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. When a developer contributes to a project by submitting PR, the maintainer of the project needs to deal with it by taking all the opinions of reviewers into consideration and merging into the main branch. It is a time-consuming process especially when the project is large. 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 have purposed 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 PR ID to find the potential reviewers from the developer's crowd. Our approach can reach up to 83% of average accuracy for the top 5 recommended reviewers.

KEYWORDS

github, reviewer, projection, bipartite, developer, pagerank, similarity, recommendation

ACM Reference Format:
Madhav Poudel. 2020. Github Pull Request Reviewer Recommendation. In *Northern Arizona University, Flagstaff, AZ*. 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 50 million developers across 195 countries using github and collaborating on more than 100 million projects [2]. Pull Request(PR) is a primary method for thousands of developers to contribute github.

Once a PR is opened by a contributor by providing a detailed description of the code changes and the reasons for those changes, other developers or contributors can review open PR, add review comments, contribute to the PR 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 project maintainers/core developers based on their expertise and knowledge. Finally, the project maintainers 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 receive nearly 100 open PR per day from external contributors [14]. The PRs received in the project are commonly discussed among the reviewers to ensure the quality of software. However, these PRs need to wait for the project maintainers to do the final review and merging into the main branch. It is a challenge for project maintainers to assign appropriate reviewers for code review to expedite the reviewing process [8]. Moreover, github doesn't provide the functionality to recommend appropriate reviewers for the open PRs.

To reduce the workload cost for the project maintainers, we have purposed a reviewer recommendation system in python that recommends potential reviewers to an open PR so that workload in the repository can be distributed efficiently. We have combined the topic modeling techniques and social network analysis to recommend highly relevant reviewers. To be more precise, we have used the popular Term Frequency - Inverse Document Frequency (TF-IDF) and Latent Dirichlet allocation (LDA) algorithms for topic modeling and bipartite graph projection and page rank algorithm for social network analysis. Further, in an experiment on arbitrary open-source github project, we have demonstrated the average accuracy of up to 83% when five reviewers are recommended.

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 source of data being used in the project. Section 4 explains the methodology of the recommendation system. Section 5 explains the findings from the experimental evaluation. Section 6 discusses the experiment result and the overall recommendation system. Finally, section 7 concludes the report along with future works.

2 BACKGROUND

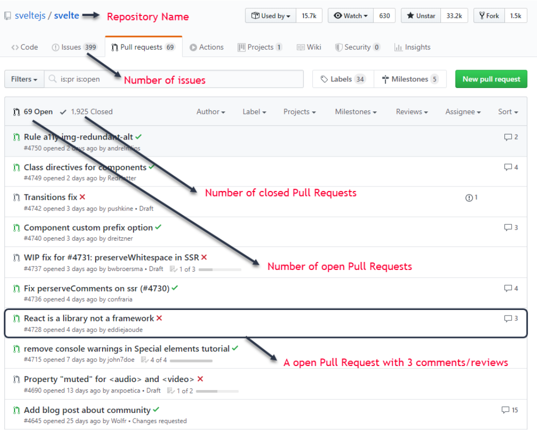
2.1 Pull Request in Github

The advent of a distributed version control system has boosted the rise of a new model for distributed software development. It has encouraged more and more external developers to contribute their code and provide suggestions to core developers in open source projects[18]. Many code hosting platform has adopted this paradigm to facilitate pull-based development offering workflow

Northern Arizona University, Jan-May, 2020, Flagstaff, AZ
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

support tools like code reviewing, issue tracking, and deployment [11].

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 called git. Besides, it has features like bug tracking, issue management, project management, wikis, and so on [4]. One of the important features of github is Pull Request (PR) that 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 PR page of an arbitrary repository with various components is shown in figure 1.



Repository Name

Number of issues

Number of closed Pull Requests

Number of open Pull Requests

A open Pull Request with 3 comments/reviews

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 typical PR tab in a github repository showing various components

It is a tedious task for the project maintainers to manage all the PRs they receive in the repository especially when the project is large. Recommending appropriate reviewers to the open PR can help them to manage the repository effectively.

