# GitHub Users Recommendations Based on Repositories and User Profile


**R. Nagaraj, G. R. Ramya, and S. Yougesh Raj**

## 1 Introduction

By offering code hosting services and team-based software development frameworks, GitHub has gained the attention of many software engineers globally as a prominent software development platform, a social coding, and a web-based Git repository hosting service that enables the developers to engage in open-source project documentation, design, coding, and testing in a socio-cultural context. The important feature of this platform is that it allows following other users or organizations, so that they may get alerts about their public activity. This feature has enhanced user collaboration by encouraging them to explore more repositories and promoting the conversation on GitHub.

Moreover, GitHub is considered as a social media platform designed specifically for developers where they can duplicate the relevant projects and then customize the features of those projects to match their requirements. Also, users can ask for assistance from or have a conversation with people who are on their following lists when they run into technical difficulties while developing a repository, bug patches, code restructuring, and feature enhancements.

Even though the developers have the option to search for the projects or do the above things, it takes a lot of time and might disappoint them. So, to overcome this we analyzed a method to identify and recommend the followers working on similar domains to the user by using different centrality measures of network analysis, link prediction between the users, and recommending followers based on the content-based filtering technique.

The paper has five sections where Sect. 2 describes the related works and emphasizes the key characteristics used by us to implement the different measures of

R. Nagaraj · G. R. Ramya (✉) · S. Yougesh Raj
Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Coimbatore, India
e-mail: gr_ramya@cb.amrita.edu



N. R. Shetty et al. (eds.), *Advances in Communication and Applications*, Lecture Notes in Electrical Engineering 1105, https://doi.org/10.1007/978-981-99-7633-1_10

social network analysis and some techniques to identify the prominent followers. Section 3 has a detailed explanation of methodology analysis and the experimental findings from the implementation used to evaluate the effectiveness of the analysis are documented in Sect. 4. Finally the conclusion is given in Sect. 5.

## 2 Related Works

Initially we need to understand how GitHub works and what features it offers. As a result, we referred to a number of research studies conducted on GitHub with various objectives, where (Joy et al. 2018; Sarwar 2021; Koskela et al. 2018) focused on understanding and analyzing the various parameters of the open-source software GitHub. They were able to identify the key parameters as the number of contributions, programming languages, project size, and project age, and they investigated their significance by developing a conceptual model and experimenting with GitHub data. They were able to confirm that the above parameters have a greater impact on project performance as a result of this.

In paper Radha et al. (2017) proposed the implemented of different centrality measures where the USA road network to attain congestion-free shipment. They analyzed the road network based on the performance indicators and identified ways to reduce traffic issues that could aid in traffic management.

Thangavelu and Jyotishi (2018), Sharma and Mahajan (2017), and Gajanayake et al. (2020) proposed a detailed analysis of GitHub work and its projects. They primarily focused on identifying the characteristics of the firm behind the influence of the performance of the open-source project at different stages. In order to experiment they developed a model with four stages such as issues, forks, pull requests, and releases. Through this, they were able to examine the factors of success of the open-source project at different levels. We were able to gain an understanding of how GitHub works and the role of parameters as a result of these works.

Identification of a key person in the network using the well-known centrality measures such as degree, betweenness, closeness, katz, and Eigenvector centrality was proposed by Landher et al. (2010) and Yadav et al. (2019). They were able to conclude that Katz centrality is better for the network they chose based on the distance between nodes, the number of shortest paths, and ranking by implementing the above measure.

Farooq et al. (2018) implemented the above measures as well as and also the coefficient clustering and page rank to a covert network and were able to visualize the metrics that assisted them in identifying the most influential node. These two works served as a foundation for our research because they provided insight into how to implement those measures in our network.

Liu et al. (2019) proposed a method for predicting links between paper citations based on keywords, times, and author information. A series of experiments were also performed on the real-time dataset. They were able to correlate two papers and create a link in the citation network by assigning a value to a pair that was as accurate as

possible. As a result, the dataset validated the feasibility of their proposal. We got an idea for implementing the link prediction to our dataset from this work.

Furthermore, to gain a better understanding of other methods for determining the social network's most significant node, we referred to the work of Ramya and Bagavathi Sivakumar (2021), who demonstrated that DC-FNN classification outperforms fixed clustering and NLP-based sentiment analysis in identifying the influential node in the Twitter dataset based on categories such as pricing, service, and timeliness. They tested these approaches on the likelihood function using iterative logistic regression analysis and the temporal influential model (TIM), and the results show that they are more accurate than other methods in terms of precision, recall, and f-measure. We gained an understanding of the topic's role in a node's influence level as a result of this research.

## 3 Methodologies of Analysis

### *3.1 Data Description*

We considered the essential GitHub values such as user, followers, following, gists, and numrepos, because we used real-time data from GitHub users. The values were adequate for our analysis, and a more detailed explanation is provided in the following sections.

### *3.2 Overview of the Work Flow Model*

In this diagram, we depicted the flow of methods that assisted us in obtaining the required analysis. There are six divisions in total, with the first two illustrating GitHub access and follower analysis, and the outputs obtained being passed to the other four divisions for detailed analysis. The first division represents the calculation of centrality measures, and the second division represents prediction and follower analysis, which leads to the recommendation of the top 10 followers to the user. Although these two divisions are related, we separated them for clarity. Finally, the fourth division represents the prediction of future links between followers. As needed for our research, the output from each division is considered for analysis in other divisions (Fig. 1).

