Berrios, M., Ramírez, A., Feldt, R., Romero, J. R.: A methodology for explaining Learning-to-rank models for test case prioritization. In: Rodríguez Luaces, M. A. (ed.) Actas de las XXVIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2024). Sistedes (2024). https://hdl.handle.net/11705/JISBD/2024/110

A methodology for explaining Learning-to-rank models for test case prioritization

Mario Berrios[1], Aurora Ramírez[1], Robert Feldt[2], and José Raúl Romero[1]

[1] University of Córdoba, Córdoba, Spain
{i12bercm, aramirez, jrromero}@uco.es
[2] Chalmers University of Technology, Goteborg, Sweden
robert.feldt@chalmers.se

Abstract. In machine learning-based test case prioritization (TCP), there is a need for explainable methods that can shed light on the internal mechanisms of the models and clarify why certain test cases are deemed more likely to fail. To address this need, we propose an experimental methodology designed to generate and analyze global explanations for models in TCP. This methodology can help testers and researchers to understand how the impact of different features shifts as the software system evolves.

Keywords: Test case prioritization · Explainable machine learning

1 Introduction

In the context of regression testing, test case prioritization (TCP) serves to decide the order in which test cases should be executed. The application of machine learning (ML) is a recent trend that generates predictive models to automate the TCP process, using features extracted from the system under test (SUT) and its testing process. More specifically, Learning-to-rank (LtR) models are able to rank all the test cases within a test suite based on the rankings observed in the past. This training process can be repeated each time a new build becomes available, in search of more accurate predictive models.

As machine learning (ML) is increasingly used in software engineering and testing, there is a need also for so called explainable artificial intelligence (XAI) [3]. XAI aims to demystify the internal mechanisms of models (providing global explanations) and to articulate the rationale behind specific predictions (offering local explanations)[1]. These XAI methods have recently been discussed in the context of TCP [2]. In this paper, we propose a specific experimental methodology to understand the global behavior of LtR models for TCP, emphasizing the need to study the evolution of the importance of features as more builds become available for learning.

2 Proposed experimental methodology

First, we introduce the concepts and problem formulation necessary to approach TCP from an ML perspective. For a given SUT, a sequence of builds $(b_1...b_n)$

This work is licensed under a Creative Commons “Attribution-NonCommercial-NoDerivatives 4.0 International” license.

all need to be tested, progressively. If, for each build, the k test cases ($tc_1...tc_k$) of its test suite can be ranked this can help make testing more efficient, i.e. problems are found earlier and costs decrease. Such a ranking can be automatically determined by a ML algorithm, considering p properties of the test cases ($f_1...f_p$), such as its coverage, past outcomes and execution time, etc. In the scope of this paper, we focus on pairwise LtR algorithms, which produce rankings —comparing each pair of test cases— instead of instance-level predictions.

As depicted in Figure 1, we define the following steps to evaluate model performance and generate global explanations in the context of ML-based TCP:

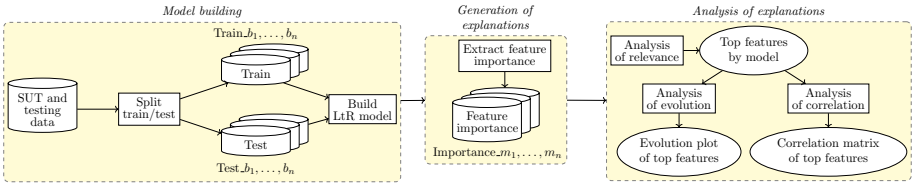

Fig. 1. The methodological steps proposed for global explainability in TCP.

Model building. The LtR algorithm is run to predict the order of test cases for a build b_i, using the data collected from previous builds ($b_1...b_{i-1}$) for training. If the process is repeated for each new build, we will train n models ($m_1...m_n$), where each model provides a relevance score for each test case selected to test b_i. If two test cases have the same predicted relevance score, the test case with a shorter execution time occupies a higher position as a way to reveal failures earlier. A common metric for measuring performance in LtR is the NDGC (Normalized Discounted Cumulative Gain) metric, while test-specific metrics (e.g. APFD) can be calculated on the resulting test case prioritization.

