The Corrective Commit Probability Code Quality Metric Supplementary Materials

We provide here the supplementary materials of “The Corrective Commit Probability Code Quality Metric”.

The supplementary materials include:

* Data flow – instructions for reproducing the research. Longer description of the components appears in the rest of the file.
* The labeling protocol
* Data
  + Labeled samples
  + Queries results
  + Analysis results (e.g., confusion matrices)
* Code
  + Sql queries designed for the BigQuery GitHub repository
  + Python scripts for data extraction and analysis

Data flow

1. Using the query “repo\_properties.sql” the table repos2018 was generated. Table was exported to the file repos2018.
2. Using the Python file extract\_projects\_properties.py, the GitHub API was queried on the files repos2018. Results were saved to repos2018\_api.
3. Run build\_repo\_file.py to generate valid\_repos, repos\_full and statistics about file scope

# Labeling protocol

Appears in the Labeling protocol\Bug fix labeling protocol file.

Describe the way a commit classification as corrective was done

# Data

## Samples

Appear under data\samples.

Contains commits that were used for the research.

* Commit\_train\_11\_aug\_18 - 840 samples, 40 are duplicated. Classified for corrective. Used as the train set.
* Commits\_test\_12\_aug\_18 - 452 samples, 52 duplicated. Classified for corrective. Used as part of the train set.
* English samples - 285 random samples, 83 duplicated. Were classified for being in English in order to evaluate the English model. Commits were also classified as corrective and were part of the test set.
* Bq\_test2\_general\_full\_v8 - 413 samples, 13 duplicated. This data set includes the annotation of three annotators. It was used for subjectiveness estimation, bounds and as part of the test set.
* Model\_validation\_samples - 500 samples, 100 duplicates used for the maximum likelihood validation
* Low\_bound\_2016\_samples - 500 samples sampled randomly from project in the scope of the research, whose CCP was below the valid domain, from the year 2016. Samples were not labeled but the predictions of the bug and English model are provided and so is the message length.

## Analysis results

* Fix confusion matrix - model performance with respect to various data sets.
* Bootstrap\_diff - differences validation (note, this is generated with randomness due to the bootstrap)
* Contribution\_twins\_stat - twin experiment on development speed

## Queries results

Queries were run on BigtQuery GitHub repositories, available at <https://bigquery.cloud.google.com/dataset/bigquery-public-data:github_repos>

### Project properties

Results by project

* Repositories\_properites - repositories in scope. The output of the query “repo properties”. Github properties (e.g., stars) that were added using the “extract\_project\_porpeties” were merged in.
* Rep\_no\_fork.csv - the repositories with forks removed
* Rep\_with\_fork.csv - fork true/false/none indication per repository
* Users\_per\_project - number of users per project with activity per year
* Avg\_commit\_size\_per\_project - coupling data
* Commits\_per\_user\_cap - development speed

## Distributions

* Hit\_rate\_2016 - CCP distribution
* Commits\_per\_user\_dist - Distribution of commits per user (how many made 1 commit, 2 commits, etc.)
* User\_repo\_contribution\_with\_file\_scoped\_repos\_dist - as Commits\_per\_user\_dist but for projects in scope
* English Distribution - English linguistic model distribution
* Extension\_file\_size - files per extension
* Files\_per\_commit\_dist - coupling
* Programming\_file\_size\_dist - file size distribution of programming languages file. The files are aggregated together, not separated per language.
* Extension\_file\_size - file size by extension (of language)

# Code

## 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.sql - Repositories in scope (including forks and those whose CCP is not in the valid range)
* Commits sample - examples of commits for labeling
* Low bound 2016 samples - Sample commits from projects with negative CCP
* File size analysis - extracting the data needed for file size analysis
* Files per commit distribution - Generates the files per commit distribution for the coupling analysis
* Twins - coupling and development speed twin experiments
* User commits - Commit per user, used for development speed analysis

## Python

* Requirements.txt - python libraries needed for the scripts
* Extract\_projects\_properties.py - These scripts extract the repositories properties from the GitHub api. The output was merged into the Repositories\_properites data file. Using the API requires a user and password at git, which can be obtained for free.
* Defect\_fix\_analyzer.py - Python implementation of linguistic models. Word cloud generation.
* Prepare\_hit\_rate\_dist.py - Compute the CCP distribution and analyze it.
* Boostrap.py - Performs Bootstrap in order to evaluate MLE sensitivity to data
* Bootstrap\_figure\_plotly.py - create the bootstrap figure
* Bootstrap\_models.py - Using the bootstrap model in order to compare resulting models.
* Star\_analyis.py - analyze the stars and quality correlation.
* File\_size\_analysis.py - Analyze file size with respect to quality
* File\_size\_extension\_analysis.py - Analyze file size per language and generates the relevant figure.
* Coupling\_analysis.py - Analyze coupling and quality.
* Commits\_per\_user.py - Analyze development speed with respect to quality.
* Ccp\_per\_lang\_figure.py - generate language quality comparison figure
* Length\_per\_lang\_figure.py - generate length quality comparison figure
* Speed\_and\_quality\_per\_language\_figure\_and\_table.py - Table (and a figure) of speed and CCP per language
* Language\_ccp\_anova.py - Anova test computation of languages CCP