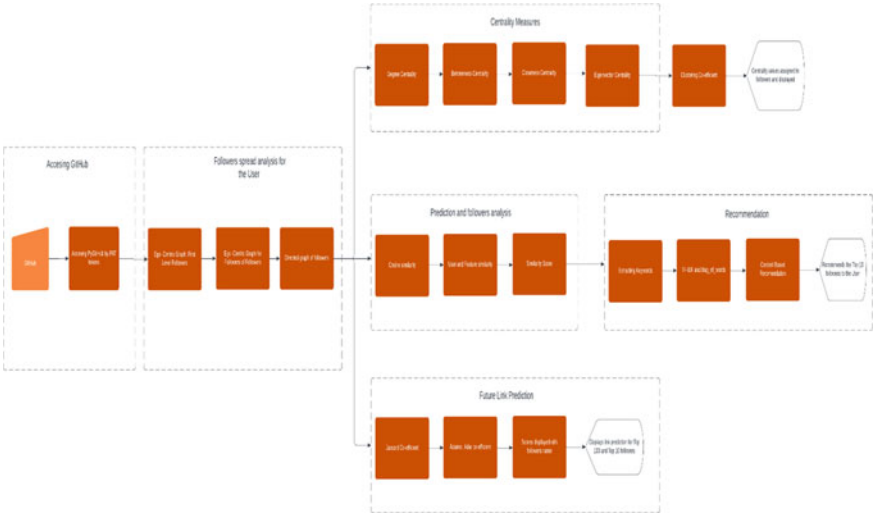


**Fig. 1** Overview of the work flow model

## *3.3 Centrality Measures*

Our methods consist of different stages where the results obtained from each stage are used in other stages to obtain an efficient output. Initially, we explored and understood the spread of followers for a particular user or developer by establishing a connection to the GitHub API (PyGitHub) using Personal Access Token, and then the username of a particular user (here the user is 'Anshul Mehta') was given to get the ego-centric graph of the first level of followers in Fig. 2. For better understanding we explored the followers of the followers and plotted the directed graph.

As our first objective, we implemented centrality measures to identify the prominent follower in the network. Basically, centrality is used in a big network to estimate an individual's properties. The degree of interpersonal interaction and communication within a social network is also measured by centrality. More connections between network members result in greater centrality and always the community development in any online social network is mainly driven by individual nodes having high centrality.

### 3.3.1 Degree Centrality Measure

Degree centrality (DC) is a metric of exposure that evaluates a node's network connectedness using the number of direct contacts to the node.

$$Dv = aiv^n i = 1 \tag{1}$$

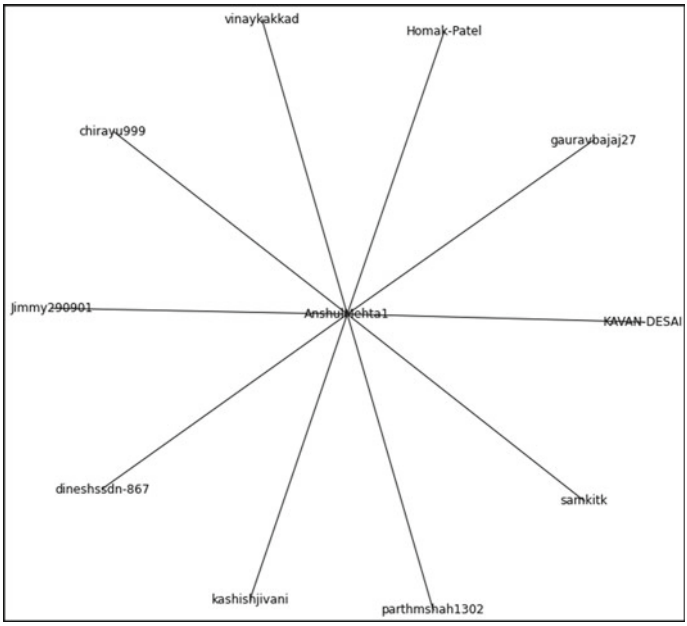


**Fig. 2** Followers graph of Anshul Mehta (developer)

In Eq. (1), the adjacency matrix *aiv* was used to determine the degree centrality of node *v*.

### 3.3.2 Betweenness Centrality Measure

Betweenness centrality (BC) measures the length of time a user or node acts as a bridge between two nodes in a sizable community network while taking into account the quickest path.

$$C_B v = \frac{\sigma_{st}(v)}{\sigma_{st}} \quad s \neq v \neq t \in V \tag{2}$$

In Eq. (2), the terms $\sigma_{st}$ and $\sigma_{st}(v)$ stand for the total number of shortest routes from node *s* to node *t* and the number of paths that passes through *v,* respectively.

### 3.3.3 Closeness Centrality Measure

Closeness centrality (CC) is a measure of how long it takes for information to go from one node to every other node in a network and it is also the length of the shortest path between any two nodes in a complicated network graph.

$$Cx = \frac{1}{y^{d(y,x)}} \tag{3}$$

In Eq. (3), $d(x, y)$ is the measured distance between the $x$ and $y$.

### 3.3.4 Eigenvector Centrality Measure

Eigenvector centrality (EC) is used to assess a node's effect over other nodes in the network. A node's influence will be stronger if it is connected to significant nodes, but its impact on other nodes will be low if it is connected to less important nodes. So, the significance of the node's neighbours determines its relevance

$$\mathrm{EC}_v = v_x = \frac{1}{\tau_{\max}(A)^{j=\frac{n}{1}a_{jx}v_j}} \tag{4}$$

In Eq. (4), $v = (v1, 2, \ldots)$ is the eigenvector for the highest eigenvalue $\tau_{\max}(A)$ of the adjacent matrix $A$.

