Evaluating a Tool for Creating Bug Report Assignment Recommenders

Disha Devaiya, John Anvik, Meher Bheree, Farjana Yeasmin Omee
Department of Mathematics and Computer Science
University of Lethbridge, Alberta, CANADA
E-mail: devaiya86@gmail.com, [john.anvik, bheree, omee]@uleth.ca

Abstract

Large software development projects that use bug tracking systems can become overwhelmed by the number of reports filed. To assist in reducing the workload of project members, researchers have proposed the use of bug report assignment recommenders. To assist project members with the creation of assignment recommenders, we proposed a web-based tool called the Creation Assistant for Supporting Triage Recommenders *(CASTR). This paper presents the results of both a laboratory and field study of CASTR. We found that CASTR can create assignment recommenders with accuracy as high as 95%, 80%, and 70% for Top-1, Top-3 and Top-5, respectively. The field study showed that 60% of the participants found CASTR easy to use, whereas the remaining participants found CASTR moderately or slightly easy to use.*

1. Introduction

An issue tracking system plays an important role in the development of a high-quality software product. Such systems record information for a bug report or feature request. These records include the name of the developer that resolved the issue and other relevant development activity. Issue tracking systems are particularly important when team members are globally distributed [3].

During the testing phase of software development, a developer or tester confirms that the software is working per the specifications. If irregularities are found, a triager marks it as a bug in the issue tracking system and includes such information as the steps to reproduce and screenshots. A triager will then assign the reported bugs to the appropriate developer based on the view of the developer's ability or their bug fixing history. In the case of large projects, a large number of new bugs can be submitted daily [2]. For a software project that uses a manual triage process, the bug triagers can become overwhelmed. To address such problems, researchers have proposed the use of bug report assignment recommenders [3, 4, 6, 7, 8].

However, the creation of a bug report assignment recommender for a software project can be a complex and time-consuming process. To address this problem, we previously proposed the Creation Assistant for Easy Assignment (CASEA) [1]. CASEA was further refined into a web-based tool called the *Creation Assistant for Supporting Triage Recommenders* (CASTR) [5]. CASTR allows a project member to create an assignment recommender for a project by specifying such items as labelling heuristics, a machine learning algorithm and a data imbalance technique. They can also use a "specify and verify" approach to determine the optimal configuration settings for a project-specific assignment recommender. By providing a little project knowledge, a project member can produce a bug report assignment recommender in a short period.

This paper presents an evaluation of the CASTR system to answer the following research questions:

1. **RQ1: Does CASTR create assignment recommenders that make accurate recommendations?** If CASTR creates assignment recommenders that make accurate recommendations, then a triager need not examine the report as deeply. Using such recommenders changes the triager's role from making decisions relying on their knowledge, experience, intuition or information they can gain from existing tools to confirming decisions made by the recommender. This shift changes, and hopefully, reduces triager cognitive load.

2. **RQ2: Can human triagers make effective use of information presented by CASTR?** If CASTR creates assignment recommenders that assist human triagers then time can be saved by not assigning bug reports to the appropriate developer manually. Some of the human resources consumed by the triage process can be then directed elsewhere in the project.

This work was funded by the Natural Sciences and Engineering Research Council of Canada.
DOI reference number: 10.18293/SEKE2021-163

Our analytical evaluation conducted using bug report datasets from the Plasmashell[1], LibreOffice[2] and Firefox[3] projects showed that recommenders with good accuracy could be created. A field study with ten participants from different technical backgrounds gave evidence that they were able to make effective use of the information provided and that most of them are likely to use CASTR in creating an assignment recommender.

2. Creation Assistant for Supporting Triage Recommenders (CASTR)

CASTR [5] is a platform-independent web-based tool that assists a project member with the creation of bug report assignment recommenders. It provides a web interface for downloading a dataset from a Bugzilla repository. Information about the collected dataset is displayed by a Configuration tab. CASTR assists with setting project-specific heuristics for labelling reports with the names of the developer to be recommended. As not all of the developer names may be valid, CASTR allows the user to select a minimum threshold of resolution activity to eliminate developers that have resolved a small number of bug reports. CASTR provides the option of choosing one of four supervised machine learning algorithms: Support Vector Machines (SVM), Multinomial Naïve Bayes, C4.5, and Rules. We chose these algorithms as they represent different categories of supervised machine learning algorithms. CASTR also provides three different approaches to handle imbalanced data: oversampling using SMOTE, manual oversampling and undersampling using Expectation-Maximization (EM). When the user clicks the "Recommender" button in the Configuration tab, a request is sent to the CASTR web service and redirects the user to an Analysis tab that presents progress information, such as the time to train the recommender and evaluation results. The Analysis tab displays the average Top-1, Top-3 and Top-5 precision and recall values for a testing set. A user can tune a recommender by comparing these values with the last five generated recommenders to create the best assignment recommender for their project.

