Towards Connecting Requirements with Developer Artifacts in a Local Context: Supplemental Material

Sonora Halili, Smith College, Northampton, MA USA
Karenna Kung, Smith College, Northampton, MA USA
Paola Spoletini, Kennesaw State University, Marietta, GA USA
Alicia M. Grubb, Smith College, Northampton, MA USA

Purpose

This repository contains the supplemental information for the paper: “Towards Connecting Requirements with Developer Artifacts in a Local Context”, which was accepted to the 31st International Working Conference on Requirement Engineering: Foundation for Software Quality (REFSQ’25) in 2025.

This repository includes issues and requirements from three projects from the PURE dataset. It also contains code for the propsed pipeline discussed in the paper and the manual clusters and matches that were treated as the ground truth in the paper.

Contents

- **RQ-1.ipynb** contains the code to generate the information in Table 1, and saves four files that begin with **[projectname]-clusters**
- **RQ-2.ipynb** contains the code to generate the information in Table 2, and saves four files that begin with **[projectname]-matches**
- **[projectname]-issues.csv** files contain issue IDs and the issues themselves for a given project
- **[projectname]-reqs.csv** files contain requirement IDs and the requirements themselves for a given project
- **[projectname]-reqs-cluster-oracle.csv** files contain ground truth cluster numbers for each requirement as created by the oracle researcher for a given project, with arbitrary cluster numbers
- **[projectname]-reqs-matches-oracle.csv** files contain ground truth matches between issues and requirements as created by the oracle researcher for a given project

The **results** folder contains the data presented in the tables in the paper.

- **[projectname]-clusters-brute.csv** files contain requirement IDs and their arbitrary cluster numbers, generated by the deterministic brute force algorithm
- **[projectname]-clusters-kmeans[#ofmanclusters].csv** files contain requirement IDs and their arbitrary cluster numbers, generated via the KMeans clustering method in which the number of clusters is set to the number of clusters created manually
- **[projectname]-clusters-kmeans[#ofbruteclusters].csv** files contain requirement IDs and their arbitrary cluster numbers, generated via the KMeans clustering method in which the number of clusters is set to the number of clusters generated by the brute force algorithm
- **[projectname]-clusters-agg.csv** files contain requirement IDs and their arbitrary cluster numbers, generated via the agglomerative clustering method
- **[projectname]-matches-byclustering.csv** files contain issue IDs and their matched requirement IDs, generated by clustering all issues and requirements
- **[projectname]-matches-kmeans[#ofmanclusters].csv** files contain issue IDs and their matched requirement IDs, generated by clustering all requirements via the KMeans clustering method in which the number of clusters is set to the number of clusters created manually, and then matching issues to the nearest requirement cluster center
- **[projectname]-matches-kmeans[#ofbruteclusters].csv** files contain issue IDs and their matched requirement IDs, generated by clustering all requirements via the KMeans clustering method in which the number of clusters is set to the number of clusters generated by the brute force algorithm, and then matching issues to the nearest requirement cluster center

- [**projectname**]**-matches-agg.csv** files contain issue IDs and their matched requirement IDs, generated via the agglomerative clustering method, and then matching issues to the nearest requirement cluster center

Note that clustering methods are nondeterministic, so running the code files will not necessarily generate the exact files in the **results** folder. Also note that cluster numbers are arbitrary, so clusters that contain the same requirement IDs but have different cluster IDs are equivalent.

The `projectname` can be any of `gammaj`, `keepass`, or `peering`.

Setup and Usage

The following packages are used in the Jupyter notebook files:

- `pandas`
- `numpy`
- `sentence_transformers`
- `sklearn`

This code was originally run on Google Colab. Place all the files in one folder. As indicated by the inline comments, update the file path to this folder in **RQ-1.ipynb** and **RQ-2.ipynb**. Additionally, update the project line (`gammaj`, `keepass`, or `peering`). Clusters generated via the four investigated methods are saved when **RQ-1.ipynb** is run, and can be found in files beginning with [**projectname**]**-clusters**. Compare these results to the clusters in the project's [**projectname**]**-reqs-cluster-oracle.csv** file. Matches generated between issues and requirements are saved when **RQ-2.ipynb** is run, and can be found in files beginning with [**projectname**]**-matches**. Compare these results to the clusters in the project's [**projectname**]**-matches-oracle.csv** file.