Software Ranker: A New Comprehensive Software Ranking Approach

Yarong Zeng, Yue Yu, Xunhui Zhang, Yaozong Li, Tao Wang, Gang Yin, Huaimin Wang
National Laboratory for Parallel and Distributed Processing
National University of Defence Technology
Changsha, China
{zengyarong16, yuyue, zhangxunhui, liyaozong17, taowang2005, yingang, hmwang}@nudt.edu.cn

Abstract—**With the popularity of the open source, the open source community has accumulated a large number of open source project. While these massive projects provide developers with rich reusable resource, they also bring difficulties for users to choose the appropriate software. Therefore, it is very meaningful to rank the software and tell users which is better. In this paper, we propose a novel approach that ranks software based on a global perspective different from traditional software evaluation and ranking methods . We evaluate the software from four dimensions, namely community popularity, development activity, software health and team health. Each dimension contains some metrics. We demonstrate the effectiveness of our method through comparative experiments. This method has been integrated into to the OSSEAN platform to form a software leaderboard.**

Keywords-open source software; software evaluation; software ranking; multidimensional evaluation;

I. INTRODUCTION

Due to the characteristics of open communication, collaborative participation, rapid prototyping and transparency, Open source software has become the trend of software development nowadays. More and more developers are hosting projects in the open source communities, such as Github[1], Bitbucket[2]. As of May 2018, Github has accumulated 86 million open source projects. On the one hand, the massive number of open source projects provide developers with rich reusable resources, which are conducive to rapid iterative development and the creation of innovative software. On the other hand, due to the large number, and the uneven quality of these projects, it is difficult for users to choose the appropriate software. Therefore, it is meaningful to tell users which software is better, which can help them find the desired software and save time at the same time.

Researchers have been interested in evaluating software and have generated many evaluation models, such as CapGemini maturity model[1], Navica maturity model[2], OpenBRR[3], QSOS[4], SQO-OSS[5], etc. Most of these models took advantage of software production, including source code, documentation and the data from software development process, which place great emphasis on software itself. In addition, recent studies have found that user feedback has a significant impact on the effective evaluation of software quality, and claimed that it's meaningful and helpful to software and developers to take their adaptation and evolution decisions [6,7,8]. Some open source communities, such as Openhub[3], Sourceforge[4] also focus on crowd wisdom and attempt to rank the software through user feedback data accumulated by the platform. However, these evaluation methods are all very simple, which only focus on software production or user feedback. They measure the quality of software from a certain dimension and do not evaluate it from a global perspective. Moreover, for some new excellent projects, there may not be enough user feedback and software process data accumulated, which will lead to having less chance to get a fair evaluation.

In this paper, we propose a novel approach called *Software Ranker*, which ranks software based on a global perspective. We evaluate the software from four dimensions, each dimension contains some metrics, and finally integrates all the metrics to score the software. The main contributions of the study include:

1) We define four dimensions, each dimension contains some metrics, and we evaluate the software separately from different dimensions.

2) We proposed a new ranking approach and verified the effectiveness of the approach through experiments.

3) Our method has been integrated into OSSEAN[5] to form a software leaderboard.

II. RELATED WORK

The ranking of open source software is difficult due to the uniqueness of the software product in the development and the use environment. There are several established approaches to evaluate and rank software. Initially, researchers used development process data to evaluate software quality. Samoladas et al. constructed the SQO-OSS model to support an automated software evaluation system, the model mainly evaluates aspects of open source projects development, including the product (code) and the community[5]. Fagerholm et al. constructed a quality model suitable for use in a Free and

[1] https://github.com/
[2] https://bitbucket.org/
[3] https://www.openhub.net/
[4] https://sourceforge.net/
[5] http://ossean.trustie.net/

Open Source Software (FOSS) context, which includes both process and product quality metrics, and takes into account the tools and working methods commonly used in FOSS projects[9].

These traditional methods of software evaluation rely heavily on developers, and mostly use the data from the software development process to rank the software. However, software evaluation is a task that is performed to ensure that software meets its design objectives efficiently and correctly, and the stakeholders of software evaluation include users who utilize the software to reach their requirements[10]. Therefore, some researchers advocate that users' acceptance and views is a main subject of evaluation, and focus on ranking software through user feedback. Fan et al. connected open source software from different communities with user feedback data in StackOverflow[6], and rank open source software through using information of connected posts in StackOverflow[11]. They verified the effectiveness of the method by comparing ranking result with several influential ranking results. Pagano et al. found that developers need to analyze the gathered feedback and assess potential impact, which is mostly accomplished manually and consequently requires high effort[12]. Thus, they demonstrated the importance of tool support to consolidate, build, analyze, and track user feedback, and summarize developers' expectations towards tool support for user involvement.

