Github Pull Request Reviewer Recommendation

Madhav Poudel
mp2525@nau.edu
School of Informatics, Computing Cyber Systems
Northern Arizona University
Flagstaff, AZ, U.S.A.

ABSTRACT

Pull Request (PR) is used in Github for the contribution of code from thousands of developers in millions of repository. PR reviewing is an essential activity in software development to maintain the quality of software projects. The time between the submission of the PR and reviewing the PR can be reduced if we can assign new open PRs to appropriate reviewers. However, this feature is not available in Github. To this extent, We purpose a reviewer recommendation system, which predicts the highly relevant reviewers for the forthcoming open PRs. Our method combines the topic modeling and social network analysis taking full advantage of the textual semantic of historical PRs and social relationships between developers. We have implemented a python script that takes an input of the repository name and the new open pull request ID to find the potential reviewers from the crowd. Our approach can reach up to 66% of accuracy for the top 5 recommended reviewers.

KEYWORDS

Github, reviewer, projection bipartite, developers, pagerank

ACM Reference Format:
Madhav Poudel. 2020. Github Pull Request Reviewer Recommendation. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Github is a source code hosting and version control platform that allows collaboration from many software developers. There are over 40 million developers across 195 countries using github and collaborating on various projects. Pull Request(PR) is a primary method for thousands of developers for code contributions.

Once a PR is opened by a contributor, Other developers or contributors can review open PR, add review comments, contribute to the pull request discussion, and even add commits to the PR. [1]. The developers who review on PR are called Reviewers. They can freely discuss the PR with contributors and core developers in terms of their expertise and knowledge. Finally, The core developers decide on the final changes to the PR considering the opinions and discussion from reviewers. Thus, The PR reviewing process is an important activity to ensure the quality of the software.

A recent study shows that popular projects in Github received nearly 100 open PR per day from external contributors [9]. These PRs need to wait for core developers to do the merging and do the final review. This results in a heavy workload to the core developers consuming a large amount of time. Thus, To reduce the workload cost for the core developers, we have purposed a reviewer recommendation script in python that recommends potential reviewers to an open PR so that workload in the repository can be distributed efficiently. We have combined the text mining and social network analysis to recommend highly relevant reviewers. Our method uses the popular Term Frequency - Inverse Document Frequency (TF-IDF) and Latent Dirichlet allocation (LDA) algorithms for topic modeling. Similarly, Our method uses a bipartite network projection function and page rank algorithm for social network analysis. We have also demonstrated the efficiency of the system on the popular open-source repository by showing a recommendation of 5 reviewers and calculating the accuracy up to 67%.

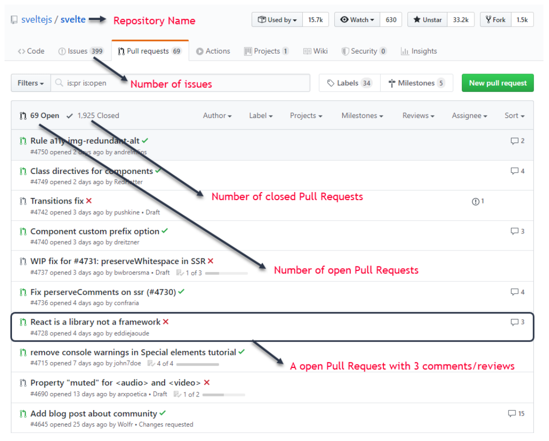
The structure of the project report is organized as follows. Section 2 explains the background of the technologies we have used in the project. Section 3 explains the data used for the project. Section 4 explains the methodology of the recommendation system. Section 5 explains the findings from the experimental in an arbitrary repository and evaluation of average accuracy with a changing number of recommendations. Section 6 discusses about the experiment and overall recommendation system. Finally, Section 7 concludes the report along with future works.

