Inspiring AI-Infused Innovations in Adoptium AQAvit and Openj9

Md Afif Al Mamun[♣], Gias Uddin[♠], Lan Xia[♦], Longyu Zhang[♦]
University of Calgary[♣], York University[♠], IBM Canada[♦]

Abstract

In today's fast-paced technological landscape, maintaining efficiency, quality, and reliability is critical for any project development. Embark on a journey of inspiration as we explore the fusion of AI with Adoptium AQAvit and Openj9. We will dive into visionary AI concepts and initiatives within both projects, illuminating pathways for transformative change – component and issue assignment, duplicate issue detection, issue creation. Moreover, we will explore the tangible benefits and pragmatic implications of infusing AI into AQAvit testing and Openj9 workflows, including accelerated testing and release cycles, heightened fault detection capabilities, and diminished manual workload. Through real-world case studies with CAS and Semesters of Code project, we would like to share our story of how AI can revolutionize testing and triaging, paving the way for more robust and reliable software development practices.

Testing Challenges for Eclipse Adoptium AQAvit and Eclipse Openj9

- Adoptium AQAvit is a comprehensive open Java testing standard used by multiple java vendors.
- Eclipse Openj9 is the Java VM used by IBM Semeru Runtimes, which runs millions of tests from AQAvit nightly to ensure the quality.
- Therefore, a large amount of testing data are generated daily, and **need to leverage AI to improve the efficiency of test failure triaging, assigning, and debugging**.

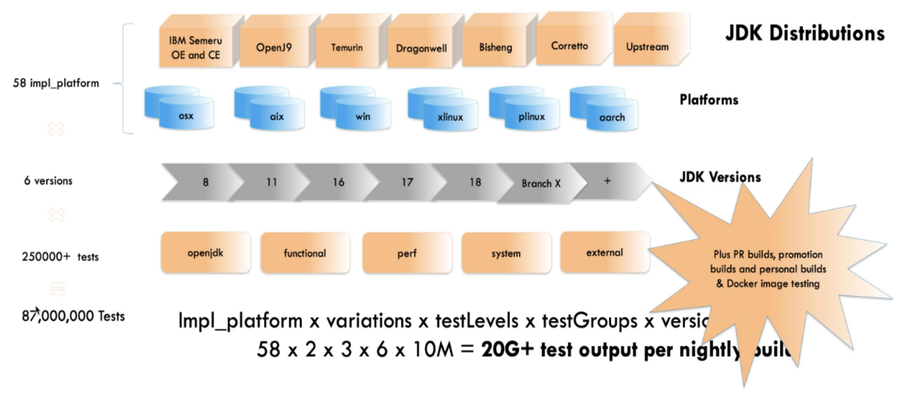

Figure 1: AQAvit Comprehensive Test Coverage For Platforms, Versions, Levels, and Groups.

Intelligent Developer Prediction System with IBM CAS TriagerX Model and Eclipse Foundation Semesters of Code

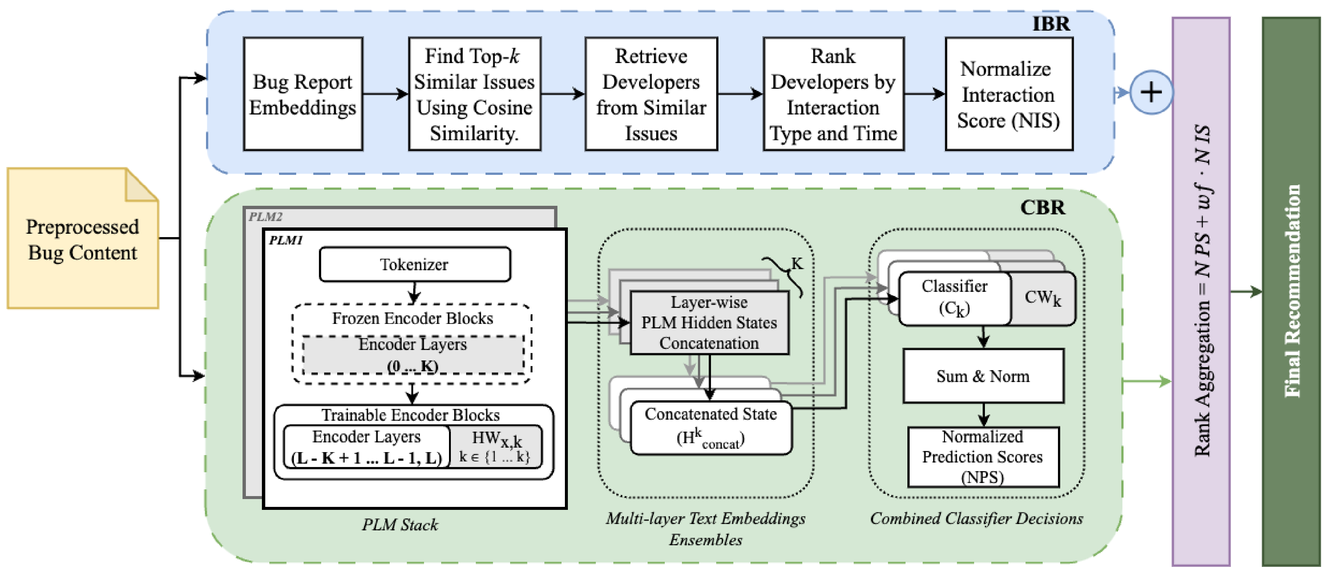

Figure 2: Architecture of TriagerX developer prediction model including Content-based Ranker (CBR) and Interaction-based Ranker (IBR).

