Adaptive software search toward users' customized requirements in GitHub

Jinze Liu, Zhixing Li, Tao Wang, Yue Yu and Gang Yin
College of Computer Science
National University of Defense Technology
Changsha, Hunan, China
jinze_liu@qq.com, {lizhixing,taowang2005,yueyu,yingang}@nudt.edu.cn

***Abstract*—Because of a tremendous growth of Open Source Software (OSS) scale and the diversity of users' requirements, users now face the problem of finding OSS that meets their expectations in a huge number of OSS resources. However, current GitHub-provided search service has a shortage in adapting to user needs. When facing diverse users' requirements, it cannot always return satisfactory results. In this paper, we provide a more efficient search service for OSS on GitHub. We first design a multi-dimensional measurement model for OSS, which forms a corresponding metric system and quantitative measurement method. Then we propose a ranking algorithm based on fuzzy synthetic evaluation in order to implement an adaptive metric ranking method that is oriented to user requirements. We verify that our work is useful by setting up experiments. The experiment results show that compared with GitHub-provided search service (searching by "Best Match" & searching by "Most Stars"), the effectiveness of our method improved by 97.6% and 13.8% respectively, which means our method returns search results which meet users' expectations more, and has high self-adaptive ability.**

Keywords-Open Source Software; Search; Multi-dimensional Measurement; Fuzzy Synthetic Evaluation; GitHub

I. INTRODUCTION

With the rise of the open source movement, OSS has made tremendous growth and the global OSS resources have become an Internet-scale repository. Especially in recent years, the rapid development of Internet technology has greatly enhanced the influence of OSS around the world. Many well-known OSS hosting platforms have emerged and the most successful platform among them is GitHub. GitHub is a social programming and code hosting platform [1], officially launched on April 10, 2008, after which countless open source projects began to migrate to this platform, and the number of hosted software showed the situation of "explosive" growth. As of December 2016, GitHub has hosted more than 35 million projects and attracted more than 14 million developers to participate in open source activities [2].

OSS resources are already a huge and diverse ecosystem. At the same time, the users' requirements of OSS is also diverse. For example, some users like the popular OSS currently. Some of them are more concerned about whether the authors of OSS have high development enthusiasm and make good maintenance of their work. And others may look for open-source software that still maintains high activity. Therefore, users now face the problem of finding OSS that suits their diverse requirements in a huge number of OSS resources.

In a web search, as stated by studies [3] [4], people often prefer results located on the first page, or at most, the first three pages. Later search results people tend to no longer be concerned about, so a good search service needs to make more important results come in front. For this reason, GitHub provides ranking indicators based on the text matching, popularity, etc. of OSS, but the results are not satisfactory. For example, if the text matching degree is used as a ranking indicator, GitHub only considers the matching between the software name or description, and the users' search keywords, thus the quality of the software is not guaranteed. Due to the low threshold of creating OSS in open source community and the different ability of authors of OSS, there are a large number of OSS with low quality in GitHub. So if the search service does not consider the quality of OSS, the search results will have very limited help to users. If we choose popularity as the ranking indicator, GitHub ranks only by the number of "Star" of the repository of OSS. Other attributes of OSS are not taken into account at all, which makes the search results still less helpful.

Although GitHub provides an advanced search service, allowing users to propose multiple search criteria, from the point of actual use, GitHub simply combines multiple search criteria together. For example, if users choose both text matching degree and popularity as ranking indicators, the final search results are only based on the search results of text matching degree, and delete the results which popularity do not meet the requirements. So if what users want is being more interested in the software which is more popular among all the related software, the search results provided by GitHub still cannot give them a satisfied answer. All in all, GitHub currently offers a weak search service. On the one hand, it provides users with less choice of ranking indicators. On the other hand, it cannot adapt to complex users' requirements. When users select an indicator to rank, other indicators will be ignored, and this often does not meet users' expectations.