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 where node usually represents the individual or actor or thing and the edge represents the relationship between the nodes [7]. The connection between the nodes creates a structure of the network called the social network. Social network structures and topological attributes have been used in several studies to understand the nuances and importance of human behavior in social networks [9]. SNA can be applied to a wide range of areas [13] to study the nature of the social network.

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, project management, and so on [15]. 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 SNA techniques in github, we can find the relationship between the developers who are involved in the PR review process in a source code repository [16]. 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 open PR 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.

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 called parts of the graph [5]. 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 [6].

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 more information from the original bipartite graph. The projection with custom weight can be used to retain the information about the importance of reviewers' collaboration in PRs 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 [17]. 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 with the page rank score assigned to it. The use of custom weight function in bipartite graph projection additionally contributes to the score of the page rank algorithm.

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: the 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 [10]. 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 [12].

In the 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 [3]. 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 between PRs are generated using the result from topic modeling with LDA and TF-IDF. The cosine similarity result is used to build a subgraph of a bipartite network connecting PRs and reviewers.

3 DATA COLLECTION

The data required in the project is gathered 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 necessary 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 data, 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 network from data collected from a repository provided as input from the user. The recommendation system demands a repository of a large size consisting of several PRs and collaborators for effective results.

As it demands a huge amount of data, the repositories with no or fewer PRs are irrelevant for reviewer recommendations. The case is true for every recommendation system.

The PRs data is collected by querying the repository using the github API library. By the same token, reviewers are collected by querying the comments of each PRs. The PR with no comments are discarded as it doesn't connect with any reviewers. Also, as we consider reviewers as external developers, the maintainers of the project and the PR requester are discarded from the reviewer's list. Similarly, there is the widespread usage of a bot in github for tracking PRs and issues. We have discarded bots from the reviewer's 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 is 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 within weight is shown in 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.

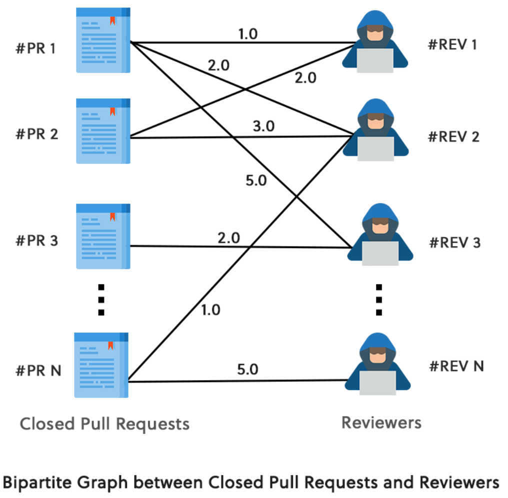


Figure 2: Bipartite graph between closed PRs and Reviewers with weight

4.2 Topic Modeling and Similarly Calculation

For the topic modeling, the algorithm collects textual documents from all the closed PRs. Each of the documents consists of a title, description, and comment contents. The PR and comment content uses 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 a 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.

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 the transformation function to get TF-IDF scores. Further, we trained the LDA model in the bags of words and run the LDA algorithm using the result from TF-IDF. The result of the LDA function is then 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. We have used 0.2 as a default value for the similarity threshold.

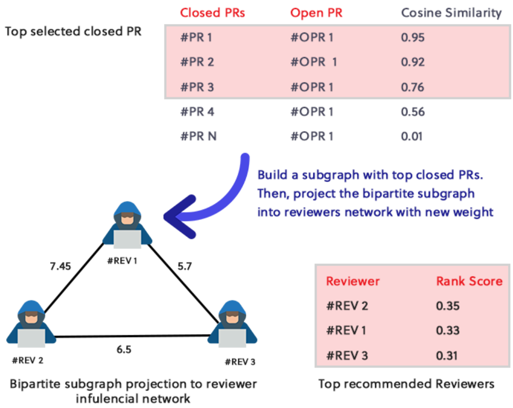
4.3 Sub graph generation and Bipartite Graph Projection

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 the 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 similar to open PR. The influence of the reviewers in the graph is further used to recommend the appropriate reviewers for a particular open PR.



