infomap. We also use this library to

[7] https://ipython.org/ipython-doc/stable/parallel/parallel_intro.html

[8] http://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/What%20is%20the%20Jupyter%20Notebook.html

evaluate our community structure through the Modularity (Q) and Normalized Mutual Information (NMI) metric functionalities available in it. Please visit [9] to check out the different functionalities the library provides.

A.4 Matplotlib

Matplotlib is a 2D python plotting library for plotting quality figures. We use the library in our project to plot our results. For further information to use it, please visit[10]

A.5 Gelphi

Gelphi is a potent tool to visualize and explore large graphs and networks. It provides a way to understand the structure of the network or graph visually. We use gelphi to look at the different network that we generated using the LFR benchmark network generator. For further information on how to use this tool, visit[11]

A.6 Pandas

Pandas is a python based library which provides different data structure and data analysis functionalities for python based data analysis. We use this package to read in our community ground truths that we extracted manually from the LFR networks to calculate the NMI score. Visit[12] for learning how to use it for data preprocessing.

A.7 Numpy

It is a python based package for scientific computing. We use it in our project to access some mathematical functionalities and specially in using the numpy arrays which can be sliced and diced the way we want to manipulate the data we want to analyze. Visit[13] for learning more about it.

[9] http://igraph.org/python/
[10] https://matplotlib.org/
[11] https://gephi.org/
[12] https://pandas.pydata.org/
[13] http://www.numpy.org/

Appendix 2: Code Listings

For the ease of set up and perform experiments in an interactive manner, as introduced in the appendix A, we make use of Jupyter Notebook as our IDE, and different Python-based packages (also mentioned in Appendix A) like Pandas and Numpy for data wrangling; networkx and python-igraph for analyzing social graphs using graph analysis functionalities in the form of Implementation of different Community detection algorithms and also validating measures. The code snippets are, therefore, in the ***.ipynb** which can be run individually. The structure of the experiment undertaken is shown by the logical flow diagram of the *.ipynb (code) files below:

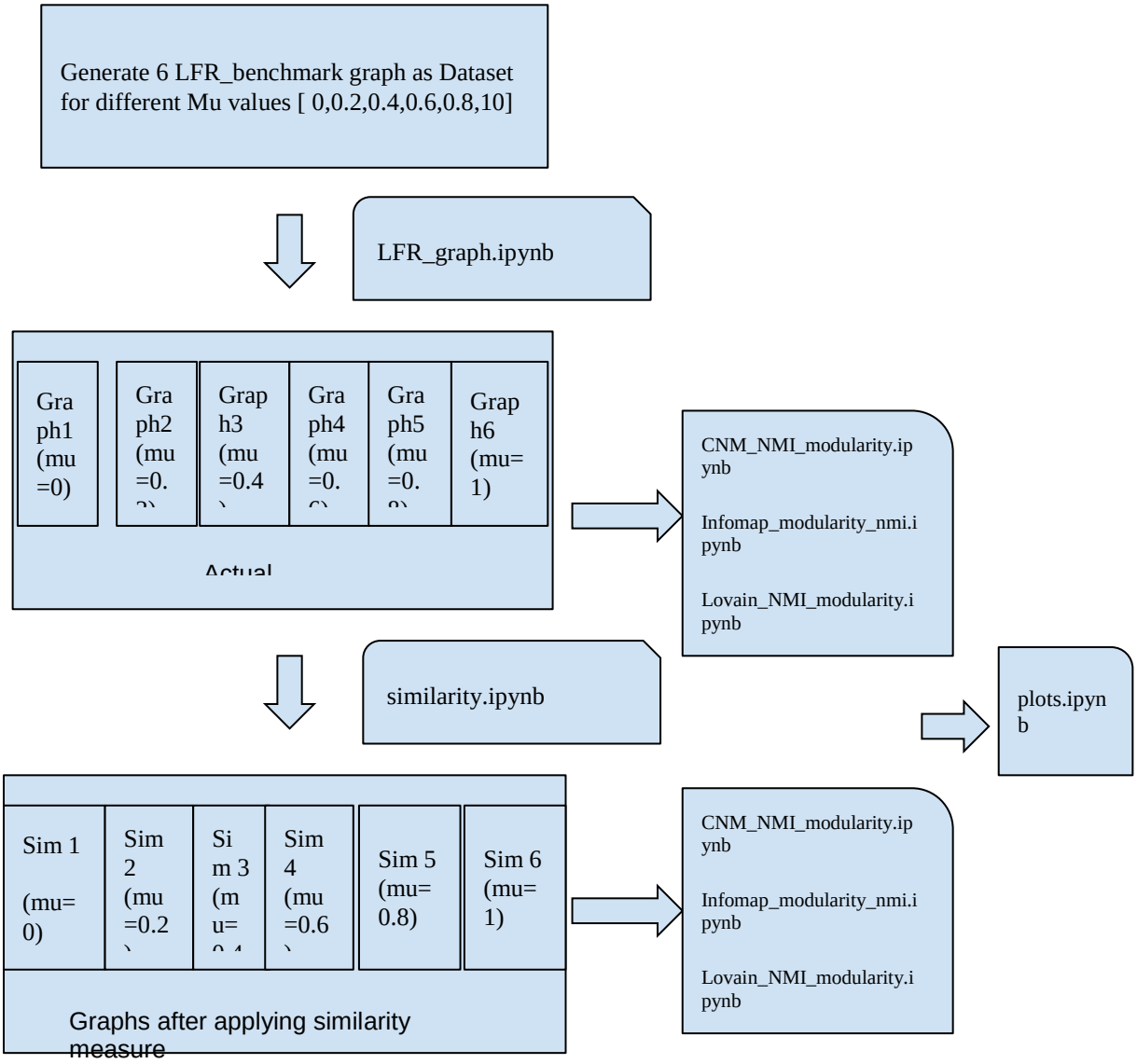

Figure B.1: Logical follow of the code files

Listing 1: LFR_graph.ipynb

```
# coding: utf-8
import networkx as nx
import numpy as np
from networkx.algorithms.community import LFR_benchmark_graph
import matplotlib.pyplot as plt
get_ipython().magic(u'matplotlib inline')

# # Generate Simulated Network using the LFR Benchmark for different Mixing parameter
n = 1000
tau1= 2
tau2= 1.5
mu = 0
G = LFR_benchmark_graph(n, tau1, tau2, mu,average_degree=15,
max_degree=50,min_community=20,
max_community=60,seed=10)
nx.write_edgelist(G,"bench_mark_network_zero.txt",encoding='utf=8',data=False)
communities_zero = {frozenset(G.nodes[v]['community']) for v in G}

# In[ ]:

n = 1000
tau1= 2
tau2= 1.5
mu = 0.2
G = LFR_benchmark_graph(n, tau1, tau2, mu,average_degree=15,
max_degree=50,min_community=20,
max_community=60,seed=10)
nx.write_edgelist(G,"bench_mark_network_two.txt",encoding='utf=8',data=False)
communities_two = {frozenset(G.nodes[v]['community']) for v in G}

# In[ ]:

n = 1000
tau1= 2
tau2= 1.5
mu = 0.4
G = LFR_benchmark_graph(n, tau1, tau2, mu,average_degree=15,