There are also some researchers who do not adopt any specific perspective to software quality, but use domain knowledge and demand information to evaluate the software. Rosqvist et al. introduced a framework for software measurement and evaluation based on expert judgement, which supports the software developer and/or assessor to arrange the quality control of the software development process and the software product[13]. Liu et al. thought the software market demand is of great significance to the research of software evaluation, they proposed a innovative approach that ranking software based on the market requirements[14].

Our work is different from the methods described above. We aim to rank the software based on a global perspective, which means evaluate the software from multiple dimensions, such as software development process, popularity, user feedback, software activity, and so on.

III. APPROACH

Different from commercial software, open source software provides process data covering the entire life cycle for research and analysis. The rich software development process data and user feedback data can be obtained from the open source community. Thus, the evaluation of open source software is facilitated by the availability of the transparency of these software-related data.

Our study propose a new comprehensive software ranking approach called *Software Ranker*. We evaluate software from four dimensions to make full use of the development process data and user feedback data, including community popularity, development activity, software health and team health. Each dimension contains several metrics. First, we obtain an evaluation score by standardizing and averaging the several metrics score for each dimension. Then, we calculate the total score of software in four dimensions. Finally, we rank the software based on the evaluation result.

Next, we describe in detail the four dimensions and their corresponding metrics. And we give a specific calculation of the evaluation metrics for the software on Github. The *Norm* function in the formula represents a normalization operation, and the *Avg* function represents an averaging operation.

Community Popularity:

The open source communities can be divided into two groups: software production communities, such as Github, Sourceforge, OpenHub etc., and software consumption communities, such as StackOverflow, CSDN[7], etc.[15]. The former contains structured software artifacts and development process data, such as source code, pull request and issues, while the latter contains rich user feedback text documents. The popularity of software is a relatively intuitive measure of the software quality. Some open source communities have tried to rank software depending on popularity, for example, the Sourceforge rank software based on the number of downloaded, and the OpenHub also gives a ranking result based on the number of users of the software.

This dimension is used to measure the popularity of software in the above two types of communities. Specifically include two metrics, the popularity of the software in the production community (PSPC), and the popularity of the software in the consumer community (PSCC). For the latter metric, we use the number of software-related technical discussion posts to measure. And in the experiment of our study, we directly use the discussion post data in OSSEAN, which collected through building connection between production communities and consumption communities[15]. The specific calculations for PSPC and PSCC are respectively expressed in Equation 1, 2. The final score for this dimension (CPScore) is expressed as Equation 3. The Num_{fork} means the number of forks of the software. We can also use the number of stars of the software to measure the popularity of the project in reality. The $Num_{relatedPost}$ means the number of related discussion posts of the software.

$$PSPC = Norm(Num_{fork}) \quad (1)$$

$$PSCC = Norm(Num_{relatedPost}) \quad (2)$$

$$CPScore = Avg(PSPC + PSCC) \quad (3)$$

Development Activity:

Compared to some mature software, some new software do not accumulate enough historical development process data and user feedback. It is difficult to get a fair ranking result through traditional evaluation methods. In crowd-sourcing collaborative social programming communities, open source projects with continuous development activities should be promoted. For example, Github has a fork mechanism. The

[6] https://stackoverflow.com/

[7] https://www.csdn.net/

original project with the largest number of stars may not be the best, because the root fork is no longer being developed. In this case, we prefer to use a secondary fork which still has many development improvements. Therefore, we argue that whether the project has the latest development activities is very important for the effective evaluation of the software.