2 BACKGROUND

2.1 Pull Request in Github

Github is a leading source code hosting platform that supports version control and collaborative software development. Since its foundation in 2008, it has grown to be the largest host provider for source code. Github uses the features of the version control system git. Besides, It has features like bug tracking, issue management, project management, wikis, and so on [3]. The important feature of github is Pull Request (PR) which allows massive collaborations between developers. The concept of PR originates from Git, where a developer can pull a branch from the repository of another developer and request changes to it. The changes are reviewed by other members collaborating on the repository and finally merged by the owner/maintainer of the repository. Many github projects use PR to manage changes from contributors and initiate a discussion about the set of changes before it is merged into the main branch of the repository. A typical pull request page of an arbitrary repository with various components is shown in figure 1.

Conference'17, July 2017, Washington, DC, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn



69 Open ✓ 1,925 Closed

Rule a11y img-redundant-alt ✓ 2
#4750 opened 2 days ago by andrelmlins

Class directives for components ✓ 4
#4749 opened 2 days ago by RedHatter

Transitions fix ✗ 1
#4742 opened 3 days ago by pushkine • Draft

Component custom prefix option ✓ 3
#4740 opened 3 days ago by dreitzner

WIP fix for #4731: preserveWhitespace in SSR ✗
#4737 opened 3 days ago by bwbroersma • Draft 1 of 3

Fix perserveComments on ssr (#4730) ✓ 4
#4736 opened 4 days ago by confraria

React is a library not a framework ✗ 3
#4728 opened 4 days ago by eddiejaoude

remove console warnings in Special elements tutorial ✓
#4715 opened 7 days ago by john7doe 4 of 4

Property "muted" for <audio> and <video> ✗
#4690 opened 13 days ago by arxpoetica • Draft 1 of 2

Add blog post about community ✓ 15
#4645 opened 25 days ago by Wolfr • Changes requested

Figure 1: A Pull Requests tab of github repository

It is a tedious task for the maintainer of the repository to manage all the pull requests they receive in the repository especially when the project is large with many contributors. Recommending appropriate reviewers to the open PR would help the maintainer to manage the repository.

2.2 Social Network Analysis

Social network analysis (SNA) is the process of representing social structures in a graph by the use of network science and graph theory. It characterizes networked structures in terms of nodes and edges. In a social network, a node usually represents the individual or actor or thing and the edge represents the relationship between the nodes [6]. The connection between the nodes creates a structure of the network. These structures and topological attributes have been used in several studies to understand the nuances and importance of human behavior in social networks [7].

Github implements a social network structure where developers are equipped to broadcast their activities to the public who are interested in and followed them. Likewise, It integrates many functionalities like issue tracking, PR submitting, code reviewing, wiki, fork, and so on [10]. The different activities in the github are directly related to the behavior of developers in the github. Users on social media or nodes take a variety of structural positions in a network, holding different types of influence in terms of information flow. These structural position defines the role of users in the network and is identified by measurement of the connectivity.

Using social network analysis technique in github, we can find the relationship between the developers who are involved in the PR review process in a source code repository. There are many methods to measure the influence of users in a network. Using these measures, we can find the reviewers who have knowledge and expertise to review the particular closed PRs based on their activities and influence on past closed PRs. To this extent, we have used the bipartite network to project the PR-Reviewer network into the reviewer network and the page rank algorithm to determine potential reviewers with a score. The insights gained from the social relationship between the developers are used in our software to build a reviewer recommendation system.

2.2.1 *Bipartite graph.*

A bipartite graph is a graph whose vertices can be divided into two disjoints and independent groups in such a way that vertex in one group connects to a vertex in other group and have no connection within the group. The group of these vertices is generally called parts of the graph [4]. Due to its unique graph structure, It is used in diverse applications like finding movie preferences, correcting error codes, matching problems, and so on.

Bipartite graph projection is a method of compressing information about the bipartite network into a network of one group of nodes. Since the one-mode projection is less informatic than the bipartite representation, weighting functions are required to better retain the original information. The use of appropriate weighting methods minimize the information loss and conform to the designer's purpose [5].

We can generate a bipartite network between the PRs in the repository and the reviewers with a weight in the link representing the number of times they have commented on a specific PR. This bipartite network is projected to form a reviewer's network with the consideration of weight. Generally, bipartite projection uses Newman's weighted projection function which helps to retain node attributes and connect the resulting graph if it has an edge to a common node in the original bipartite graph. It is also possible to use a custom weight function to retain graph information. The projection with custom weight can be used to retain the information about the importance of reviewers' collaboration with the PR and the similarity result between closed PRs and open PR.

