Journal on Big Data
DOI:10.32604/jbd.2020.010000
Article

An Importance Assessment Model of Open-Source Community Java Projects Based on Domain Knowledge Graph

Chengrong Yang[1], Rongjing Bu[2], Yan Kang[2], Yachuan Zhang[2], Hao Li[2,*], Tao Li[2] and Junfeng Li[2]

[1]Liupanshui Normal University, Liupanshui, 553000, China
[2]Yunnan University, Kunming, 650500, China
[*]Corresponding Author: Hao Li. Email: lihao707@ynu.edu.cn
Received: 20 June 2020; Accepted: 25 October 2020

Abstract: With the rise of open-source software, the social development paradigm occupies an indispensable position in the current software development process. This paper puts forward a variant of the PageRank algorithm to build the importance assessment model, which provides quantifiable importance assessment metrics for new Java projects based on Java open-source projects or components. The critical point of the model is to use crawlers to obtain relevant information about Java open-source projects in the GitHub open-source community to build a domain knowledge graph. According to the three dimensions of the Java open-source project's project influence, project activity and project popularity, the project is measured. A modified PageRank algorithm is proposed to construct the importance evaluation model. Thereby providing quantifiable importance evaluation indicators for new Java projects based on or components of Java open-source projects. This article evaluates the importance of 4512 Java open-source projects obtained on GitHub and has a good effect.

Keywords: GitHub open-source community; Java open-source project; domain Knowledge Graph; importance assessment

1 Introduction

In recent years, with the deep integration of software collaborative development technology and social networks, the social development paradigm has occupied an indispensable position in the current software development process 0. As of April 2018, the Java programming language topped the TIOBE ranking with 15.777% usage, up 0.21% from April 2017. Meanwhile, in Java project development, most project developers choose project construction tools when developing Java projects [2]. According to the statistics of the DZone website, Apache Maven [3], Apache Ant [4], and Gradle [5] are currently widely used project development tools. Based on the mechanisms such as watch, fork, and star provided by the GitHub open-source community, combined with the pom.xml, build.xml, and build.gradle attributes configuration files in the Java project development tools, the domain knowledge graph of Java open-source projects were constructed [6], and the relationship between projects was more intuitively displayed by using the visualization technology.

As an open-source community, GitHub for groups of all kinds, they are from different industries, different positions, interests, experiences, and level of education also varies, to participate in the project of development with its field do not match, lead to this widespread group creation behavior [7] tend to have some disadvantages. Project developers sometimes need to use the open-source community project to secondary development as the basic version of the project, and there may be multiple open-source projects with the same or similar functions. Due to the lack of evaluation basis, if the selected project maintenance personnel are few, and the reliability is low, there may be many problems in the iterative development

process of the project developer. Therefore, the development of the open-source community will be affected more or less if the public creative development activities are not properly guided [8–10].

Based on the above problems, this paper presents an open-source community Java project importance evaluation model based on domain knowledge graph, which measures Java open-source projects from three dimensions of project influence, project activity, and project popularity, and evaluates its importance with PageRank algorithm. Specifically, the main contribution of this paper is to build a knowledge graph of open-source community Java projects and propose a project importance assessment model. The research significance of this paper mainly includes the following two aspects:

(1) Assist the open-source community in quantitative analysis of project importance. Through the quantitative evaluation and scoring of each Java open-source project, and then ranking and comparing them, it provides a reference for project developers;

(2) Guide project development. When project developers select projects from several open-source projects as the basic version for iterative development, it can recommend high-quality projects to developers through project importance assessment and quantitative analysis of project reliability and importance.