max_degree=50,min_community=20,
max_community=60,seed=10)
nx.write_edgelist(G,"bench_mark_network_four.txt",encoding='utf=8',data=False)
communities_four = {frozenset(G.nodes[v]['community']) for v in G}


# In[ ]:

n = 1000
tau1= 2
tau2= 1.5
mu = 0.6
G = LFR_benchmark_graph(n, tau1, tau2, mu,average_degree=15,
max_degree=50,min_community=20,
```

```
max_community=60,seed=10)
nx.write_edgelist(G,"bench_mark_network_six.txt",encoding='utf=8',data=False)
communities_six = {frozenset(G.nodes[v]['community']) for v in G}


# In[ ]:


n = 1000
tau1= 2
tau2= 1.5
mu = 0.8
G = LFR_benchmark_graph(n, tau1, tau2, mu,average_degree=15,
max_degree=50,min_community=20,
max_community=60,seed=10)
nx.write_edgelist(G,"bench_mark_network_eight.txt",encoding='utf=8',data=False)
communities_eight = {frozenset(G.nodes[v]['community']) for v in G}


n = 1000
tau1= 2
tau2= 1.5
mu = 1
G = LFR_benchmark_graph(n, tau1, tau2, mu,average_degree=15,
max_degree=50,min_community=20,
max_community=60,seed=10)
nx.write_edgelist(G,"bench_mark_network_ten.txt",encoding='utf=8',data=False)
communities_ten = {frozenset(G.nodes[v]['community']) for v in G}
```