Therefore, we provide a more efficient search service for OSS on GitHub by a multi-dimensional measurement model and a ranking algorithm based on fuzzy synthetic evaluation. In this paper, we first build a GitHub OSS information database that contains information about the attributes of OSS

DOI reference number: 10.18293/SEKE2018-064

such as name, description, number of "Watch", situation of "Pull Request" (PR), and so on. Second, we get preliminary search results by keywords matching. Third, we extract ranking indicators and design a multi-dimensional measurement model according to users' requirements. Fourth, we propose and implement a ranking algorithm based on fuzzy synthetic evaluation. This algorithm calculates the synthetic evaluation score of OSS according to the weight of OSS attributes, so that we can get final search results from the order of evaluation scores. At last, by setting up experiments with different users' customized requirements scenarios, we compare our search results with those of GitHub-provided search service and verify that our method returns search results which meet users' expectations more, and has high self-adaptive ability.

The key contributions of this study include the following:

- We analyze the key factors when users choose OSS from four dimensions: popularity of software, collaborative development of software, development attitude of software author and activeness of software, and propose a detailed quantitative measurement method.
- We allow users to set their personal search requirements and provide users with enhanced, easier-to-use search service.
- We propose and implement an OSS ranking algorithm based on fuzzy synthetic evaluation, so that the OSS can obtain the corresponding evaluation score according to different users' customized requirements so as to make the search results more in line with users' expectations, greatly improving the self-adaptive ability of the search service.

The rest of this paper is organized as follows. Section II introduces related research on software ranking and fuzzy mathematics theory. Section III elucidates the approach of our study. Section IV elaborates our experimental process and results. Section V concludes this paper and introduces future work.

II. Related Work

A. Software Ranking

In the field of software engineering, researchers evaluate software quality through software evaluation models, and then rank the software based on software quality. Researchers have proposed Capgemini maturity model [5], Navica maturity model [6], OpenBRR model [7], QSOS model [8], and SQO-OSS model [9] and so on.

In the field of open source community, open source communities typically use the feedback from community users to rank the OSS. For example, GitHub provides the function of ranking OSS according to the "Most Star" indicator. The higher the number of "Star" of a software, the more popular it is in the community. SourceForge uses the information which is collected from users about the download and comment situation of OSS to calculate the popularity of them, and then rank the software. OpenHub has designed a button named "I use it" for each OSS, which users can click to mark this software was used. Then the platform ranks by the number of users of them.

However, these ranking methods only consider a single ranking indicator, not comprehensive enough to meet more complex users' customized requirements.

B. Fuzzy Synthetic Evaluation

Fuzzy synthetic evaluation is one of the most widely used methods for multi-index synthetic evaluation [10]. With the help of the theory of membership degree of fuzzy mathematics, it turns qualitative evaluation into quantitative evaluation, which can solve the problem of fuzzy and hard to quantify. The basic steps of fuzzy synthetic evaluation shows in Table I [11].

TABLE I: Basic Steps of Fuzzy Synthetic Evaluation

1. Determine the evaluation indicator set and evaluation level set of the evaluation target.
2. Determine the weight of each evaluation indicator and its membership vector in order to get the fuzzy evaluation matrix.
3. Make fuzzy operation between the fuzzy evaluation matrix and the weight set of evaluation indicators.
4. Normalize the calculation results in step 3 and get the evaluation results.

III. Approach

The goal of our work is providing users with OSS search service which has high self-adaptive ability. As shown in Figure 1, we first get development history data of OSS from GitHub to build an information database. Then we get preliminary search results by keywords matching. Third, we extract ranking indicators and design a multi-dimensional measurement model according to users' requirements. Fourth, we get final search results by a ranking algorithm based on fuzzy synthetic evaluation. In the following sections, we will elaborate each step in detail.