2 Background

In the open-source community, project developers release a wide variety of projects, and it is not easy to search the most popular and representative projects among the numerous open-source projects. If we can integrate various attributes of open-source projects in GitHub and evaluate the importance of open-source projects, we can save a lot of time for project developers, and provide a scientific reference. OSSEAN [11] is an analytical search platform for global open-source software. It has crawled many open-source software resources from the collaborative development community and the knowledge sharing community of open-source software, ranked open-source software according to the group wisdom contained in the discussions and feedback of participants, and then recommend high-quality open-source software. In recent years, Google knowledge graph technology has attracted widespread attention, and "open knowledge graph" has also emerged. The openness of open-source community and knowledge graph have similar relationships, and common goal pursuit, both of them are to integrate knowledge. However, due to the limited public disclosure technology of knowledge graph, there are few related researches on knowledge graph in the software domain. However, the knowledge graph in the software domain has a high degree of deficiency. Knowledge graph can effectively guide software development, update, maintenance, and so on. Liu et al. [12] made a comprehensive analysis of the key technologies involved in the construction of knowledge graph based on the definition and technical framework of knowledge graph. The importance assessment of complex network node has always been the focus of many scholars. Zhang et al. [13] drew on the PageRank algorithm, and combined the characteristics of node importance evaluation in complex networks, reflected the influence of the overall link relationship on node importance in directed weighted complex networks. Zhang [16] proposed an algorithm for quantitative evaluation of the importance of complex network nodes, which took the grey relational degree as a measure to evaluate the relevance of each node in the network with the ideal "core node". An evaluation of the importance of open-source projects can provide important clues for developers to search for high-quality projects.

3 Importance Assessment Model

This section first introduces the construction process of the domain knowledge graph, then introduces the impact factors of project importance assessment, and finally gives the improved PageRank importance assessment method.

3.1 Building Domain Knowledge Graph

According to the difference and highly scattered characteristics between the development experience of the project developer and the open-source project, the difficulty of the importance assessment of the

open-source project lies in how to reasonably make use of the relevant attribute data in GitHub for assessment, to provide a reference for project developers.

The main idea of the knowledge graph construction in this paper is to make full use of the activity records of project developers in GitHub open-source community, and integrate the data such as the project's attributes, the correlation between projects and the participation of project contributions into the relationship network diagram, to construct domain knowledge graph. It is mainly divided into the following three steps:

(1) Obtain Java open-source project data. Firstly, got all Java project data developed by Apache Maven, Apache Ant, and Gradle in the GitHub open-source community, including watch, fork, star, commits, etc. Secondly, analyzed the configuration files of three project development tools: pom.xml, build.xml, and build.gradle to acquire the data that determined the unique identity, dependencies, and project description of the project.

(2) Build relationship network diagram. With Java open-source project as the node, the dependency relationship between the projects is used to build the directed relationship network diagram, and the in-degree and the out-degree are determined through the dependencies between the projects. Taking Fig. 1 as an example, the in-degree and out-degree of PMD of project node are 0 and 13, respectively.

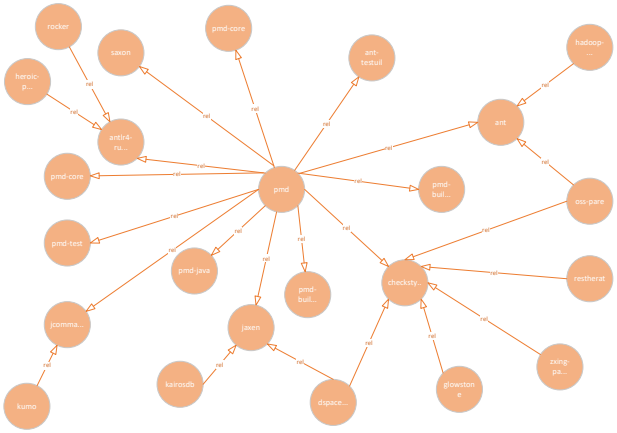

Figure 1: Relationship network diagram

(3) Construct domain knowledge graph. With the help of Neo4j graph database [17] constructed domain knowledge graph and the influence degree of the project in the Java open-source project can be roughly judged by the in-degree and out-degree. According to the acquired project data, the attribute values of the project nodes in the graph are enriched.

3.2 Obtain the Influential Factors of Importance

In this paper, Java open-source projects are measured from three dimensions of project influence, project activity and, project popularity. The specific acquisition methods of each dimension are as follows.