Listing 2 : Node_similarity.ipynb

```
import networkx as nx
import numpy as np
#read the benchmark network as Graph
G= nx.read_edgelist("bench_mark_network_ten.txt",nodetype=int)

def similarity(G):
      # function the similarity_graph
    sim = [[0 for u in G.nodes()] for v in G.nodes()]
    for u in G.nodes():
        for v in G.nodes():
            if u==v:
                 sim[u][v]= 0.0
            else:
                 adj = 0.0
            neighbors_u = len(list(G.neighbors(u)))
            neighbors_v= len(list(G.neighbors(v)))
            com_neighbours= len(list(nx.common_neighbors(G,u,v)))
```

```
            if G.has_edge(u,v)==True:
                adj= 1.0
            sim[u][v] = float((adj + float(com_neighbours)))/float(neighbors_u + neighbors_v)
        return np.array(sim)

# create a new network using a node_similarity function
similarity_matrix = similarity(G)
similarity_edge_list = similarity_matrix.argmax(axis=0)
node_list = G.nodes()
edge_tuple= zip(node_list,similarity_edge_list)
edge_list= list(edge_tuple)
G.add_edges_from(edge_list)

nx.write_edgelist(G,"similarity_enriched_ten.txt",encoding='utf=8',data=False)
nx.write_edgelist(G,"similarity_enriched_ten.csv",encoding='utf=8',data=False)
```