2.2.2 *Page Rank algorithm.*

Page rank algorithm is a variant of eigenvector centrality designed for ranking web content, using hyperlink between pages as a measure of importance. The fundamental concept of this algorithm can be leveraged in any kind of network. it provides each of the nodes a rating based on its importance to the network. The relative ranking score is assigned to all nodes in the network based on the notion that connections to high-scoring nodes provide more to the score of the node in question than equal connections to low-scoring nodes. It helps to uncover influential nodes in the network whose reach stretches beyond just their direct connections.

The page rank algorithm can be used to find the influential reviewer from the reviewer's network whose score can be calculated beyond the direct connection by leveraging the projected weight of the connection.

2.3 Topic Modeling

Topic modeling is a process of identifying topics in a set of documents using unsupervised machine learning technique that is capable of scanning a collection of documents, detecting word and phrase patterns, and automatically cluster word into a combination of similar expressions. There are many existing methods of topic modeling. We have used the combination of Term Frequency - Inverse Document Frequency and Latent Dirichlet allocation algorithm to develop the topic modeling of PR contents.

2.3.1 *Term Frequency - Inverse Document Frequency (TF-IDF).*

TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. The measure is achieved by multiplying two metrics: Number of times a word appears in a document and inverse document frequency of word across a set of documents. The process involves the increasing proportionally to the number of times a word appears in a document and offset by the number of documents that contain the word. The words that are common in every document gets the low rank even though it appears multiple times since they don't significantly contribute to the meaning of the particular document. In contrast, The word which appears many times in one document but doesn't appear in other documents would mean a high rank for that word.

The TF-IDF for a word in a document is calculated by multiplying two metrics:

- Term Frequency (TF): The term frequency is calculated by counting the raw instances of a word appearing in a document. The frequency is then adjusted by the length of the document and the frequent occurrence of the word in the document.
- Inverse Document Frequency (IDF): It is a measure of how common or rare a word is in the entire document set. If the value is closer to 0 then, It means the word is common and if the value is closer to 1 then, the word is rare. The final metric is calculated by taking a total number of documents, dividing by the number of documents that contain a word, and calculating the logarithm function.

Multiplying these two metrics gives the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document.

2.3.2 Latent Dirichlet allocation (LDA). latent Dirichlet allocation (LDA) is an unsupervised learning and generative statistical model that views documents as bags of words proposing that each word in the document is attributable to one of the document's topics. The LDA model determines the different topics that documents represent and how much of each topic is present in a document. The working mechanism of the algorithm for each document is given below.

- First, topics from the document are extracted and distributed across the document by assigning each word a topic
- For each word in a document, the topic is assumed wrong but every word is assigned to the correct topic
- Probabilistically assign word a topic based on the topic in the document and occurrence of word assigned to a particular topic across all documents
- The process is repeated for each document several times

The algorithm reduces the large collection of documents to some commonly used occurring words or phrases, grouped according to the topics. Topic modeling with LDA is used in different applications of natural language processing, text mining, social media analysis, and information retrieval [8].

In our project, we have combined both the TDF-IDF and LDA algorithm for topic modeling gaining the advantages provided by both of the algorithms.

2.4 Cosine Similarly

Cosine similarity is a measure of how similar the documents are despite the variation of their size. It is often used to measure the document similarity in text analysis and information retrieval.

Mathematically, it measures the similarity between two vectors projected in a multidimensional space from cosine of the angle between them. The smaller the angle, the higher is the cosine similarity [2]. The cosine similarity is advantageous even if the two similar documents are far apart by the Euclidean distance because of the size as they could still have a smaller angle between them.

In our project, the vectors for the cosine similarity is generated from topic modeling with LDA followed by the calculation of term frequency by the TF-IDF algorithm.

3 DATA COLLECTION

The data required in the project is mined from the github API. Github has provided a python library called "githubpy" which offers the querying of API endpoints using library functions. This library is used to pull the necessary data required for the project. Since the library can be used to retrieve data in real-time, we have not collected data beforehand. Instead, We have made the implicit call from the library to gather data and processes according to the input provided by the user. Using the library, we have gathered the repository information, open and closed PRs list, metadata of the PRs, comments in the PRs, and reviewers in the PRs.

