Option B (Empirical evaluation)

What is the problem? Why is it important?

Predicting the build failure type without executing the build will help the developer identify issues earlier in the development process. Also, it allows the CI services to prioritize the builds and the automatic correction tools to suggest a fix. There are studies in the literature proposing approaches to predict the result of a build [1,2]. They use logistic regression and random forest to predict the output of the build. As a result, they collect metrics from the code change to feed to the model. However, in this study, we want to use a deep neural network to predict the reason for build failure using the changed code itself. These reasons could be a missing dependency, configuration error, or typo in the build script or source files.

Techniques to tackle the problem

We will first gather build info for the Java projects from TravisCI API. Using code2vec [4] and git diff, we will feed the changed code to an LSTM. Then, we will put a fully connected neural network in front of the LSTM and narrow the network down to the number of reasons a build may fail and then put a single neuron in front of it. First, we train the network to classify if a build fails or not. Then, we remove that single node and use a few samples to train the categories obtained using the method described in Gallaba et al.'s study [3].

[1] Hassan, Ahmed E., and Ken Zhang. "Using decision trees to predict the certification result of a build." *21st IEEE/ACM International Conference on Automated Software Engineering (ASE'06)*. IEEE, 2006.

[2] Hassan, Foyzul, and Xiaoyin Wang. "Change-aware build prediction model for stall avoidance in continuous integration." *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2017.

[3] Gallaba, Keheliya, et al. "Noise and heterogeneity in historical build data: an empirical study of Travis CI." *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 2018.

[4] Alon, Uri, et al. "code2vec: Learning distributed representations of code." *Proceedings of the ACM on Programming Languages* 3.POPL (2019): 1-29.