## *3.4 Similarity Measures*

In addition to that coefficient clustering was calculated to identify how closely the followers were present in a network. The obtained values of each follower were stored as a data frame with basic values of a GitHub such as the number of repositories, gists, number of followers, and number of following to get an idea of data spread and analyze the influential parameters.

As a part of the next research, we implemented the link prediction using the cosine distance for the obtained network. Link prediction is used to find a collection of missing or future linkages between the users by forecasting and estimating the likelihood of each non-existing link in the network.

$$\cos(x, y) = \frac{x.y}{||x|| * ||y||} \tag{5}$$

In Eq. (5), the terms $x.y$ and $||x||$ and $||y||$ are the dot product and the length of the vectors $x$ and $y$, whereas the cross product of the vectors $x$ and $y$ is $||x|| * ||y||$.

Whereas, the cosine of the angle created by the two vectors is used to calculate how similar two users or objects are when they are treated as two vectors in m-dimensional space. During filtering, it is often used to compare two documents, and subsequently more and more regularly to compare two individuals or other aspects instead of simply documents. Since it is often utilized in positive space, it determines how close 0 and 1 are in value.

| | user | followers | following | gists | numrepos | degree centrality | in degree cen | betweenness | closeness | eigenvector | clustering coefficient |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Hrutika Patel | 16 | 27 | 0 | 1 | 0.062893 | 0.000000 | 0.005965 | 0.516234 | 0.128450 | 0.666667 |
| 1 | Varshil Shah | 17 | 13 | 0 | 12 | 0.603774 | 0.597484 | 0.391897 | 0.713004 | 0.399630 | 0.041447 |
| 2 | uditk | 8 | 6 | 0 | 9 | 0.666667 | 0.000000 | 0.492901 | 0.750000 | 0.418723 | 0.035220 |
| 3 | Alpay Yildirim | 7733 | 100012 | 1 | 58 | 0.188679 | 0.000000 | 0.065581 | 0.531773 | 0.204822 | 0.213793 |
| 4 | Vishwa Raval | 22 | 36 | 0 | 8 | 0.182390 | 0.000000 | 0.020995 | 0.530000 | 0.206596 | 0.219212 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 95 | Abdeen Mohamed | 1431 | 32708 | 6 | 7 | 0.031447 | 0.000000 | 0.000000 | 0.490741 | 0.088424 | 1.000000 |
| 96 | Harvish | 71 | 107 | 0 | 19 | 0.006289 | 0.000000 | 0.000000 | 0.417323 | 0.023649 | 0.000000 |
| 97 | Priyank Sangani | 21 | 20 | 0 | 10 | 0.006289 | 0.000000 | 0.000000 | 0.417323 | 0.023649 | 0.000000 |
| 98 | Nikhil Balwani | 34 | 36 | 0 | 35 | 0.018868 | 0.000000 | 0.000000 | 0.463557 | 0.060653 | 1.000000 |
| 99 | Saanvi Tayal | 7 | 14 | 0 | 6 | 0.006289 | 0.000000 | 0.000000 | 0.429730 | 0.024779 | 0.000000 |

**Fig. 3** Data frame created from the calculated values

Then to get a clear idea of similarities among the followers, we employed the follower similarity which identifies the similarities between the followers based on ratings given by them and also found the feature similarities in which the similar features among the followers were evaluated. The data frame consisting of these values was formed in Fig. 3. In addition, the pairwise cosine was found to get a clear idea of similarities between the followers. By using these data, similar followers were found based on the aggregate similarity score (based on the number of gists, repositories, followers, and following) with a developer.

As we were able to identify similar followers now, we focused on finding the potential followers to recommend the user based on the content filtration technique. Here the keywords of the repositories are considered as content. The reason for the content-based recommendation is that it recommends based on the user preferences and this would help us in recommending the followers to the developer.

To implement the recommendation, we initially found the keywords of repositories by stage-wise like name, description, and language. Here comes the natural language processing (NLP) where the strings were converted to lowercase and by stemming, we found the keywords. This was done in two approaches such as bag of words and TF-IDF.

$$\text{TF-IDF} = \text{TF}(t, d) + \text{IDF}(t) \tag{6}$$

In Eq. (6), TF($t$, $d$) is the term frequency of term '$t$' that appears in a document '$d$' and IDF($t$) is the inverse document frequency.

Bag of words is an NLP model used to count the frequency of each word in a corpus for that document, whereas TF-IDF is used to quantify the importance of the relevance of string representation in documents among the collection of documents. As we obtained the words, now again the cosine distance was checked in a bag of words and TF-IDF to find the similarity. These words were then stored and sorted with the corresponding indexes. Now we developed a recommender where the index fetched from the similarity array and sorted the distance in descending order to get the top 10 followers for both bag of words and TF-IDF.

As we found the followers for the user now, we identified the list of followers who can be recommended to the user. In addition, this was done to all the first-level followers of the developer to check whether our method worked properly or not. From this, we obtained the end result of recommending the followers to the user working on similar domains.

Finally, the future prediction between the user's followers was calculated using the Jaccard coefficient and Adamic Adar coefficient, and the results were visualized to get an idea of links. We discovered the network's common triads to gain a better understanding of the links and influence of followers.

## 4 Experimental Results

This section carries out a thorough experimental and quantitative study of the analysis undertaken and the results were shown with an explanation. Initially, the graph was obtained and then the centrality measures were implemented on the follower's network in Figs. 4 and 5 depicts the centrality of common neighbours.

