Which Refactoring Reduces Bug Rate?

Supplementary Materials

We provide here the supplementary materials of the “Which Refactoring Reduces Bug Rate?”.

This is a temporary location, for the review process.

The supplementary materials include:

* The labeling protocol
* Data
  + Labeled samples
  + Queries results
  + Analysis results (e.g., confusion matrices)
* Code
  + Sql queries desigein for the BigQuery GitHub repository
  + Python scripts for data extraction and analysis

# Data

## Samples

* Train - used for labeling commit type for training. Based on taxonomy enhancement of the CCP paper data sets English\_samples and model\_validation\_samples
* Ccp\_not\_improving\_refactors.csv - samples of refactor commits that didn’t reduce the CCP (not labeled). Produced by commit\_by\_refactor\_efficiency\_samples
* Ccp\_improving\_refactors.csv - samples of refactor commits that didn’t reduce the CCP (not labeled). Produced by commit\_by\_refactor\_efficiency\_samples

## Analysis results

* repos2018 -repositories in scope. The output of the query “repo properties”.
* repos2018\_api- The result of the GitHub API query. Used to identify forks.
* Ccp\_changing\_tokens.csv - token, their frequency and CCP reducing refactor probability. Produced by compare\_word\_dist.py

## Distributions

* File\_commit\_dist - commits per source code file in 2018
* File\_hit\_rate\_dist - source code hit rate distribution
* File\_hit\_rate\_dist\_atleast\_10 - source code files with at least 10 commits during 2018 distribution
* Refacor\_per\_repo\_dist.csv - refactor per repo (in 2018)
* refactor\_fix\_adaptive\_CM - is the refactor also fix or adaptive

# Code

The file data flow explain how the data was generated

## Queries

The queries are run on the [GitHub database available in Google's BigQuery](https://bigquery.cloud.google.com/dataset/bigquery-public-data:github_repos) infrastructure.

Most queries were run directly on the GitHub tables.

Some flows included intermediate tables. These table are stored to a schema that is unique per user. When running these queries, the schema name should be named to the schema name given to your user.

* Repo properties - select the repositories in scope and collects the linguistic models results on them
* ETL - Extract, Transfer and Load: Moving the data to a convenient structure
* File major extension - extensions distribution
* Hotspots - hot spots and file ccp analysis
* classify\_valid\_commits - label al valid commits (breaked into file level) with linguistic models
* Create\_commit\_size - number of files (of different types) in a commit
* Create\_refactor\_stats - computes before/after statistics of refactors
* Refactor\_descreptive\_stats - descriptive statistics of the 2018 refactors

## Python

* Extract\_projects\_properties.py - queries the GitHub API. Identifies forks.
* file\_hit\_rate\_dist.py - compute file\_ccp distribution
* File\_ccp\_stability.py - compute file ccp stability
* C\_projects.py - set the scope of C projects
* Commit\_type\_model - python implementation of commit type linguistic model
* Linguistic\_models\_performance - utilities for linguistic model evaluation and improvement
* Compare\_word\_dist.py - token, their frequency and CCP reducing refactor probability