Listing 3: CNM_NMI_Modularity.ipynb

```
from igraph import *
import pandas as pd
import numpy as np
import networkx as nx


# # Read the Simulated Graphs generated using LFR Benchmark
# Read the simulated LFR graphs
g0= Graph.Read_Edgelist('bench_mark_network_zero.txt', directed= False)
g2= Graph.Read_Edgelist('bench_mark_network_two.txt', directed= False)
g4= Graph.Read_Edgelist('bench_mark_network_four.txt', directed= False)
g6= Graph.Read_Edgelist('bench_mark_network_six.txt', directed= False)
g8= Graph.Read_Edgelist('bench_mark_network_eight.txt', directed= False)
g10= Graph.Read_Edgelist('bench_mark_network_ten.txt', directed= False)



# # Fast Greedy ( CNM clustering Algorithm)
# create community using the CNM algorithm
p0 = g0.community_fastgreedy().as_clustering(n=30)
p2 = g2.community_fastgreedy().as_clustering(n=30)
p4 = g4.community_fastgreedy().as_clustering(n=30)
p6 = g6.community_fastgreedy().as_clustering(n=30)
p8 = g8.community_fastgreedy().as_clustering(n=30)
p10 =g10.community_fastgreedy().as_clustering(n=30)


print ("cluster for mu 0:",p0.membership)
print ("cluster for mu 2:",p2.membership)
```

```
print ("cluster for mu 4:",p4.membership)
print ("cluster for mu 6:",p6.membership)
print ("cluster for mu 8:",p8.membership)
print ("cluster for mu 10:",p10.membership)

# # Ground Truth communities of the Benchmark network

a = pd.read_excel("GT_0.xlsx",header=None)
gt_0 = a[1].values.tolist()

b = pd.read_excel("GT_2.xlsx",header=None)
gt_2 = b[1].values.tolist()

c = pd.read_excel("GT_4.xlsx",header=None)
gt_4 = c[1].values.tolist()

d = pd.read_excel("GT_6.xlsx",header=None)
gt_6 = d[1].values.tolist()

e = pd.read_excel("GT_8.xlsx",header=None)
gt_8 = e[1].values.tolist()

f = pd.read_excel("GT_10.xlsx",header=None)
gt_10 = f[1].values.tolist()

# # NMI and Modularity for Fast greedy
Q0 = g0.modularity(p0)
Q2 = g2.modularity(p2)
Q4 = g4.modularity(p4)
Q6 = g6.modularity(p6)
Q8 = g8.modularity(p8)
Q10 = g10.modularity(p10)

nmi0 = compare_communities(p0.membership,gt_0,method="nmi")
nmi2 = compare_communities(p2.membership,gt_2,method="nmi")
nmi4 = compare_communities(p4.membership,gt_4,method="nmi")
nmi6 = compare_communities(p6.membership,gt_6,method="nmi")
nmi8 = compare_communities(p8.membership,gt_8,method="nmi")
nmi10 = compare_communities(p10.membership,gt_10,method="nmi")
# Modularity calculation for all graph with different mu [0,0.2,0.4,0.6,0.8,1] for Fast Greedy
(CNM)
print(Q0)
print(Q2)
print(Q4)
```

```
print(Q6)
print(Q8)
print(Q10)
print("\n")
# NMI calculation for all graph with different mu [0,0.2,0.4,0.6,0.8,1] for Fast Greedy (CNM)
against its GT
print(nmi0)
print(nmi2)
print(nmi4)
print(nmi6)
print(nmi8)
print(nmi10)
# # NMI and Modularity for Fast Greedy ( Similarity-CNM clustering Algorithm)
h0= Graph.Read_Edgelist("similarity_enriched_zero.txt", directed= False)
h2= Graph.Read_Edgelist("similarity_enriched_two.txt", directed= False)
h4= Graph.Read_Edgelist("similarity_enriched_four.txt", directed= False)
h6= Graph.Read_Edgelist("similarity_enriched_six.txt", directed= False)
h8= Graph.Read_Edgelist("similarity_enriched_eight.txt", directed= False)
h10= Graph.Read_Edgelist("similarity_enriched_ten.txt", directed= False)
q0 = h0.community_fastgreedy().as_clustering(n=30)
q2 = h2.community_fastgreedy().as_clustering(n=30)
q4 = h4.community_fastgreedy().as_clustering(n=30)
q6 = h6.community_fastgreedy().as_clustering(n=30)
q8 = h8.community_fastgreedy().as_clustering(n=30)
q10 = h10.community_fastgreedy().as_clustering(n=30)
# Modularity calculation for all graph with different mu [0,0.2,0.4,0.6,0.8,1] for Similarity-
Fast Greedy (CNM)
s_Q0 = h0.modularity(q0)
s_Q2 = h2.modularity(q2)
s_Q4 = h4.modularity(q4)
s_Q6 = h6.modularity(q6)
s_Q8 = h8.modularity(q8)
s_Q10 = h10.modularity(q10)
# NMI calculation for all graph with different mu [0,0.2,0.4,0.6,0.8,1] for Similarity Fast
Greedy (CNM) against GT
s_nmi0 = compare_communities(q0.membership,gt_0,method="nmi")
s_nmi2 = compare_communities(q2.membership,gt_2,method="nmi")
s_nmi4 = compare_communities(q4.membership,gt_4,method="nmi")
s_nmi6 = compare_communities(q6.membership,gt_6,method="nmi")
s_nmi8 = compare_communities(q8.membership,gt_8,method="nmi")
s_nmi10 = compare_communities(q10.membership,gt_10,method="nmi")
# print modularity values
print(s_Q0)
print(s_Q2)
print(s_Q4)
print(s_Q6)
print(s_Q8)
print(s_Q10)
print("\n")
#print nmi values
print(s_nmi0)
print(s_nmi2)
print(s_nmi4)
print(s_nmi6)
print(s_nmi8)
print(s_nmi10)
```