Github API is accessed by passing tokens to the library's initializing function. Github has set a limit to the number of API requests per hour based on the types of request functions. To cope with this limitation, we have used the API request functions at the beginning of the algorithm and stored in variables for further usage. This said, using algorithms several times an hour may result in an error about the request rate limit.

4 METHODOLOGY

4.1 Pull Request and Reviewer Network

First of all, we construct the Pull Request and Reviewer graph network from a repository provided as input from the user. The algorithm considers that the repository is large enough to have several PRs and collaborators. As our algorithm demands a huge amount of data, the repositories with no or fewer PRs are irrelevant. The case is true for every recommendation system.

The PRs information are collected by querying the repository using the Github API library. By the same token, PR comments are queried to collect the reviewers. The PR with no pull requests are discarded. Since we consider reviewers as external developers, the maintainers of the project and the PR requester are discarded from the reviewers list. Similarly, There is widespread usage of a bot in github for tracking pull requests and issues. We have discarded bots from the reviewers list as well. Finally, these reviewers are connected with PR to form a graph. The graph generated is a bipartite graph where one group or part consists of PRs and the other consists of reviewers. Each of the links between the PR and reviewer nodes are assigned with weight. The weight is calculated by measuring the number of times a reviewer comments on a particular PR. Example: If a reviewer has commented on a PR 5 times, then the weight of the link between reviewer and PR is 5. We assume that weight signifies the interest and expertise of the developers in the particular domain of PR. A typical bipartite network of PRs and reviewers withn weight is shown in the figure 2. Furthermore, The PRs and reviewers data collected during this process is saved in the variables for further usage to prevent the rate limit error from github API.

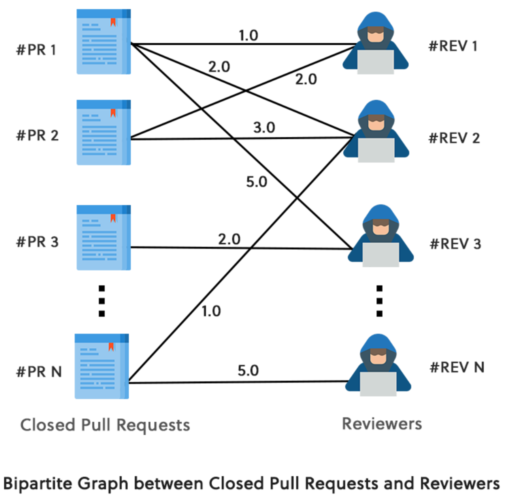


Figure 2: The structure of bipartite network with weight

4.2 Topic Modeling and Similarly Calculation

For the topic modeling, the algorithm collects textual document from all the closed PRs. Each of the documents consists of a title, description, and comment contents. The PR and comment content use markdown text in Github and usually it consists of source code, user mentions, and website links. We have filtered the document with these texts while keeping only the text that describes the particular PR in plain language. The documents from closed PRs are managed in a dictionary called corpus. The algorithm takes an input of one open PR to determines the recommended reviewers. The corpus of the open PR is also extracted using the similar technique.

The documents collected from PRs are preprocessed in the following steps:

- Tokenization: First, The texts from the sentences are split into words. The punctuations are removed from the text and all the texts are converted into lowercase. Also, the words that have fewer than 3 characters are removed.
- Stopword removal: Second, Stop words like a, an, the, in, are, etc in the English language which occurs commonly are removed from the texts.
- Lemmatization: Third, the words that represent the third person are changes into first-person, and verbs in the past and future tense are converted to present tense.
- Stemming: Finally, the words are reduced into its root form. Example: car, cars, car's, cars' are reduced to the car and am, are, is reduced to be.

These preprocessing steps of the corpus document are performed by using gensim and nltk python library using appropriate functions.

Once the corpus document is preprocessed, the bag of words is created with a dictionary containing words and the occurrence in the document. It is further processed with the TF-IDF model by applying transformation to get TF-IDF scores. Further, we train the LDA model in the bags of words and run the LDA algorithm using the result from TF-IDF. This process is performed with 100 topics and 4 workers running in parallel.