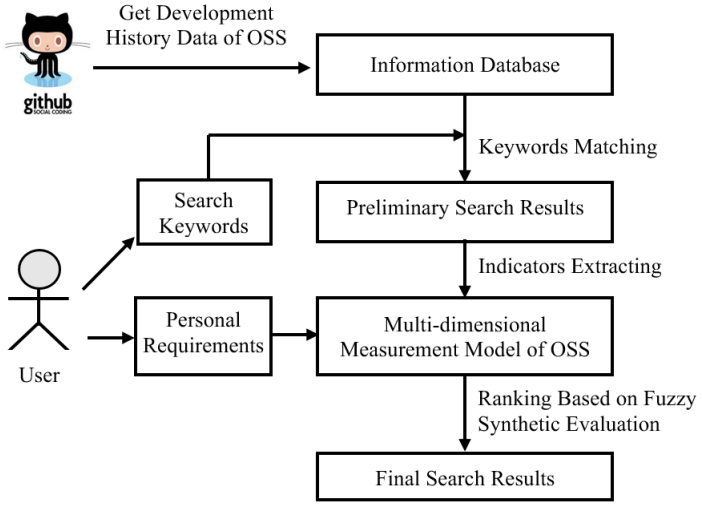

Fig. 1: Overall Framework of Our Method

A. Database Building

Dataset collection. In this paper, we use the "GHTorrent" project, which monitors the GitHub public event timeline, to get GitHub open source data [2]. Whenever an event occurs on GitHub, the project retrieves its content and all its dependencies and stores them, then releases the data regularly for users to download.

Dataset cleaning. GitHub allows users to "Fork" the origin OSS project to create a new branch based on the main branch of the project. The main branch and the "Fork" branch actually refer to the same project. However, there are multiple records stored in the database. Therefore, in order to avoid duplication of processing, all "Fork" branches need to be removed and the corresponding main branch should be reserved.

B. Preliminary Search Based on Keywords Matching

In this section, we get all OSS related to users' search keywords as preliminary search results by matching keywords with the name and description of software. First, we use WordNet to make synonym expansion of keywords in order to improve the recall rate. Second, we set whether software name or description contains keywords as the standard of whether software is related to keywords.

C. Multi-dimensional Measurement Model for OSS

By collecting lots of posts of IT Q & A communities (such as Stack Overflow etc.) and analyzing the community users' comments on their needs for OSS, a multi-dimensional measurement model is presented for OSS. As shown in Figure 2, this model includes four dimensions, and each dimension has one or more indicators. Each indicator is described by one or more online attributes through the Github open source project repository.

Popularity of software. In general, the software of higher popularity degree often means more famous and has higher quality. So an indicator named *software popularity degree* is introduced to describe how popular is an OSS.

In GitHub, if developers in the community are interested in an OSS project, their first choice is to "Watch" the software. Then, whenever the software has any dynamic information, developers can receive the corresponding data at the first time, which like following celebrities interested in Facebook.

Therefore, the number of "Watch" of an OSS can describe its *software popularity degree*, the higher the number of "Watch", on behalf of the developers concern more about this software. This indicator can be calculated as Equation 1.

$$P_{popularity}(j) = Nor(watch(j)) \tag{1}$$

where $Nor()$ is normalization and can be calculated as bellow:

$$Nor(watch(j)) = \frac{watch(j) - \min(watch)}{\max(watch) - \min(watch)} \tag{2}$$

Collaborative development of software. GitHub is an open source collaborative development community. For the software the developers are interested in, developers can work with the related authors. Developers who involved in collaborative development are called as contributors to this OSS. So an indicator named *software coordination degree* is introduced to describe the enthusiasm of contributors to participate in the collaborative development of an OSS project.

In GitHub, the development mechanism based on "Fork→Pull Request" is widely used [12]. If developers want to participate in the collaborative development of a software, they can clone the main branch of this software to developers' local branch through "Fork". After that, they can implement some new features or fix bugs based on their personal repository cloned from the latest version of project repository. When their work is finished, the patches are packaged as a PR submitted to GitHub and reviewed by software authors, which indicates that developers request that their work be merged into the main branch of the original project.

Therefore, the number of PR of an OSS can describe its *software coordination degree*, the higher the number of PR, on behalf of the more requests are sent and the higher enthusiasm of contributors to participate in the collaborative development. This indicator can be calculated as Equation 3.

$$P_{coordination}(j) = Nor(pull\_request(j)) \tag{3}$$

Development attitude of software author. When developers want to participate in a collaborative development of OSS, they do not prefer those OSS developed by authors who are indifferent to their work. When they encounter problems during the development, the authors will not actively communicate with them and solve the problem, thus affecting the user experience. This dimension describes how positive is an author in his work.

