Github Project Recommendation System

Prof. Girija Chiddarwar[1] Rugved Vengurlekar[2] Somesh Narwade[3] Shrikant Pansare[4] Kunal Patil[5]
[1,2,3,4,5]Department of Computer Engineering
[1,2,3,4,5]Sinhgad College of Engineering Pune, India

Abstract— In the open source community such as GitHub, developers usually need to find projects similar to their work, with the aim to reuse their functions and explore ideas of features that could be possibly added into their project at hand. Traditional text search engine can help detect similar resources. However, it is difficult for developers to use in open source community because a few query words cannot describe the whole features of a project. In this paper, we present a practical software recommendation system, which is used to recommend personalized software projects in GitHub. According to the features of projects created by developers and their behaviour to other known projects, the project recommends the top N relevant and personalized software projects.

Keywords: Collaborative Filtering, GitHub, Stochastic, Recommendation System

I. INTRODUCTION

In open source software community, developers are the initiator of all software repositories. They not only create a repository for their own project but also explore other projects they are interested in. For their own projects, they do not have to build everything from scratch. More often, they make copies of their relevant projects and then tailor the features of these projects to meet their needs [1]. At the same time, they are willing to contribute their skills. They can make a suggestion, fix a bug, or contribute code to other developers' projects. If the software repository community can recommend similar and relevant projects, it can improve the developing efficiency by reusing the relevant models of others projects and explore ideas for possible features from other projects. However, developers usually need to waste a lot of time in detecting suitable projects for usage [2].

Typically, developers use the traditional searching engine to find similar projects. But traditional search engines usually focus on text matching—it is not suitable in open source software repositories [3], as software projects are holistic, and a few words cannot accurately describe the characteristics of a software project. If there are some recommended projects that are similar or relevant to their projects, they can freely compare and select suitable projects that they want to use [4]. In the open source community, such as GitHub, projects are usually recommended based on their popularity. Obviously, it ignores the developer's purpose and the content of the developer's projects at hand for recommendation. So this recommendation is not tailored to the individual developer's needs, which not only wastes the developer's time to judge but also decreases the developer's trust in the recommendation.

Hence, recommendation of the software projects in the open source community needs to be personalized as different developers have different requirements for the same project. In this project, we develop a recommendation system to recommend projects that developers care about.

II. LITERATURE SURVEY

Tadej et al. Analyzed different link prediction methods on a graph,that is sampled from a Github network, in order to determine which method is the best and can be used for GitHub open source project recommendation system. To evaluate the performance of link prediction methods they used Area Under ROC Curve(AUC).

Collin proposed an approach, Exemplar, which uses user's query as the input, and returns similar applications. They used domain analysis, page rank and Coupling between Objects in order to achieve a more accurate result. Collin et al. developed a recommendation system, CLAN, which uses a whole project as the input and returns one or more similar projects based on Latent Semantic Indexing.

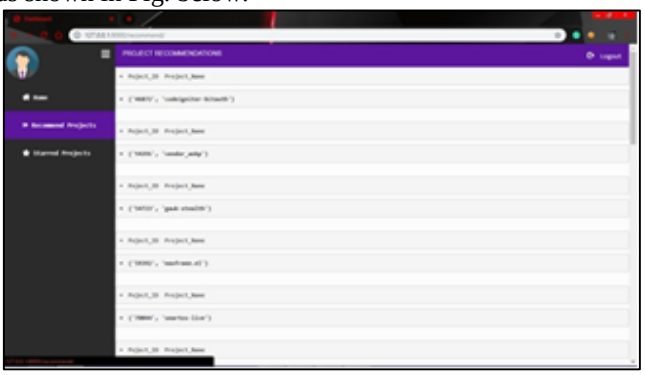
Some studies used user's query as the input, and recommended similar code snippets like Codebroker and Codefinder. Codebroker calculates the similarity between methods depending on the name of the variable and class, and comments in the source code. Codefinder is another recommendation system which compares the similarity mainly on projects's descriptions. It is different from us in that Codefinder does not consider the similarity of source code, which leads to a lower precision of the returned results.

III. SYSTEM OVERVIEW

The GitHub is an open-source platform where one can deploy the projects and also explore other projects on cloud storage. The GitHub only provides the trending projects in market in recommendation. The GitHub project recommendation system will provide the relevant projects of user based on the ratings provided by the user for particular project.

Using those rankings for relating the interest of user, the recommendation system will suggest the projects to the user. The system will consist of a website as a front end and the recommendation system running in the back end. This system will improve its performance for users with more interaction with user by finding the more relevant projects as per the relevance.