Listing 4 : Infomap_modularity_nmi.ipynb

```
# coding: utf-8

from igraph import *
import pandas as pd
import numpy as np
import networkx as nx
# # Read the Simulated Graphs generated using LFR Benchmark
# Read the simulated LFR graphs
g0= Graph.Read_Edgelist('bench_mark_network_zero.txt', directed= False)
g2= Graph.Read_Edgelist('bench_mark_network_two.txt', directed= False)
g4= Graph.Read_Edgelist('bench_mark_network_four.txt', directed= False)
g6= Graph.Read_Edgelist('bench_mark_network_six.txt', directed= False)
g8= Graph.Read_Edgelist('bench_mark_network_eight.txt', directed= False)
g10= Graph.Read_Edgelist('bench_mark_network_ten.txt', directed= False)

# # Infomap (clustering Algorithm)
# create community using the infomap algorithm
p0 = g0.community_infomap()#.as_clustering(n=30)
p2 = g2.community_infomap()#.as_clustering(n=30)
p4 = g4.community_infomap()#.as_clustering(n=30)
p6 = g6.community_infomap()#.as_clustering(n=30)
p8 = g8.community_infomap()#.as_clustering(n=30)
p10 =g10.community_infomap()#.as_clustering(n=30)


print ("cluster for mu 0:",p0.membership)
print ("cluster for mu 2:",p2.membership)
print ("cluster for mu 4:",p4.membership)
print ("cluster for mu 6:",p6.membership)
print ("cluster for mu 8:",p8.membership)
print ("cluster for mu 10:",p10.membership)

# # Ground Truth communities of the Benchmark network

a = pd.read_excel("GT_0.xlsx",header=None)
gt_0 = a[1].values.tolist()

b = pd.read_excel("GT_2.xlsx",header=None)
gt_2 = b[1].values.tolist()

c = pd.read_excel("GT_4.xlsx",header=None)
gt_4 = c[1].values.tolist()

d = pd.read_excel("GT_6.xlsx",header=None)
gt_6 = d[1].values.tolist()

e = pd.read_excel("GT_8.xlsx",header=None)
gt_8 = e[1].values.tolist()
```

```
f = pd.read_excel("GT_10.xlsx",header=None)
gt_10 = f[1].values.tolist()

# # NMI and Modularity for Infomap without Similarity

Q0 = g0.modularity(p0)
Q2 = g2.modularity(p2)
Q4 = g4.modularity(p4)
Q6 = g6.modularity(p6)
Q8 = g8.modularity(p8)
Q10 = g10.modularity(p10)

nmi0 = compare_communities(p0.membership,gt_0,method="nmi")
nmi2 = compare_communities(p2.membership,gt_2,method="nmi")
nmi4 = compare_communities(p4.membership,gt_4,method="nmi")
nmi6 = compare_communities(p6.membership,gt_6,method="nmi")
nmi8 = compare_communities(p8.membership,gt_8,method="nmi")
nmi10 = compare_communities(p10.membership,gt_10,method="nmi")

# Modularity calculation for all graph with different mu [0,0.2,0.4,0.6,0.8,1] for infomap
print(Q0)
print(Q2)
print(Q4)
print(Q6)
print(Q8)
print(Q10)
print("\n")
# NMI calculation for all graph with different mu [0,0.2,0.4,0.6,0.8,1] for infomap against
its GT
print(nmi0)
print(nmi2)
print(nmi4)
print(nmi6)
print(nmi8)
print(nmi10)

# # NMI and Modularity for infomap( with similarity)

h0= Graph.Read_Edgelist("similarity_enriched_zero.txt", directed= False)
h2= Graph.Read_Edgelist("similarity_enriched_two.txt", directed= False)
h4= Graph.Read_Edgelist("similarity_enriched_four.txt", directed= False)
h6= Graph.Read_Edgelist("similarity_enriched_six.txt", directed= False)
h8= Graph.Read_Edgelist("similarity_enriched_eight.txt", directed= False)
h10= Graph.Read_Edgelist("similarity_enriched_ten.txt", directed= False)

q0 = h0.community_infomap()#.as_clustering(n=30)
q2 = h2.community_infomap()#.as_clustering(n=30)
q4 = h4.community_infomap()#.as_clustering(n=30)
q6 = h6.community_infomap()#.as_clustering(n=30)
q8 = h8.community_infomap()#.as_clustering(n=30)
q10 = h10.community_infomap()#.as_clustering(n=30)
```

```
# Modularity calculation for all graph with different mu [0,0.2,0.4,0.6,0.8,1] for Similarity-
infomap
s_Q0 = h0.modularity(q0)
s_Q2 = h2.modularity(q2)
s_Q4 = h4.modularity(q4)
s_Q6 = h6.modularity(q6)
s_Q8 = h8.modularity(q8)
s_Q10 = h10.modularity(q10)
# NMI calculation for all graph with different mu [0,0.2,0.4,0.6,0.8,1] for Similarity
similarity-infomap against GT
s_nmi0 = compare_communities(q0.membership,gt_0,method="nmi")
s_nmi2 = compare_communities(q2.membership,gt_2,method="nmi")
s_nmi4 = compare_communities(q4.membership,gt_4,method="nmi")
s_nmi6 = compare_communities(q6.membership,gt_6,method="nmi")
s_nmi8 = compare_communities(q8.membership,gt_8,method="nmi")
s_nmi10 = compare_communities(q10.membership,gt_10,method="nmi")
# print modularity values
print(s_Q0)
print(s_Q2)
print(s_Q4)
print(s_Q6)
print(s_Q8)
print(s_Q10)

print("\n")

#print nmi values
print(s_nmi0)
print(s_nmi2)
print(s_nmi4)
print(s_nmi6)
print(s_nmi8)
print(s_nmi10)
```