This evaluation dimension specifically includes two metrics: the number of recent code submissions (RCS), and the growth of recent development tasks (RDT). Note that what we count here is the amount of recent development activities rather than the total amount. In reality, we can extract development activities from software hosting community, for example, in Github, these two metric correspond to the increment of commits ($Num_{recentCommits}$) and the increment of pull requests ($Num_{recentPullRequest}$), and in this paper, we use the increment in the past three months. The specific calculations for RCS and RDT are respectively expressed as Equation 4, 5. The final score for this dimension (DAScore) is expressed as Equation 6.

$$RCS = Norm(\ Num_{recentCommits}\ ) \quad (4)$$

$$RDT = Norm(\ Num_{recentPullRquest}) \quad (5)$$

$$DAScore = Avg(\ RCS\ + RDT\ ) \quad (6)$$

Software Health:

Software health is the most essential factor in evaluating software. The software product with quality defects must not be placed on the market. This dimension includes three metrics: defect resolution rate (DSR), defect repair ratio (DRR), and static code quality (SCQ). The defect resolution rate metric is mainly used to determine whether the submitted defect can be quickly responded by the project manager. The faster the resolution rate, the healthier the project is. The defect repair ratio metric refers to the ratio of defects that are repaired. A higher repair ratio indicates that the project is healthier because the software is well maintained by developers. Equations 7, 8 give the specific calculations of the DSR and DRR of the software, The *n* in *DSR* indicates the number of issues of the software. For static code quality, there are several open source tools that try to analysis static code quality of a software by examining several aspects of it, such as PMD[8] , Findbugs[9] , SonarQube [10]. In this study, we choose SonarQube as our source code analyzer. Because it can detect code quality from multiple dimensions and support multiple programming languages through the plug-ins form. We map the analysis results (5 levels) of SonarQube to a [1-50] score with intervals of 10, and use this score as SCQ score. The final score for this dimension (SHScore) is expressed as Equation 9.

$$DRR = Norm(\frac{Num_{issue\_closed}}{Num_{issue\_total}}) \quad (7)$$

$$DSR = Norm(\frac{1}{Avg(\sum_{i=1}^{n}(issueCloseTime_i - issueOpenTime_i))}) \quad (8)$$

$$SHScore = Avg(\ DRR + DSR + SCQ\ ) \quad (9)$$

[8] https://pmd.github.io/
[9] http://findbugs.sourceforge.net/
[10] https://www.sonarqube.org/

Team Health:

The behavior of developers is also an important factor affecting software quality. Many recent studies[16,17,18] have also examined the effects of social interaction and ownership of developer on software quality. As a result of the popularity of crowd-sourcing collaborative development models, the relationship between the contributors of open source software is increasingly complex and often changes dynamically (the contributor of software means the participant who submitted the code). We believe that the dynamic changes of contributors will have an impact on the software quality.

This dimension specifically includes two metrics: continuous contribution rate (CCR), growth contribution rate (GCR). Assuming that the contributors in the last month forms a set $Contri_{cur}$, and in the current month forms a set $Contri_{cur}$, then the continuous contribution rate and the growth contribution rate are expressed as Equation 10, 11. The final score for this dimension (THScore) is expressed as Equation 12.

$$CCR = Norm(\frac{Contri_{pre} \cap Contri_{cur}}{Contri_{pre}}) \quad (10)$$

$$GCR = Norm(\frac{Contri_{cur} - (Contri_{pre} \cap Contri_{cur})}{Contri_{cur}}) \quad (11)$$

$$THScore = Avg(\ CCR + GCR\ ) \quad (12)$$

The total evaluation score of software is shown in Equation 13, which combine the evaluation results in four dimensions.

$$TotalScore = Avg(CPScore + DAScore + SHScore + THScore) \quad (13)$$

IV. EXPERIMENT

A. Experiment Setup

In order to validate the effectiveness of our approach, we choose some categories of software, and ranking them by using our approach. Then, we use the metrics of the search ranking algorithm to compare our ranking results with some authoritative ranking websites.

We use software of database management system (DBMS) and deep learning tools in Github to verify the effectiveness of our method. Here we choose the GHTorrent[11] dump released on January 2018. About how to select a certain category of software, we use the search matching algorithm in OSSEAN which through identifying the relevance between category keywords and software keywords (software titles, tags and description information).

1) *DBMS:* Nowadays, the DBMS has become an indispensable part of online platform and daily application, some relational and NoSQL databases are widely used, such as MySQL, PostgreSQL and MongoDB. The DB-Engines[12] website is a knowledge base of relational and NoSQL DBMS. This website provides professional ranking results of DBMS software which has been referenced by some agencies and studies[11]. In this paper, we regard it as a standard ranking result and compare it

[11] http://ghtorrent.org/downloads.html
[12] https://db-engines.com/en/

with our ranking result. In order to maintain data consistency, we also take the ranking result of DBEngines in January.