Further, the link prediction between the followers was identified using cosine distance and values for the corresponding followers were stored in the data frame as in Fig. 6.

The cosine similarity was calculated between the followers and the features of the data frame which helped us to get an idea of the relations between the followers as in Fig. 7.

The above values were then added to the data frame as in Figs. 8 and 9.

Based on these values the similar feature of the user was calculated by considering the ratings given and it was mapped to the gists as a similar score. So, when we give the gists number the corresponding features get displayed as in Fig. 10.

Based on the score of similar features, the aggregate score was calculated which helped us to find similar followers to the user as in Fig. 11.

```
dfc.head(100)
```

| | user | followers | following | gists | numrepos | degree centrality | in degree cen | betweenness | closeness | clustering coefficient |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Saahil Doshi | 14 | 17 | 0 | 5 | 0.064103 | 0.000000 | 0.006005 | 0.516556 | 0.666667 |
| 1 | Gizachew | 605 | 1959 | 0 | 56 | 0.615385 | 0.608974 | 0.398523 | 0.718894 | 0.041447 |
| 2 | Nand Patel | 1 | 16 | 0 | 4 | 0.673077 | 0.000000 | 0.491054 | 0.753623 | 0.035897 |
| 3 | mirchandani-mohnish | 47 | 24 | 0 | 27 | 0.185897 | 0.000000 | 0.054275 | 0.530612 | 0.229064 |
| 4 | None | 12 | 13 | 0 | 4 | 0.185897 | 0.000000 | 0.020794 | 0.530612 | 0.219212 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 95 | Jevin Jivani | 41 | 84 | 0 | 11 | 0.032051 | 0.000000 | 0.000000 | 0.490566 | 1.000000 |
| 96 | Mananshi | 48 | 55 | 0 | 29 | 0.006410 | 0.000000 | 0.000000 | 0.419355 | 0.000000 |
| 97 | Manav Vagrecha | 33 | 39 | 0 | 20 | 0.006410 | 0.000000 | 0.000000 | 0.419355 | 0.000000 |
| 98 | Martand Javia | 67 | 69 | 0 | 15 | 0.019231 | 0.000000 | 0.000000 | 0.465672 | 1.000000 |
| 99 | None | 6 | 26 | 0 | 2 | 0.006410 | 0.000000 | 0.000000 | 0.430939 | 0.000000 |

**Fig. 4** Values obtained from the centrality measures

```
('kkalaria16', 'AbdeenM', 10.666666666666664)
('kkalaria16', 'kashvi05', 10.666666666666664)
('Gizachew29', '15pratik', 10.666666666666664)
('Gizachew29', 'harshilmehta67', 10.666666666666664)
('Gizachew29', 'DipikaPawar12', 10.666666666666664)
('Gizachew29', 'UditKapadia', 10.666666666666664)
('Gizachew29', 'VidishJoshi', 10.666666666666664)
('Gizachew29', 'im3dabasia', 16.799999999999997)
('Gizachew29', 'patelabhi23', 16.799999999999997)
('Gizachew29', 'mrchocha', 10.666666666666664)
('Gizachew29', 'vatsal-dp', 10.666666666666664)
('Gizachew29', 'dhruvshah01', 10.666666666666664)
('Gizachew29', 'digant15803', 10.666666666666664)
('Gizachew29', 'MRJ35', 10.666666666666664)
('Gizachew29', 'caped-crusader16', 10.666666666666664)
('Gizachew29', 'MuskanM1', 10.666666666666664)
('Gizachew29', 'samkitk', 16.799999999999997)
('Gizachew29', 'PClub-Ahmedabad-University', 10.666666666666664)
('Gizachew29', 'viju3038', 10.666666666666664)
('Gizachew29', 'RidhamShah', 10.666666666666664)
('Gizachew29', '22388o', 10.666666666666664)
('Gizachew29', 'AYIDouble', 16.799999999999997)
('Gizachew29', 'manjunath5496', 10.666666666666664)
('Gizachew29', 'coderaditya3', 16.799999999999997)
('Gizachew29', 'naruhitokaide', 10.666666666666664)
('Gizachew29', 'tirthPatel177', 10.666666666666664)
('Gizachew29', 'Stepwell-Radio', 10.666666666666664)
('Gizachew29', 'srushti-03', 10.666666666666664)
('Gizachew29', 'YashLongani29', 10.666666666666664)
('Gizachew29', 'tirth8205', 16.799999999999997)
('Gizachew29', 'rajmehta18', 10.666666666666664)
('Gizachew29', 'kashishjivani', 10.666666666666664)
('Gizachew29', 'Poojan987', 16.799999999999997)
('Gizachew29', 'Priyank31', 10.666666666666664)
```

**Fig. 5** Common neighbours centrality measure

```
def normalize(row):
    new_row=(row-row.mean())/(row.max()-row.min())
    return new_row
dfc_normalize=dfcu.apply(normalize)
dfc_normalize.head()
```

| | followers | following | gists | numrepos | degree centrality | in degree cen | betweenness | closeness | clustering coefficient |
|---|---|---|---|---|---|---|---|---|---|
| 0 | -0.031839 | -0.021417 | -0.031083 | -0.017522 | 0.051323 | -0.023597 | -0.004642 | 0.163133 | 0.156399 |
| 1 | 0.027249 | -0.005023 | -0.031083 | 0.004301 | 0.878246 | 0.881165 | 0.794696 | 0.623264 | -0.468820 |
| 2 | -0.033139 | -0.021425 | -0.031083 | -0.017950 | 0.964784 | -0.023597 | 0.983130 | 0.702240 | -0.474370 |
| 3 | -0.028540 | -0.021357 | -0.031083 | -0.008108 | 0.234015 | -0.023597 | 0.093658 | 0.195097 | -0.281203 |
| 4 | -0.032039 | -0.021450 | -0.031083 | -0.017950 | 0.234015 | -0.023597 | 0.025476 | 0.195097 | -0.291056 |

