HAAG Weekly Report Week 11

Mercedes Quintana

Time-Log

What did you do this week?

- Worked to connect backend and frontend
- Emailed Oday Abdulazeez about a possible server
- Updated website
- Prepared and had computation meeting

What are you going to do next week

- Keep website updated
- Meet with Ayush about help with connecting the front and backend

Blockers, things you want to flag, problems, etc.

- None

Abstracts:

Link: https://dl.acm.org/doi/10.1145/3705309

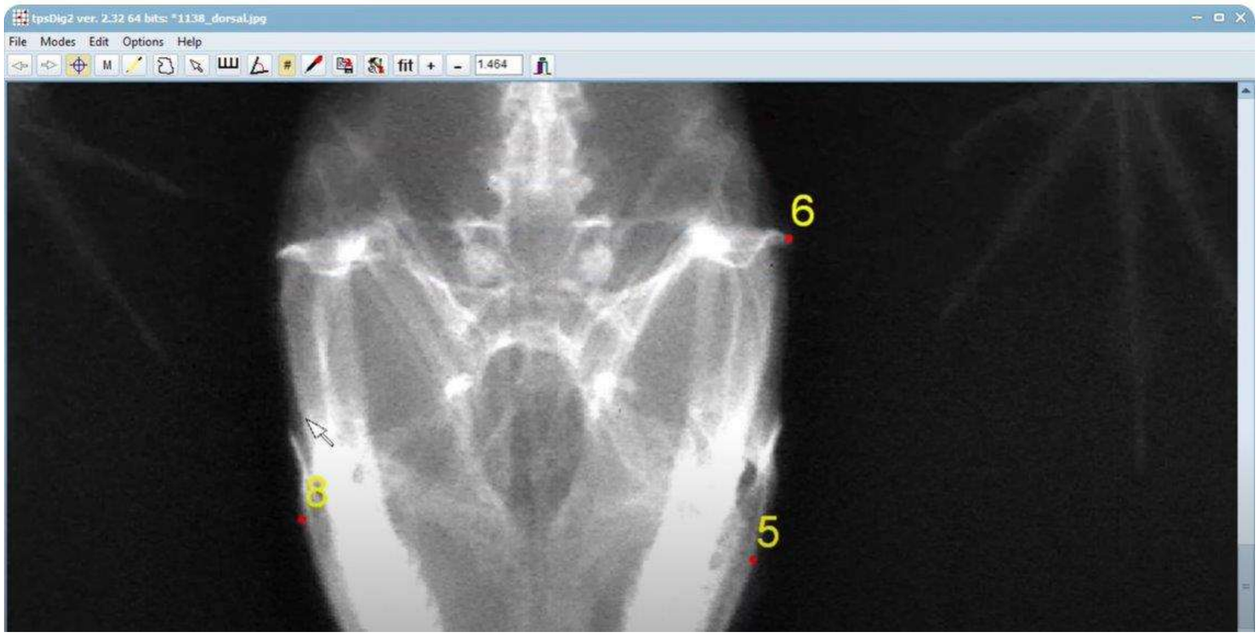
Detecting Refactoring Commits in Machine Learning Python Projects: A Machine Learning-Based Approach

Refactoring aims to improve the quality of software without altering its functional behaviors. Understanding developers' refactoring activities is essential to improve software maintainability. The use of machine learning (ML) libraries and frameworks in software systems has significantly increased in recent years, making the maximization of their maintainability crucial. Due to the data-driven nature of ML libraries and frameworks, they often undergo a different development process compared to traditional projects. As a result, they may experience various types of refactoring, such as those related to the data. The state-of-the-art refactoring detection tools have not been tested in the ML technical domain, and they are not specifically designed to detect ML-specific refactoring types (e.g., data manipulation) in ML projects; therefore, they may not adequately find all potential refactoring operations, specifically the ML-specific refactoring operations. Furthermore, a vast number of ML libraries and frameworks are written in Python, which has limited tooling support for refactoring detection. PyRef, a rule-based and state-of-the-art tool for Python refactoring detection, can identify 11 types of refactoring operations with relatively high precision. In contrast, for other languages such as Java, state-of-the-art tools are capable of detecting a much more comprehensive list of refactorings. For example, Rminer can detect 99 types of refactoring for Java projects. Inspired by previous work that leverages commit messages to detect refactoring, we introduce MLRefScanner, a prototype tool that applies ML techniques to detect refactoring commits in ML Python projects. MLRefScanner detects commits involving both ML-specific refactoring operations and additional refactoring operations beyond the scope of state-of-the-art refactoring detection tools. To demonstrate the effectiveness of our approach, we evaluate MLRefScanner on 199 ML open source libraries and frameworks and compare MLRefScanner against other refactoring detection tools for Python projects. Our findings show that MLRefScanner outperforms existing tools in detecting refactoring-related commits, achieving an overall precision of 94% and recall of 82% for identifying refactoring-related commits. MLRefScanner can identify commits with ML-specific and additional refactoring operations compared to state-of-the-art refactoring detection tools. When combining MLRefScanner with PyRef, we can further increase the precision and recall to 95% and 99%, respectively. MLRefScanner provides a valuable contribution to the Python ML community, as it allows ML developers to detect refactoring-related commits more effectively in their ML Python projects. Our study sheds light on the promising direction of leveraging machine learning techniques to detect refactoring activities for other programming languages or technical domains where the commonly used rule-based refactoring detection approaches are not sufficient.

Summary: Keeping up with software updates in applications in Python is more difficult than in other languages, where the refactoring software is not as plentiful. This group created a ML approach which detects refactoring, and present a prototype to deal with refactoring for ML libraries and non-ML libraries.

What did you do and prove it

This week I worked to connect the front and backend, emailed about getting a server, and had my bi-weekly computation meeting. I have been struggling connecting the backend to the frontend. Even though I was able to get help from my advisor, to get one component to the frontend, I still struggled to make any more progress. I have a meeting with Ayush this afternoon, and I am hopeful he can help me because he has a lot of experience with web development. Below is a capture of the video Jon sent to show his current process, we are happy our current zoom practices show address his concerns about his current method.


tpsDig2 ver. 2.32 64 bits: *1138_dorsal.jpg
File Modes Edit Options Help
fit
1.464
6
8
5