Further, the result of the LDA function is applied to the cosine similarity function to get the similarity between the closed and open PRs documents. The similarity matrix between the chosen open PR and all the closed PRs are calculated during this process. Then, we choose top similar closed PRs to form a subgraph. The top similar closed PRs are selected by using a similarity threshold value provided by the user. Example: If the similarity threshold value is 0.2 then the top 20% of the closed PRs are chosen to form subgraph. The default value of the similarity threshold is 0.2.

4.3 Sub graph generation and Bipartite Graph Projection

From the chosen top PRs based on the similarity threshold, a subgraph is formed discarding the rest of the PR nodes. The links of removed PR nodes are also removed during the node removal process.

The resulting bipartite subgraph is projected to form a reviewer graph considering the weight in the links. Moreover, we used custom weight function during the projection where the weight from the bipartite graph is multiplied with similarity result to form new weight. For N number of PR nodes connected to u and v reviewer's node, custom weight function is given in equation 1.

$$\sum_{i=0}^{N}(weight(u,pr_i)+weight(v,pr_i))*similarityscore(pr_i) \quad (1)$$

The preservation of weight during the projection process helps to retain the information we had in the original bipartite graph and include the similarity score. In the projected graph, nodes are the reviewers and the links are the influence of reviewers in closed PRs. The influence of the reviewers in the graph can be used to recommend the appropriate reviewers for a particular open PR.

4.4 Reviewers Infulence Measurement

We have used the page rank algorithm to measure the influence of the node in the projected graph. Since weight has significant information, the use of a page rank algorithm provides significant advantages in the reviewer's influence calculation. The resultant score from the page rank algorithm is the recommendation of the potential reviewers for the particular PRs. The projection of graph from the similarity results and ranking for recommendation is shown in the figure 3.

In this way, By the combination of topic modeling and network science, the algorithm recommends reviewers for a particular PR in a github repository.

5 EXPERIMENTAL EVALUATION AND ANALYSIS

We have chosen a github repository named sveltejs/svelte with over 293 contributors, 1926 closed PRs, and 70 open PRs that is sufficient

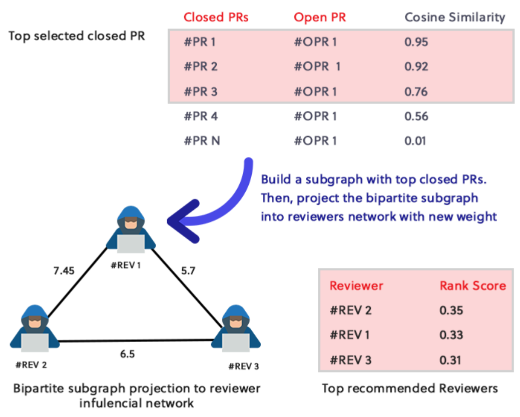


Figure 3: Cosine similarity projection, bipartite projection, and recommended reviewers

enough for our algorithm to experiment. We have used a medium scale source code project for the experiment to prevent the Github API limitation.

First of all, we scarped the repository data from Github API. The informations retrieved are closed PRs, open PRs, comments on PRs, and reviewers reviewing those PRs. The PRs with comments less than one is filtered. Similarly, We have filtered the reviewer who are maintainer, PRs requester, bot, and empty or ghost user. The reviewer commenting multiple times on a PR is considered as weight of the edge between closed PR and reviewers node. We have generated a bipartite network from the information between the reviewer and PR which shown in the figure 4.