2) *Deep Learning tools:* Deep learning is beginning to change our work and life. It can be applied to various fields such as automatic driving, language translation, traffic prediction and so on. Many deep learning tools are widely used by companies and researchers to quickly implement algorithms. Recently, the famous data science website KDnuggets[13] released the investigation results of the use of deep learning tools in 2018. In this experiment, we use this result as a standard ranking result to verify our ranking results.

B. Evaluation Metrics

To evaluate the ranking results, some measures of search ranking algorithms are commonly used, such as *MAP*(mean average precision), *MRR* (mean reciprocal rank) and *NDCG* (Normalized Discounted Cumulative Gain). In this paper, in order to get an overview of the global quality of the ranking result, we use *NDCG* to evaluate the ranking result, which takes into account the effects of ranking positions. Equation 15 describes the formula to derive *NDCG*.

First calculate the *DCG* (Discounted Cumulative Gain) as Equation 14 to evaluate the current ranking results, where *i* represents the index position in the ranking list, *p* denotes the length of the ranking list, and rel_i denotes the relevant grading score of the *i*-th result. Then normalize the evaluation score, the *IDCG* represents ideal *DCG*. For *NDCG*, the higher the value is, the better the ranking results.

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i}-1}{log_2(i+1)} \qquad (14)$$

$$nDCG_p = \frac{DCG_p}{IDCG_p} \qquad (15)$$

V. RESULTS AND DISCUSSIONS

1) DBMS: We use the ranking results of DBMS from DBEngines as a standard ranking result and compare them with our ranking result. The result is shown in Table I. We can find that the top 10 DBMS software on DBEngines is also in our ranking list except InfluxDB and CouchDB. In order to evaluate our ranking results more intuitively, we compare our result with Google Trends. The result in Fig.1 shows the NDCG scores of our method and Google Trends. We can find the ranking result using our method outperformed Google trend, and is comparable to the standard rankings on DBEngines.

2) Deep Learning tools: All ranking result of deep learning tools are shown in Table II. As expected, we can find our rank is similarity to KDnuggets. The top five software in our ranking results, except for PyTorch, other software rankings are consistent with KDnuggets, and besides Apache MXnet and TFLearn, all popular deep learning tools in KDnuggets can be found in top 10 of our rank. Fig.2 shows the NDCG scores of different method of deep learning tools. From this comparison, we confirm that our method can effectively evaluate the software and give a reasonable ranking result.

TABLE I. THE RANKING RESULTS OF DBMS SOFTWARE

DBMS software	DB-Engines	Our Rank	Google Trends
Redis	1	1	6
Cassandra	2	3	7
Elasticsearch	3	2	5
Solr	4	6	2
Hbase	5	5	10
Hive	6	4	9
Neo4j	7	7	3
Memcached	8	10	1
CouchDB	9	16	4
InfluxDB	10	19	8

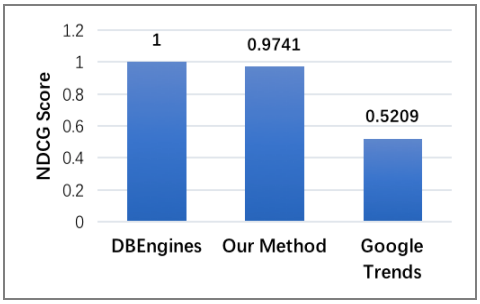

Figure 1. The NDCG score of different method of DBMS

Through the above two experiments, we have proved the effectiveness of our method. Moreover, Our method has been encapsulated into an API , which is integrated into the OSSEAN platform. Because there is no comparability between different types of software, we rank the software separately according to the type of software in OSSEAN. The platform not only shows the overall score of the software, but also shows the score of the software in four dimensions and metrics. Compared to evaluating the software from a global perspective, some people may pay more attention to a certain dimension. For example, some people are very concerned about software health. In this case, you can refer to the ranking result of software in a certain dimension. In addition, since the software has been dynamically updated over time, the leaderboards in OSSEAN will be reanalyzed and evaluated monthly to maintain synchronization with the state of the software.