**Fig. 6** Values obtained from the cosine distance measure

```
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import jaccard_score
from scipy.spatial.distance import pdist, squareform
user_similarity=cosine_similarity(dfcu) #user similarity
feature_similarity=cosine_similarity(dfcu.T) #feature similarity

user_similarity

array([[1.        , 0.90726441, 0.81884068, ..., 0.62889206, 0.68552595,
        0.95145563],
       [0.90726441, 1.        , 0.94720727, ..., 0.43516906, 0.68886148,
        0.74499801],
       [0.81884068, 0.94720727, 1.        , ..., 0.56689569, 0.8527325 ,
        0.67409133],
       ...,
       [0.62889206, 0.43516906, 0.56689569, ..., 1.        , 0.87196473,
        0.7708928 ],
       [0.68552595, 0.68886148, 0.8527325 , ..., 0.87196473, 1.        ,
        0.68360133],
       [0.95145563, 0.74499801, 0.67409133, ..., 0.7708928 , 0.68360133,
        1.        ]])
```

**Fig. 7** Implementation to find user and feature similarity

| user | Saahil Doshi | Gizachew | Nand Patel | mirchandani-mohnish | 0 | 0 | Dhruv prajapati | Priyanshu Pathak | Nancy Radadia | Aatman Vaidya | ... | 0 | Nisarg Thoriya | Kushal Gandhi | mithilesh thakkar | Parth Sarkhelia | Chirayu Vithalani | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| user | | | | | | | | | | | | | | | | | | |
| Saahil Doshi | 1.000000 | 0.907264 | 0.818841 | 0.896785 | 0.998284 | 0.982526 | 0.998536 | 0.944269 | 0.982093 | 0.807412 | ... | 0.888245 | 0.990033 | 0.373613 | 0.943184 | 0.995648 | 0.673548 | 0.9 |
| Gizachew | 0.907264 | 1.000000 | 0.947207 | 0.632970 | 0.885235 | 0.818486 | 0.926956 | 0.723799 | 0.815092 | 0.578240 | ... | 0.637462 | 0.888721 | 0.125737 | 0.928931 | 0.916149 | 0.557006 | 0.9 |
| Nand Patel | 0.818841 | 0.947207 | 1.000000 | 0.549581 | 0.787024 | 0.739245 | 0.836656 | 0.642049 | 0.725351 | 0.618157 | ... | 0.595198 | 0.758451 | 0.288578 | 0.940058 | 0.804542 | 0.705356 | 0.8 |
| mirchandani-mohnish | 0.896785 | 0.632970 | 0.549581 | 1.000000 | 0.914945 | 0.962277 | 0.873236 | 0.992252 | 0.963708 | 0.924645 | ... | 0.991161 | 0.887982 | 0.623635 | 0.796995 | 0.874469 | 0.724157 | 0.8 |
| 0 | 0.998284 | 0.885235 | 0.787024 | 0.914945 | 1.000000 | 0.988276 | 0.995010 | 0.956987 | 0.989108 | 0.813592 | ... | 0.901942 | 0.993167 | 0.379918 | 0.926637 | 0.994899 | 0.661877 | 0.9 |

**Fig. 8** Values obtained from the user similarity

| | followers | following | gists | numrepos | degree centrality | in degree cen | betweenness | closeness | clustering coefficient |
|---|---|---|---|---|---|---|---|---|---|
| followers | 1.000000 | 0.931039 | 0.193257 | 0.176093 | 0.053440 | 0.032573 | 0.020341 | 0.225889 | 0.224934 |
| following | 0.931039 | 1.000000 | 0.206733 | 0.112252 | 0.034044 | 0.016431 | 0.006773 | 0.184273 | 0.213722 |
| gists | 0.193257 | 0.206733 | 1.000000 | 0.043657 | 0.060534 | 0.095661 | 0.000034 | 0.269474 | 0.302779 |
| numrepos | 0.176093 | 0.112252 | 0.043657 | 1.000000 | 0.048952 | 0.135944 | 0.015763 | 0.198213 | 0.121508 |
| degree centrality | 0.053440 | 0.034044 | 0.060534 | 0.048952 | 1.000000 | 0.375797 | 0.938467 | 0.439090 | 0.212582 |

**Fig. 9** Values obtained from the feature similarity

As a next step for recommending the followers, we used the content-based filtering approach where the keywords of repositories, such as name, description, and language were extracted and added to the data frame by converting strings to lowercase and stemming as in Fig. 12.

```
def get_similar_features(feature_name,rating_given):
    similar_score=fsdf[feature_name]*rating_given
    similar_score=similar_score.sort_values(ascending=False)
    return similar_score
print(get_similar_features("gists",26))
```

```
gists                    26.000000
clustering coefficient    7.872243
closeness                 7.006312
following                 5.375051
followers                 5.024686
in degree cen             2.487182
degree centrality         1.573897
numrepos                  1.135070
betweenness               0.000886
Name: gists, dtype: float64
```