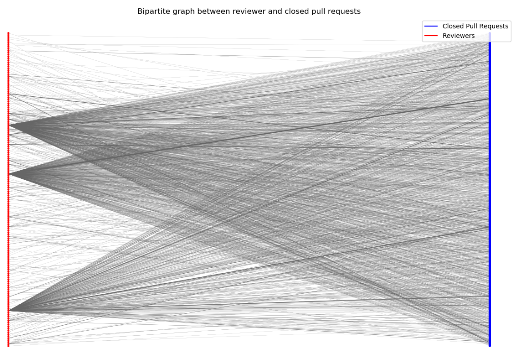


Figure 4: Bipartite graph between reviewers and closed pull requests

The gererated bipartite graph between PRs and reviewers has the following properties:

- Number of nodes: 1488
- Number of edge (reviews): 1830
- Number of PRs filtered: 1327
- Number of reviewers: 161
- Average Degree: 2.4597

Second, we determine an open PR with id 1410 to get the reviewer recommendations. Using the open and closed PRs data, we build a document corpus for topic modeling. The topic modeling is achieved by TF-IDF and LDA algorithm. The result from the topic modeling is subjected to cosine similarity function to generate similarities between all the closed PR in the graph to the open PR. Top 20% (0.2 similarity threshold) closed PRs are chosen for further analysis. The top 5 results of cosine similarity are given in table 1.

Closed PRs	Open PRs	Similarity score
PR #4758	PR #1410	0.8635911438068727
PR #4753	PR #1410	0.8350713392420019
PR #4739	PR #1410	0.8178997026707061
PR #4724	PR #1410	0.7934759953696884
PR #4719	PR #1410	0.7915135635426369

Table 1: Table showing top 5 similarity matrix result

There were 131 top PR nodes with cosine similarity threshold of 0.2. We used these PR nodes to generate a subgraph from a bipartite graph. During the subgraph generation, 1184 PR nodes and the their link of to reviewers were removed.

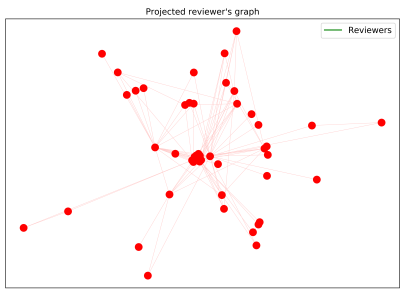


Figure 5: Projected graph of reviewers

The bipartite subgraph is then projected to form a reviewers only network considering the weights in the edges. Further, Isolated nodes with no links are removed from the projected graph as they serve no purpose to our algorithm. The projected graph generated is shown in the figure 5 with the properties given below.

- Number of nodes: 46
- Number of edge: 96
- Average Degree: 4.1739
- Average clustering: 0.606484533991833

Thrid, the projected graph is used to find influential reviewers with expertise and knowledge in the particular open PR. We use the page rank algorithm to determine the infulencial reviewersand the top reviewers were recommended as potential reviewers. The result from the page rank algorithm is given in table 3.

To evaluate the result, we scarped the actual reviewers who were reviewing the PR. The were: stalkerg, antony and Conduitry. We used the actual result and predicted result to measure the accuracy of the recommendation. We got the accuracy of 67%. The graph with pagerank result is shown in figure 6

Recommended reviewer	Page rank score
Conduitry	0.21293204796229653
Rich-Harris	0.10618037165190555
codecov-io	0.05653590244003036
antony	0.04195710147687927
tanhauhau	0.03964471209395288

Table 2: Top 5 recommendation from page rank algorithm

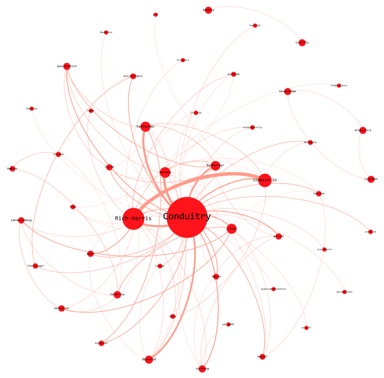


Figure 6: Projected graph of reviewers with Page Rank result

Finally, We took 3 other open pull requests to find the accuracy with a changing number of recommendations. The figure 7 shows the plot of recommendation accuracy with changing number of recommended reviewers.

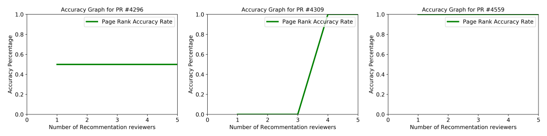


Figure 7: Recommendation accuracy plot

Futhermore, We calculated average accuracy for recommended reviewers these 3 open PRs with IDs 4309, 4559 and 4296 which are shown the table 3.

