An Exploratory Study on Improving Automated Issue Triage with Attached Screenshots

Ethem Utku Aktas
Softtech Inc., Research and Development Center, 34947 Istanbul, Turkey
utku.aktas@softtech.com.tr

Cemal Yilmaz
Faculty of Engineering and Natural Sciences, Sabanci University, 34956 Istanbul, Turkey
cemal.yilmaz@sabanciuniv.edu

ABSTRACT

Issue triage is a manual and time consuming process for both open and closed source software projects. Triagers first validate the issue reports and then find the appropriate developers or teams to solve them. In our industrial case, we automated the assignment part of the problem with a machine learning based approach. However, the automated system's average accuracy performance is 3% below the human triagers' performance. In our effort to improve our approach, we analyzed the incorrectly assigned issue reports and realized that many of them have attachments with them, which are mostly screenshots. Such issue reports generally have short descriptions compared to the ones without attachments, which we consider as one of the reasons for incorrect classification. In this study, we describe our proposed approach to include this new piece of information for issue triage and present the initial results.

CCS CONCEPTS

• **Software and its engineering → Software verification and validation**.

KEYWORDS

issue triage, issue report assignment, optical character recognition, text mining

ACM Reference Format:
Ethem Utku Aktas and Cemal Yilmaz. 2020. An Exploratory Study on Improving Automated Issue Triage with Attached Screenshots. In *42nd International Conference on Software Engineering Companion (ICSE '20 Companion), October 5–11, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3377812.3390805

1 INTRODUCTION

For automating issue triage, previous work concentrate on the summary and description attributes of the issue reports [1, 3], which are one line short title for the issue report and a more detailed explanation written in natural language. With text mining and machine learning techniques, the assignees of the issue reports can be predicted. Support Vector Machines (SVM) is a successful algorithm for classifying the textual data present in issue reports [1–3].

Our industrial case is a large scale industrial software company, Softtech[1], serving the largest bank in Turkey, IsBank[2], where around 350 software related issue reports are received at Softtech daily from the bank.

We automated the issue assignment process in Jan 2018 at Softtech with state of the art techniques (*tf-idf* for vectorization and SVM algorithm for prediction). However, we occasionally receive complaints asking why a specific issue report was assigned to their team. When we check such issue reports, we often observe that they have attachments, where the user does not explain the issue much in the description part. Furthermore, most of these attachments are screenshots of the errors they receive when using the related software product.

In this work, we explain our approach for including the information in screenshots for automated issue report assignment. The rest of the paper is organized as follows: In Section 2, we describe our approach to extract the textual data in screenshots and how we include this extra information for classification, in Section 3 we represent the results of the study whether including the textual information in screenshots may improve accuracy of the system with the proposed approach and in Section 4, we conclude and discuss the future work to conduct.

2 APPROACH

2.1 Data Extraction

We use Optical Character Recognition (OCR) to extract the textual information in these screenshots. We first retrieve the "ID, summary, description and attachment URL" features of the issue reports. If the issue report has any attachments, we then check its extension. If the attachment is an image file, or if it is a document sheet that includes an image file in it, we get the image and utilize Py-tesseract [5] as the OCR engine to get the textual data in it. We do not perform data cleaning due to the errors that may occur in OCR translation. Finally, we use the text from screenshots as another textual attribute of the issue report. We apply tokenization and remove stop words on all the textual data.

With this method, we extract three months of data for training (17604 issue reports), created in June, July and August of 2019 and 1 week of data for testing (1451 issue reports), created in the first week of September 2019, where 735 of them have attached screenshots and 716 of them do not.

Among these reports, 56% of them have screenshots. With the deployed system, we observe that 81% and 88% of the ones with and without screenshots are assigned correctly, respectively. So, an upper bound for the improvement that can be obtained by including

[1]https://softtech.com.tr
[2]https://www.isbank.com.tr

this artefact could be around 4%, if we could obtain the same level of accuracy with and without screenshots.

