1 Preprocessed MLCQ dataset

This repository contains the replication package for the paper " Automatic detection of Long Method and God Class code smells through neural source code embeddings," accepted for publication in the *Expert Systems with Applications* journal.
We used the MLCQ dataset for God Class and Long Method code smell detection in our experiments:
Madeyski, L. and Lewowski, T., 2020. MLCQ: Industry-relevant code smell data set. In Proceedings of the Evaluation and Assessment in Software Engineering (pp. 342-347).
publicly available at https://zenodo.org/record/3666840#.YnOJ1ehBwuU. We extracted the following features:

- code2vec neural source code embeddings – we used the implementation provided by Alon et al. https://github.com/tech-srl/code2vec.
- code2seq neural source code embeddings – we used the implementation provided by Alon et al. https://github.com/tech-srl/code2seq.
- CuBERT neural source code embeddings – we used the pre-trained Java model available at https://github.com/google-research/google-research/tree/master/cubert.
- Source code metrics – we extracted the metrics values by using the following metric extraction tools:
  - CK Tool https://github.com/mauricioaniche/ck/
  - RepositoryMiner https://github.com/antoineBarbez/RepositoryMiner/.

1.1 code2vec

We extracted the 384-dim vectors for God Class and Long Method code snippets in the MLCQ dataset. The embeddings are available in pickle DataFrames (column embedding):

- God Class – */Data/God_Class_CuBERT_embeddings.pkl*
- Long Method – */Data/Long_Method_CuBERT_embeddings.pkl*

1.2 Code2seq

We extracted the 320-dim vectors for God Class and Long Method code snippets in the MLCQ dataset. The embeddings are available in pickle DataFrames (column embedding):

- God Class
  - Training set - */Data/God_Class_code2seq_train.pkl*
  - Test set - */Data/God_Class_code2seq_test.pkl*
- Long Method
  - Training set - */Data/Long_Method_code2seq_train.pkl*
  - Test set - */Data/ Long_Method_code2seq_test.pkl*

1.3 CuBERT neural source code embeddings

We extracted the 1024-dim vectors for God Class and Long Method code snippets in the MLCQ dataset. The embeddings are available in pickle DataFrames (column embedding):

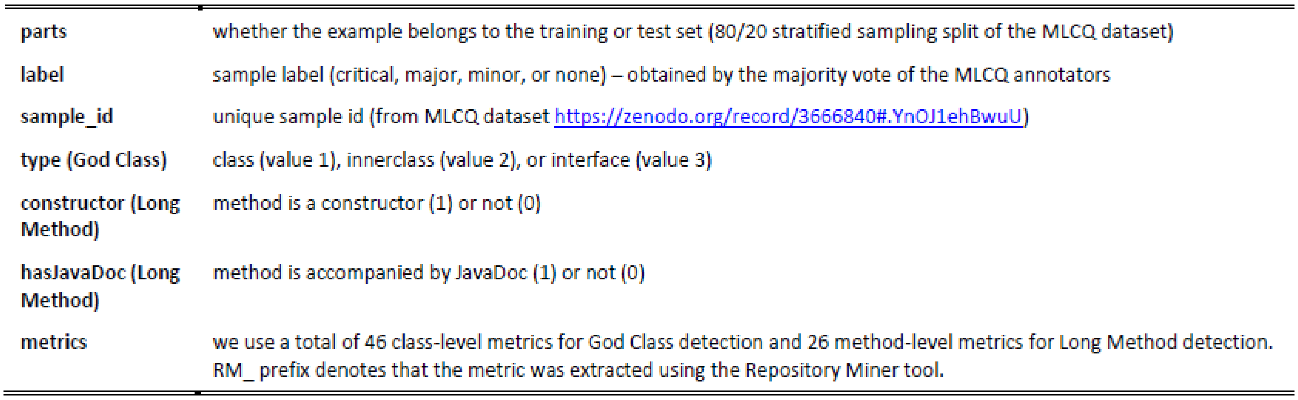
- God Class – */Data/God_Class_CuBERT_embeddings.pkl*
- Long Method – */Data/Long_Method_CuBERT_embeddings.pkl*

1.4 Source code metrics

We provide two excel files with original metrics values:

- God Class – */Data/God_Class_code_metrics_values.xlsx*
- Long Method – */Data/Long Method/Long_Method_code_metrics_values.xlsx*

Table 1 Column desciption



parts	whether the example belongs to the training or test set (80/20 stratified sampling split of the MLCQ dataset)
label	sample label (critical, major, minor, or none) – obtained by the majority vote of the MLCQ annotators
sample_id	unique sample id (from MLCQ dataset https://zenodo.org/record/3666840#.YnOJ1ehBwuU)
type (God Class)	class (value 1), innerclass (value 2), or interface (value 3)
constructor (Long Method)	method is a constructor (1) or not (0)
hasJavaDoc (Long Method)	method is accompanied by JavaDoc (1) or not (0)
metrics	we use a total of 46 class-level metrics for God Class detection and 26 method-level metrics for Long Method detection. RM_ prefix denotes that the metric was extracted using the Repository Miner tool.

We apply the heuristic-base detectors on the original metric values. However, for training Machine Learning classifiers, we apply additional preprocessing:

- God Class metrics TCC and LCC: The value of these metrics should be between 0 and 1. Value -1 denotes that the metric cannot be calculated (classes with 0 or 1 method). Semantically, value -1 is closer to 1 (perfect cohesion).
- We normalize all columns (z-normalization obtained using the mean and standard deviation of the training data) except: type, constructor, and hasJavaDoc.
- Using the column parts, we separate the dataset into the training (train_X) and test (test_X) features. We exclude the following columns from these features: parts, label, sample_id.
- We create the corresponding labels train_y and test_y (row *k* in train_y corresponds to row *k* in train_X). We binarize these labels (“none” corresponds to FALSE (non-smell), while “minor”, “major”, and “critical” correspond to TRUE (smell)).

Source code metrics preprocessed for applying ML classifiers are available in folders:

- God Class - */Data/ MLCQ metrics_processed*
- Long Method - */Data/MLCQ metrics_processed*.