3. Evaluation

In evaluating CASTR, we sought to answer our two research questions. We answered RQ1 using an analytical evaluation of the recommenders created by CASTR and investigated RQ2 through a field study.

[1]https://bugs.kde.org

[2]https://bugs.documentfoundation.org

[3]https://bugzilla.mozilla.org

3.1. An Analytic Evaluation of the Recommenders

We selected bug reports with the resolution *Fixed* from three open source projects: Plasmashell, LibreOffice and Firefox. Also, we removed bug reports for developers who fixed less than 20 reports. We used a 90%/10% split between training and testing sets. Table 1 shows the number of bug reports used for the three projects. The third column shows the number of bug reports remaining filtering.

We evaluate the performance of a recommender using the metrics of precision[4] and recall[5]. This requires us to know the set of developers who could have been accurately assigned to a bug report. We approximated this information using the names of developers who fixed reports in the same component as the testing bug report.

We explored all possible combinations of machine learning and data imbalance algorithms provided by CASTR. As Table 2 shows, we found that LibreOffice produced the highest precision values (97/95/91) with the Naïve Bayes/SMOTE combination whereas for Firefox the Naïve Bayes/undersampling with EM technique was the best (55/36/36). On the other end, Plasamshell had the best results (96/83/73) using an SVM algorithm without any imbalance technique applied. The low recall values are likely a result of overestimating the set of possible developers. For example, in the Firefox project, it was not uncommon for the estimated set of developers to be 30+ developers, meaning that the best recall value for a single recommendation would be about 3% (1/30).

3.2. A Field Study of CASTR

To answer our second research question, we conducted a field study with experienced software developers, project managers, bug triagers and graduate students. The study contained ten (10) participants: 3 project managers, 5 application developers and 2 graduate students.

The field study was conducted by first asking participants to complete an initial survey that collected demographic information and technical background details. The demographic questions were for general analysis to break down the response data into meaningful groups. For example, we found that Indian participants took the survey more than the participants with other nationalities and the majority were in the age range of 26 to 39. Most participants completed a graduate-level of schooling, and most of them belong to the job function *Application Developer*. For technical background, 60% reported having a lot of prior experience with issue tracking systems, and 33% had a good amount of experience with contributing to large open-source projects.

[4]Precision measures how often the approach makes a relevant recommendation for a report

[5]Recall measures how many of the relevant recommendations are truly recommended

Projects	Original Dataset Size	After Filtering Dataset Size	# of Bug Reports in Training Set	# of Bug Reports in Testing Set
Plasmashell	1112	532	479	53
LibreOffice	2500	1725	1553	172
Firefox	1000	777	699	78

Table 1. Training and testing set sizes for evaluating recommenders.

Project	Algorithm	Sampling Technique	Precision			Recall		
			Top 1	Top 3	Top 5	Top 1	Top 3	Top 5
Plasmashell	SVM	None	96	83	73	11	26	38
LibreOffice	Naïve Bayes	SMOTE	97	95	91	2	7	11
Firefox	Naïve Bayes	UnderSampling/EM	55	36	36	11	15	9

Table 2. Evaluation results of assignment recommender

Overall, participants had low bug triaging experience[6] and low familiarity with the machine learning algorithms.

After completing the initial survey, participants were given a user manual of CASTR and asked to create an assignment recommender for the Plasmashell project. The Plasmashell dataset used was identical to that used for the analytical experiment.

In the field study, a total of 71 recommenders were created by the participants using different heuristic configurations provided by CASTR. Table 3 shows the quantitative results for the best recommender created by each participant. The first column identifies the unique participant. The second column presents the machine learning algorithm selected by the participants for creating their best assignment recommender. The next two columns present the minimum and maximum threshold that the participants selected before creating their most accurate assignment recommender using CASTR. Half of the participants chose values greater than or equal to 10 for the minimum threshold and the other half used values less than 10. By default, the maximum threshold is set to the largest activity value depending on the set project-specific heuristics. These values were left unchanged by the participants. The next three columns show the Top-1, Top-3 and Top-5 precision and recall values for the best assignment recommender created by the participants. Most of the assignment recommenders were created using the SVM machine learning algorithm. In the case of Top-5, two scenarios have no values as the recommender suggested less than 5 developers because of threshold value settings.

As a part of the recommender evaluation, CASTR provides information about how long the tool takes to create a recommender. For most users, the average time to create a recommender using any of the algorithms was less than 30 seconds. In general, the results show that assignment recommenders created using the C4.5 and SVM algorithms took more processing time than the assignment recommender created using the Naïve Bayes and Rules algorithms. Two notable exceptions were for Users 1 and 7, whose average recommender creation times using C4.5 was between 160 and 200 seconds. A possible reason for the large processing time is that both participants had set the minimum threshold value as 1, meaning that CASTR considered all of the possible developers for classification, which led to a substantial increase in the processing time.