# of recommendations	Average accuracy
1	0.5
2	0.5
3	0.5
4	0.83
5	0.83

Table 3: Average recommendation accuracy for 3 open PRs

The result shows that the average accuracy is 50% for 1, 2 and 3 recommendations, and 83% for 4 and 5 recommendations. We infer from our experiment that the accuracy of our recommendation system is above 60%.

6 DISCUSSION

By using the combination of social network analysis and topic modeling, we built a reviewer recommendation system for github to ease the process of maintaining large projects by maintainers. We used the topic modeling to get the similarities between the closed PRs and open PR which implies the similarity of the pull requests. Since these PRs are similar to each other, recommending the expert and knowledgable reviewer to review the open PR would drastically expediate the process of PR review and project maintainence. The experiment we conducted with an arbitrary github repository shows accuracy upto 67% when the predicted recommended reviewers were compared to the actual reviewers.

While we have achieved 67% of average accuracy in our experiments, the accuracy might deviate from different conditions. These conditions sets limit to our algorithm. Some of these limitations are:

- The recommendation system needs lots of data for topic modeling and graph generating. We need to choose a repository with a higher number of pull requests to get better results.
- All the pull requests don't necessarily have enough content to be used for finding similarities. Insufficient data in pull request can deviate the accuracy value.
- The document generated from PR increases over time with increased activities in github repository. It would affect the cosine similarity result with time.

7 CONCLUSION

In this project, we implemented a recommendation system in python using social network analysis and topic modeling. The experiment we conducted showed the recommendation with over accuracy of 67% when 5 reviewers are recommended. We have used the custom weight function while projecting a bipartite graph to the reviewer's influential graph. The weight plays a significant role in the recommendation of potential reviewers. Because of the lack of data in topic modeling and graph generation, there could be an impact on accuracy in the recommendation. We have naively used the similarity matrix result in our weight function. It could be furher optimize to build a robust weight function. The use of similarity matrix result in weight function remains as the future work of the algorithm.

ACKNOWLEDGMENTS

I would like to thank Dr. Morgan Vigil-Hayes for providing an oppurtunity to work on a project. It has been a great learning experience for me. It has helped me to broaden my knowledge about network science and it's the usage with topic modeling. Although the project doesn't include different network visualization and approaches to calculate the social infulence, I got a chance to explore and learn a lot from it. Overall, I would like to thank you for this project to happen.

REFERENCES

[1] [n. d.]. About pull requests - GitHub Help. https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-pull-requests
[2] 2018. Cosine Similarity - Understanding the math and how it works? (with python). https://www.machinelearningplus.com/nlp/cosine-similarity/

[3] 2018. Microsoft to acquire GitHub for $7.5 billion. https://news.microsoft.com/2018/06/04/microsoft-to-acquire-github-for-7-5-billion/

[4] 2020. Bipartite graph. https://en.wikipedia.org/w/index.php?title=Bipartite_graph&oldid=951042828 Page Version ID: 951042828.

[5] 2020. Bipartite network projection. https://en.wikipedia.org/w/index.php?title=Bipartite_network_projection&oldid=945808568 Page Version ID: 945808568.

[6] 2020. Social network analysis. https://en.wikipedia.org/w/index.php?title=Social_network_analysis&oldid=946499547 Page Version ID: 946499547.

[7] Peter Balogh, József Popp, Judit Oláh, Monika Harangi-Rakos, and Peter Lengyel. 2017. Social Network Analysis of Scientific Articles Published by Food Policy Journal.

[8] Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. 2018. Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey. *Multimedia Tools and Applications* 78 (11 2018). https://doi.org/10.1007/s11042-018-6894-4

[9] Jakub Lipcak and Bruno Rossi. 2018. A Large-Scale Study on Source Code Reviewer Recommendation. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, Prague, 378–387. https://doi.org/10.1109/SEAA.2018.00068

[10] Ferdian Thung, Tegawendé Bissyandé, David Lo, and Lingxiao Jiang. 2013. Network Structure of Social Coding in GitHub. *Proceedings of the Euromicro Conference on Software Maintenance and Reengineering, CSMR*, 323–326. https://doi.org/10.1109/CSMR.2013.41