Closed PRs	Open PR	Cosine Similarity
#PR 1	#OPR 1	0.95
#PR 2	#OPR 1	0.92
#PR 3	#OPR 1	0.76
#PR 4	#OPR 1	0.56
#PR N	#OPR 1	0.01

Reviewer	Rank Score
#REV 2	0.35
#REV 1	0.33
#REV 3	0.31

Top selected closed PR
Build a subgraph with top closed PRs. Then, project the bipartite subgraph into reviewers network with new weight
#REV 1
#REV 2
#REV 3
7.45
5.7
6.5
Bipartite subgraph projection to reviewer infulencial network
Top recommended Reviewers

Figure 3: Cosine similarity between open PR and closed PRs, bipartite graph projection, and top recommended reviewers from page rank score

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 information about the knowledge and expertise of reviewers, the use of a page rank algorithm provides significant advantages as it considers weight in its score calculation. The score from the page rank algorithm considered as the recommendation of the potential reviewers for the particular PRs. The projection of the graph from the similarity results and ranking for recommendation is shown in 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 for the experiment. It has over 293 contributors, 1926 closed PRs, and 70 open PRs sufficient enough for our algorithm to produce an effective result. We have used a medium scale source code project for the experiment to prevent the Github API limit issue.

First of all, we collected the repository data from the github API. The pieces of data 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 is 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 figure 4.

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

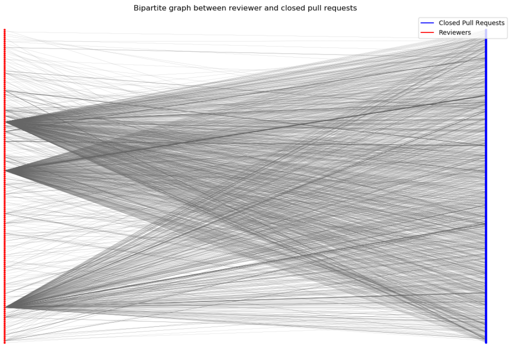


Figure 4: Bipartite graph between reviewers and closed PRs

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 a 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 their link to reviewers were removed.

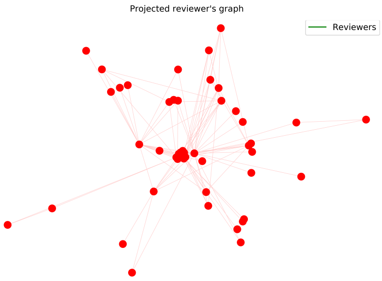


Figure 5: Reviewer's only network after bipartite subgraph projection

The bipartite subgraph is then projected to form a reviewer's 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 figure 5 with the properties given below.

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

Third, 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 influential reviewers and the top reviewers were recommended as potential reviewers. The result from the page rank algorithm is given in table 2.

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

Table 2: Table showing top 5 recommended reviewers with page rank score

To evaluate the result, we collected the actual reviewers who were reviewing the PR from the github API. They were: **stalkerg, antony and Conduitry**. We used the actual result and predicted results to measure the accuracy of the recommendation. We got the accuracy of 67%. The graph with page rank result is shown in figure 6

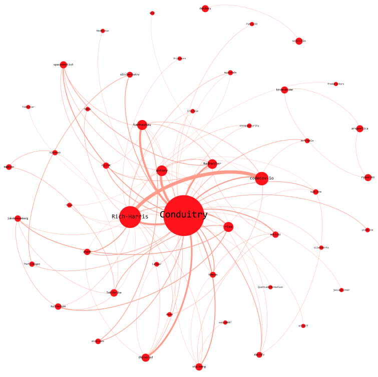


Figure 6: Reviewer's only network showing page Rank result by node size

Finally, we took 3 other open PRs with id **#4309**, **#4559**, and **#4296** to find the accuracy with a changing number of recommendations. Figure 7 shows the plot of recommendation accuracy with a changing number of recommended reviewers.

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

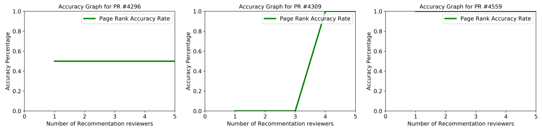


Figure 7: Plot showing average accuracy for reviewers recommendation in 3 open PRs

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

Table 3: Table showing average recommendation accuracy for 3 open PRs with changing number of recommended reviewers

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 can reach up to 83% when applied to a repository with sufficient data.