2.2 Classification

Machine learning based issue triage techniques use previously solved issue reports for training and regards the problem as a supervised classification problem. First, the textual data is pre-processed (tokenized and stop-words are removed, however stemming is not applied in our case since it has minor effect) and then vectorized to obtain the $tf - idf$ (term frequency - inverse document frequency) vectors.

$$tf(t,i) = \frac{\textit{Number of times term t occurs in issue report i}}{\textit{Total number of terms in issue report i}} \quad (1)$$

$$idf(t,I) = log(\frac{\textit{Total number of issue reports in repository I}}{\textit{Number of issue reports having term t}}) \quad (2)$$

$$tf - idf(t,i,I) = tf(t,i) \times idf(t,I) \quad (3)$$

We then use Support Vector Machines (SVM) as the classifier, which is a commonly used technique in issue triage. We utilize Linear SVC implementation of scikit-learn tool [4] and use the default parameters.

Depending on the selected feature to be used as input for the pipeline, we have four machine learning models trained with SVM: ML_s is trained by using "summary" feature only; ML_{sa} is trained by using the concatenated feature "summary" and "text from image"; ML_{sd} is trained by using the concatenated feature "summary" and "description"; ML_{sda} is trained by using the concatenated feature "summary", "description" and "text from image". We have one final model, that is an ensemble of two models: ML_{sd} and ML_{sa}. We multiply the prediction probabilities of the two models with their weights (which sum up to 1), and select the class that has the highest resultant probability. We use 0.6 for the weight of ML_{sd} and 0.4 for the weight of ML_{sa}, which are obtained after some initial training and testing with the current data.

We use accuracy, precision, recall and f-measure as the evaluation metrics for our multi-class classification problem and report the results of predictions on our separate test data (1451 issue reports).

3 STUDY

3.1 Research Questions

We investigate the following questions in our study:

RQ1: Does using the extra feature, text from images, has any contribution in prediction accuracy? We compare ML_s and ML_{sa} to see how much we improve by adding "text from attachments", compared to using only the "summary" attribute.

RQ2: Does adding the extra textual information from screenshots improve the overall performance? Our baseline model is ML_{sd} and we compare it with ML_{sda} and $ML_{ensemble}$ to see whether the overall performance improves when we add the textual data in screenshots.

3.2 Study Results

Accuracy (A), weighted precision (P), recall (R) and F-measure (F) results on the test data set are presented in Table 1.

Table 1: Results on the test issue report dataset

Classifier	A	P	R	F
ML_s	0.70	0.72	0.70	0.69
ML_{sa}	0.74	0.75	0.74	0.73
ML_{sd}	0.86	0.84	0.86	0.84
ML_{sda}	0.86	0.84	0.85	0.84
$ML_{ensemble}$	0.86	0.85	0.86	0.85

RQ1: Does using the extra feature, text from images, has any contribution in prediction accuracy? ML_{sa} performs better than ML_s on the test dataset. A 0.04 performance improvement is achieved in accuracy, precision and f-score and 0.03 performance improvement in recall. Although we applied minimal pre-processing on the textual data extracted from images, the performance improvement implies that attached screenshots may have extra useful information to improve issue triage.

RQ2: Does adding the extra textual information from screenshots improve the overall performance? The performance of ML_{sd}, ML_{sda} and $ML_{ensemble}$ are very similar as seen in Table 1. So, with our approach, we cannot conclude that we could improve in overall. The possible reasons are, the techniques used in this study are fairly simple and mainly based on analysis of the raw textual data on which little processing is applied. As future work, we plan to study on what kind of textual data in screenshots is actually useful for improving issue triage.

4 CONCLUSION

In this paper, we argued that useful information exists in issue report attachments that could improve automated issue triage. We proposed to use OCR technique to extract the textual information in these attachments. We applied minimal processing on this data and used it with summary and description for the prediction of software teams to solve these issue reports. Our initial results suggest that the attached image files may have useful information for issue triage, however with the proposed approach the results are not substantial and significant.

In the future, we plan to extend our work such that we work on more innovative ways to exploit the useful textual data in attachments to improve issue management.

REFERENCES

[1] John Anvik, Lyndon Hiew, and Gail C Murphy. 2006. Who should fix this bug?. In *Proceedings of the 28th international conference on Software engineering*. ACM, 361–370.

[2] Pamela Bhattacharya, Iulian Neamtiu, and Christian R Shelton. 2012. Automated, highly-accurate, bug assignment using machine learning and tossing graphs. *Journal of Systems and Software* 85, 10 (2012), 2275–2292.

[3] Leif Jonsson, Markus Borg, David Broman, Kristian Sandahl, Sigrid Eldh, and Per Runeson. 2016. Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts. *Empirical Software Engineering* 21, 4 (2016), 1533–1578.

[4] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[5] Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Vol. 2. IEEE, 629–633.