VI. CONCLUSIONS

This paper proposes a new approach to measure software from multiple dimensions. We evaluate two categories of

[13] https://www.kdnuggets.com/2018/05/new-poll-software-analytics-data-mining-data-science-machine-learning.html

software in experiment, namely DBMS and deep learning tools, and verify the effectiveness of our approach through comparing our ranking result with two professional website. In addition, our approach has been applied to practice and provide services normally. For the massive amounts of open source software, we believe that our approach can provide users with some additional information and help them select the appropriate software.

TABLE II. THE RANKING RESULTS OF DEEP LEARNING TOOLS

Deep Learning tools	*KDnuggets*	Our Rank	Google Trends
Tensorflow	1	1	3
Keras	2	2	2
PyTorch	3	6	7
Theano	4	4	5
DeepLearning4J	5	5	10
CNTK	6	3	6
Apache MXnet	7	15	11
Caffe	8	9	1
Caffe2	9	8	8
TFLearn	10	19	9
Torch	11	7	4

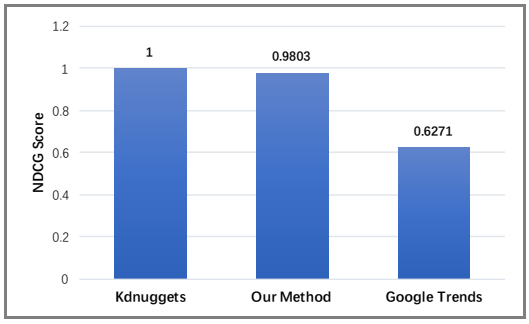

Figure 2. The NDCG score of different method of deep learning tools

ACKNOWLEDGMENT

The research is supported by the National Natural Science Foundation of China (Grant No.61432020) and National Grand R&D Plan (Grant No. 2016-YFB1000805).

REFERENCES

[1] Duijnhouwer F. W., Widdows C. Open Source Maturity Model[EB/OL], http://www.seriouslyopen.org, 2015-07-10.

[2] Bernard Golden. Succeeding with Open Source [M], Addison-Wesley Information Technology Series, 2005.

[3] OpenBrr, Business Readiness Rating for Open Source [EB/OL], http://www.openbrr.org, 2015-07-10.

[4] Origin, Atos. "Method for Qualification and Selection of Open Source software (QSOS), version 1.6." *Disponible en Internet: http://www. qsos. org/download/qsos-1.6-en. pdf* (2006).

[5] Samoladas, Ioannis, et al. "The SQO-OSS quality model: measurement based open source software evaluation." *IFIP International Conference on Open Source Systems*. Springer, Boston, MA, 2008.

[6] Atoum, Issa, and Chih How Bong. "A framework to predict software "quality in use" from software reviews." *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*. Springer, Singapore, 2014.

[7] Pagano, Dennis, and Bernd Brügge. "User involvement in software evolution practice: a case study." *Software Engineering (ICSE), 2013 35th International Conference on*. IEEE, 2013.

[8] Ali, Raian, et al. "Social adaptation: when software gives users a voice." (2012).

[9] Fagerholm, Fabian. "Measuring and tracking quality factors in Free and Open Source Software projects." (2007).

[10] Sherief, Nada. "Software evaluation via users' feedback at runtime." Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. ACM, 2014.

[11] Fan, Qiang, et al. "Ranking open source software based on crowd wisdom." Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference on. IEEE, 2015.

[12] Pagano, Dennis, and Bernd Brügge. "User involvement in software evolution practice: a case study." *Software Engineering (ICSE), 2013 35th International Conference on*. IEEE, 2013.

[13] Rosqvist, Tony, Mika Koskela, and Hannu Harju. "Software quality evaluation based on expert judgement." *Software Quality Journal* 11.1 (2003): 39-55.

[14] Liu, Bingxun, et al. "Software Ranking and Analysis based on Mining Market Requirements and Characteristics." *Proceedings of the 7th Asia-Pacific Symposium on Internetware*. ACM, 2015.

[15] Yin, Gang, et al. "OSSEAN: Mining crowd wisdom in open source communities." *Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium on*. IEEE, 2015.

[16] Bettenburg, Nicolas, and Ahmed E. Hassan. "Studying the impact of social interactions on software quality." *Empirical Software Engineering* 18.2 (2013): 375-431.

[17] Bird, Christian, et al. "Don't touch my code!: examining the effects of ownership on software quality." Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering. ACM, 2011.

[18] Nagappan, Nachiappan, Brendan Murphy, and Victor Basili. "The influence of organizational structure on software quality." *Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on*. IEEE, 2008.