6 DISCUSSION

By using the combination of social network analysis and topic modeling, we have built a reviewer recommendation system for github to ease the process of maintaining large projects by the maintainers. We have used the topic modeling to get the similarities between the closed PRs and open PR and social network analysis to get the recommended reviewers. Since these PRs are similar to each other, recommending some expert and knowledgable reviewer to review the open PR would drastically expedite the process of PR review and project management.

We conducted an experiment with an arbitrary github repository which results in an accuracy of recommendation up to 83% compared to the actual reviewers. While we have achieved 83% of average accuracy in our experiments, the accuracy might deviate because of several conditions. These conditions sets limit to our recommendation system. 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 PRs to get better results.
- All the PRs don't necessarily have enough content for finding PR similarities. Insufficient data in PR can deviate the accuracy result.
- The document generated from PR increases over time due to increased activities in the 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 result with an accuracy of up to 83% when 5 reviewers are recommended. We have used the custom weight function while projecting a bipartite graph to the reviewer's influential graph which plays a significant role to retain the information of the reviewers and PR similarities. We have naively used the similarity matrix result in our weight function. It could be further optimized to build a robust weight function. Moreover, Lack of data in topic modeling and graph generation could impact the accuracy of the recommendation. So, It is suggested to use the script with a large repository consisting of enough PR data. Similarly, the recommendation system cannot be used in realtime as it needs to process a large amount of data to produce a result. However, it can be used as a backend engine where the processing of the recommendations takes place whenever a new PR is published in the repository, and the result is stored so that it can be returned whenever the client requests. It remains the future work for the project.

ACKNOWLEDGMENTS

I would like to thank Dr. Morgan Vigil-Hayes for providing me an opportunity to work on this project. It has been a great learning experience for me. It not only helped me to broaden my knowledge about network science but also it's an application in another area of computing. Although the project doesn't include different network visualization and approaches for social influence calculation, I didn't miss the chance to explore and learn about them. 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] [n.d.]. Build software better, together. https://github.com

[3] 2018. Cosine Similarity - Understanding the math and how it works? (with python). https://www.machinelearningplus.com/nlp/cosine-similarity/

[4] 2018. Microsoft to acquire GitHub for $7.5 billion. https://news.microsoft.com/2018/06/04/microsoft-to-acquire-github-for-7-5-billion/

[5] 2020. Bipartite graph. https://en.wikipedia.org/w/index.php?title=Bipartite_graph&oldid=951042828 Page Version ID: 951042828.

[6] 2020. Bipartite network projection. https://en.wikipedia.org/w/index.php?title=Bipartite_network_projection&oldid=945808568 Page Version ID: 945808568.

[7] 2020. Social network analysis. https://en.wikipedia.org/w/index.php?title=Social_network_analysis&oldid=946499547 Page Version ID: 946499547.

[8] Vipin Balachandran. 2013. Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation. *Proceedings - International Conference on Software Engineering*, 931–940. https://doi.org/10.1109/ICSE.2013.6606642

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

[10] David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3 (05 2003), 993–1022. https://doi.org/10.1162/jmlr.2003.3.4-5.993

[11] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An Exploratory Study of the Pull-Based Software Development Model. In *Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE 2014)*. Association for Computing Machinery, New York, NY, USA, 345–355. https://doi.org/10.1145/2568225.2568260

[12] 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

[13] Clement Lee and Darren Wilkinson. 2018. A Social Network Analysis of Articles on Social Network Analysis.

[14] 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

[15] 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

[16] 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

[17] Z. Yang, X. Huang, J. Xiu, and C. Liu. 2012. SocialRank: Social network influence ranking method. In *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, Vol. 02. 591–595.

[18] Yang Zhang, Gang Yin, Yue Yu, and Huaimin Wang. 2014. Investigating Social Media in GitHub's Pull-Requests: A Case Study on Ruby on Rails. https://doi.org/10.1145/2666539.2666572