Listing 5 : lovain_NMI_Modularity.ipynb

```
# coding: utf-8

from igraph import *
import pandas as pd
import numpy as np
import networkx as nx

# # Read the Simulated Graphs generated using LFR Benchmark
# Read the simulated LFR graphs
g0= Graph.Read_Edgelist('bench_mark_network_zero.txt', directed= False)
g2= Graph.Read_Edgelist('bench_mark_network_two.txt', directed= False)
g4= Graph.Read_Edgelist('bench_mark_network_four.txt', directed= False)
g6= Graph.Read_Edgelist('bench_mark_network_six.txt', directed= False)
g8= Graph.Read_Edgelist('bench_mark_network_eight.txt', directed= False)
```

```
g10= Graph.Read_Edgelist('bench_mark_network_ten.txt', directed= False)

# # Lovain (clustering Algorithm)
# create community using the lovain algorithm
p0 = g0.community_multilevel()#.as_clustering(n=30)
p2 = g2.community_multilevel()#.as_clustering(n=30)
p4 = g4.community_multilevel()#.as_clustering(n=30)
p6 = g6.community_multilevel()#.as_clustering(n=30)
p8 = g8.community_multilevel()#.as_clustering(n=30)
p10 =g10.community_multilevel()#.as_clustering(n=30)

print ("cluster for mu 0:",p0.membership)
print ("cluster for mu 2:",p2.membership)
print ("cluster for mu 4:",p4.membership)
print ("cluster for mu 6:",p6.membership)
print ("cluster for mu 8:",p8.membership)
print ("cluster for mu 10:",p10.membership)

# # Ground Truth communities of the Benchmark network

a = pd.read_excel("GT_0.xlsx",header=None)
gt_0 = a[1].values.tolist()

b = pd.read_excel("GT_2.xlsx",header=None)
gt_2 = b[1].values.tolist()

c = pd.read_excel("GT_4.xlsx",header=None)
gt_4 = c[1].values.tolist()

d = pd.read_excel("GT_6.xlsx",header=None)
gt_6 = d[1].values.tolist()

e = pd.read_excel("GT_8.xlsx",header=None)
gt_8 = e[1].values.tolist()

f = pd.read_excel("GT_10.xlsx",header=None)
gt_10 = f[1].values.tolist()

# # NMI and Modularity for Lovain without Similarity

Q0 = g0.modularity(p0)
Q2 = g2.modularity(p2)
Q4 = g4.modularity(p4)
Q6 = g6.modularity(p6)
Q8 = g8.modularity(p8)
Q10 = g10.modularity(p10)

nmi0 = compare_communities(p0.membership,gt_0,method="nmi")
nmi2 = compare_communities(p2.membership,gt_2,method="nmi")
nmi4 = compare_communities(p4.membership,gt_4,method="nmi")
nmi6 = compare_communities(p6.membership,gt_6,method="nmi")
nmi8 = compare_communities(p8.membership,gt_8,method="nmi")
```

```
nmi10 = compare_communities(p10.membership,gt_10,method="nmi")

# Modularity calculation for all graph with different mu [0,0.2,0.4,0.6,0.8,1] for Lovain
(Multilevel)
print(Q0)
print(Q2)
print(Q4)
print(Q6)
print(Q8)
print(Q10)
print("\n")
# NMI calculation for all graph with different mu [0,0.2,0.4,0.6,0.8,1] for Lovain
(Multilevel) against its GT
print(nmi0)
print(nmi2)
print(nmi4)
print(nmi6)
print(nmi8)
print(nmi10)

# # NMI and Modularity for lovain( with similarity)

h0= Graph.Read_Edgelist("similarity_enriched_zero.txt", directed= False)
h2= Graph.Read_Edgelist("similarity_enriched_two.txt", directed= False)
h4= Graph.Read_Edgelist("similarity_enriched_four.txt", directed= False)
h6= Graph.Read_Edgelist("similarity_enriched_six.txt", directed= False)
h8= Graph.Read_Edgelist("similarity_enriched_eight.txt", directed= False)
h10= Graph.Read_Edgelist("similarity_enriched_ten.txt", directed= False)

q0 = h0.community_multilevel()#.as_clustering(n=30)
q2 = h2.community_multilevel()#.as_clustering(n=30)
q4 = h4.community_multilevel()#.as_clustering(n=30)
q6 = h6.community_multilevel()#.as_clustering(n=30)
q8 = h8.community_multilevel()#.as_clustering(n=30)
q10 = h10.community_multilevel()#.as_clustering(n=30)

# Modularity calculation for all graph with different mu [0,0.2,0.4,0.6,0.8,1] for
lovain(Multilevel)
s_Q0 = h0.modularity(q0)
s_Q2 = h2.modularity(q2)
s_Q4 = h4.modularity(q4)
s_Q6 = h6.modularity(q6)
s_Q8 = h8.modularity(q8)
s_Q10 = h10.modularity(q10)
# NMI calculation for all graph with different mu [0,0.2,0.4,0.6,0.8,1] for similarity-
lovain(Multilevel)
s_nmi0 = compare_communities(q0.membership,gt_0,method="nmi")
s_nmi2 = compare_communities(q2.membership,gt_2,method="nmi")
s_nmi4 = compare_communities(q4.membership,gt_4,method="nmi")
s_nmi6 = compare_communities(q6.membership,gt_6,method="nmi")
s_nmi8 = compare_communities(q8.membership,gt_8,method="nmi")
s_nmi10 = compare_communities(q10.membership,gt_10,method="nmi")
```

```
# print modularity values
print(s_Q0)
print(s_Q2)
print(s_Q4)
print(s_Q6)
print(s_Q8)
print(s_Q10)
print("\n")
#print nmi values
print(s_nmi0)
print(s_nmi2)
print(s_nmi4)
print(s_nmi6)
print(s_nmi8)
print(s_nmi10)
```