Suppose a user who might have starred the project and want to receive recommendations will look on interface as shown in Fig. below.



IV. METHODOLOGY

The Overall system works in different phases. These phases are descried below:-

A. Sync of User's GitHub Data with Our System:

In this step the retrieved data will be stored in our database and the frontend (Django Framework) will send this data to backend for further processing. Synchronize will update the database by accessing the GitHub profile by using the internet.

B. Evaluating User-Item Matrix:

In this phase, by using the data of collected in last phase, the user-item matrix is formed. In this phase, the projects that user will mark as starred will be considered as interested project in users sense. These will be Binary values.

C. Finding Neighbors using Cosine Distance:

As the data in the matrix will be binary, we could have used Jacard Distance but also the fact that it is very much sparsed lead us to find similar projects using Cosine Distance.

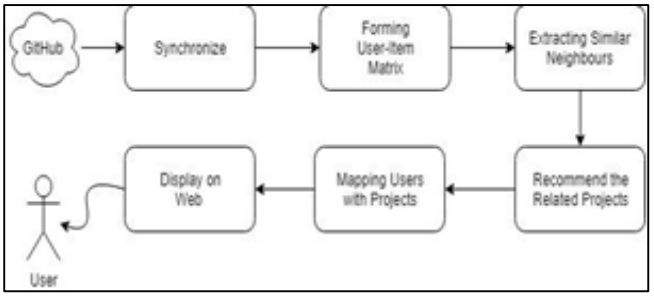
D. Generating Recommendations:

Using the similar projects for user, the value is predicted whether the user will be interested in the project and will mark it as starred. These values are used to recommend the projects for specific user.

E. Showing the Generated Recommendations on User's Dashboard

In this final step the recommendations generated in the previous step are converted to the web format suitable for viewing on the web. The user will receive a list of repositories with their language used, description and a link to the GitHub Repository.

V. SYSTEM ARCHITECTURE



VI. ADVANTAGES

Our overall model has the following advantages.

- The System synchronize data frequently and provide better recommendations by minimizing error using Stochastic Gradient Descent (SGD).
- In advancement, we can consider the weight-age of frequent words used in description and source code of projects.
- The System can be easily scaled by using parallelized databases like AWS.

VII. CONCLUSION

The recommendation system mentioned in the paper recommends projects on GitHub according to relevance of user and might be helpful for the developers to find the projects in which they can contribute and explore of their interests in easy manner. Also, the system can be implemented in real life applications but might need high processing platform and great database structure and handling. The problem of sparsity and fetching of data can be resolved with more data collected from the user and fast, efficient database management like AWS. The content-based collaborative filtering also be developed in order to get recommendations for the person having first visit on the website.

REFERENCES

[1] X. Sun, B. Li, Y. Duan, W. Shi, and X. Liu, "Mining software repositories for automatic interface recommendation," Scientific Programming, vol. 2016, 2016.
[2] C. McMillan, N. Hariri, D. Poshyvanyk, J. Cleland-Huang, and B. Mobasher, "Recommending source code for use in rapid software prototypes," in Proceedings of the 34th International Conference on Software Engineering, ser. ICSE '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 848–858. [Online].
[3] C. McMillan, M. Grechanik, and D. Poshyvanyk, "Detecting similar software applications," in Proceedings of the 34th International Conference on Software Engineering, ser. ICSE '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 364–374. [Online]
[4] W. Shi, X. Sun, B. Li, Y. Duan, and X. Liu, "Using feature-interface graph for automatic interface recommendation: A case study," in Advanced Cloud and Big Data, 2015 Third International Conference on. IEEE, 2015, pp. 296–303.
[5] Miika Koskela, Inka Simola, and Kostas Stefanidis, "Open Source Software Recommendations Using GitHub."
[6] Mohamed Guendouz, Abdelmalek Amine, Reda Mohamed Hamou, "Recommending relevant GitHub repositories: a collaborative-filtering approach", 2nd International Conference on Networking and Advanced Systems May 6-7, 2015.
[7] Wenyuan Xu, Xiaobing Sun, Jiajun Hu, Bin Li, REPERSP: Recommending Personalized Software Projects on GitHub, 2017 IEEE International Conference on Software Maintenance and Evolution.
[8] Mingsheng Fu, Hong Qu, Zhang Yi, Li Lu and Yongsheng Liu, A Novel Deep Learning-Based Collaborative Filtering Model for Recommendation System, IEEE TRANSACTIONS ON CYBERNETICS.