Generation of explanations. To understand the behavior of m_i, a global explanation is required. One approach is to analyze the importance of the features used to build the model [1], which also provides information on how their changes might influence model performance. For tree-based learning algorithms, one way to assign an importance to each feature is to count how many times the feature is used in split nodes, i.e., nodes that fork new branches after a decision is made.

Therefore, we extract an array of feature importance for each model. In the end, we have a matrix p x n of feature importance (M_{FI}), where the rows correspond to the features and the columns are the models sorted by build history. Note that the specific importance values may present a different range of values for each model. For ease of analysis, each cell value $v_{i,j}s$ expresses the ranking of f_i in m_j when ordered by importance.

Analysis of explanations. M_{FI} can be analyzed in three ways:

Berrios, M., Ramírez, A., Feldt, R., Romero, J. R.: A methodology for explaining Learning-to-rank models for test case prioritization. In: Rodríguez Luaces, M. A. (ed.) Actas de las XXVIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2024). Sistedes (2024). https://hdl.handle.net/11705/JISBD/2024/110

This work is licensed under a Creative Commons “Attribution-NonCommercial-NoDerivatives 4.0 International” license.

Berrios, M., Ramírez, A., Feldt, R., Romero, J. R.: A methodology for explaining Learning-to-rank models for test case prioritization. In: Rodríguez Luaces, M. A. (ed.) Actas de las XXVIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2024). Sistedes (2024). https://hdl.handle.net/11705/JISBD/2024/110

1. *Analysis of relevance.* Identification of the most relevant features at a given point in the software testing process by inspecting the ranking of all features within a model m_i. In this process, we can also identify the top features, those that usually appear in the sequence of models with high importance.
2. *Analysis of evolution.* Visualization of the evolution of the importance assigned to each feature, analyzing how its position in the ranking varies throughout the models. This analysis serves to detect trends in the relevance of the data collected as the testing process progresses.
3. *Analysis of correlation.* Study of the correlation of features in terms of how they influence a series of models. This analysis can reveal whether some pairs of features show the same trend with respect to their importance scores.

3 Preliminary experimental results

We apply our experimental methodology to a project (*facebook@buck*, 129 builds) from a TCP dataset with 150 features [4] and an average of 622 test cases/build. The LtR models are trained using the LambdaMart algorithm (as implemented in the LightGBM library). Compared to other projects, this one reported the best average performance in terms of the NDCG metric across all builds.

Analysis of relevance. The union of the top-5 features in terms of importance obtained for all models ($m_1 ... m_{129}$) is composed of 11 features of diverse origin: 1) features related to the complexity of the test code (TES_COM), which are *CountLineCodeExe*, *CountLineComment*, *CountStmtExe* and *RatioCommentToCode*; 2) features related to the test code development process (TES_PRO), such as *CommitCount*, *OwnersContribution*, *OwnersExperience* and *AllCommitersExperience*; and 3) features related to the test case execution records, such as *Age*, *RecentAvgExeTime* and *TotalAvgExeTime*. The fact that the top-5 undergoes some changes from build to build suggests that the evolution of the SUT influences the relevance of LtR model features.

Analysis of evolution. As shown in Figure 2, two behaviors are observed. A group of features (*Age*, *AllCommitersExperience*, *OwnersExperience*, *TotalAvgExeTime* and *OwnersContribution*) show a very stable behavior, which means that they are consistently relevant to the LtR models. Among them, *Age* has the highest variation in terms of relevance score. Within this group, three out of the four TES_PRO features are included. This suggests the important role that the test case development process plays in predicting test case failures. The remaining features experience noticeable ranking fluctuations, specially in the case of the feature *CommitCount*. This seems to indicate that its relevance is subject to specific moments in the SUT development history.