- We introduce TriagerX, a novel hybrid bug triaging framework that leverages a dual-transformer architecture to generate a content-based ranking (CBR) of candidate developers through multi-layered bug report representations, further refined with an Interaction-Based Ranking (IBR) method.
- Our approach surpasses state-of-the-art PLM-based methods like LBT-P[1], earlier DBRNN-A[2], and traditional techniques using large PLM variants (e.g., RoBERTa-large, DeBERTa-large) or TF-IDF. TriagerX's compact design enables faster training, making it more suitable for deployment than larger PLM-based models (i.e., DeBERTa-large-based CNN classifiers has over **50% more parameters** than TriagerX CBR). TriagerX also outperforms other approaches on literature datasets[2].

Training & Evaluation

We train and evaluate TriagerX using issue data from 50 developers (each with at least 20 assigned issues) from the OpenJ9 repository, comprising 3,375 training samples and 382 test samples.

Table 1: Top-K accuracy of TriagerX compared to other methods in the Openj9 dataset.

Method	K=1	K=3	K=5	K=10
TriagerX-Full	**0.327**	**0.533**	**0.633**	**0.807**
TriagerX CBR	0.272	0.476	0.601	0.780
TriagerX IBR	0.284	0.488	0.585	0.699
DeBERTa-L (FCN)	0.178	0.418	0.547	0.698
RoBERTa-L (FCN)	0.191	0.418	0.586	0.743
BERT-L (FCN)	0.168	0.393	0.507	0.694
CodeBERT (FCN)	0.129	0.331	0.476	0.689
DeBERTa-L (CNN)	0.170	0.374	0.503	0.675
RoBERTa-L (CNN)	0.206	0.403	0.531	0.670
BERT-L (CNN)	0.181	0.323	0.445	0.652
CodeBERT (CNN)	0.100	0.253	0.409	0.595
LBT-P	0.211	0.407	0.501	0.631
DBRNN-A	0.127	0.300	0.454	0.627
TF-IDF + SVM	0.189	0.357	0.484	0.665

Deployment & Future Works

- The framework is currently deployed in the Eclipse OpenJ9 GitHub repository, where it predicts both developer and component labels for each new issue created. The framework operates in a Docker container for efficient deployment and maintenance, providing recommendations through GitHub bot comments and Slack notifications. The deployment pipeline is
- Future work will focus on (1) **developing a feedback-driven, auto-adaptive bug triaging framework** that learns from user input to enhance its recommendations, and (2) **providing contextual explanations** for recommendations to boost triagers' confidence.

Figure 3 shows the integration of the framework into the OpenJ9 repository along with an example of recommendation for an existing issue.

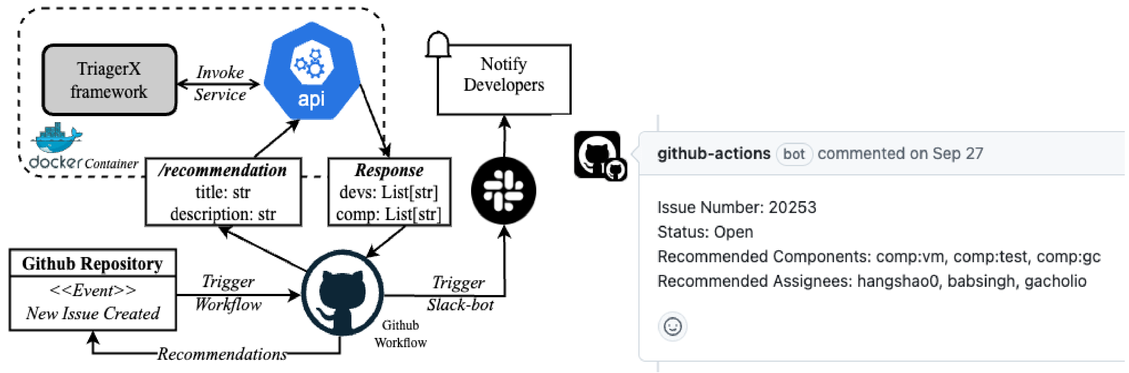

Figure 3: (Left) Deployment pipeline of TriagerX. (Right) Example of a recommendation made by the bot for an issue in the OpenJ9 repository.

References

1. Jaehyung Lee, Kisun Han, and Hwanjo Yu. 2023. A Light Bug Triage Framework for Applying Large Pre-trained Language Model. In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22). Association for Computing Machinery, New York, NY, USA, Article 3, 1–11.
2. Senthil Mani, Anush Sankaran, and Rahul Aralikatte. 2019. DeepTriage: Exploring the Effectiveness of Deep Learning for Bug Triaging. In Proceedings of the ACM India Joint International Conference on Data Science and Management of Data (CODS-COMAD '19). Association for Computing Machinery, New York, NY, USA, 171–179.

Md Afif Al Mamun :: LinkedIn/afifaniks
Gias Uddin :: LinkedIn/giasuddin
Lan Xia (IBM) :: LinkedIn/lanxia
Longyu Zhang (IBM) :: LinkedIn/LongyuZhang