3.2.1 Project Influence

Java open-source projects have situations where they depend on or are dependent on other projects. According to the degree centrality theory [18], the more neighbors a node has, the greater its influence will be. Based on this theory, the project relational network diagram is first constructed according to the dependencies between projects. In this network, each node represents a Java open-source project and each edge represents the dependency between two projects. Then, the influence degree of the project is calculated by using in-degree and out-degree of the project node. The specific formula is as follows:

$$EF(i) = \frac{K_i}{\mathrm{n}-1} \tag{1}$$

where, the degree of the project node i, denoted as k_i, refers to the number of nodes directly connected to the node i, $K_i = \sum_i a_{ij}$, In the adjacency matrix A, if two nodes are connected by edges, the value is 1; otherwise, the value is 0. a_{ij} represents the element of row i and column j in the adjacency matrix A. n represents the number of nodes in the network. The denominator $n-1$ represents the maximum possible value of the node.

3.2.2 Project activity

As an open-source community of social programming, GitHub provides various social mechanisms for project developers to interact, of which commit mechanism [19] is one. Commit is an essential operation in open-source project version control. Every commit can be considered as a modified node or a targeted operation. The number of commits submitted by the commit and the latest commit time of the project can reflect the active degree of the project.

This paper designed Eq. (2) and Eq. (3) to calculate project activity:

$$t_i = \frac{\Delta d}{30} months \ ; \ \mathrm{T}(t_i) = \begin{cases} 1, & t_i \le 2months \\ 0.8, & 2months < t_i \le 6months \\ 0.6, & 6months < t_i \le 12months \\ 0.4, & 12months < t_i \le 24months \\ 0.2, & t_i > 24months \end{cases} \tag{2}$$

$$AC(i, t_i) = \alpha 1 * nor\big(commits(i)\big) + \beta 1 * T(t_i) \tag{3}$$

where, t_i represents the time interval from the last commit time to now. $\mathrm{T}(t_i)$ represents the weight coefficient corresponding to the commit time interval. Through continuous tracking of open-source projects on GitHub, it is most reasonable to take the above time node as the failure cycle. $\mathrm{AC}(i, t_i)$ represents the activity of open-source project i,$commits\ (i)$ represents the number of commits for project i. In order to make the activity measurement more accurate, first normalized the commit number, then assigned different weights based on the impact of the commit number and the latest update time on project activity, and finally made linear combination.

3.2.3 Project Popularity

In GitHub open-source community, the mechanism of watch, fork, and star are provided. If the project developers are interested in an open-source project, they can use the mechanism of watch, fork, and star to track the project and develop it locally. This paper uses these three mechanisms to measure project popularity.

Watch: The most straightforward way to be interested in an open-source project is to focus on all the dynamics of the project. With the help of watch mechanism, users' notification center will receive the updated information of the project once the project that users follow changes, such as other project developers submitted a pull request, initiated issue comments and so on. The watch number of open-source projects can reflect the level of project developers' attention to the project.

Fork: Fork is the number of copies of the project. Using fork mechanism, users can make a copy of the project to their repository to facilitate bug repair or continue to optimize the project. The fork number of open-source projects can reflect the popularity of the project.

Star: Star can be colloquially translated as "like". The star mechanism gives users the ability to collect projects for later lookup. Users can use the personal center's "Your Stars" feature to view all items in their collections. The star number of open-source projects can reflect the degree of support from users.

Yang et al. [20] proposed a method to calculate the popularity of an open-source project based on watch number and fork number. In order to better quantify the popularity of an open-source project, this paper designed the following Eq. (4) to calculate the popularity of an open-source project by integrating the above three mechanisms of watch, fork, and star.

$$PP(i) = \alpha 2 * nor\big(watch(i)\big) + \beta 2 * nor\big(fork(i)\big) + \gamma 2 * nor\big(star(i)\big) \quad (4)$$

Among them, $\mathrm{pp}(i)$ indicates the popularity of open-source project i, $watch(i)$, $fork(i)$ and $star(i)$ represent the number of watches, forks, and stars corresponding to the open- source project i, respectively. In order to make the measurement more accurate, the data of watch, fork, and star were normalized at first. Secondly, due to the different influence degrees of the three mechanisms of watch, fork and, star on the open-source projects, they were given different weights respectively, and finally, the sum operation was conducted.