**Fig. 10** Similar feature of gists 26

```
def get_similar_users(userName,aggregate_score):
    similar_score1=usdf[userName]*aggregate_score #aggregate score based on :followers, gists, repos and followings
    similar_score1=similar_score1.sort_values(ascending=False)
    return similar_score1
anshullist=get_similar_users("Anshul Mehta",26)
```

```
for i in anshullist.index:
    print(i)
```

```
Anshul Mehta
Chirayu Vithalani
Vashisth Bhushan
Kavan Desai
Gaston Roldan
Kirtan Kalaria
```

**Fig. 11** Found similar followers for user Anshul Mehta

```
UserList=[]
TwodList=[]
for u in followers_of_followers:
  UserList.append(u.name)
  repo=u.get_repos()
  userRepoList=[]
  # For Lopp for getting the Keywords of Repo
  for r in repo:
    # Add a try catch here to avoid errors
    try:
      userRepoList.append(str(r.name))
      userRepoList.append(str(r.description))
      userRepoList.append(str(r.language))
      # userRepoList.append(r.get_topics())
    except:
      continue
  TwodList.append(userRepoList)
```

**Fig. 12** Extracting keywords from repositories

By the method of TF-IDF and bag of words, the frequency of words was found and then cosine similarity was used to identify the similarity between the TF-IDF vectors which was then stored in the sorted list as in Fig. 13.

From the above diagram, we can infer some of the known keywords such as JavaScript, stake, dart, etc. We may thus conclude that extraction was successful since we have now identified the followers who, in light of the list of followers, can be suggested to the user as in Figs. 14 and 15.

| | TF-IDF |
|---|---|
| dart | 0.426639 |
| thisisyourdailypersonalplannermadeusingreactjsa... | 0.312552 |
| monk | 0.312552 |
| thisrepoisforanappwhichrecordsyourdailyexpenses... | 0.312552 |
| thisisarecipeappmadeusingyummlyapi | 0.312552 |
| thisismyrepoforthedeli | 0.312552 |
| ox | 0.312552 |
| personal_plann | 0.312552 |
| wise | 0.288950 |
| food | 0.208255 |
| javascript | 0.085421 |
| stance | 0.000000 |
| starknet | 0.000000 |
| stars | 0.000000 |
| stakenet | 0.000000 |
| start | 0.000000 |
| stake | 0.000000 |
| started | 0.000000 |
| startercodefortheworkshop | 0.000000 |
| starter | 0.000000 |
| starterfiles | 0.000000 |
| startherepositoryifyoufinditinteresting | 0.000000 |
| stak | 0.000000 |
| staskistofindthisdeepersymmetry | 0.000000 |
| stat | 0.000000 |

**Fig. 13** Frequency of keywords extracted from the TF-IDF

```
sna.head()
```

| | user | Topics |
|---|---|---|
| 0 | Saahil Doshi | delim thisismyrepoforthedeli-mealsappcomprisin... |
| 1 | Gizachew | -coursera-blockchain-basics-university-at-buff... |
| 2 | Nand Patel | ala_project none none drum-kit-using-javascrip... |
| 3 | mirchandani-mohnish | audoc aresourcebookbuiltanddesignedbythestuden... |
| 4 | None | dsa-lab-work thisismydsaprogramsthatiimplement... |

**Fig. 14** Keywords obtained were stored to the corresponding user

```
def recommend(user):
    user_index=sna[sna['user']==user].index[0]
    distances_bow=similarity_bow[user_index]
    distances_tfidf=similarity_tfidf[user_index]
    users_list_bow=sorted(list(enumerate(distances_bow)),reverse=True,key=lambda x:x[1])[1:11]
    users_list_tfidf=sorted(list(enumerate(distances_tfidf)),reverse=True,key=lambda x:x[1])[1:11]
    print("Users recommended by Bag of Words Method")
    for i in users_list_bow:
        print(sna.iloc[i[0]].user)
    print("Users recommended by TF-IDF")
    for i in users_list_tfidf:
        print(sna.iloc[i[0]].user)
```

**Fig. 15** Finding the top 10 users from both the methods

The follower's list closer to the user is printed from the data frame where we can see some followers are the same in both the methods. Figure 16 shows the list of top 10 followers from both the methods.

At last, the objective of our research is to recommend the top 10 followers to the user based on the working of followers in similar technologies or domains. So, the recommendation is done as in Fig. 17.

Additionally, in order to identify the future link prediction between the followers we find Jaccard coefficient and Adamic Adar coefficient measures and also visualized

**Fig. 16** List of top 10 followers from both methods

```
recommend('Anshul Mehta')

Users recommended by Bag of Words Method
Jay Shah
Arnab Dey
Parth Maniyar
Kaushal
Yashraj Kakkad
Nancy Radadia
Chirayu Vithalani
Varshil Shah
dineshssdn-867
Nikhil Balwani
Users recommended by TF-IDF
Jay Shah
Arnab Dey
Chirayu Vithalani
Kavan Desai
Parth Maniyar
Kaushal
Nancy Radadia
Dincer Dogan
Varshil Shah
Yashraj Kakkad
```

```
[ ]  recommendrtlist('Anshul Mehta')

     ['Jay Shah',
      'Arnab Dey',
      'Chirayu Vithalani',
      'Kavan Desai',
      'Parth Maniyar',
      'Kaushal',
      'Nancy Radadia',
      'Dincer Dogan',
      'Varshil Shah',
      'Yashraj Kakkad']
```

**Fig. 17** List of top 10 followers recommended to the developer

the prediction for the top 100 and top 10 followers. Figure 18 depicts scores of Adamic Adar and Jaccard coefficient.

