Capstone Project Outline

I will be testing active learning approaches for use in source code analysis, particularly the classification (True Positive, False Positive, Indeterminate) of CERT coding rule alerts, in a similar manner to this paper - <https://resources.sei.cmu.edu/asset_files/ConferencePaper/2018_021_001_524697.pdf>.

The paper outlines a procedure they used to develop models for classification with the Random Forest, XGBoost, CART, and Lasso Logistic Regression algorithms. They however, built the model from the complete training data and then used the model to predict on their testing set. Instead, I will be researching a method to do “active learning” to classify source code analysis alerts, essentially labeling the data as being a True or False prediction as we go along building the model. This involves a process for also prioritizing the unlabeled alerts that are not yet part of our model, to choose which of the next alert is the “best” one to audit and reveal whether the prediction was True or not.

Since we do not have the resources to full audit alerts that are flagged for a large codebase, we will be using three training datasets located at <https://samate.nist.gov/SATE6ClassicTrack.html>, which are preinjected with bugs and has correct labels for True or False positives for different tool outputs.

Initially, we can build complete models on the whole training set in order to set a benchmark for correct classification %. We can then test different active learning approaches to test how quickly different approaches converge to the benchmark, as well as tracking what the associated cost of each different approach is. A good starting point would be Alipy (<http://parnec.nuaa.edu.cn/huangsj/alipy/index.html>), which is an open source module for active learning. In particular, I would like to target active learning approaches for the Random Forest and XGBoost machine learning algorithms, since they were the best overall performing algorithms in the CERT Paper.

Further along the project, depending on progress, I would also like to start building a framework for the tradeoff an auditor of alerts would make between auditing an alert that would provide the most information to the model (improving it the most), and simply resolving the most risk associated error in the code.

Steps

1. Download Samate datasets
2. Run the samate datasets through SCALe, using all of the static analysis tools
3. Analyze the datasets with lizard, a code complexity tool
4. Split each of the datasets, together with the coding alerts that related to that split into training sets, and use the alerts+code complexity output to build a model on the training sets.
5. Test on the testing set to attain benchmark rates for True Positive, False Positive, and indeterminate.
6. Use various different active learning approaches to estimate how quickly and at what cost they approach the benchmark performance.

Deliverable: Technical paper outlining research, and the performance of different active learning approaches with regards to source code analysis.