3.3 Importance Assessment Model Construction

The construction of the importance assessment model mainly includes two steps: Data preprocessing and model calculation. Firstly, ranked them according to the dependency complexity and public participation of different Java open-source projects, and then conduct feature extraction [21], which mainly includes project influence data, project activity data and project popularity data of the project itself. After that, this paper proposed an improved form of PageRank as the importance assessment method. By labeling the ranking data set, it trained the relationship between feature set and labeled value, optimized the parameters between features and constructed the importance assessment model.

PageRank algorithm is based on web link analysis to process keyword matching search results [22]. According to the basic idea of the PageRank algorithm for sorting web pages, the algorithm is improved. User access to open-source community project in behavior have random, and there is an interdependent relationship between open-source projects, but because the PageRank algorithm without considering the influence of dynamic information of sorting, led to the sort will be disturbed by historical data. Therefore, this paper introduced the feedback factor to solve this problem, and the feedback factor is a linear combination of project influence, project activity, and project popularity. The specific formula is as Eq. (5).

$$S(i) = \alpha 3 * EF(i) + \beta 3 * AC(i) + \gamma 3 * PP(i) \quad (5)$$

where, $\mathrm{S}(i)$ represents the feedback factor of open-source project i. $EF(i)$ represents the influence degree of project i. $AC(i)$ represents the activity of project i. $PP(i)$ represents the popularity of project i.

Improved PageRank algorithm. We combine the feedback factor with the PageRank algorithm, the evaluation of project importance can be carried out more accurately. Based on the above ideas, this paper designs Eq. (6) to improve the PageRank algorithm.

$$PR(i) = \frac{1-d}{C_{total}} + d\sum_{j=1}^{n}\frac{PR(T_j)}{N(T_j)+1} + S(i) \quad (6)$$

$PR(i)$ means the importance of level about open-source project i. $\mathrm{T}_j(\mathrm{j} = 1,2,\cdots,\mathrm{n})$ represents other items that depend on item i. d represents probability of random access to items for users, it is between 0 and 1. C_{total} represents the number of all project nodes. $N(T_j)$ is the number of projects which project T_j depends on.

$\frac{PR(T_j)}{N(T_j)+1}$ represents PR value given to i by the dependent item T_j. We assume that the initial PR value for each project is 1.

Above all, we can get a flowchart of the importance assessment model of the open-source community Java project based on the domain knowledge graph, it is shown in Fig. 2.