The number of submitted PR per day on GitHub is enormous, leading to the update iterative efficiency of OSS largely dependents on whether PR can be reviewed in a timely manner. Study [13] showed that the review of PR is a very important way for distributed software development community like GitHub to maintain the quality of code of OSS.

Therefore, two indicators including response degree of PR ($P_{response}$) and response speed of PR ($S_{response}$) are introduced, which can be calculated as Equation 4.

$$\begin{aligned} P_{response}(j) &= Nor\left(\frac{pull\_request_{response}(j)}{pull\_request(j)}\right) \\ S_{response}(j) &= Nor\left(\frac{1}{T_{latency}(j)}\right) \\ &= Nor\left(\frac{1}{\underset{1\leqslant i\leqslant M}{Mid}\left(T^i_{closed}(j) - T^i_{opened}(j)\right)}\right) \end{aligned} \tag{4}$$

where $P_{response}$ is the ratio of the number of responded PR to the total number of PR. $S_{response}$ is the reciprocal of the medium number of PR evaluation latency, which is the time difference between the PR opened and closed.

Activeness of software. The less active OSS often means that its author has not updated it for a long time and has even abandoned its development. So we use the reciprocal of the

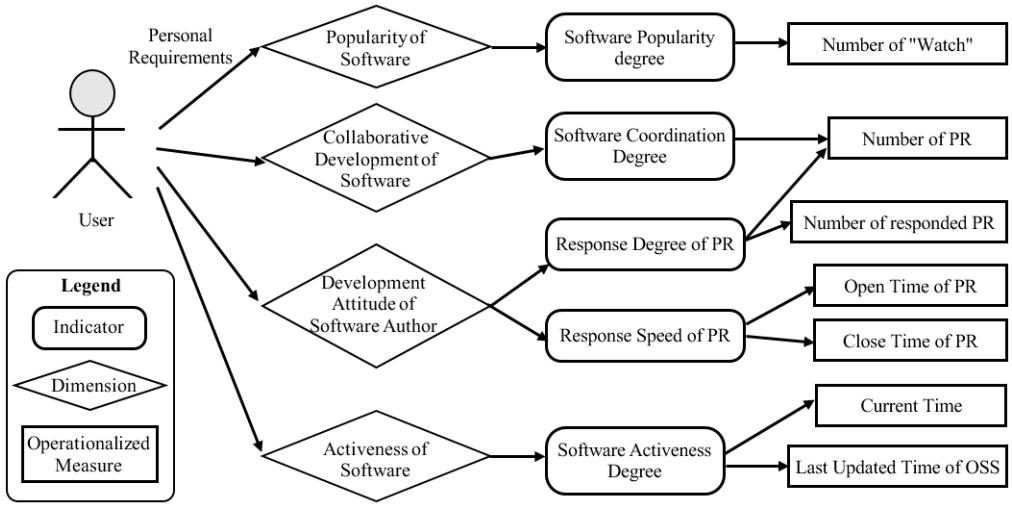

Fig. 2: Multi-dimensional Measurement Model for OSS

time difference between current time and the last updated time of OSS to describe an indicator named *software activeness degree*, which can be calculated as Equation 5.

$$P_{activeness}(j) = Nor\left(\frac{1}{T_{current} - T_{updated}(j)}\right) \quad (5)$$

For those dimensions above, users can choose one or more of them and set the corresponding importance order. For example, if users want to find OSS in great quality, they would choose *popularity of software* as the first important dimension. Then we determine the weight of dimensions according to their requirements. After many experiments, we select a set of weight values that have the better search performance, and the results shows in Table II.

TABLE II: Weight of Dimensions

	The order of importance			
Number of dimensions	First	Second	Third	Fourth
4	0.4	0.3	0.2	0.1
3	0.5	0.3	0.2	
2	0.7	0.3		
1	1.0			

As mentioned before, there are two indicators (*response degree of PR* and *response speed of PR*) to describe the dimension: *development attitude of software author*. So the weight of both indicators will be the half of the weight of this dimension.

D. Ranking Based on Fuzzy Synthetic Evaluation

We build a fuzzy synthetic evaluation model of OSS to rank the preliminary search results by using evaluation score of software.