Although participants were provided with a video presentation of CASTR and a brief tutorial of the recommender creation process at the beginning of the field study, participants encountered some problems related to understanding concepts, specifically with labelling bug reports and setting an appropriate minimum threshold value. Most of the participants did not initially understand how to select the appropriate label for bug report resolution and which machine learning algorithm to use. Also, the meaning of the precision and recall metrics was not initially well understood by participants. However, once their meaning was understood, participants felt they made more intelligent choices.

The field study results show that 60% of the participants found CASTR easy to use whereas the remaining participants found CASTR moderately or slightly easy to use. We received positive responses about recommending CASTR for creating a recommender for bug report assignment with 50% of the participants responding “very likely” or “extremely likely”, and the remaining participants responding “moderately likely”. When asked whether they believed that the assignment recommenders created using CASTR would reduce the time to triage bug reports, all participants responded either “extremely likely” (2), “very likely” (5), or ”moderately likely” (3).

Based on observations while analyzing the field study result, participants were found to employ two strategies for as-

[6] A possible reason for this is that most of the participants were part of a large software development project team where their responsibilities are limited to within specific modules or feature development.

Identifier	Algorithm	Trials to Best	Threshold		Top 1 (%)		Top 3 (%)		Top 5 (%)	
			Min	Max	Precision	Recall	Precision	Recall	Precision	Recall
User1	SVM	2	5	108	92	9	83	25	72	34
User2	SVM	1	10	130	96	11	83	25	77	38
User3	SVM	1	26	112	94	11	85	30	-	-
User4	Naïve Bayes	5	49	73	87	9	78	21	-	-
User5	SVM	1	10	130	96	11	83	25	77	38
User6	SVM	2	1	141	74	17	58	33	56	49
User7	SVM	4	1	61	91	7	85	21	74	29
User8	Naïve Bayes	1	1	117	50	5	37	10	33	14
User9	SVM	1	10	115	94	10	82	24	76	38
User10	SVM	7	1	102	87	14	65	27	60	42

Table 3. Best assignment recommenders created by participants.

signment recommender creation using CASTR. Some participants were found to be very experimental in their approach, making many changes before creating a new recommender. Other users were more methodical, making small changes and testing the results. Most of the time, participants changed the heuristic configurations and not the minimum threshold. Out of 71 recommenders, 42 were created with the minimum threshold value as 1, 27 recommenders were created with a threshold more than or equal to 10 and the remaining 2 recommenders were created with thresholds of 3 and 5. The majority of recommenders were created using the SVM machine learning algorithm (32), with Naïve Bayes being the second most used algorithm (19) and the C4.5 and Rules algorithm each used 10 times.

4. Conclusions

This paper presented the results of our evaluation of CASTR, a tool to assist software development projects with the creation of bug report assignment recommenders. Our analytical evaluation showed that CASTR can create recommenders with good accuracy, answering RQ1.

Our field study demonstrated that users were able to create accurate bug report assignment recommenders in less than 10 trials. This indicates that they were able to make good use of the information provided by CASTR, answering RQ2. Also, users found the tool was generally easy to use.

Although the results we obtained have shown that a CASTR assists the project members with the creation of assignment recommenders based on feedback and the results of the user study, several future improvements were identified. These include extending CASTR to collaborate with the other issue tracking systems, having the tool use information from duplicate bug reports, supporting the creation of other types of triage recommenders, and a field study by dataset project members.

References

[1] J. Anvik, M. Brooks, H. Burton, and J. Canada. Assisting software projects with bug report assignment recommender creation. *In Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering*, pages 470–473, 2014.

[2] J. Anvik, L. Hiew, and G. C. Murphy. Who should fix this bug? In *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, pages 361–370, New York, NY, USA, 2006. ACM.

[3] J. Anvik and G. C. Murphy. Reducing the effort of bug report triage: Recommenders for development-oriented decisions. *ACM Trans. on SE and Methodology*, 20, 2011.

[4] P. Bhattacharya and I. Neamtiu. Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging. In *Proceedings of the 2010 IEEE International Conference on Software Maintenance*, ICSM '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

[5] D. Devaiya. Castr: A web-based tool for creating bug report assignment recommenders. Master's thesis, University of Lethbridge, Lethbridge, Alberta, CANADA, 2019.

[6] G. Jeong, S. Kim, and T. Zimmermann. Improving bug triage with bug tossing graphs. In *Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ESEC/FSE '09, pages 111–120, New York, NY, USA, 2009. ACM.

[7] S. Kim and E. J. Whitehead, Jr. How long did it take to fix bugs? In *Proceedings of the 2006 International Workshop on Mining Software Repositories*, MSR '06, pages 173–174, New York, NY, USA, 2006. ACM.

[8] R. Shokripour, J. Anvik, Z. M. Kasirun, and S. Zamani. Why so complicated? simple term filtering and weighting for location-based bug report assignment recommendation. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 2–11, Piscataway, NJ, USA, 2013. IEEE Press.