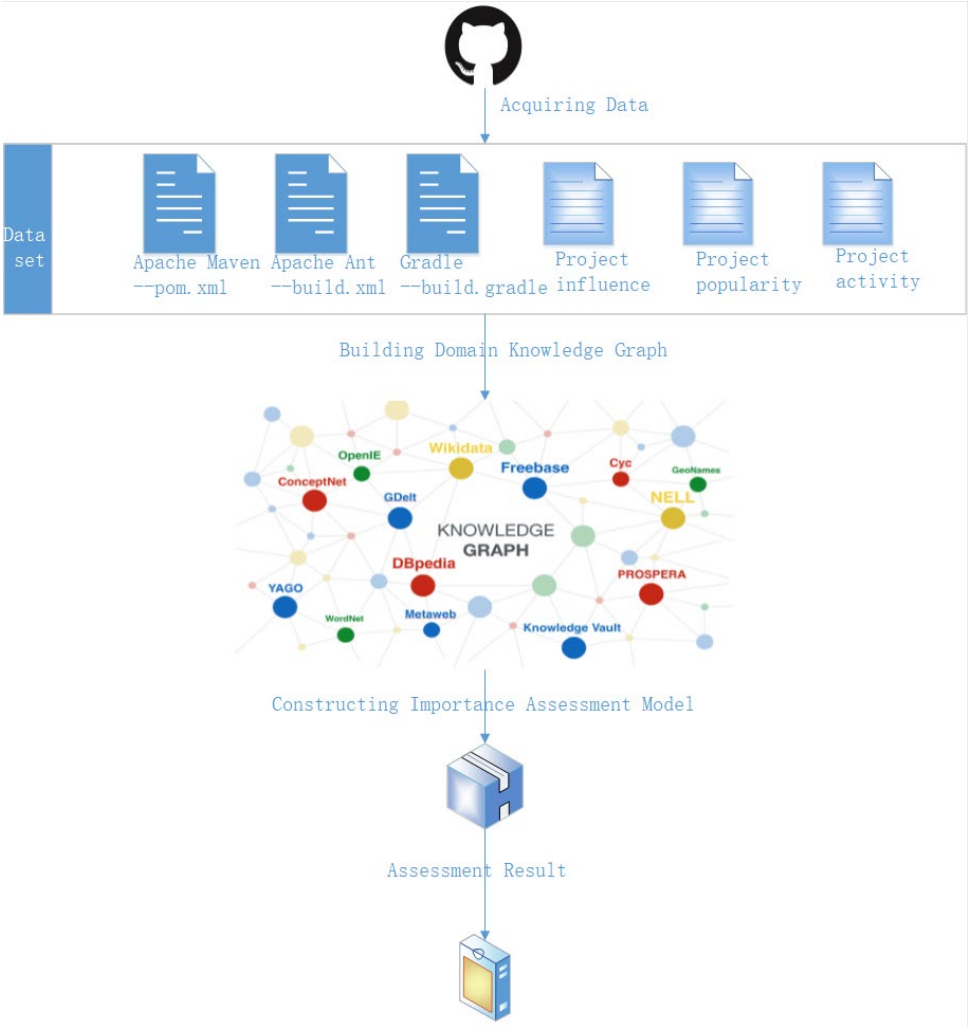

Figure 2: The flowchart of the importance assessment model

4 Experiment Result and Analyze

4.1 Dataset

This paper use python [23] to obtain information about Java open-source projects in GitHub open-source community to build a data set. In the experiment process, representative experimental samples were selected by removing repeated data, and we evaluate the model by analyzing and comparing these data.

When acquiring data, we get the data from GitHub, all Java projects developed with three project development tools: Apache Maven, Apache Ant, and Gradle. Then we analyze the configuration files of three kinds of project development tools: pom.xml, build.xml, and build.gradle. In the end, the unique identification of the project was used to obtain data on watch, fork, star, and commits on GitHub to complete the dataset. Through simple screening of acquisition projects (at least one non-empty project attribute was selected) as the experimental data set, and 4512 open-source projects were finally selected.

4.2 Knowledge Graph

The obtained Java open-source project is taken as the network node, the dependencies between projects as the edge, and the project attribute as the node attribute. Furthermore, the Neo4j graph database is used to build the knowledge graph, as shown in Fig. 3.

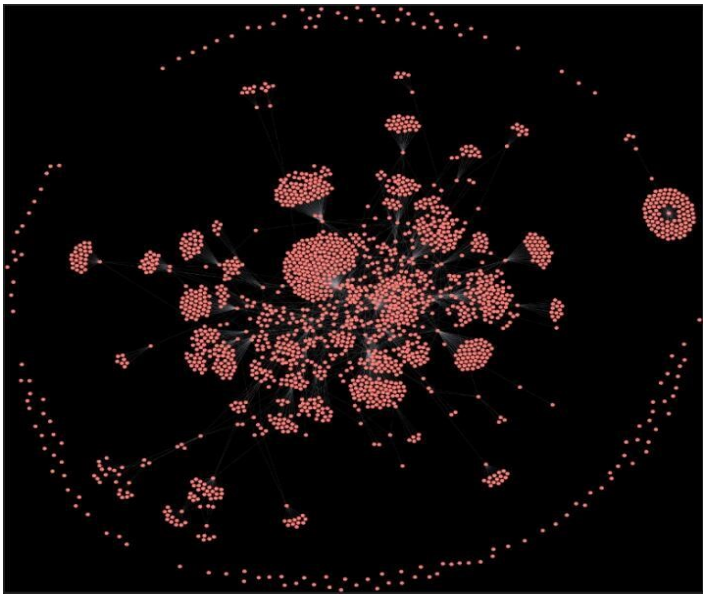

Figure 3: Domain knowledge graph

4.3 Result Analyzes

PageRank algorithm is iterative[24], and the improved PageRank importance assessment model in this paper still has the characteristic of iterative until finally reaching a steady state. We calculated for the first time according to the set initial PR value of 1 for each project and then iterated until the data is stable. The PR value initial calculation comparison is shown in Fig. 4.