Evaluation indicator set and evaluation level set. In this paper, the evaluation indicator set U is:

$$U = \{u_1, u_2, u_3, u_4, u_5\} \quad (6)$$

where u_1 represents *software popularity degree*, u_2 represents *software coordination degree*, u_3 represents *response degree of PR*, u_4 represents *response speed of PR* and u_5 represents *software activeness degree*.

We choose "excellent", "good", "general", "bad" and "awful" as the evaluation level set V:

$$V = \{v_1, v_2, v_3, v_4, v_5\} \quad (7)$$

where v_1, v_2, v_3, v_4, v_5 represents "excellent", "good", "general", "bad" and "awful" respectively.

Membership function. We use the triangle function as membership function in our evaluation model. $\mu_{v_1}, \mu_{v_2}, \mu_{v_3}, \mu_{v_4}, \mu_{v_5}$ is respectively the membership function of "excellent", "good", "general", "bad" and "awful".

$$\mu_{v_1}(x) = \begin{cases} 4x-3 & x \in [0.75,1] \\ 0 & others \end{cases}$$

$$\mu_{v_2}(x) = \begin{cases} 4x-2 & x \in [0.5,0.75] \\ -4x+4 & x \in (0.75,1] \\ 0 & others \end{cases}$$

$$\mu_{v_3}(x) = \begin{cases} 4x-1 & x \in [0.25,0.5] \\ -4x+3 & x \in (0.5,0.75] \\ 0 & others \end{cases} \quad (8)$$

$$\mu_{v_4}(x) = \begin{cases} 4x & x \in [0,0.25] \\ -4x+2 & x \in (0.25,5] \\ 0 & others \end{cases}$$

$$\mu_{v_5}(x) = \begin{cases} -4x+1 & x \in [0,0.25] \\ 0 & others \end{cases}$$

Fuzzy evaluation matrix. For each indicator, we get its single factor evaluation according to the membership function.

Then we get the fuzzy evaluation matrix bellow:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} \\ r_{21} & r_{22} & r_{23} & r_{24} & r_{25} \\ r_{31} & r_{32} & r_{33} & r_{34} & r_{35} \\ r_{41} & r_{42} & r_{43} & r_{44} & r_{45} \\ r_{51} & r_{52} & r_{53} & r_{54} & r_{55} \end{bmatrix} \tag{9}$$

where $\boldsymbol{r_i} = (r_{i1}, r_{i2}, r_{i3}, r_{i4}, r_{i5})$ represents the single factor evaluation of u_i.

Evaluation indicator weight set. In this paper, the evaluation indicator weight set W is:

$$W = \{w_1, w_2, w_3, w_4, w_5\} \tag{10}$$

where w_i is the weight of indicator u_i.

Evaluation model and evaluation score. We get the fuzzy evaluation set S bellow:

$$\begin{aligned} S &= W * R \\ &= (w_1, w_2, w_3, w_4, w_5) * \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} \\ r_{21} & r_{22} & r_{23} & r_{24} & r_{25} \\ r_{31} & r_{32} & r_{33} & r_{34} & r_{35} \\ r_{41} & r_{42} & r_{43} & r_{44} & r_{45} \\ r_{51} & r_{52} & r_{53} & r_{54} & r_{55} \end{bmatrix} \\ &= (s_1, s_2, s_3, s_4, s_5) \end{aligned} \tag{11}$$

where $*$ is fuzzy synthesis operator and there are four common operators: $M(\wedge, \vee)$, $M(\bullet, \vee)$, $M(\wedge, +)$ and $M(\bullet, +)$.In this paper, we use $M(\bullet, +)$: $s_j = \sum_{i=1}^{5} w_i \bullet r_{ij}$

Finally we get the evaluation score bellow:

$$Score_k = \frac{\sum_{i=1}^{n} c(v_i) \bullet s_i^k}{\sum_{i=1}^{n} s_i^k} \tag{12}$$

where $c(v_i)$ is the quantified value of each evaluation level, and can be expressed as bellow:

$$\{c(v_1), c(v_2), c(v_3), c(v_4), c(v_5)\} = \{5,4,3,2,1\} \tag{13}$$

IV. Experiment

A. Experimental Setup

Stack Overflow is the most popular developer community of asking and answering questions and it is a platform which reflects the real requirements of developers. As shown in Table III, we summarize several hot software types by analyzing the tags in Stack Overflow.