As per the scores, the graph was plotted where the lines between followers shows the future link predictions for the top 100 and top 10 followers. Figures 19 and 20 depict visualization for Adamic Adar and Jaccard coefficient for top 10 followers.

From these graphs, we could be able to infer that there are high chances of future links between the followers of users. As the outliers are the same in these graphs, all graphs are nearly giving the same prediction of links. Figure 21 depicts the scores between the followers.

The outer clusters remain the same and all the inner clusters are concentrated for a few users forming the bigger circle. On basis of scores obtained from the Jaccard coefficient and Adamic Adar coefficient, the top 10 most predicted links between the user and his followers were evaluated.

| | To | From | Score |
|---|---|---|---|
| 12277 | dineshssdn-867 | parthmshah1302 | 5.314758 |
| 11130 | kashishjivani | dineshssdn-867 | 3.846408 |
| 12002 | KAVAN-DESAI | parthmshah1302 | 2.451613 |
| 10289 | samkitk | KAVAN-DESAI | 2.451613 |
| 3485 | AnshulMehta1 | ankitdevani17 | 2.080227 |
| ... | ... | ... | ... |
| 6636 | DincerDogan | DipikaPawar12 | 0.000000 |
| 6635 | DincerDogan | harshilmehta67 | 0.000000 |
| 6634 | DincerDogan | 15pratik | 0.000000 |
| 6632 | harshmange44 | AbdeenM | 0.000000 |
| 12351 | zenilsanghvi | kashvi05 | 0.000000 |

| | To | From | Score |
|---|---|---|---|
| 3187 | parth-27 | UditKapadia | 1.0 |
| 8414 | dhruvildave | D3VAR5H | 1.0 |
| 1133 | Kaushal1011 | jinesh0109 | 1.0 |
| 10209 | caped-crusader16 | aneri0x4f | 1.0 |
| 9154 | JeetKaria06 | nisarg14 | 1.0 |
| ... | ... | ... | ... |
| 6636 | DincerDogan | DipikaPawar12 | 0.0 |
| 6635 | DincerDogan | harshilmehta67 | 0.0 |
| 6634 | DincerDogan | 15pratik | 0.0 |
| 6632 | harshmange44 | AbdeenM | 0.0 |
| 12351 | zenilsanghvi | kashvi05 | 0.0 |

