PyART: Python API Recommendation in Real-Time

Xincheng He*, Lei Xu†, Xiangyu Zhang‡, Rui Hao*, Yang Feng* and Baowen Xu*
* State Key Laboratory for Novel Software Technology, Nanjing University, China
† Purdue University, USA
xinchenghe2016@gmail.com, xlei@nju.edu.cn, xyzhang@cs.purdue.edu,
rui.hao.gm@gmail.com, fengyang@nju.edu.cn, bwxu@nju.edu.cn

Abstract—This is the research artifact of the paper titled 'PyART: Python API Recommendation in Real-Time'. PyART is a real-time API recommendation tool for Python, which includes two main functions: data-flow analysis and real-time API recommendation for both incomplete and complete Python code context. Compared to classical tools, PyART has two important particularities: it is able to work on real-time recommendation scenario, and it provides data-flow analysis and API recommendation for dynamic language. Classical tools often fail to make static analysis in real-time recommendation scenario, due to the incompletion of syntax. And the dynamic features of Python language also bring challenges to type inference and API recommendation. Different from classical tools, PyART derives an optimistic data-flow that is neither sound nor complete but sufficient for API recommendation and cost-effective to collect, and provides real-time API recommendations based on novel candidate collection, context analysis and feature learning techniques. The artifact evaluation experiments of PyART include three main aspects: data-flow analysis, intra-project API recommendation and across-project API recommendation. We assume users of the artifact is able to use Linux Ubuntu Operating System.

Index Terms—API recommendation, context analysis, data flow analysis, real-time recommendation, Python

I. INSTALLATION

Installation of PyART. The artifact can be downloaded from the website https://github.com/PYART0/PyART-demo, with the ‘git clone’ command.

Requirements for Starting the artifact. PyART is evaluated on Ubuntu 18.04 operating system. The operating system can be downloaded on https://ubuntu.com/download/desktop, or you can download *.iso file from http://mirrors.163.com/ubuntu-releases/18.04/ and install it on vmware tools. The following software environments are needed: Java (default version) and Python (version 3.6+). Since there is Python 3.6+ installed in original ubuntu 18 OS, users just need to install Java environment. Users can use the ‘apt-get install’ command to make it. Several Python libraries are needed: pylearn, sklearn, pandas, joblib and nltk. Users can use the ‘pip install’ command to make it. Specific installation instructions can be found in the README file of the Github website.

II. ARTIFACT CONTENT

After loading the artifact, the PyART-demo directory contains the following contents:

• A README file that describes basic information of PyART, including execution requirements, installation instructions, expected results and so on.
• A LICENSE file that describes the license of PyART.
• The working directory of data-flow analysis evaluation (PyART-demo/DataflowEvaluation), including data, results, baseline, code implementation and running scripts. The structure is listed as follows:
  – data folder, including input projects.
  – results folder, including output of the baseline tool Pysonar2 and PyART.
  – get_dataflow.py, the implementation file of PyART for data-flow analysis for Python.
  – main_dataflow.py, the entry file of data-flow analysis of PyART.
  – runpystyping.sh, the entry file of data-flow analysis of baseline Pysonar2.
• The working directory of API recommendation evaluation (PyART-demo/PyART), including data, results, baseline, code implementation and running scripts. The structure is listed as follows:
  – output folder, including intermediate files and output files.
  – srilm-1.7.2 folder, including N-GRAM model training tool.
  – testdata folder, including testing projects.
  – testJson folder, including PROJ.json files that contain API definition information of the testing project PROJ.
  – traindata folder, including training projects.
  – trainJson folder, including PROJ.json files that contain API definition information of the training project PROJ.
  – traincsv folder, including feature vectors and labels for training in *.csv files and constructed models in *.pkl for intra-project recommendation.
  – traincsv-1 folder, including feature vectors and labels for training in *.csv files and constructed models in *.pkl for across-project recommendation.
  – aget_train_kfold.py, the entry for the training process of intra-project recommendation, which extracts feature vectors from ( 1-k-1)th-fold of projects in testdata folder and outputs vectors and labels to traincsv/*.csv.