TABLE III: Hot Software Types in Stack Overflow

Software type	Number of tags
database	138070
machine learning	20603
web crawler	6560

So we choose "database", "machine learning" and "web crawler" as the search keywords in the experiment.

In addition, we also summarize two representative users' customized requirements:

Requirement I. Users want to find OSS resources for software reuse. As for software reuse, users need OSS in better quality and more mature. When the OSS has a high degree of popularity and remains active, the quality of such OSS can be better guaranteed. Therefore, *popularity of software* is the first important dimension and *activeness of software* is the second important dimension.

Requirement II. Users want to find OSS which is suitable for collaborative development. For this need, users need OSS of which contributors participate in passionately. These software authors communicate with contributors frequently, timely, and actively maintain software. Therefore, *collaborative development of software* is the first important dimension and *development attitude of software author* is the second.

B. Evaluation Metrics

In user study, in order to compare the effectiveness of our search service and GitHub-provided search service, volunteers will be asked to evaluate whether the top 10 OSS that in the search results offered by our method and GitHub (include searching by "Best match" and searching by "Most stars") is what they are looking for. The evaluation is a Likert-type scale with a more detailed expression for each choice [14]. The respondents are asked to choose one of three candidate response items, that is, our evaluation process is a three-point Likert scale. Table IV describes these three candidates.

TABLE IV: Likert Scale Response Categories

Scale	Response category
3	perfect expectations
2	general expectations
1	few expectations

To do the evaluation, twenty individuals with different backgrounds were invited to assess the result. Among them, nine are master students, six are Ph.D. students and five engineers with at least three years software development experience.

At last, we believe that software which is more in line with users' requirements should be ranked in more front. So according to the rank of software, we use $weighted\_Likert\_score$ to describe the effectiveness of search results.

$$weighted\_Likert\_score = \sum_{i=1}^{10} scale_i * weight_i \tag{14}$$

For the software ranked in the first, its weight is 1.0. For the software ranked in the second, its weight is 0.9. And for the last, the weight is 0.1. So the full marks of $weighted\_Likert\_score$ is 16.5.

C. Experimental Results

Volunteers will evaluate OSS separately according to Requirement I and Requirement II.

As for Requirement I, after volunteers tried out these OSS, they evaluated whether the OSS was mature and whether the quality reached their expectations.

As for Requirement II, after participating in the collaborative development of these OSS, volunteers assessed the

enthusiasm of other contributors and whether the software authors had good communication with them.

Figure 3 shows the evaluation results of user study in Requirement I.[1]

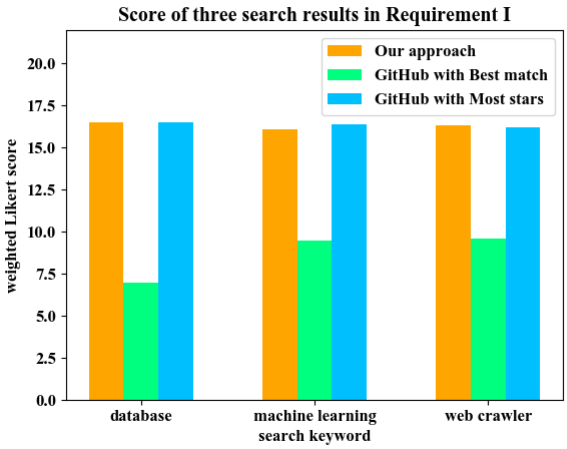

Fig. 3: Score of three search results in Requirement I

Figure 4 shows the evaluation results of user study in Requirement II.