Listing 6 : plots.ipynb

```
import matplotlib.pyplot as plt

# # CNM Algorithm
# # Modularity Plot

x= [0,0.2,0.4,0.6,0.8,1]
acnm= [0.956,0.578,0.314,0.189,0.175,0.175]
simcnm=[0.956,0.592,0.335,0.198,0.182,0.176]

plt.gca().set_color_cycle(['red', 'blue'])
plt.plot(x, acnm)
plt.plot(x, simcnm)
plt.legend(['Actual CNM', 'Similarity CNM'], loc='upper right')
plt.xlabel("mixing parameter mu ")
plt.ylabel(" Modularity Q")
plt.title("Modularity for CNM Algorithm")

plt.show()

# # NMI plot

nmi_x= [0,0.2,0.4,0.6,0.8,1]
nmi_acnm= [0.985,0.751,0.476,0.145,0.069,0.066]
nmi_simcnm=[0.956,0.756,0.530,0.136,0.072,0.061]

plt.gca().set_color_cycle(['red', 'blue'])
plt.plot(nmi_x, nmi_acnm)
plt.plot(nmi_x, nmi_simcnm)
plt.legend(['Actual CNM', 'Similarity CNM'], loc='upper right')
plt.xlabel("mixing parameter (mu) ")
plt.ylabel(" Normalized Mutual Info (NMI)")
plt.title("  NMI for CNM algorithm")
```

```
plt.show()

# # Lovain Algorithm
# # Modularity
x= [0,0.2,0.4,0.6,0.8,1]
alov= [0.956,0.664,0.412,0.216,0.185,0.190]
slov=[0.956,0.676,0.424,0.221,0.187,0.190]

plt.gca().set_color_cycle(['red', 'blue'])
plt.plot(x, alov)
plt.plot(x, slov)
plt.legend(['Actual Lovain', 'Similarity Lovain'], loc='upper right')
plt.xlabel("mixing parameter mu ")
plt.ylabel(" Modularity Q")
plt.title(" Modularity for Lovain Algorithm")

plt.show()

# # NMI

nmi_x= [0,0.2,0.4,0.6,0.8,1]
nmi_alov= [0.985,0.996,0.935,0.314,0.061,0.042]
nmi_slov=[0.985,0.996,0.940,0.256,0.066,0.052]

plt.gca().set_color_cycle(['red', 'blue'])
plt.plot(nmi_x, nmi_alov)
plt.plot(nmi_x, nmi_slov)
plt.legend(['Actual Lovain', 'Similarity Lovain'], loc='upper right')
plt.xlabel("mixing parameter mu ")
plt.ylabel(" Normalized Mutual Info (NMI)")
plt.title(" NMI for Lovain Algorithm")

plt.show()

# # Infomap
# # modularity

x= [0,0.2,0.4,0.6,0.8,1]
ainfo= [0.956,0.664,0.402,0.00,0.00,0.00]
sinfo=[0.956,0.676,0.424,0.00,0.00,0.00]

plt.gca().set_color_cycle(['red', 'blue'])
plt.plot(x, ainfo)
plt.plot(x, sinfo)
plt.legend(['Actual Infomap', 'Similarity Infomap'], loc='upper right')
plt.xlabel("mixing parameter (mu) ")
plt.ylabel(" Modularity Q")
plt.title("Modularity for Lovain Algorithm")
plt.show()

# # NMI
```

```
nmi_x= [0,0.2,0.4,0.6,0.8,1]
nmi_ainfo= [0.985,1.00,0.967,0.00,0.000,0.00]
nmi_sinfo=[0.985,1.00,0.969,0.00,0.00,0.00]

plt.gca().set_color_cycle(['red', 'blue'])
plt.plot(nmi_x, nmi_ainfo)
plt.plot(nmi_x, nmi_sinfo)
plt.legend(['Actual Infomap', 'Similarity Infomap'], loc='upper right')
plt.xlabel("mixing parameter (mu) ")
plt.ylabel(" Normalized Mutual Info (NMI)")
plt.title(" NMI for Infomap Algorithm")

plt.show()

# # ALL three algorithm comparisions
# # Modularity

x= [0,0.2,0.4,0.6,0.8,1]
cnm= [0.956,0.592,0.335,0.198,0.182,0.176]
lovain=[0.956,0.676,0.424,0.221,0.187,0.190]
infomap=[0.956,0.676,0.424,0.00,0.00,0.00]

plt.gca().set_color_cycle(['red', 'blue','green'])
plt.plot(x, cnm)
plt.plot(x, lovain)
plt.plot(x, infomap)
plt.legend(['CNM', 'Lovain','Infomap'], loc='upper right')
plt.xlabel("mixing parameter (mu) ")
plt.ylabel(" Modularity (Q)")
plt.title(" Clustering Algorithm performance interms of Modularity")
plt.show()
# # NMI
nmi_x= [0,0.2,0.4,0.6,0.8,1]
nmi_cnm= [0.956,0.756,0.530,0.136,0.072,0.061]
nmi_lovain=[0.985,0.996,0.940,0.256,0.066,0.052]
nmi_infomap=[0.985,1.00,0.969,0.00,0.00,0.00]

plt.gca().set_color_cycle(['red', 'blue','green'])
plt.plot(nmi_x, nmi_cnm)
plt.plot(nmi_x, nmi_lovain)
plt.plot(x, nmi_infomap)
plt.legend(['CNM', 'Lovain','Infomap'], loc='upper right')
plt.xlabel("mixing parameter (mu) ")
plt.ylabel(" Modularity (Q)")
plt.title(" Clustering Algorithm performance interms of NMI")
plt.show()
```