• The working directory of API recommendation evaluation (PyART-demo/PyART), including data, results, baseline, code implementation and running scripts. The structure is listed as follows:
  – output folder, including intermediate files and output files.
  – srilm-1.7.2 folder, including N-GRAM model training tool.
  – testdata folder, including testing projects.
  – testJson folder, including PROJ.json files that contain API definition information of the testing project PROJ.
  – traindata folder, including training projects.
  – trainJson folder, including PROJ.json files that contain API definition information of the training project PROJ.
  – traincsv folder, including feature vectors and labels for training in *.csv files and constructed models in *.pkl for intra-project recommendation.
  – traincsv-1 folder, including feature vectors and labels for training in *.csv files and constructed models in *.pkl for across-project recommendation.
  – aget_train_kfold.py, the entry for the training process of intra-project recommendation, which extracts feature vectors from ( 1-k-1)th-fold of projects in testdata folder and outputs vectors and labels to traincsv/*.csv.
- generateclf.py, the file that takes traincsv/*.csv files as input and outputs trained model files to traincsv/*.pkl.
- aget_test_result.py, the entry for the testing process of intra-project recommendation evaluation, which takes the kth-fold of the projects in testdata folder as input and outputs expected API, Top-k accuracy and MRR value for each recommendation point.
- get_train_kfold.py, the entry for the training process of across-project recommendation, which extracts feature vectors from training files in traindata folder and outputs vectors and labels to traincsv-1/*.csv.
- ac_generateclf.py, the file that takes traincsv-1/*.csv files as input and outputs trained model to files traincsv-1/*.pkl.
- bget_test_result.py, the entry for the testing process of across-project recommendation evaluation, which takes projects in testdata folder as input and outputs expected API, Top-k accuracy and MRR value for each recommendation point.
- getudbs.py, the file that gets *.udb file from original *.py files.
- getJson.pl, the file that gets *.json file from original *.udb files.
- trainfile.lm, the pre-trained ngram model for data-flow feature generation.
- built-in.txt, including built-in APIs.
- typeshed.txt, including typeshed APIs.

III. EVALUATION

The evaluation of PyART includes two tasks, Task1 (Dataflow evaluation) and Task2 (API recommendation evaluation).

To run PyART for Task1, use the command 'python3 main_dataflow.py'. Users can use the command 'java -Xms3550M -jar pystyping-3.0-milestone.jar data/PROJ results/PROJ -clearcache' or './runpystyping.sh' to run the baseline Pysonar2 for Task1, in which PROJ denotes project names. If users want to reproduce Task1 for other projects, they can store their projects in data folder and change the 'root path' variable to their project names in main_dataflow.py in line 39.

To run the training process of intra-project recommendation in Task2, use the 'python3 aget_train_kfold.py' command. The result of the command, which is a set of feature vectors and labels, is output to *.csv files in traincsv dictionary. Then use the 'python3 generateclf.py' command to generate RandomForest model, stored as *.pkl in traincsv dictionary. To reproduce training for other projects, users should store their projects in testdata folder, change the 'CURRENT_PROJ' in aget_train_kfold.py in line 1329, and change the 'proj_name' variable in generateclf.py in line 6 as their own project names. To run the testing process of intra-project recommendation in Task2, use the 'python3 aget_test_result.py' command. The recommendation results are printed in real time. To reproduce testing for other projects, users should store their projects in testdata folder, change the 'CURRENT_PROJ' in aget_test_result.py in line 1463.

To run training process of across-project recommendation in Task2, use the 'python3 ac_generateclf.py' command. The result of the command, which is a set of feature vectors and labels, is output to *.csv files in traincsv-1 dictionary. Since the training data for across-project is large, we only put a single project in traindata folder, users can get our all training projects from Github with TrainDataList.txt. Then use the 'python3 ac_generateclf.py' command to generate RandomForest model, stored as total.pkl in traincsv-1 dictionary. To run testing process of across-project recommendation in Task2, use the 'python3 bget_test_result.py' command. To reproduce testing for other projects, users should store their projects in testdata folder, change the 'CURRENT_PROJ' in bget_test_result.py in line 1463.

IV. RESULTS

For the data-flow analysis evaluation, the evaluation metrics include Precision, Recall and F1-score. Since there is not any tool that can give completely accurate analysis for the task, we take manual check records as the ground truth, recorded in PyART-demo/DataFlowEvaluation/ManaulCheckResults.xlsx. Results of testing cases show that PyART's data-flow results align well with the human baseline (92.06%-98.66% precision and 92.58%-98.03% recall).

For the API recommendation analysis evaluation, the evaluation metrics include Top-k accuracy (k=1,2,3,4,5,10) and MRR. In the process of running test cases, PyART prints Top-10 recommendation results for each recommendation point and gives lastest Top-k accuracy and MRR values. Results of testing cases show that the Top-1 accuracy of PyART is generally high (i.e., up to 70.49%) for intra-project recommendation, and PyART is still effective for across-project recommendation.