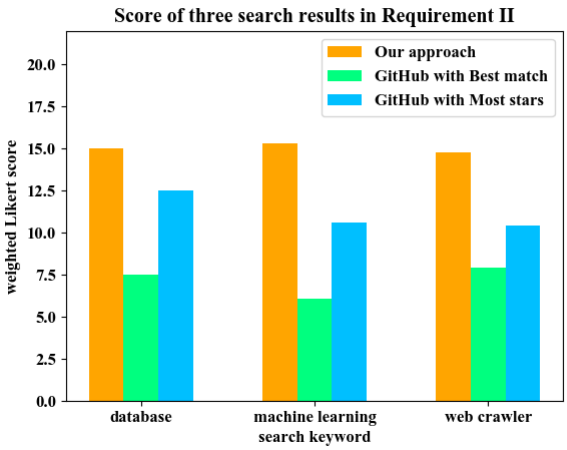

Fig. 4: Score of three search results in Requirement II

Through Figure 3 and Figure 4 we conclude the following conclusions:

- Whatever users' requirements are, searching by "Best match" has limited help with users.
- Searching by "Most stars" has better performance in finding software in high quality than finding suitable software for collaborative development. Because it does not take other attributes into account.
- Our method shows good results under different users' customized requirements and reflects a high self-adaptive ability.

[1]Complete experimental results can be found at: https://www.trustie.net/projects/4009/boards

- Compared with searching by "Best match", the average score of our method increased by 97.6%.
- Compared with searching by "Most stars", the average score of our method increased by 13.8%.

V. Conclusion and Future Work

In this paper, we first introduce a problem that users have to find OSS that suits their diverse requirements in a huge number of OSS resources and analyze why the current GitHub-provided search service is weak. Then we propose our own approach, which is providing a more efficient search service for OSS on GitHub by a multi-dimensional measurement model and a ranking algorithm based on fuzzy synthetic evaluation. At last, we verify that our method returns search results which meet users' expectations more, and has high self-adaptive ability.

The software data hosted on GitHub is huge today and will continue to grow rapidly. So in the future, we plan to optimize our algorithm and improve its efficiency.

References

[1] A. Begel, J. Bosch, and M.-A. Storey, "Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder," *IEEE Software*, vol. 30, no. 1, pp. 52–66, 2013.

[2] G. Gousios, "The ghtorent dataset and tool suite," in *Proceedings of the 10th working conference on mining software repositories*. IEEE Press, 2013, pp. 233–236.

[3] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1, no. 1.

[4] A. Aula, P. Majaranta, and K.-J. Räihä, "Eye-tracking reveals the personal styles for search result evaluation," in *IFIP Conference on Human-Computer Interaction*. Springer, 2005, pp. 1058–1061.

[5] F. Duijnhouwer and C. Widdows, "Open source maturity model. capgemini expert letter," *EU QualOSS project (grant number: 033547, IST-2005-2.5. 5)*, 2003.

[6] B. Golden, *Succeeding with open source*. Addison-Wesley Professional, 2005.

[7] A. Wasserman, M. Pal, and C. Chan, "The business readiness rating model: an evaluation framework for open source," in *Proceedings of the EFOSS Workshop, Como, Italy*, 2006.

[8] R. Semeteys, "Method for qualification and selection of open source software," *Open Source Business Resource*, no. May 2008, 2008.

[9] B. Russo, E. Damiani, S. Hissam, B. Lundell, and G. Succi, *Open Source Development, Communities and Quality: IFIP 20th World Computer Congress, Working Group 2.3 on Open Source Software, September 7-10, 2008, Milano, Italy*. Springer Science & Business Media, 2008, vol. 275.

[10] U. Höhle and S. E. Rodabaugh, *Mathematics of fuzzy sets: logic, topology, and measure theory*. Springer Science & Business Media, 2012, vol. 3.

[11] X. Xue and X. Yang, "Seismic liquefaction potential assessed by fuzzy comprehensive evaluation method," *Natural hazards*, vol. 71, no. 3, pp. 2101–2112, 2014.

[12] Y. Yu, H. Wang, G. Yin, and C. X. Ling, "Reviewer recommender of pull-requests in github," in *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*. IEEE, 2014, pp. 609–612.

[13] Y. Yu, H. Wang, V. Filkov, P. Devanbu, and B. Vasilescu, "Wait for it: Determinants of pull request evaluation latency on github," in *Mining software repositories (MSR), 2015 IEEE/ACM 12th working conference on*. IEEE, 2015, pp. 367–371.

[14] S. Jamieson *et al.*, "Likert scales: how to (ab) use them," *Medical education*, vol. 38, no. 12, pp. 1217–1218, 2004.