**Fig. 18** Scores of Adamic Adar and Jaccard coefficient

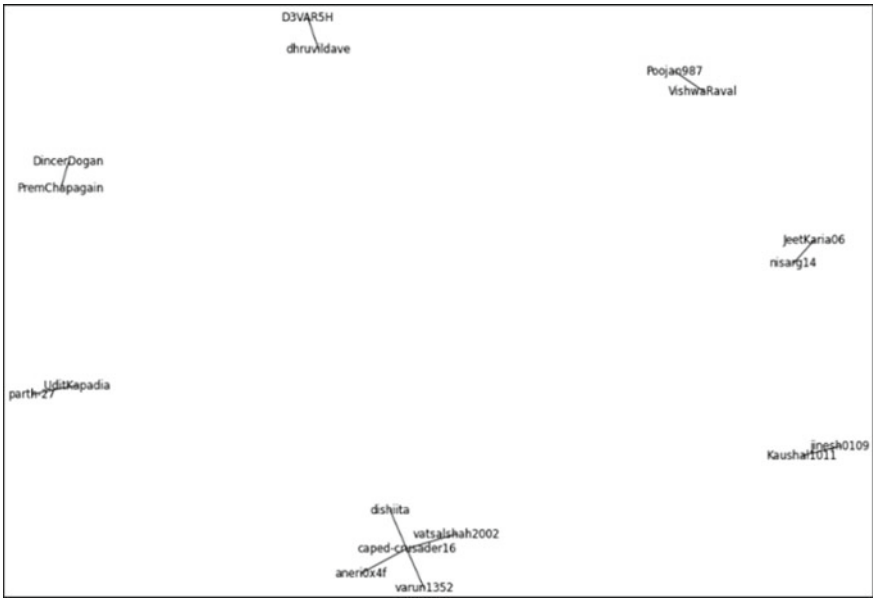


**Fig. 19** Adamic Adar coefficient for top 10 followers

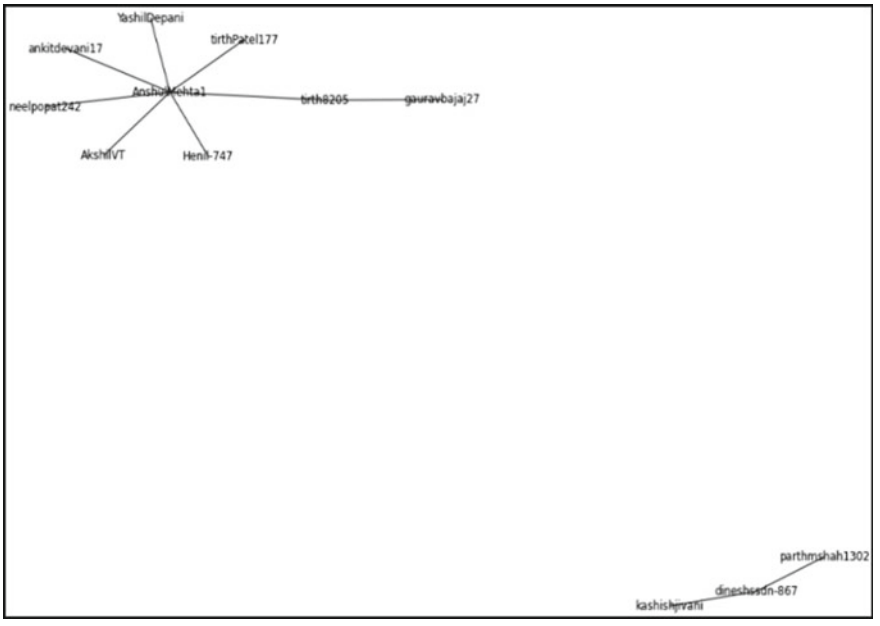


**Fig. 20** Jaccard coefficient for top 10 followers

| | To | From | Score |
|---|---|---|---|
| **3439** | AnshulMehta1 | neelpopat242 | 21.6 |
| **3474** | AnshulMehta1 | tirthPatel177 | 21.6 |
| **3478** | AnshulMehta1 | tirth8205 | 21.6 |
| **3485** | AnshulMehta1 | ankitdevani17 | 21.6 |
| **3407** | AnshulMehta1 | Henil-747 | 21.6 |

**Fig. 21** Scores between the followers

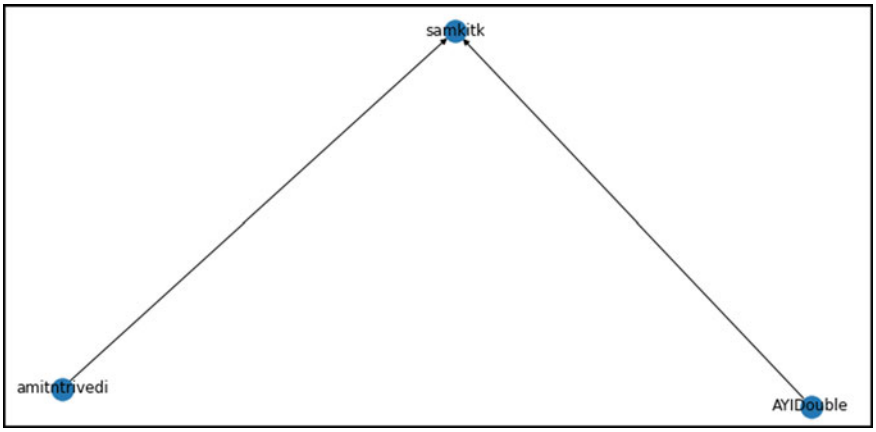


**Fig. 22** Triad of developer and followers

The common triads in the network were found to get an idea of links and influence followers. Triads are a set of three actors and the smallest community possible in a network. Through this, we were able to identify that user and closest follower present in all triads as in Fig. 22.

## 5 Conclusion

Every day, more people join GitHub, but identifying users working in a similar domain is critical for developers so that they can collaborate and work together. We discussed the recommendation system designed specifically for the user and their followers in this paper. We were able to achieve our goals by employing approaches

such as centrality measures, link prediction with standard algorithms, and content-based recommendations based on the developer's real-time data. The results of the tests and a thorough quantitative analysis indicate that the metrics used are effective in suggesting notable followers to the network user.

## References

Farooq A, Joyia GJ, Uzair M, Akram U (2018, Mar) Detection of influential nodes using social networks analysis based on network metrics. In: 2018 International conference on computing, mathematics and engineering technologies (iCoMET). IEEE, pp 1–6

Gajanayake RGUS, Hiras MHM, Gunathunga PIN, Supun EJ, Karunasenna A, Bandara P (2020, Dec) Candidate selection for the interview using GitHub profile and user analysis for the position of software engineer. In: 2020 2nd International conference on advancements in computing (ICAC), vol 1. IEEE, pp 168–173

Joy A, Thangavelu S, Jyotishi A (2018, Feb) Performance of GitHub open-source software project: an empirical analysis. In: 2018 Second international conference on advances in electronics, computers and communications (ICAECC). IEEE, pp 1–6

Koskela M, Simola I, Stefanidis K (2018, Sept) Open-source software recommendations using GitHub. In: International conference on theory and practice of digital libraries. Springer, Cham, pp 279–285

Landherr A, Friedl B, Heidemann J (2010) A critical review of centrality measures in social networks. Bus Inf Syst Eng 2(6):371–385

Liu H, Kou H, Yan C, Qi L (2019) Link prediction in paper citation network to construct paper correlation graph. EURASIP J Wirel Commun Netw 2019(1):1–12

Radha D, Kavikuil K, Keerthi R (2017, Dec) Centrality measures to analyze transport network for congestion free shipment. In: 2017 2nd International conference on computational systems and information technology for sustainable solution (CSITSS). IEEE, pp 1–5

Ramya GR, Bagavathi Sivakumar P (2021) An incremental learning temporal influence model for identifying topical influencers on Twitter dataset. Soc Netw Anal Min 11(1):1–16

Sarwar MU (2021) Recommending whom to follow on GitHub. Doctoral dissertation, North Dakota State University

Sharma S, Mahajan A (2017) Suggestive approaches to create a recommender system for GitHub. Int J Inf Technol Comput Sci 9(8):48–55

Thangavelu S, Jyotishi A (2018, June) Determinants of open source software project performance: a stage-wise analysis of GitHub projects. In: Proceedings of the 2018 ACM SIGMIS conference on computers and people research, pp 41–42

Yadav AK, Johari R, Dahiya R (2019, Oct) Identification of centrality measures in social network using network science. In: 2019 International conference on computing, communication, and intelligent systems (ICCCIS). IEEE, pp 229–234