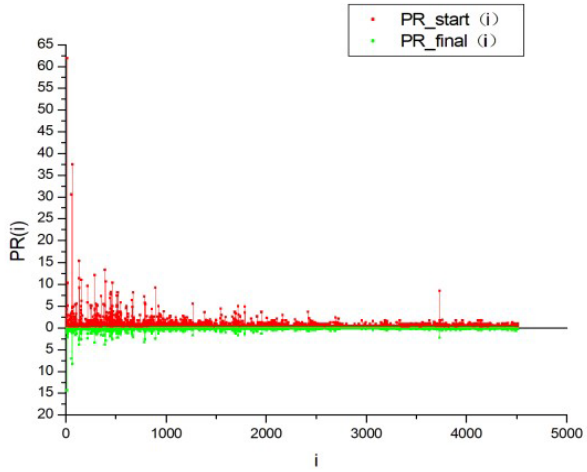

Figure 4: PR comparison graph

To ensure the accuracy of the assessment model, so we make the relationship between PR value and feedback factor $S(i)$ is shown in Fig. 5. The data in the graph is divided into two parts. In part 1, when $S(i) \in (0,0.007)$, all item attributes in the interval contain only the in-degree and out-degree, and all other attributes are empty; in this case, the project influence has a more significant correlation with the feedback factor $S(i)$ and the importance of level PR value. In part 2, when $S(i) = (0.007,0.6)$, all items within the interval contain multiple attributes; in this case, the project's importance assessment is affected by project influence, project activity, and project popularity. After multiple hyperparameter adjustments, the experimental results show that $\gamma 3 > \alpha 3 > \beta 3$, the model accuracy rate is higher, indicating that the project

popularity has the most significant impact on feedback factor S(i).

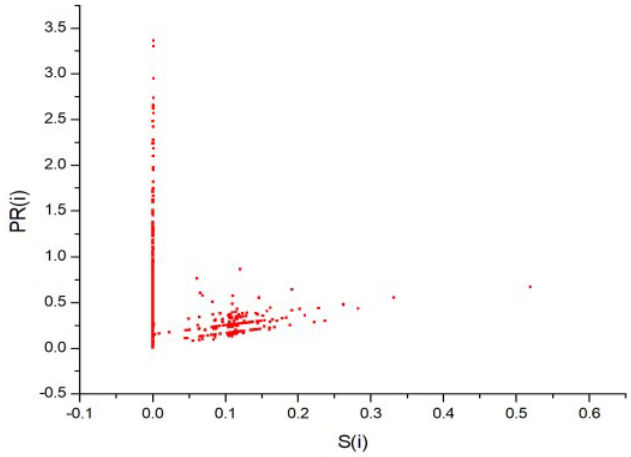

Figure 5: The relationship graph between feedback factor $S(i)$

The experimental data in the two intervals were analyzed; the experimental results show that the PR value increased with the increase of S(i). Top-5 data of the two intervals are shown in Tab. 1 and Tab. 2.

Table 1: The relationship between PR value and feedback factor in $\mathbf{S}(\boldsymbol{i}) \in (\mathbf{0}, \mathbf{0.007})$

$S(i)$	0.006057007	0.00368171	0.003087886	0.001603325	0.001543943
PR	14.3718608	8.303603224	7.098486797	3.828167775	3.788973489

Table 2: The relationship between PR value and feedback factor in $\mathbf{S}(\boldsymbol{i}) = (\mathbf{0.007}, \mathbf{0.6})$

$S(i)$	0.863770505	0.764199489	0.669487246	0.641412639	0.605614479
PR	0.519487246	0.1915298	0.120227621	0.065506568	0.060978557

When S(i) = (0.007,0.6), the TOP-1 project name is "jsqlparser", this project influence degree is 0.001039192, activity is 0.510770559, popularity is 0.044144581, all of which were high in the influence factor. Moreover, both the project which dependent and dependent projects have a high PR value, which confirms the correlation between hot projects.