Analysis of correlation. None of the features show a strong Pearson correlation: 3 pairs of features are moderately correlated, 18 are weakly correlated, and 34 are very weakly correlated. The more relevant are: *OwnersContribution-CountLineCodeExe* (-0.41), *OwnersContribution-TotalAvgExeTime* (-0.41), and

This work is licensed under a Creative Commons “Attribution-NonCommercial-NoDerivatives 4.0 International” license.

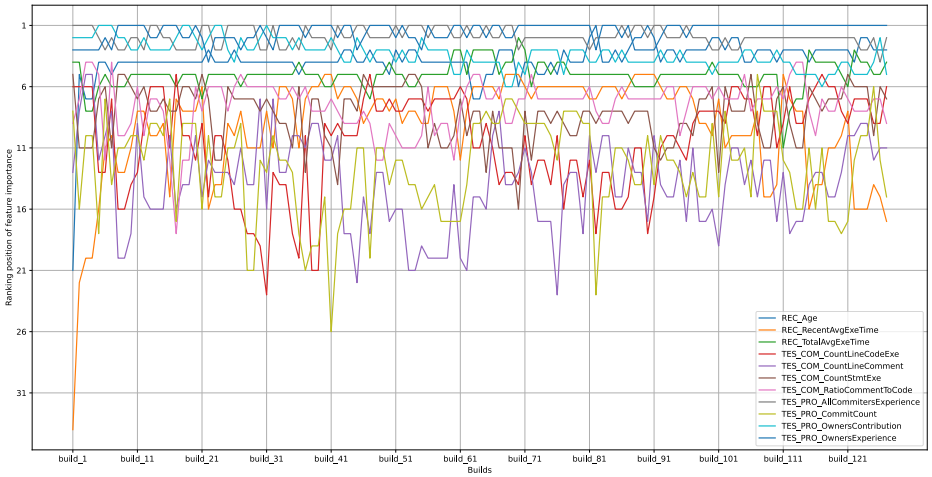

Fig. 2. Evolution of the relevance of 11 features (expressed as ranking positions) in the LtR models trained over the SUT history. Lower values mean more importance.

CountLineCodeExe-*CountStmtExe* (0.47). This analysis suggests that the contribution of each feature to model prediction does not follow common patterns.

4 Discussion and open issues

A more detailed analysis of the feature values could reveal the causes of their variations in relevance. This would help to better understand how different features are related. A clustering analysis could identify more complex patterns in terms of feature relevance, and whether it has any relation to the origin of the features used in TCP. To draw more general conclusions, the methodology should be applied to more projects, other ML paradigms, and possibly validated by the software testers to whom the explanations are addressed. The analysis can be complemented with local explanations to understand specific predictions.

Acknowledgments. Work supported by the project PID2020-115832GB-I00 and the RED2022-134647-T network funded by MICIU/AEI/10.13039/501100011033.

References

1. Biecek, P., Burzykowski, T.: Explanatory Model Analysis: Explore, Explain, and Examine Predictive Models. Chapman and Hall/CRC (2021)
2. Ramírez, A., Berrios, M., Romero, J.R., Feldt, R.: Towards Explainable Test Case Prioritisation with Learning-to-Rank Models. In: AIST@ICST. pp. 66–69 (2023)
3. Tantithamthavorn, C., Cito, J., Hemmati, H., Chandra, S.: Explainable AI for SE: Challenges and Future Directions. IEEE Software **40**(3), 29–33 (2023)
4. Yaraghi, A.S., Bagherzadeh, M., Kahani, N., Briand, L.: Scalable and Accurate Test Case Prioritization in Continuous Integration Contexts. IEEE TSE pp. 1–24 (2022)

Berrios, M., Ramírez, A., Feldt, R., Romero, J. R.: A methodology for explaining Learning-to-rank models for test case prioritization. In: Rodríguez Luaces, M. A. (ed.) Actas de las XXVIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2024). Sistedes (2024). https://hdl.handle.net/11705/JISBD/2024/110

This work is licensed under a Creative Commons “Attribution-NonCommercial-NoDerivatives 4.0 International” license.