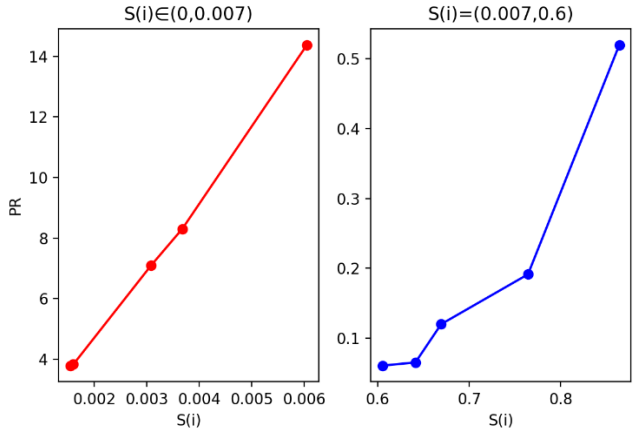

Figure 6: The Top-5 project data

It can be seen from the above figure that feedback factor $S(i)$ and importance of level PR value present a non-linear positive correlation. When project A can be linked to project B, it is considered that project B has obtained the value of project A's contribution to it. The value of this value depends on the importance assessment of project A itself; therefore, projects with high PR value are more likely to link to projects with high PR value.

From the above data, it can be found that the influence, activity, and popularity of a project significantly influence its importance assessment. Moreover, there are dependencies among hot projects. The importance assessment model designed in this paper can be used to measure Java open-source projects and also show a good effect.

5 Conclusion

In recent years, with the rise of open-source software, software and related technologies have significantly development. GitHub as a social development platform allows developers to create and browse the code, and it also provides community-based software development function. Users interact by watch, pull request and issue mechanisms, and they can fix bug, optimization and perfection for the open-source project in GitHub.

In this paper, we present an importance assessment model of open-source community Java projects based on the domain knowledge graph. This model makes full use of multiple associations between Java open-source projects, projects, and project developers; it combines attributes of multiple dimensions and improved the PageRank algorithm to evaluate the importance of open-source projects. Specifically, this article first constructs a domain knowledge graph through the dependencies between open-source projects and the attributes of related mechanisms in the GitHub open source community. Then integrate the three dimensions of the Java open source project's project influence, project activity, and project popularity to measure the project, and use the improved PageRank algorithm to establish an open-source project importance evaluation model. Finally, use the test data set to conduct an importance assessment experiment for a given project. The method proposed in this paper comprehensively considers the multi-dimensional factors that affect the project importance assessment and can evaluate the importance of open-source projects with high accuracy.

In the future, we will consider further work from other aspects. Now, the data we obtained are all real-time data; the life cycle of open-source projects is not taken into consideration. There will be misevaluation for the finished projects. Secondly, this paper only uses the activity records of users participating in the project, but does not delve into the specific contents of the project, such as technical dependence, technical parameters and so on. The next step is to explore the social connection between the project lifecycle and the project developers; then, we gradually improve the project importance assessment model.

Funding Statement: This work has been supported by the National Science Foundation of China Grant No. 61762092, "Dynamic multi-objective requirement optimization based on transfer learning," and the Open Foundation of the Key Laboratory in Software Engineering of Yunnan Province, Grant No. 2017SE204, "Research on extracting software feature models using transfer learning," and the National Science Foundation of China Grant No. 61762089, "The key research of high order tensor decomposition in a distributed environment".

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. A. Storey, C. Treude and A. van Deursen, "The impact of social media on software engineering practices and tools," in *FSE/SDP Workshop on Future of Software Engineering Research*, pp. 359-364, 2010.

[2] R. Winder and G. Roberts, "Java software development," *Computer CD Software and Applications*, 2008.

[3] J. J. Li, "Applying maven in developing enterprise java application," *Computer Knowledge and Technology*, vol. 7, no. 7, pp. 1562-1565, 2011.

[4] H. F. Dang, "Three key players in java web development: Eclipse + Tomcat + Ant integrated development," Beijing, China: Electronic Industry Press, 2008.

[5] Y. M. Wang, "Gradle applications in large Java projects," *Practical Electronics*, vol. 2, 2015.

[6] C. L. Jiang, W. B. Du, J. B. Li, "Economy papers map od co-occurrence analysis based on CSSCI," *Journal of Intelligent Manufacturing*, vol. 27, no. 9, PP. 78–80, 2008.

[7] H. M. Wang, G. Yin and B. Xie, "Research on network-based large-scale collaborative development and evolution of trustworthy software," *Scientia Sinica Informationis*, vol. 44, pp. 1–19, 2014.

[8] H. C. Qi, D. C. Huang and Y. F. Zheng, "An improved PageRank algorithm with time feedbacking," *Journal of Zhejiang University of Technology*, vol. 33, no. 3, pp. 272–275, 2005.

[9] W. Lin, "Distributed algorithms for fully personalized PageRank on large graphs," in *The World Wide Web Conf.,* San Francisco, United States, pp. 1084–1094, 2019.

[10] J. Q. Wang, F. Wang and J. Wang, "A new importance assessment method for risk-informed SSC categorization," *International Journal of Energy Research*, vol. 42, no. 4, pp. 1779–1786, 2018.

[11] G. Yin, T. Wang and H. Wang, "OSSEAN: Mining crowd wisdom in open source communities//service-oriented system engineering (SOSE)," in *2015 IEEE Sym. on IEEE*, Dublin, Ireland, pp. 367–371, 2015.

[12] Q. Liu, Y. Li and H. Duan, "Knowledge graph construction techniques," *Journal of Computer Research and Development*, vol. 53, no. 3, pp. 582–600, 2016.

[13] K. Zhang, P. P. Li and B. P. Zhu, "Evaluation method for node importance in directed-weighted complex networks based on PageRank," *Journal of Nanjing University of Aeronautics & Astronautics*, vol. 45, no. 3, pp. 429-434, 2013.

[14] F. U. Zambuk, A. Y. U. Gital and S. Boukary, "Evaluation of iterative PageRank algorithm for web page ranking," in *2019 4th Int. Conf. on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*, *IEEE*, Mysore, pp. 365-370, 2019.

[15] Z. D. Shen, M. U. Rehman, W. Y. Chen, Y. D. Liu, J. Liu *et al.,* "A Method based on modified PageRank-Algorithm for measuring and rating android malwares," *Procedia Computer Science*, pp. 252-255, 2020.

[16] Y. Zhang, "Quantitative evaluation algorithm for node importance of complex networks," *Computer Engineering*, vol. 37, no. 20, pp. 87–88, 2011.

[17] A. Vukotic, N. Watt and T. Abedrabbo, "Neo4j in Action," *Manning Publications Co*, 2014.

[18] X. L. Ren and L. Y. La, "Review of ranking node in complex networks," *Chinese Science Bulletin (Chinese Science Bulletin)*, vol. 13, pp. 1175–1195, 2014.

[19] E. Guzman and Y. Li, "Sentiment analysis of commit comments in GitHub: an empirical study," in *MSR 2014: Proc. of the 11th Working Conf. on Mining Software Repositories*, vol. 8, no. 399, pp. 352–355, 2014.

[20] C. Yang, Q. Fan, T. Wang, G. Yin, X. H. Zhang *et al.,* "RepoLike: A multi-feature based personal recommendation approach for opensource project," *Journal of Software*, vol. 28, no. 6, pp. 1357–1372, 2017.

[21] C. Y. Cui and J. Y. Shi, "Analysis of feature extraction in 3D model retrieval," *Journal of Computer-Aided Design & Computer Graphics*, vol. 16, no. 7, pp. 882–889, 2004.

[22] Z. Y. Li, W. Yang and Z. J. Xie, "Research on PageRank algorithm," *Computer Science*, pp. 185–188, 2011.

[23] J. H. Liu and Y. L. Lu, "Survey on topic-focused Web crawler," *Application Research of Computer*, vol. 24, no. 10, pp. 26–29, 2007.

[24] H. C. Qi, D. C. Huang and Y. F. Zheng, "An improved PageRank algorithm with time feedbacking," *Journal of Zhejiang University of Technology*, vol. 33, no. 3, pp. 272–275, 2005.

[25] W. Lin, "Distributed algorithms for fully personalized PageRank on large graphs," in *The World Wide Web Conf.,* San Francisco, United States, pp. 1084–1094, 2019.